Simulation Photoréaliste de l'éclairage en Synthèse d'Images

Nicolas Holzschuch

Simulation Photoréaliste de l'Éclairage en Synthèse d'Images

Nicolas Holzschuch

Mémoire présenté pour l'obtention de l'Habilitation à Diriger des Recherches de l'Université Joseph Fourier — Grenoble I, Spécialité Mathématiques et Informatique

Composition du jury :

Sumanta Pattanaik	Rapporteur
Bernard Péroche	Rapporteur
Hans-Peter Seidel	Rapporteur
George-Pierre Bonneau	Président
George Drettakis	Examinateur
Claude Puech	Examinateur
François Sillion	Examinateur

Habilitation préparée au sein de l'équipe ARTIS du laboratoire LJK. Le LJK est l'UMR 5224, un laboratoire commun au CNRS, à l'INPG, à l'Université Joseph Fourier et à l'Université Pierre Mendès-France. ARTIS est une équipe du LJK et un projet de l'INRIA Rhône-Alpes.

À Myriam,
À Henry, Erik et Lena,
ma famille qui me supporte... et qui me soutient.

Table des matières

Table des figures

1.
Introduction

Les techniques de rendu en Synthèse d'Images produisent des images d'une scène virtuelle en prenant comme point de départ la *définition* de cette scène : les objets qui la composent, leur position, leurs matériaux, mais aussi la position de l'observateur.

Parmi les techniques de rendu, on distingue les techniques de rendu *Photo-Réalistes*, qui cherchent à produire une image aussi proche que possible de la réalité, en simulant les échanges lumineux à l'intérieur de la scène. On obtient ainsi une image de la scène avec les effets d'éclairage, à la fois directs et indirects, les reflets, les ombres...

La recherche dans le domaine de la simulation de l'éclairage a énormément progressé au cours des dernières années, de telle sorte que la production d'images photoréalistes est désormais un objectif accessible pour le grand public. Plusieurs applications industrielles tirent parti de cette génération d'images photoréalistes : visite virtuelle de bâtiments, jeux vidéo, prototypage virtuel, effets spéciaux, design architectural...

Ces applications industrielles ont un effet d'entraînement sur la recherche : les utilisateurs (et les industriels) sont demandeurs d'effets toujours plus réalistes, et les chercheurs sont mis à contribution. Le décalage entre la date de publication d'un nouvel algorithme et son emploi dans un produit industriel s'est considérablement réduit, passant de plus de 10 ans dans les années 1990 à quelques années seulement en 2006. Non seulement ce dynamisme augmente les possibilités d'application industrielle pour nos recherches, mais encore il ouvre de nouvelles directions de recherche, pour combler les besoins accrus en interactivité et en réalisme des utilisateurs.

Dans ce mémoire, nous allons nous intéresser à ces problèmes de simulation photo-réaliste de l'éclairage. En particulier, nous allons présenter : la simulation de l'éclairage par des méthodes d'éléments finis multi-échelles (radiosité par ondelettes), la détermination des caractéristiques de la fonction d'éclairage (dérivées, fréquence), et la simulation en temps-réel ou interactif de plusieurs effets lumineux (ombres, reflets spéculaires, éclairage indirect). Ces trois domaines recouvrent l'ensemble de nos travaux pendant cette période.

- Dans le domaine de la simulation de l'éclairage par éléments finis, nous avons travaillé sur la méthode de radiosité hiérarchique, puis sur la méthode de radiosité par ondelettes. Nous avons montré l'efficacité de la représentation hiérarchique, mais aussi ses limites : la hiérarchie est conditionnée par la modélisation initiale de la scène. Nous avons montré comment s'affranchir de cette limitation. Nous avons également montré que l'extension de la méthode de radiosité aux ondelettes d'ordre élevé imposait une modification radicale de l'algorithme, puis comment combiner efficacement les ondelettes d'ordre élevé avec un maillage de discontinuité.
- Un des problèmes commun à plusieurs méthodes de simulation de l'éclairage, c'est qu'il est nécessaire d'adapter l'échantillonnage employé aux caractéristiques de la fonction

d'éclairage. Mais ces caractéristiques sont, par définition, inconnues au début de la simulation. Nous avons montré comment calculer localement certaines caractéristiques de la fonction d'éclairage : d'une part ses dérivées, d'autre part la fréquence de ses variations.

– Enfin, on remarque qu'une part très importante des calculs en simulation de l'éclairage est consacrée à l'aspect visuel du résultat, plus qu'à la précision physique des calculs. Il est possible de calculer une image physiquement acceptable d'une scène en un temps très court, mais calculer une image visuellement réaliste de la même scène multiplie approximativement par 10 le temps de calcul. Cette règle expérimentale vaut pour plusieurs algorithmes de simulation, comme la radiosité ou le photon mapping. Mais le réalisme visuel n'est, par définition, nécessaire que pour la partie *visible* de la scène. Nous avons développé plusieurs algorithmes qui permettent de calculer en temps réel certains effets essentiels pour le réalisme visuel (ombres, reflets spéculaires). En combinant ce calcul temps-réel avec un calcul séparé des effets d'éclairage indirect, notre but est d'obtenir une simulation en temps-réel de l'éclairage global dans une scène dynamique.

1.1 Structure du mémoire

Nous avons rédigé trois principaux chapitres, couvrant les travaux de chaque thème. Après chaque chapitre en français se trouvent les articles eux-mêmes, donnant le détail des travaux. Nous présentons ensuite les conclusions et quelques perspectives pour l'avenir.

Les travaux décrits dans ce mémoire ont tous été réalisés en collaboration avec des collègues ou dans le cadre des stages de DEA ou des thèses que j'ai co-dirigés. Les noms de mes collaborateurs sont cités dans les endroits appropriés.

2.

Modélisation multi-échelles de l'éclairage

Les techniques de radiosité hiérarchiques ou par ondelettes utilisent une représentation multi-échelle de l'éclairage. Cette représentation hiérarchique permet d'accélérer les calculs. Cependant, la représentation hiérarchique est basée sur le modèle géométrique de la scène ; l'influence de ce modèle géométrique peut ralentir la simulation et en diminuer la qualité. Nous présentons deux méthodes pour s'affranchir du modèle géométrique original. Nous présentons également une analyse de l'algorithme de radiosité par ondelettes. L'emploi de fonctions de base hiérarchiques d'ordre 2 ou 3, par opposition aux fonctions de base constantes par morceaux, impose d'adapter l'algorithme pour tenir compte des coûts proportionnels à l'ordre des fonctions de base élevé puissance n. Nous montrons qu'avec une adaptation fine de chacune des étapes de l'algorithme, ces ondelettes d'ordre élevé permettent de réduire la complexité et d'accélérer les calculs.

2.1 La méthode de radiosité hiérarchique

Dans les techniques de simulation de l'éclairage, on cherche à calculer l'aspect d'une scène donnée pour un observateur virtuel. La scène étant définie par sa géométrie, les matériaux et les caractéristiques (position, émission) des sources lumineuses, le calcul de l'éclairage revient à résoudre l'équation de l'éclairage[1] :

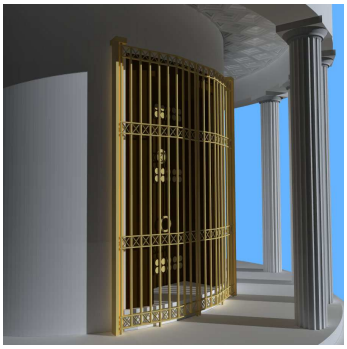
$$L(x, \theta_0, \phi_0) = L_e(x, \theta_0, \phi_0) + \int_\Omega \rho_{bd}(x, \theta_0, \phi_0, \theta, \phi) L_i(x, \theta, \phi) \cos\theta \, \mathrm{d}\omega \tag{2.1}$$

La radiance en un point x dans la direction (θ_0, ϕ_0) est simplement la somme de la radiance émise en propre (L_e) et de la radiance réfléchie en ce point. Il existe plusieurs méthodes de résolution de cette équation (lancer de rayon, radiosité, Monte-Carlo, photon mapping) ; parmi ces méthodes, la méthode de radiosité[2] résout l'équation 2.1 en simplifiant le problème (en supposant toutes les surfaces diffuses), puis en discrétisant l'équation obtenue. On aboutit à une équation matricielle :

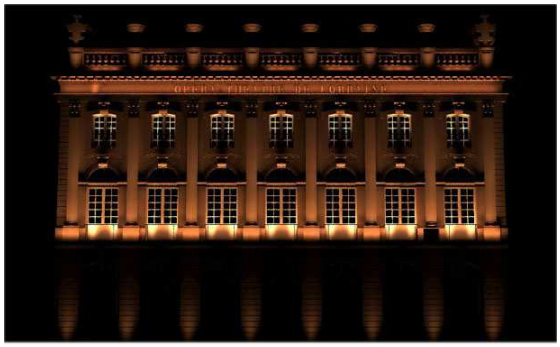
$$\boldsymbol{B} = \boldsymbol{E} + \boldsymbol{M}\boldsymbol{B} \tag{2.2}$$

dont la complexité est en $O(n^2)$ par rapport à la discrétisation de la scène. Comme les surfaces de la scène sont diffuses, le résultat obtenu est indépendant du point de vue, ce qui permet ensuite de se déplacer dans la scène en temps-réel (voir figures 2.1 et 2.2).

1. James T. K . « The Rendering Equation ». *Computer Graphics (ACM SIGGRAPH '86 Proceedings)*, 20(4):143–150, août 1986.
2. Cindy M. G , Kenneth E. T , Donald P. G et Bennett B . « Modelling the Interaction of Light Between Diffuse Surfaces ». *Computer Graphics (ACM SIGGRAPH '84 Proceedings)*, 18(3):212–222, juillet 1984.

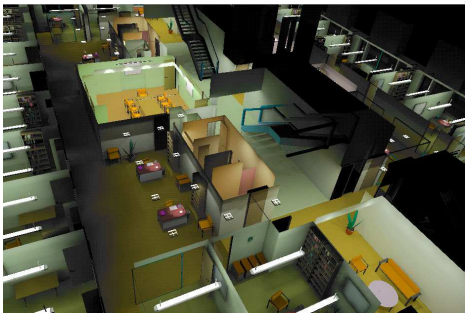


(a) Tholos, Delphe



(b) Place Stanislas, Nancy

Figure 2.1 – Images de modèles architecturaux calculées avec la méthode de radiosité. Les modèles ont été fournis par l'École d'Architecture de Nancy.



(a)

(b)

Figure 2.2 – Images du modèle du Soda Hall calculées avec la méthode de radiosité. Le modèle du Soda Hall a été fourni par Carlo Sequin.

La matrice $\boldsymbol{M}$ contient les coefficients de transfert d'énergie lumineuse entre les différentes facettes de la discrétisation de la scène. On résout l'équation 2.2 de façon itérative :

$$\boldsymbol{B} = (\boldsymbol{I} - \boldsymbol{M})^{-1} E = \sum_{k=0}^{\infty} \boldsymbol{M}^k \boldsymbol{E} \tag{2.3}$$

La taille de la matrice $\boldsymbol{M}$ est n^2, où n est le nombre de facettes issues de la discrétisation de la scène. Cette taille rend difficile son stockage en mémoire pour de scènes complexes. On utilise alors une résolution partielle, qui repose à chaque étape, sur le calcul soit d'une ligne, soit d'une colonne de la matrice (voir figure 2.3) :

- le calcul basé sur une *ligne* revient à mettre à jour la radiosité d'une facette en fonction de la radiosité de toutes les autres facettes de la scène (*gathering*).
- le calcul basé sur une *colonne* revient à mettre à jour la radiosité de toutes les facettes de

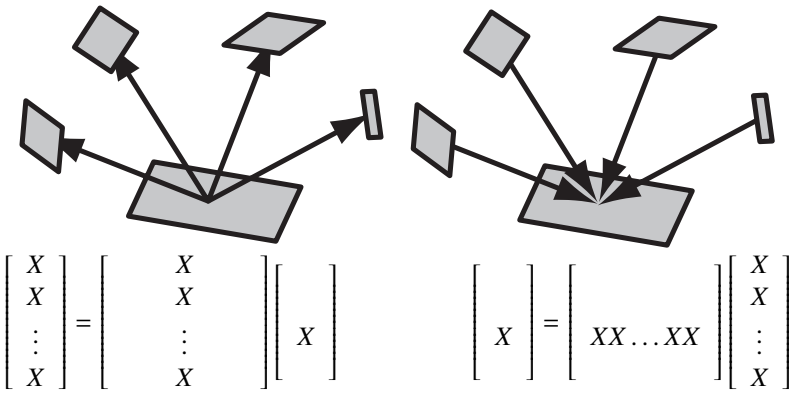


Figure 2.3 – *Gathering* et *Shooting* : utilisation d'une ligne ou d'une colonne de la matrice de transport, $\boldsymbol{M}$.

la scène en fonction de la radiosité d'une seule facette (*shooting*).
Dans la radiosité classique, le *shooting* permet d'obtenir des images exploitables plus rapidement : l'éclairage direct est obtenu dès les premières étapes, et la partie la plus importante de l'éclairage indirect est obtenue ensuite. En revanche, le temps nécessaire pour obtenir la solution après convergence est identique dans les deux méthodes.

2.2 Analyse de l'algorithme de radiosité hiérarchique

L'algorithme de radiosité hiérarchique[3, 4] utilise une représentation multi-échelle de la radiosité sur chacune des surfaces composant la scène, ce qui permet de réduire la complexité de l'algorithme à $O(n \log n)$. Sur chacun des objets composant la scène, on établit une hiérarchie de facettes, représentant la radiosité à différents niveaux de précision. Les interactions ou transferts d'énergie entre objets peuvent ensuite être établis entre les différents niveaux des hiérarchies ; ainsi, pour deux objets éloignés l'un de l'autre (et donc échangeant peu d'énergie), on placera une interaction entre les représentations de haut niveau des deux hiérarchies. En revanche, pour deux objets proches et échangeant beaucoup d'énergie, on fera établira les interactions entre des représentations précises de l'éclairage sur chaque objet.

On peut dire que la méthode correspond à stocker l'opérateur de transport $\boldsymbol{M}$ sous forme de blocs.

L'algorithme de radiosité hiérarchique calcule le transfert d'énergie lumineuse de façon itérative en plusieurs étapes :

- raffiner la représentation de l'opérateur de transfert $\boldsymbol{M}$, pour tenir compte de la radiosité existante dans la scène. Cette étape utilise un *oracle de raffinement* pour déterminer les interactions qui ne sont pas modélisées avec une précision suffisante.
- utiliser l'opérateur $\boldsymbol{M}$ pour transférer l'énergie entre les différents objets de la scène.

3. Pat H , David S et Larry A . « A Rapid Hierarchical Radiosity Algorithm ». *Computer Graphics (ACM SIGGRAPH '91 Proceedings)*, 25(4):197–206, juillet 1991.
4. Pat H et David S . « A Rapid Hierarchical Radiosity Algorithm for Unoccluded Environments ». Dans *Photorealism in Computer Graphics (Eurographics Workshop on Photosimulation, Realism and Physics in Computer Graphics)*, p. 151–171, juin 1992.

– propager l'énergie reçue par les différents niveaux hiérarchiques de chaque objet dans l'ensemble de la hiérarchie (*push-pull*).

La méthode de radiosité hiérarchique modifie en profondeur l'algorithme de radiosité : à chaque étape, on a une représentation complète de l'opérateur de transport, et on peut donc calculer un transfert de radiosité dans toute la scène en une seule étape.

Dans une étape de propagation, il est plus facile d'utiliser du *gathering* que du *shooting* : avant qu'une surface puisse envoyer son énergie dans la scène, il est nécessaire d'effectuer une étape de *push-pull*. Le *shooting* impose ainsi un trop grand nombre d'étapes de *push-pull*, ce qui le rend moins efficace que le *gathering*. Expérimentalement, nous avons aussi trouvé que la méthode de *shooting* est plus sensible à l'imprécision sur les calculs des coefficients de transfert, et divergeait avec les méthodes imprécises de calcul de ces coefficients utilisées à l'époque.

Nous avons effectué une étude en profondeur de l'algorithme de radiosité hiérarchique [16] (voir p. 26). Cette étude a montré deux choses importantes :

– la plus grande partie du temps de calcul est passé dans les tests de visibilité (plus de 80 % dans une scène complexe). Les autres étapes de l'algorithme ont une influence relativement modeste;
– la méthode construit une hiérarchie sur chacun des objets de haut niveau composant la scène . On établit d'abord un lien entre chacune de ces hiérarchies (*initial linking*), puis on raffine ces liens. La méthode a ainsi une complexité réduite par rapport au nombre de facettes générées lors du raffinement, mais reste quadratique par rapport au nombre d'objets de haut niveau composant la scène.

En nous basant sur cette étude, nous avons proposé une méthode de *lazy linking*, qui réduit le temps de calcul d'un facteur 2, et permet d'obtenir les premières images plus rapidement. Nous avons également proposé un nouvel oracle de raffinement, qui évite des raffinements inutiles, et permet de gagner encore un facteur 2 sur le temps de calcul.

2.3 Efficacité de la hiérarchie

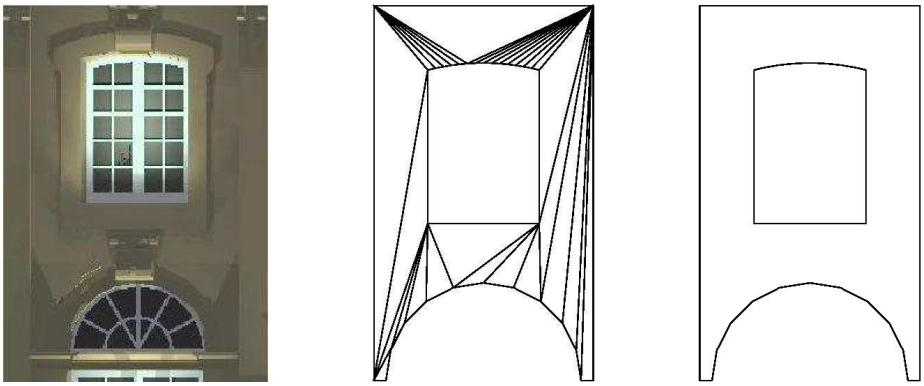
La méthode de *lazy linking* que nous avons proposé permet seulement de diminuer l'influence de l'étape de création des liens entre surfaces de haut niveau, mais elle ne supprime pas complètement cette étape, dont la complexité reste quadratique. Plusieurs méthodes ont été proposées pour réduire encore la complexité, comme le *clustering*[5] ou le *face-clustering*[6].

Une des difficultés provient de l'étape de *push-pull*, qui dépend des formes relatives entre les nœuds parents et enfants de la hiérarchie. Les calculs de l'étape de *push-pull* sont simplifiés si la hiérarchie est *régulière*, c'est-à-dire si à chaque niveau de la hiérarchie les formes et les positions des nœuds fils par rapport aux ascendants sont identiques. Une hiérarchie régulière impose que les surfaces de haut niveau qui sont soit triangulaires, soit parallélogrammes.

Cependant, dans une grande partie des scènes utilisées pour la simulation de l'éclairage, les surfaces de départ n'ont pas une forme régulière (voir figure 2.4). Avant de lancer les calculs de radiosité, il est alors nécessaire de les trianguler, mais cette triangulation augmente artificiellement le nombre de surfaces de haut niveau dans la scène. En outre les triangles résultants de la triangulation influencent négativement les calculs : ils ont souvent des formes très allongées, ce qui diminue la qualité de la simulation (voir figure 2.5) et l'influence de la hiérarchie. Enfin, des discontinuités peuvent apparaître entre les triangles.

5. Francois S . « Clustering and Volume Scattering for Hierarchical Radiosity Calculations ». Dans *Rendering Techniques '95 (Eurographics Workshop on Rendering)*, p. 105–117, juin 1994.

6. A. W , P. H et M. G . « Face Cluster Radiosity ». Dans *Rendering Techniques '99 (Eurographics Workshop on Rendering)*, p. 293–304, 1999.

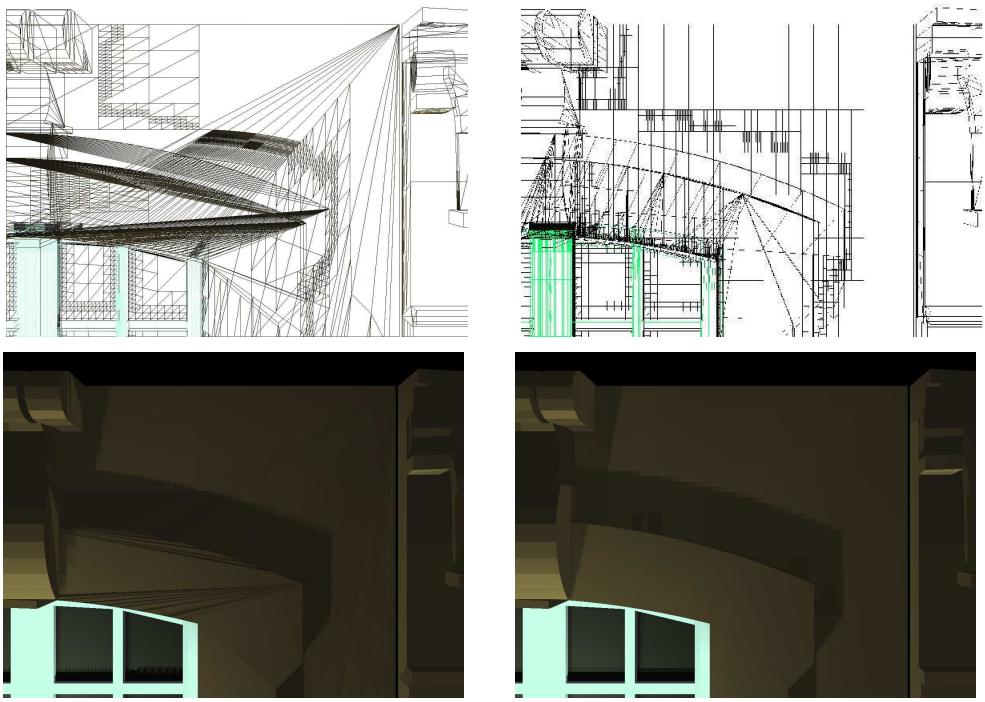


(a) Détail de la figure 2.1(b)

(b) Triangulé : 32 triangles

(c) Non-triangulé : une seule surface

Figure 2.4 – Les modèles contiennent des surfaces planes complexes, qui sont triangulées excessivement.



(a) Triangulé

(b) Non-triangulé (notre algorithme)

Figure 2.5 – Calcul de radiosité sur des surfaces planes quelconques.

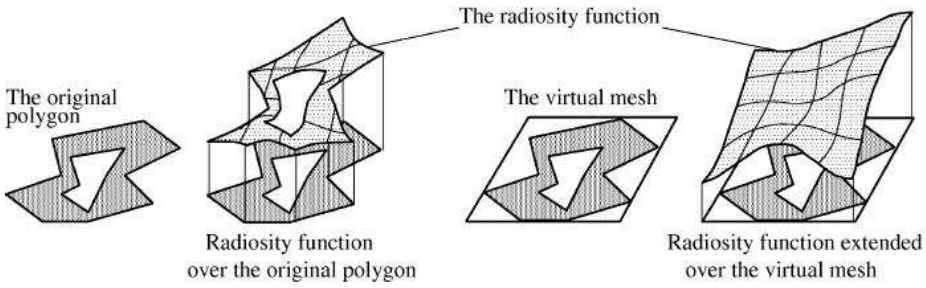


Figure 2.6 – La fonction de radiosité est définie sur un domaine plan étendu, plus simple que la surface originale.

En collaboration avec le doctorant François Cuny (co-encadré par Jean-Claude Paul et Laurent Alonso), nous avons proposé une méthode qui permet les calculs de radiosité hiérarchique sur des surfaces planes de forme quelconque, y compris les surfaces concaves ou trouées [14] (voir p. 42). Cette méthode repose sur une extension de la surface plane originale à une surface plus simple, qui la contient, et sur laquelle il est possible de construire une hiérarchie régulière (voir figure 2.6). La radiosité doit être définie de façon continue sur la surface étendue, tout en étant égale à la radiosité sur la surface de départ.

Notre algorithme permet naturellement de diminuer le nombre de surfaces initiales dans la scène, et ainsi de diminuer le coût mémoire de l'algorithme et le coût de l'étape d'*initial linking*, et ces points ont été confirmés par nos expériences. De façon plus surprenante, nous avons trouvé que notre algorithme accélère la convergence de l'algorithme de radiosité hiérarchique (voir figure 2.7) : le pourcentage d'énergie restant à propager dans la scène diminue plus rapidement. Ce résultat montre que la hiérarchie régulière employée approxime de façon plus efficace la radiosité sur la surface que la hiérarchie basée sur la triangulation des surface.

Ces résultats ont été étendus par la suite au cas des surfaces courbes paramétriques[7], puis des maillages polygonaux paramétrés[8].

2.4 Ondelettes d'ordre élevé

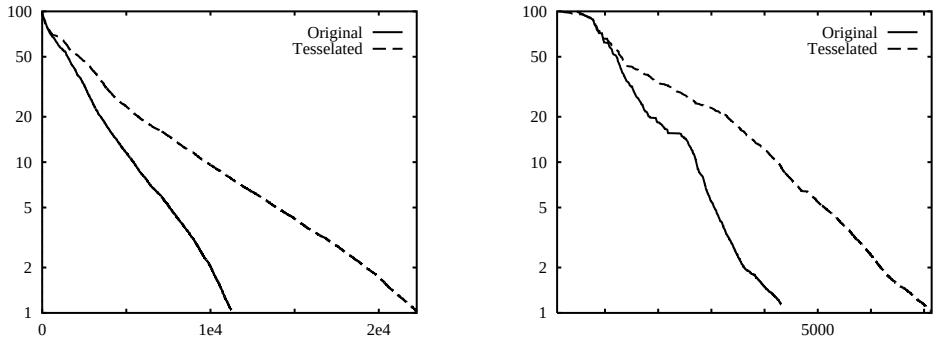
La méthode de radiosité hiérarchique repose sur une représentation multi-échelle de l'éclairage. Cette représentation a été par la suite formalisée et étendue à des fonctions d'ondelettes quelconques[9]. En théorie, ces recherches ouvraient la voie à des fonctions de base d'ordre plus élevé, par exemple linéaires ou quadratiques par morceaux, au lieu de constantes par morceaux.

En pratique, l'utilisation d'ondelettes d'ordre élevé modifie l'équilibre de l'algorithme. Le

7. Laurent A , François C , Sylvain P , Jean-Claude P , Sylvain L et Eric W . « The Virtual Mesh: A Geometric Abstraction for Efficiently Computing Radiosity ». *ACM Transactions on Graphics*, 20(3):169–201, juillet 2001.

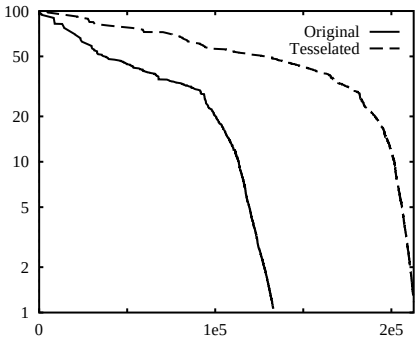
8. Gregory L , Bruno L , Laurent A et Jean-Claude P . « Master-Element Vector Irradiance for Large Tesselated Models ». Dans *Third International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia (GRAPHITE '05)*, p. 315–322, 463, novembre 2005.

9. Steven J. G , Peter S , Michael F. C et Pat H . « Wavelet Radiosity ». Dans *ACM SIGGRAPH '93*, p. 221–230, 1993.



(a) Place Stanislas (figure 2.1(b))

(b) Temple (figure 2.1(a))



(c) Soda Hall (figure 2.2)

Figure 2.7 – Taux de convergence (rapport énergie restant à propager sur énergie initiale) en fonction du temps de calcul (en secondes)

coût de stockage des coefficients attachés à des ondelettes d'ordre n est de $(n+1)^k$, où k est la dimension de la fonction échantillonnée. Ainsi, le coût lié au stockage de chaque facette de la hiérarchie évolue comme $(n+1)^2$, et le coût de stockage de chaque interaction évolue comme $(n+1)^4$. L'emploi d'ondelettes d'ordre 2 (quadratiques par morceaux) augmente donc le coût de stockage de chaque interaction de deux ordres de grandeur.

Pour cette raison, une étude expérimentale[10] a montré que les ondelettes d'ordre élevé n'apportaient pas d'amélioration significative par rapport aux ondelettes de Haar (constantes par morceaux). Cette étude était basée sur une implémentation naïve de l'algorithme de radiosité par ondelettes, en modifiant simplement la fonction de base. Avec le doctorant François Cuny (co-encadré par Jean-Claude Paul et Laurent Alonso), nous avons montré qu'il était nécessaire d'adapter chacune des étapes de l'algorithme aux spécificités introduites par les fonctions de base d'ordre élevé [8] (p. 56). Il était également nécessaire de réfléchir aux *conséquences* de chaque adaptation sur les autres étapes de l'algorithme ; ainsi, la décision de ne pas stocker les liens a eu une influence sur le choix de l'algorithme de résolution.

10. Andrew W et Paul H . « An Empirical Comparison of Progressive and Wavelet Radiosity ». Dans *Rendering Techniques '97 (Eurographics Workshop on Rendering)*, p. 175–186, 1997.

2.4.1 Améliorations de l'algorithme

Nous avons proposé les modifications suivantes à l'algorithme de radiosité par ondelettes :

- Le coût lié au stockage des interactions étant prohibitif (16 ou 81 fois plus élevé qu'avec des fonctions constantes par morceaux), nous avons abandonné le principe de stockage des liens. Il était devenu plus intéressant de recalculer les liens à chaque itération que de les stocker en mémoire.
- Une conséquence de ce choix est qu'il devenait à nouveau plus intéressant d'utiliser du *shooting* pour la propagation de l'énergie. Avec le *shooting*, la distribution d'énergie propagée dans la scène varie à chaque étape, ce qui signifie que la distribution optimale des liens varie à chaque itération. Il n'est donc pas pénalisant de les recalculer, au contraire.
- Nous avons augmenté la précision du calcul des coefficients de transfert pour chaque itération. En particulier, nous avons multiplié les tests de visibilité pour les interactions entre facettes partiellement visibles. Nous avons trouvé que l'augmentation du temps de calcul lié à ces tests supplémentaires était plus que compensée par l'augmentation de la qualité du résultat.
- Nous avons également utilisé un oracle de raffinement[11] qui tenait compte de la nature linéaire ou quadratique de la fonction d'éclairage en cours de calcul pour guider le raffinement. On ne raffine une interaction que si la fonction d'éclairage sur le récepteur s'écarte significativement de la représentation fournie par les fonctions de base.

Avec ces modifications, nous avons montré que les ondelettes d'ordre élevé permettaient d'obtenir un résultat de meilleure qualité et plus rapidement que les ondelettes de Haar. Chaque nœud de la hiérarchie a un coût de stockage plus élevé (4 ou 9 fois plus élevé selon le degré des ondelettes utilisées), mais la réduction du nombre de nœuds dans la hiérarchie compense ce surcoût.

La figure 2.8 montre le coût en mémoire de la modélisation de l'éclairage, en fonction de l'erreur commise dans la simulation, pour deux scènes de test. On voit que pour une simulation de mauvaise qualité (erreur ≈ 0.1), les ondelettes de Haar fournissent une approximation plus compacte en mémoire, mais que plus la qualité de la simulation augmente et plus les ondelettes d'ordre élevé sont avantagées. Pour chaque base d'ondelettes étudiées, il existe un niveau de qualité où elle est meilleure que les autres bases.

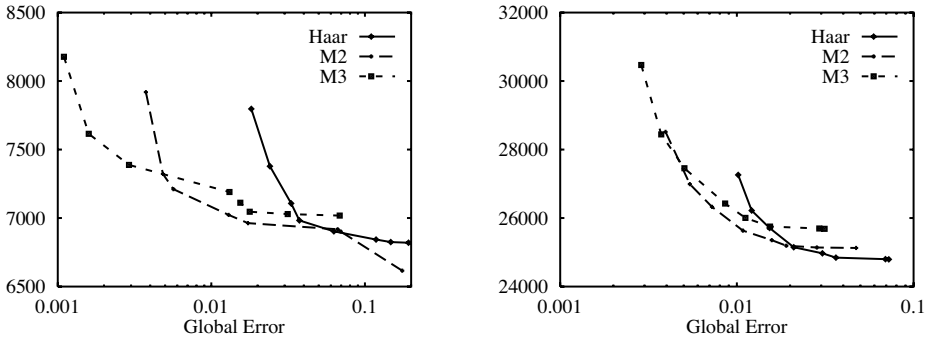
Le niveau de qualité correspondant à une simulation acceptable visuellement correspond (expérimentalement) à une erreur $\approx 5 \times 10^{-3}$, c'est-à-dire à l'intersection des courbes $\mathcal{M}_2$ et $\mathcal{M}_3$.

Nous avons également représenté l'erreur commise dans la simulation de l'éclairage, en fonction du temps de calcul, pour les trois bases d'ondelettes (voir figure 2.9). On voit que l'erreur commise diminue en fonction du temps de calcul, pour les trois bases. Pour chaque base d'ondelettes, il existe un niveau de qualité pour lequel elle fournit un résultat plus rapidement que les autres bases.

Si on analyse ces résultats, on voit que lorsqu'on raffine peu par rapport au modèle de départ, les ondelettes constantes sont avantagées, ce qui est normal : leur coût par objet est moins élevé que les autres bases. En revanche, plus on raffine, plus les fonctions de base d'ordre élevé permettent une modélisation plus précise et plus compacte de la radiosité. Sur ce point, nos travaux se complètent naturellement avec nos travaux sur la simplification de la scène (voir section 2.3 et [14]), dans la mesure où sur un modèle où les surfaces sont triangulées, le raffinement est moins efficace.

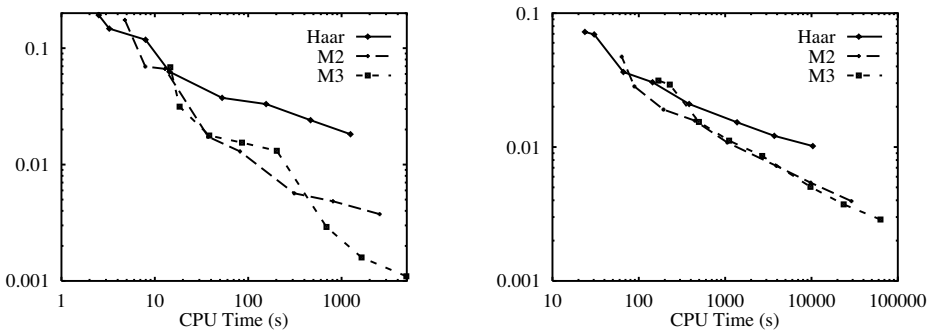
Comme conséquence inattendue, mais intéressante, de nos travaux, nous avons trouvé que notre implémentation donnait directement une fonction de radiosité continue (voir figure 2.10), sans qu'il soit nécessaire d'effectuer un post-traitement (contrairement à toutes les implémenta-

11. Philippe B et Yves W . « Error Control for Radiosity ». Dans *Rendering Techniques '96 (Eurographics Workshop on Rendering)*, p. 153–164, 1996.



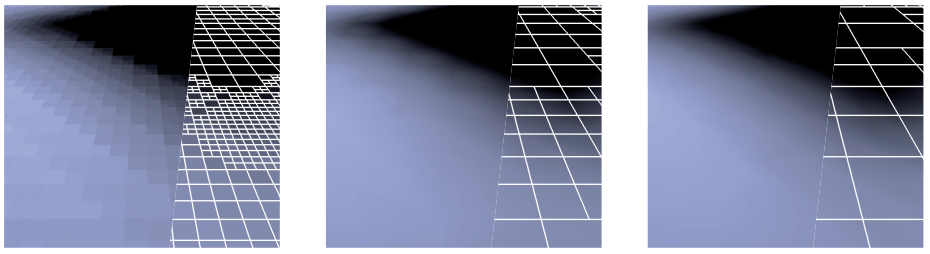
(a) Dining room (b) Classroom

Figure 2.8 – Coût mémoire (en Ko) pour les différentes fonctions de base, en fonction de l'erreur sur la simulation.



(a) Dining room (b) Classroom

Figure 2.9 – Erreur commise sur la simulation en fonction du temps de calcul (en s).



(a) Haar (b) $\mathcal{M}_2$ (c) $\mathcal{M}_3$

Figure 2.10 – Comparaison entre les ondelettes de Haar (constantes), $\mathcal{M}_2$ (linéaires) et $\mathcal{M}_3$ (quadratiques), pour le même temps de calcul.

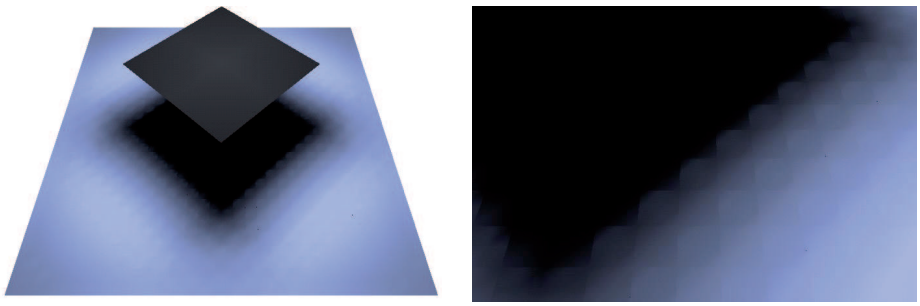


Figure 2.11 – Lissage a posteriori des ondelettes de Haar. On peut comparer la qualité avec la figure 2.10.

tions précédentes). La qualité de la fonction que nous obtenons avec des ondelettes linéaires ou quadriques sans post-traitement est supérieure à celle obtenue avec des ondelettes constantes par morceaux et un post-traitement (voir figure 2.11)

2.4.2 Maillage de discontinuités et ondelettes d'ordre élevé

Dans la simulation de l'éclairage, on rencontre fréquemment des discontinuités de la fonction d'éclairage ou de sa dérivée. Ces discontinuités sont par exemple liées au contact entre objets, ou bien aux ombres dures causées par des sources ponctuelles. Les discontinuités de la dérivée, elles, sont liées aux frontières d'ombre et de pénombre causées par des sources étendues.

Un échantillonnage par mailles régulières, comme celui produit par la radiosité par ondelettes, ne permet évidemment pas de modéliser ces discontinuités. Pour chaque discontinuité de la radiosité, l'oracle de raffinement est conduit à subdiviser indéfiniment, jusqu'à atteindre un seuil sur la taille de la facette. Avec des ondelettes $\mathcal{M}_2$ et $\mathcal{M}_2$, chaque maille a un coût plus élevé qu'avec des ondelettes constantes. Le raffinement effectué sur les discontinuités vient contrebalancer les gains obtenus par ailleurs.

Plusieurs travaux ont montré que dans le cas de la radiosité classique, un maillage adapté aux discontinuités fournit les meilleurs résultats en terme de qualité[12, 13, 14]. Ces travaux ont été étendus au cas de la radiosité hiérarchique[15, 16, 17]. Toutes ces approches commencent par calculer l'ensemble des discontinuités, par des méthodes géométriques, puis triangulent cet ensemble, et construisent enfin le maillage hiérarchique en se basant sur la triangulation.

Cette méthode a plusieurs défauts : le nombre de discontinuités des dérivées est élevé, donc le maillage généré est très complexe, et cette complexité ralentit la simulation. Certaines des dis-

12. Daniel L , Filippo T et Donald P. G . « Discontinuity Meshing for Accurate Radiosity ». *IEEE Computer Graphics and Applications*, 12(6):25–39, novembre 1992.
13. Paul H . « Discontinuity Meshing for Radiosity ». Dans *Eurographics Workshop on Rendering*, p. 203–226, mai 1992.
14. George D et Eugene F . « A Fast Shadow Algorithm for Area Light Sources Using Backprojection ». Dans *ACM SIGGRAPH '94*, p. 223–230, 1994.
15. Daniel L , Filippo T et Donald P. G . « Combining Hierarchical Radiosity and Discontinuity Meshing ». Dans *ACM SIGGRAPH '93*, p. 199–208, 1993.
16. George D et Francois S . « Accurate Visibility and Meshing Calculations for Hierarchical Radiosity ». Dans *Rendering Techniques '96 (Eurographics Workshop on Rendering)*, p. 269–278, 1996.
17. Fredo D , George D et Claude P . « Fast and Accurate Hierarchical Radiosity Using Global Visibility ». *ACM Transactions on Graphics*, 18(2):128–170, 1999.

continuités calculées n'ont par ailleurs pas d'effets visibles sur la simulation[14]. Enfin, le maillage généré est incompatible avec l'approche par ondelettes, qui nécessite une hiérarchie régulière.

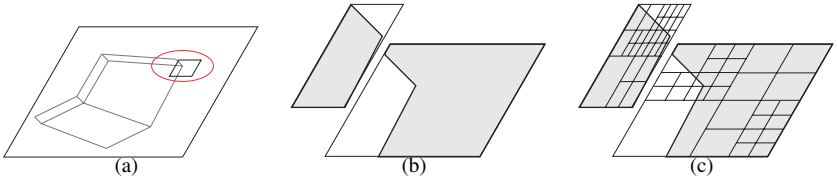


Figure 2.12 – Une maille coupée par une discontinuité est (a) décomposée en deux mailles. Pour chacune des mailles, nous identifions le parallélogramme englobant (b). Sur chaque parallélogramme, nous conduisons un algorithme de radiosité par ondelettes classique.

Nous avons développé un algorithme qui combine les ondelettes d'ordre élevé avec le maillage de discontinuité [13] (voir p. 68). Notre algorithme utilise les ondelettes d'ordre élevé autant que possible, et n'introduit les discontinuités dans le maillage que si l'oracle de raffinement établit qu'elles permettront d'en réduire la complexité. Nous avons trouvé qu'il n'est nécessaire d'introduire dans le maillage qu'un petit nombre de discontinuités seulement (voir figure 2.13). Les autres discontinuités sont approximées de façon correcte par les fonctions de base linéaires ou quadratiques.

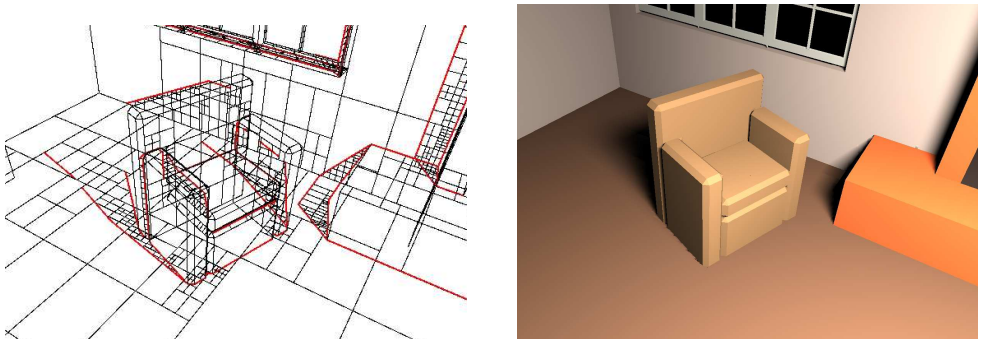
Notre travail est basé sur les travaux précédents sur la radiosité sur des surfaces planes quelconques [14], mais en modifiant l'approche sur plusieurs points :

- Les deux côtés de la discontinuité jouent un rôle dans la simulation de l'éclairage. Il faut donc créer *deux* mailles spéciales pour chaque discontinuité introduite (voir figure 2.12).
- Sur la discontinuité, il faut calculer des coefficients de push-pull particuliers, qui tiennent compte de la proportion effective de la surface qui se trouve de chaque côté de la discontinuité. Sur le reste de la hiérarchie, en amont et en aval de la discontinuité, on utilise une subdivision régulière, avec tous ses avantages.
- Il peut y avoir intersection entre plusieurs discontinuités (les frontières d'ombre causées par plusieurs sources lumineuses, ou encore les frontières d'ombre et de pénombre qui se rejoignent quand l'obstacle touche le récepteur). Il faut alors traiter les discontinuités les unes après les autres, et prévoir certains cas particuliers.

Notre approche permet une représentation très compacte de la radiosité, y compris en présence de discontinuités (voir [13] et figure 2.13). Dans les zones éclairées, les ondelettes $\mathcal{M}_3$ permettent d'utiliser des mailles très larges tout en fournissant une représentation continue, tandis que sur les frontières d'ombre, l'emploi des discontinuités permet d'interrompre rapidement le raffinement. Dans les zones de pénombre, nous avons montré qu'il n'est pas toujours nécessaire d'introduire les discontinuités, et que si la transition est suffisamment douce, les ondelettes $\mathcal{M}_3$ peuvent la modéliser correctement.

2.4.3 Radiosité spatio-temporelle

Avec le doctorant Cyrille Damez (encadré par François Sillion), nous nous sommes par ailleurs intéressé au calcul de la radiosité dans un espace à 4 dimensions (3 dimensions d'espace et une dimension de temps). En supposant connue une animation, nous voulons calculer l'éclairage pour toute cette animation, en exploitant non seulement la cohérence spatiale mais aussi la



(a) (b)

Figure 2.13 – Radiosité avec ondelettes quadriques ($\mathcal{M}_3$) et maillage de discontinuité, en présence d'une source surfacique. Une partie de la zone de pénombre du fauteuil est modélisée avec des mailles régulières, sans avoir besoin d'insérer de discontinuités. On voit aussi que des mailles très larges suffisent pour la zone éclairée au pied du fauteuil.

cohérence *temporelle* de l'éclairage, par une décomposition hiérarchique portant également sur la dimension temporelle.

Les travaux préliminaires[18] souffraient d'importantes discontinuités temporelles. Ces discontinuités existent également dans les dimensions spatiales dans le cas de la radiosité hiérarchique classique, où elles sont partiellement corrigées par un post-traitement. Les discontinuités temporelles sont à la fois beaucoup plus gênantes, parce qu'elles concernent l'ensemble de la scène, et plus difficiles à traiter.

Nos travaux sur les ondelettes d'ordre élevé avaient montré que ces ondelettes génèrent directement une solution continue dans le domaine spatial. Nous avons montré qu'en introduisant les ondelettes d'ordre élevé dans le domaine temporel [6] (p. 82), il était également possible d'éliminer les discontinuités temporelles.

2.5 Structure de la scène

Nous avons fait plusieurs fois l'expérience de relations avec des partenaires extérieurs au monde de la recherche : avec l'École d'Architecture de Nancy, avec la société OPTIS, avec les entreprises innovantes Visual Information Systems (Cape Town) et VSP-Technologies (Nancy), avec le Fraunhofer-Institut für Graphische Datenverarbeitung (Darmstadt), avec les partenaires des projets européens ARCADE et SIMULGEN. Ces partenaires nous confiaient des scènes sur lesquelles nous pouvions tester nos algorithmes de calcul (voir, par exemple, figure 2.1). Ces scènes étaient, à chaque fois, fournies sous forme d'un ensemble désordonné de polygones, sans information de connectivité ni structure interne.

Des discussions avec nos partenaires nous ont appris que la structure de la scène est, en général, présente lors du processus de modélisation, mais qu'elle est perdue dans les multiples transferts de fichiers. Cette disparition peut se produire même entre les différents services d'une même entreprise.

18. C. D et F. S . « Space-Time Hierarchical Radiosity ». Dans *Rendering Techniques '99 (Eurographics Workshop on Rendering)*, p. 235–246, 1999.

D'un autre côté, nos propres travaux et ceux d'autres chercheurs ont montré que la connaissance de la structure de la scène sous-jacente permet une meilleure qualité dans la simulation de l'éclairage, par exemple en remplaçant un ensemble de triangles déconnectés par un seul polygone plan [14], ou bien en remplaçant un ensemble de polygones par une surface paramétrée[7], ou encore en exploitant la structure de la scène pour construire une hiérarchie efficace pour le lancer de rayons, le *clustering* ou l'instantiation.

Nous avons lancé le projet SHOW, en collaboration avec François Sillion et Cyril Soler, et avec trois autres projets INRIA : ALICE, REVES et IPARLA. L'objectif du projet SHOW est de reconstruire la structure d'un grand ensemble de données désordonnées. En particulier, le doctorant Aurélien Martinet, financé par le projet SHOW et que je co-encadre avec Cyril Soler et François Sillion travaille sur l'extraction d'une structure de scène en partant d'un ensemble de polygones désordonnés.

Les premiers résultats [4] (voir p. 96) ont permis l'extraction d'ensembles de triangles connexes par arêtes, qui forment des briques de base. Nous sommes également parvenus à extraire la liste des symétries de chaque brique de base, et à identifier les briques identiques, même si leur tesselation diffère. Des travaux ultérieurs[19] permettent l'identification rapide d'objets identiques dans la scène, où les objets sont formés par assemblage de briques de base.

2.6 Discussion

Dans ce chapitre, nous avons présenté nos travaux dans le domaine de la radiosité par ondelettes. Nos contributions portent sur des améliorations de l'algorithme : faire porter la hiérarchie sur l'ensemble de la scène, adapter la méthode aux ondelettes d'ordre élevé, combiner ondelettes d'ordre élevé et maillage de discontinuité.

Ces améliorations sont importantes, voire essentielles pour toute application pratique de la méthode de radiosité. Combinées, elles permettent une représentation optimale — compacte — de l'éclairage diffus dans une scène. Mais ces travaux ont leurs propres limitations :

- on ne calcule que l'éclairage diffus, or les fonctions de réflectance complexes jouent un rôle important dans le réalisme de la scène ;
- on manque d'informations *a priori* sur le comportement de la fonction d'éclairage, ce qui ne permet pas d'adapter l'échantillonnage aux variations de la fonction ;
- une part très importante du temps de calcul est consacrée au calcul des frontières d'ombre et de pénombre dans l'éclairage direct, alors que ces frontières ne jouent qu'un rôle mineur dans le calcul de l'éclairage indirect.

Ces différents points sont partiellement liés. Ainsi, introduire des fonctions de réflectance qui ne sont ni diffuses, ni spéculaires impose d'échantillonner à la fois dans le domaine spatial et dans le domaine angulaire, augmentant ainsi la dimensionnalité du problème et le coût mémoire de l'algorithme. Une connaissance *a priori* des variations de la fonction d'éclairage permettrait d'adapter l'échantillonnage (spatial et angulaire) à ces variations, et ainsi de contrôler le coût mémoire de l'algorithme. Le prochain chapitre est consacré à l'extraction des propriétés de la fonction d'éclairage.

Le troisième point est commun à d'autres algorithmes de simulation de l'éclairage : une grande part du temps de calcul est consacrée à des effets qui sont importants pour l'aspect visuel de l'image calculée (frontières d'ombre, réflexions spéculaires) mais qui sont sans intérêt pour

7. Laurent A , François C , Sylvain P , Jean-Claude P , Sylvain L et Eric W . « The Virtual Mesh: A Geometric Abstraction for Efficiently Computing Radiosity ». *ACM Transactions on Graphics*, 20(3):169–201, juillet 2001.

19. Aurélien M . « *Structuration automatique de scènes 3D* ». Thèse, Université Joseph Fourier, 2006.

les calculs d'éclairage indirect. Cette disproportion impose de réfléchir à ce qu'on simule, et d'étudier les moyens de calculer séparément certains effets. C'est l'objet du chapitre 4.

2.7 Articles

2.7.1 Liste des articles

- An efficient progressive refinement strategy for hierarchical radiosity (EGWR '94)
- Wavelet Radiosity on Arbitrary planar surfaces (EGWR 2000)
- A novel approach makes higher order wavelets really efficient for radiosity (EG 2000)
- Combining higher-order wavelets and discontinuity meshing: a compact representation for radiosity (EGSR 2004)
- Space-time hierarchical radiosity with clustering and higher-order wavelets (CGF 2004)
- Accurate detection of symmetries in 3D shapes (TOG 2006)

2.7.2 An efficient progressive refinement strategy for hierarchical radiosity (EGWR '94)

Auteurs : Nicolas H , François S et George D
Conférence : 5e *Eurographics Workshop on Rendering*, Darmstadt, Allemagne.
Date : juin 1994

An Efficient Progressive Refinement Strategy for Hierarchical Radiosity

Nicolas Holzschuch, François Sillion, George Drettakis

iMAGIS / IMAG *

A detailed study of the performance of hierarchical radiosity is presented, which confirms that visibility computation is the most expensive operation. Based on the analysis of the algorithm's behavior, two improvements are suggested. Lazy evaluation of the top-level links suppresses most of the initial linking cost, and is consistent with a progressive refinement strategy. In addition, the reduction of the number of links for mutually visible areas is made possible by the use of an improved subdivision criterion. Results show that initial linking can be avoided and the number of links significantly reduced without noticeable image degradation, making useful images available more quickly.

1 Introduction

The radiosity method for the simulation of energy exchanges has been used to produce some of the most realistic synthetic images to date. In particular, its ability to render global illumination effects makes it the technique of choice for simulating the illumination of indoor spaces. Since it is based on the subdivision of surfaces using a mesh and on the calculation of the energy transfers between mesh elements pairs, the basic radiosity method is inherently a costly algorithm, requiring a quadratic number of form factors to be computed.

Recent research has focused on reducing the complexity of the radiosity simulation process. Progressive refinement has been proposed as a possible avenue [1], whereby form factors are only computed when needed to evaluate the energy transfers from a given surface, and surfaces are processed in order of importance with respect to the overall balance of energy. The most significant advance in recent years was probably the introduction of hierarchical algorithms, which attempt to establish energy transfers between mesh elements of varying size, thus reducing the subdivision of surfaces and the total number of form factors computed [4, 5].

Since hierarchical algorithms proceed in a top-down manner, by limiting the subdivision of input surfaces to what is necessary, they first have to establish a number of top-level links between input surfaces in an "initial linking" stage. This

* iMAGIS is a joint research project of CNRS/INRIA/UJF/INPG. Postal address: B.P. 53, F-38041 Grenoble Cedex 9, France. E-mail: Nicolas.Holzschuch@imag.fr.

results in a quadratic cost with respect to the number of input surfaces, which seriously impairs the ability of hierarchical radiosity systems to deal with environments of even moderate complexity. Thus a reformulation of the algorithm is necessary in order to be able to simulate meaningful architectural spaces of medium complexity (several thousands of input surfaces). To this end the questions that must be addressed are: What energy transfers are significant? When must they be computed? How can their accuracy be controlled?

The goal of the research presented here is to extend the hierarchical algorithm into a more progressive algorithm, by identifying the calculation components that can be delayed or removed altogether, and establishing improved refinement criteria to avoid unnecessary subdivision. Careful analysis of the performance of the hierarchical algorithm on a variety of scenes shows that the visibility calculations dominate the overall compute time.

Two main avenues are explored to reduce the cost of visibility calculations: First, the cost of initial linking is reduced by delaying the creation of the links between top-level surfaces until they are potentially significant. In a BF refinement scheme this means for instance that no link is established between dark surfaces. In addition, a form factor between surfaces can be so small that it is not worth performing the visibility calculation.

Second, experimental studies show that subdivision is often too high. This is a consequence of the assumption that the error on the form factor is of magnitude comparable to the form factor itself. In situations of full visibility between surfaces, relatively large form factors can be computed with good accuracy.

2 Motivation

To study the behaviour of the hierarchical algorithm, we ran the original hierarchical program [5] on a set of five different interior environments, varying from scenes with simple to moderate complexity (from 140 to 2355 input polygons). The scenes we used were built in different research efforts and have markedly different overall geometric properties. By using these different scenes, we hope to identify general properties of interior environments. We thus hope to avoid, or at least moderate, the pitfall of unjustified generalisation that oftens results from the use of a single scene or a class of scenes with similar properties to characterise algorithm behaviour. The scenes are: *"Full office"*, which is the original scene used in [5], *"Dining room"*, which is "Scene 7" of the standard set of scenes distributed for this workshop, *"East room"* and *"West room"*, which are scenes containing moderately complex desk and chair models, and finally *"Hevea"*, a model of a hevea tree in a room. Table 1 gives a short description and the number of polygons n for each scene. Please refer to colour section (Figs. 1, 3, 5 and 9-12) for a computed view of the test scenes.

2.1 Visibility

The first important observation we make from running the algorithm on these test scenes is the quantification of the cost of visibility calculations in the hier-

Table 1. Description of the five test scenes.

Name	n	Description
Full Office	170	The original office scene
Dining room	402	A table and four chairs
East room	1006	Two desks, six chairs
West room	1647	Four desks, ten chairs
Hevea	2355	An hevea tree with three light sources

archical algorithm. As postulated in previous work [9, 6], visibility computation represents a significant proportion of the overall computation time. In the graph shown in Fig. 1, the percentages of the computation time spent in each of the five main components of the hierarchical algorithm are presented. "Push-pull" signifies the time spent traversing the quadtree structure associated with each polygon, "Visibility" is the time spent performing visibility calculations, both for the initial linking step and subsequent refinement, "Form Factors" is the time spent performing the actual unoccluded form factor approximation calculation, "Refine" is the time spent updating the quadtree for refinement, and finally "Gather" shows the cost of transferring energy across the links created between quadtree nodes (interior or leaves) [5]. The graph in Fig. 1 shows that

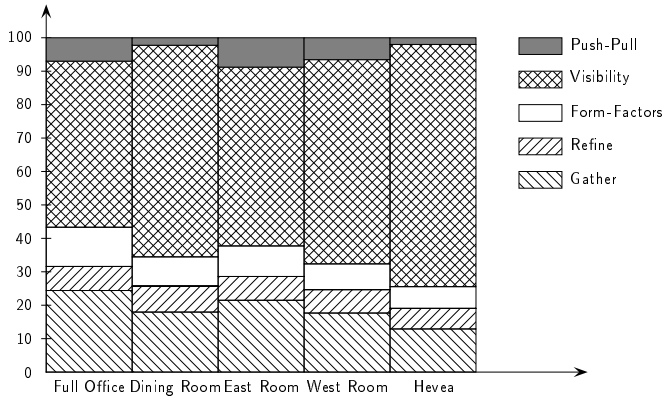


Fig. 1. Relative time spent in each procedure.

visibility calculations dominate the computation in the hierarchical algorithm[2].

[2] In its current version, the program uses a fixed number of rays to determine the mutual visibility between two polygons. The cost of visibility computation is thus roughly proportional to the number of rays used. In the statistics shown here, 16 rays were used, a relatively small number. Using more rays would increase the percentage of time devoted to visibility tests.

Of course this is relative to the algorithm used. A better approach, e.g. with a pre-processing step, as in Teller et al. [9] could probably reduce the relative importance of visibility.

2.2 Initial Linking

The second important observation concerns the actual cost of the initial linking step. As mentioned in the introduction, this cost is at least quadratic in the number of polygons, since each pair of input polygons has to be tested to determine if a link should be established. Since this step is performed before any transfer has commenced, it is a purely geometric visibility test, in this instance implemented by ray-casting. The cost of this test for each polygon pair can vary significantly, depending on the nature of the scene and the type of ray-casting acceleration structure used. In all the examples described below, a BSP tree is used to accelerate the ray-casting process.

Table 2. Total computation time and cost of initial linking (in seconds).

Name	n	Total Time	Initial Linking
Full office	170	301	5.13
Dining room	402	4824	436
East room	1006	587	194
West room	1647	1017	476
Hevea	2355	4253	1597

Table 2 presents timing results for all test scenes. The total computation time is given for ten steps of the multigridding method described by Hanrahan et al [5].[3].

These statistics show that the cost of initial linking grows significantly with the number of polygons in the scene. The dependence on scene structure is also evident, since the growth in computation time between *East room* and *West room* is actually sublinear, while on the other hand the growth of the computation time between *West room* and *Hevea* displays greater than cubic growth in the number of input polygons. For all tests of more than a thousand polygons, it is clear that the cost of initial linking becomes overwhelming. Invoking this cost at the beginning of the illumination computation is particularly undesirable, since a useful image cannot be displayed before its completion. Finally, we note that recent improvements of the hierarchical radiosity method by Smits et al. [8] and Lischinski et al. [6] have allowed significant savings in refinement time, but still

[3] The k'th step of the multigridding method is typically implemented as the k'th "bounce" of light: the first step performs all direct illumination, the second step all secondary illumination, the third all tertiary illumination etc.

rely on the original initial linking stage. Thus initial linking tends to become the most expensive step of the algorithm[4].

Another interesting observation can be made concerning the number of top-level links (links between input polygons) for which the product BF never becomes greater than the refinement threshold $\varepsilon_{\text{refine}}$ over the course of the ten refinement steps[5]. Figure 2 shows the percentage of such links during the first ten iterations. A remarkably high percentage of these links never becomes a candidate for refinement: after 10 steps, between 65% and 95% of the links have not been refined. A significant number of those links probably have very little impact on the radiosity solution.

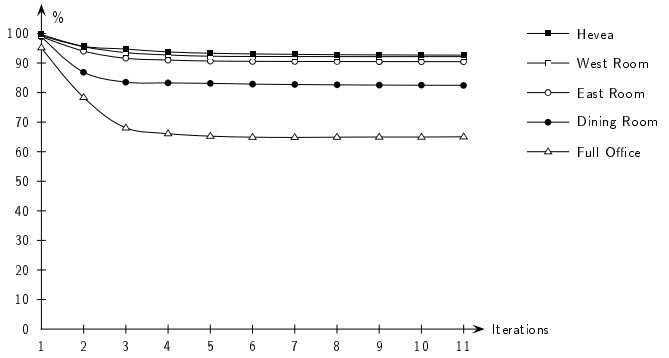


Fig. 2. Percentage of links for which BF does not exceed $\varepsilon_{\text{refine}}$.

What can be concluded from the above discussion? First, if the initial linking step can be eliminated at the beginning of the computation, a useful solution becomes available much more quickly, enhancing the utility of the the hierarchical method. Second, if the top-level links are only computed when they contribute significantly to the solution, there is the potential for large computational savings from eliminating a large number of visibility tests.

2.3 Unnecessary Refinement

The third important observation made when using the hierarchical algorithm is that unnecessary subdivision is incurred, especially for areas which do not include shadow boundaries. This observation is more difficult to quantify than the previous two. To demonstrate the problem we present an image of the *Dining room* scene, and the corresponding mesh (see colour section, Fig. 1 and 2). The simulation parameters were $\varepsilon_{\text{refine}} = 0.5$ and $MinArea = 0.001$.

As can be seen in Fig. 2 in the colour section, the subdivision obtained with these parameters is such that acceptable representation of the shadows

[4] For example Lischinski et al. report a refinement time of 16 minutes for an initial linking time of 2 hours and 16 minutes.

[5] This is the ε used in the original formulation.

is achieved in the penumbral areas caused by the table and chairs. However, the subdivision on the walls is far higher than necessary: the illumination over the wall varies very smoothly and could thus be represented with a much coarser mesh. In previous work it was already noted that radiance functions in regions of full illumination can be accurately represented using a simple mesh based on the structure of illumination [2].

If this unnecessary subdivision is avoided, significant gains can be achieved since the total number of links will be reduced, saving memory, and since an attendant reduction of visibility tests will result, saving computation time.

3 Lazy Evaluation of the Top-level Interactions

In this section a modification of the hierarchical algorithm is proposed, which defers the creation of links between top-level surfaces until such a link is deemed necessary. The basic idea is to avoid performing any computation that does not have a sizable impact on the final solution, in order to concentrate on the most important energy transfers. Thus it is similar to the rationale behind progressive refinement algorithms. At the same time it remains consistent with the hierarchical refinement paradigm, whereby computation is only performed to the extent required by the desired accuracy.

To accomplish this, a criterion must be defined to decide whether a pair of surfaces should be linked. In our implementation we use a specific threshold $\varepsilon_{\text{link}}$on the product BF. Top-level links are then created *lazily*, only once the linking criterion is met during the course of the simulation.

3.1 Description of the Algorithm

In the original hierarchical radiosity algorithm, two polygons are either mutually invisible, and thus do not interact, or at least partially visible from each other and thus exchange energy. We introduce a second qualification, whereby a pair of polygons is either *classified* or *unclassified*. A pair will be marked *classified* when some information is available regarding its interaction. Initially, all pairs of polygons are marked as unclassified.

At each iteration, all unclassified pairs of polygon are considered: First their radiosity is compared to $\varepsilon_{\text{link}}$. If they are bright enough, we check (in constant time) if they are at least partially facing each other. If not, the pair is marked as classified and no link is created. If they are facing, we compute an approximation of their form factor, without a visibility test. If the product of the form factors and the radiosity is still larger than $\varepsilon_{\text{link}}$, we mark the pair of polygons as classified, and compute the visibility of the polygons. If they are visible, a link is created using the form factors and visibility already computed. Thus a pair of polygons can become classified either when a link is created, or when the two polygons are determined to be invisible. Figure 3 shows a pseudo-code listing of both the Initial Linking phase and the Main Loop in the original algorithm [5] and Fig. 4 gives the equivalent listing in our algorithm.

```
Initial Linking
for each pair of polygons (p, q)
    if p and q are facing each other
        if p and q are at least partially visible from each other
            link p and q
Main Loop
for each polygon p
    foreach link l leaving p
        if B × F > ε_refine
            refine l
    foreach link l leaving p
        gather l
```

Fig. 3. The Original Algorithm

```
Initial Linking
for each pair of polygons (p, q)
    record it as unclassified
Main Loop
    for each unclassified pair of polygons (p, q)
        if p and q are facing each other
            if B_p > ε_link or B_q > ε_link
                compute the unoccluded FF
                if B × F > ε_link
                    link p and q
                    record (p, q) as classified
        else record (p, q) as classified
    for each polygon p
        for each link l leaving p
            if B × F > ε_refine
                refine l
        for each link l leaving p
            gather l
```

Fig. 4. Pseudo-code listing for our algorithm

The threshold $\varepsilon_{\text{link}}$used to establish top-levels interactions is not the same as the threshold used for BF refinement, $\varepsilon_{\text{refine}}$. The main potential source of error in our algorithm is an incomplete balance of energy. Since energy is transfered across links, any polygon for which some top-level links have not been created is retaining some energy, which is not propagated to the environment.

When recursive refinement is terminated because the product BF becomes smaller than $\varepsilon_{\text{refine}}$, a link is always established, which carries some fraction of this energy (the form factor estimate used in the comparison against $\varepsilon_{\text{refine}}$ is an upper bound of the form factor). On the other hand, when two top-level surfaces are not linked because the product BF is smaller than $\varepsilon_{\text{link}}$, all the corresponding energy is "lost". It is thus desirable to select a threshold such that $\varepsilon_{\text{link}} < \varepsilon_{\text{refine}}$.

In the examples shown below we used $\varepsilon_{\text{link}} = \varepsilon_{\text{refine}}/5$.

The classified/unclassified status of all pairs of input surfaces requires the storage of $\frac{n(n-1)}{2}$ bits of information. We are currently working on compression techniques to further reduce this cost[6].

3.2 Energy Balance

Since radiosity is mainly used for its ability to model light interreflection, it is important to maintain energy consistency when modifying the algorithm. An issue raised by the lazy linking strategy is that "missing" links, those that have not been created because they were deemed insignificant, do not participate in energy transfers. Thus each gather step only propagates some of the energy radiated by surfaces.

If the corresponding energy is simply ignored, the main result is that the overall level of illumination is reduced. However a more disturbing effect can result for surfaces that have very few (or none) of their links actually established: these surfaces will appear very dark because they will receive energy only from the few surfaces that are linked with them.

The solution we advocate in closed scenes is the use of an *ambient term* similar to the one proposed for progressive refinement radiosity [1]. However the distribution of this ambient term to surfaces must be based on the estimated fraction of their interaction with the world that is missing from the current set of links. The sum of the form factors associated with all links leaving a surface gives an estimate of the fraction of this surface's interactions that is actually represented. Thus, in a closed scene, its complement to one represents the missing link. Using this estimate to weight the distribution of the ambient energy, the underestimation of radiosities can be partially corrected: surfaces that have no links will use the entire ambient term, whereas surfaces with many links will be only marginally affected.

However, since form factors are based on approximate formulas, the sum of all form-factors can differ from one, even for a normally linked surface. This comes from our BF refinement strategy: we accept that the form-factor on a link between two dark surfaces be over-estimated, or under-estimated. This may results in energy loss, or creation. If the error we introduced by not linking some surfaces is of the same order – or smaller – than the one due to our lack of precision on the form-factor estimation, using the ambient term will not suffice to correct the energy inbalance.

To quantify the influence of those errors on the overall balance of energy, we compute the following estimate of the incorrect energy:

$$\mathcal{E}_{E_T} = \sum_p |1 - F_p| \, B_p \, A_p \tag{1}$$

[6] The storage cost for the classified bit represents 62 kb for a thousand surfaces, 25 Mb for twenty thousand surfaces.

where A_p is the area of polygon p, B_p its radiosity and F_p the sum of the form factors on all links leaving p. This can be compared to the total energy present in the scene:

$$E_T = \sum_p B_p A_p \tag{2}$$

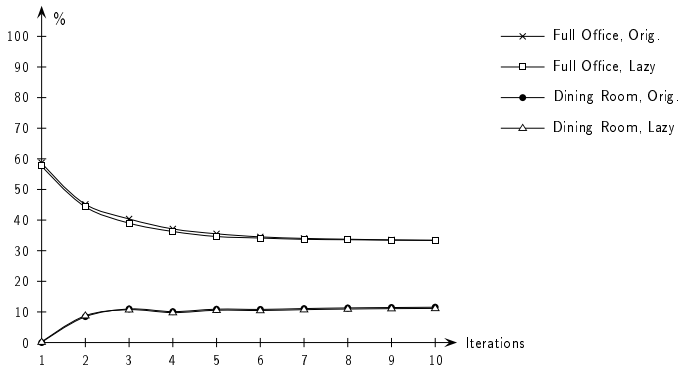


Fig. 5. Incorrect Energy $\mathcal{E}_{E_T}/E_T$

Figure 5 shows a plot of the ratio $\mathcal{E}_{E_T}/E_T$ for the *Dining Room* scene and the *Full Office*, for both the original algorithm and our algorithm. Note that the error can be significant, but is mainly due to the original algorithm.

4 Reducing the Number of Links

The refinement of a link is based on the estimation of an associated error bound. Various criteria have been used that correspond to different error metrics, including the error in the form factor [4], the error in the radiosity transfer [5], and the impact of the error in the radiosity transfer on the final image [8].

All these criteria implicitly assume that the error in the form factor estimate is equivalent to the magnitude of the form factor itself. While this is true for infinitesimal quantities, in many instances it is possible to compute a reasonable estimate of a relatively large form factor. In particular this is true in situations of full visibility between a pair of surfaces.

Consider two patches p and q. A bi-directionnal link between them carries two form factor estimates $F_{p,q}$ and $F_{q,p}$. If we refine the link by dividing p in smaller patches p_i of area A_i (e.g. in a quadtree), the definition of the form factor

$$F_{u,v} = \frac{1}{A_u} \int_{A_u} \int_{A_v} G(dA_u, dA_v) dA_u dA_v \tag{3}$$

where G is a geometric function, implies that the new form factors verify:

$$F_{p,q} = \frac{1}{A_p} \left(\sum_i A_i F_{p_i,q} \right) \tag{4}$$

$$F_{q,p} = \sum_i F_{q,p_i} \tag{5}$$

These relations only concern the exact values of the form factors. However they can be used to compare the new form factor estimates with the old ones, and determine *a posteriori* wether refinement was actually required. If the sum of the F_{q,p_i} is close to the old $F_{q,p}$, and they are not very different from one another, little precision was gained by refining p. Moreover, if $F_{p,q}$ is close to the average of the $F_{p_i,q}$, and the $F_{p_i,q}$ are not too different from one another, then the refinement process did not introduce any additional information. In this case we force p and q to interact at the current level, since the current estimates of form factors are accurate enough.

In our implementation we only allow reduction of links in situations of full visibility between surfaces. We compute the relative variation of the children form factors, which we test against a new threshold $\varepsilon_{\text{reduce}}$. We also check that the difference between the old form factor $F_{p,q}$ and the sum of the $F_{p_i,q}$, and the difference between $F_{q,p}$ and the average of the F_{q,p_i} are both smaller than $\varepsilon_{\text{reduce}}$.

If we note $F_{u,v}$ our current estimation of the form-factor between two patches u and v, and assuming we want to refine a patch p in p_i, we note:

$$F^{\min}_{p,q} = \min_i(F_{p_i,q}) \qquad F^{\min}_{q,p} = \min_i(F_{q,p_i})$$
$$F^{\max}_{p,q} = \max_i(F_{p_i,q}) \qquad F^{\max}_{q,p} = \max_i(F_{q,p_i})$$
$$F'_{p,q} = \frac{1}{A_p}\left(\sum_i A_i F_{p_i,q}\right) \qquad F'_{q,p} = \sum_i F_{q,p_i}$$

and we refine p if any of the following is true:

$$\frac{F^{\max}_{p,q} - F^{\min}_{p,q}}{F^{\max}_{p,q}} > \varepsilon_{\text{reduce}} \qquad \frac{F^{\max}_{q,p} - F^{\min}_{q,p}}{F^{\max}_{q,p}} > \varepsilon_{\text{reduce}}$$
$$\frac{\left|F'_{p,q} - F_{p,q}\right|}{F'_{p,q}} > \varepsilon_{\text{reduce}} \qquad \frac{\left|F'_{q,p} - F_{q,p}\right|}{F'_{q,p}} > \varepsilon_{\text{reduce}}$$

The decision to cancel the subdivision of a link is based purely on geometrical properties, therefore it is permanent. The link is marked as "un-refinable" for the entire simulation.

The check whether a link is worth refining involves the computation of form factor estimates to and from all children of patch p. Thus the associated cost in time is similar to that of actually performing the subdivision. If a single level of refinement is avoided by this procedure, there will be little gain in computation time, but the reduction in the number of links will yield memory savings. But if link reduction happens "early enough", several levels of refinement can be avoided. In our test scenes, an implementation of this algorithm reduced significantly the number of quadtree nodes and links (see Fig. 6), with a slightly smaller reduction in computation time because of the cost of the extra form factor estimates (see Fig. 7).

5 Results

5.1 Lazy Linking

Figures 3 in coulour section shows the same scene as in Fig. 1, computed using the lazy linking strategy of Sect. 3. Note that it is visually indistinguishable from its original counterpart. Figure 4 plots the absolute value of the difference between these two images.

5.2 Reduction of the Number of Links

To measure the performance of the reduction criterion, we computed the ratio of the number of quadtree nodes (surface elements) obtained with this criterion, to the number of nodes obtained with the original algorithm. The graph in Fig. 6a plots this ratio against the number of iterations. Note that an overall reduction by nearly a factor of two is achieved for all scenes. Figure 6b shows a similar ratio for the number of links. This global reduction of the number of objects involved leads to a similar reduction of the memory needed by the algorithm, thus making it more practical for scenes with more polygons.

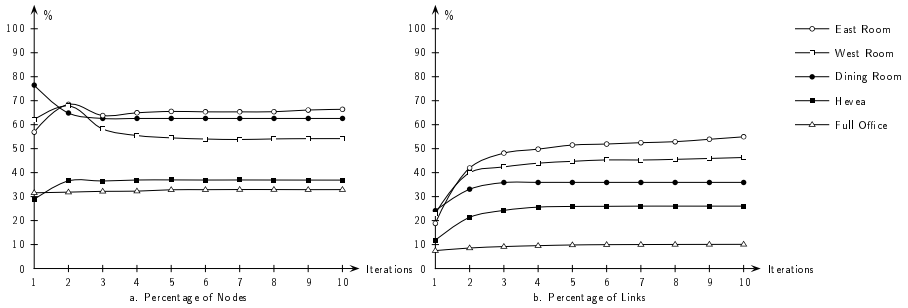


Fig. 6. Percentage of nodes and links left after reduction.

Figure 7 shows the ratio of the computation times using the improved criterion and the original algorithm. The reduction of the number of links has a dramatic impact on running times, with speedups of more than 50%.

Figure 5 and 6 in colour section shows the image obtained after link reduction. Note the variation in the mesh on the walls, and the similarity of the shaded image with the ones in Figs. 1 and 3. Figure 7 plots the absolute value of the difference between the image produced by the original algorithm and the image obtained after link reduction. Note that part of the differences are due to the lazy linking strategy of Sect. 3. So Figure 8 shows the difference between lazy linking and reduction of the number of links.

5.3 Overall Performance Gains

Timing results are presented in Table 3. As expected, a significant speedup is achieved, particularly for complex scenes. For all scenes, ten iterations with lazy

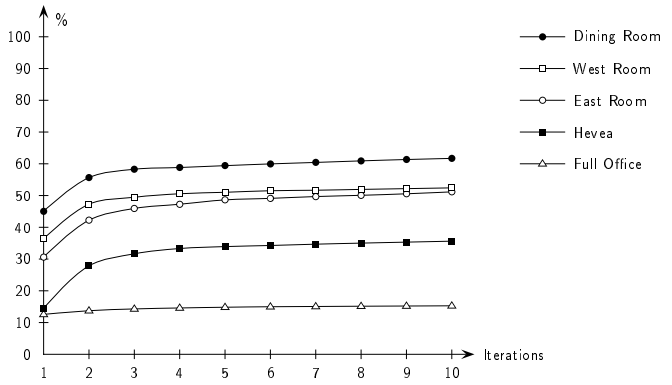


Fig. 7. Percentage of computation time using link reduction.

linking took less time to compute than the first iteration alone with the original algorithm. Finally, using lazy linking and reduction produces a useful image in a matter of minutes even for the most complex scenes in our set.

Table 3. Time needed for ten iterations (and time for producing the first image).

Name	n	Original Algorithm		with Lazy Linking...		and Reduction	
Full office	170	301 s	(242 s)	287 s	(234 s)	43 s	(30 s)
Dining room	402	4824 s	(4191 s)	4051 s	(3911 s)	657 s	(552 s)
East room	1006	587 s	(378 s)	377 s	(191 s)	193 s	(59 s)
West room	1647	1017 s	(752 s)	514 s	(277 s)	270 s	(101 s)
Hevea	2355	4253 s	(2331 s)	1526 s	(847 s)	543 s	(122 s)

6 Conclusions and Discussion

We have presented the results of an experimental study conducted on a variety of scenes, showing that visibility calculations represent the most expensive portion of the computation. Two improvements of the hierarchical algorithm were proposed. The first modification creates top-level links *lazily*, only when it is established that the proposed link will have a definite impact on the simulation. With this approach the hierarchical algorithm still remains quadratic in the number of input surfaces, but no work and very little storage is devoted to the initial linking phase. The resulting algorithm is more progressive in that it produces useful images very quickly. Note that the quadratic cost in the number of input surfaces can only be removed by clustering methods [7].

An improved subdivision criterion was introduced for situations of full visibility between surfaces, which allows a significant reduction of the number of links.

Future work will include the simplification of the hierarchical structure due to multiple sources and subsequent iterations. A surface that has been greatly refined because it receives a shadow from a given light source can be fully illuminated by a second source, and the shadow become washed in light.

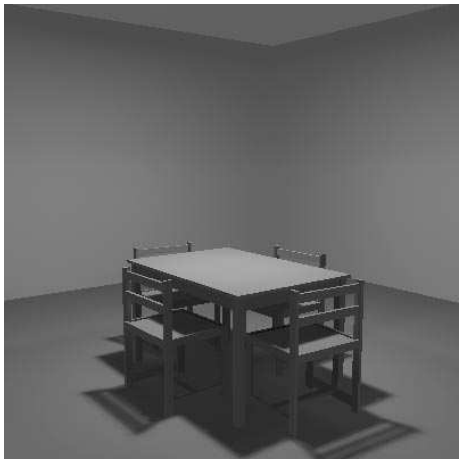
Better error bounds, both on form factor magnitude and global energy transfers, should allow even greater reduction of the number of links. Accurate visibility algorithms can be used to this end, by providing exact visibility information between pairs of surfaces.

7 Acknowledgments

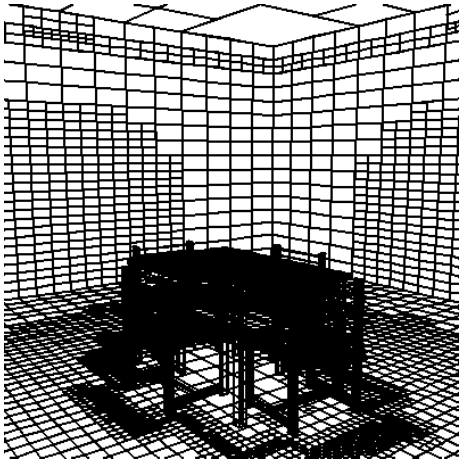
George Drettakis is a post-doc hosted by INRIA and supported by an ERCIM fellowship. The hierarchical radiosity software was built on top of the original program kindly provided by Pat Hanrahan.

References

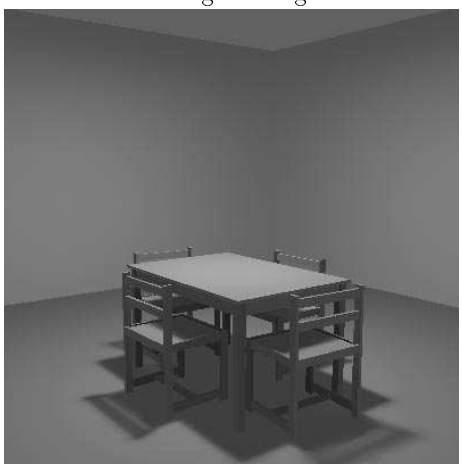
1. Cohen, M. F., Chen, S. E., Wallace, J. R., Greenberg, D. P.: A Progressive Refinement Approach to Fast Radiosity Image Generation. SIGGRAPH (1988) 75–84
2. Drettakis, G., Fiume, E.: Accurate and Consistent Reconstruction of Illumination Functions Using Structured Sampling. Computer Graphics Forum (Eurographics 1993 Conf. Issue) 273–284
3. Goral, C. M., Torrance, K. E., Greenberg, D. P., Bataille, B.: Modeling the Interaction of Light Between Diffuse Surfaces. SIGGRAPH (1984) 213–222
4. Hanrahan, P. M., Salzman, D.: A Rapid Hierarchical Radiosity Algorithm for Unoccluded Environments. *Eurographics Workshop on Photosimulation, Realism and Physics in Computer Graphics*, June 1990.
5. Hanrahan, P. M., Salzman, D., Auperle, L.: A Rapid Hierarchical Radiosity Algorithm. SIGGRAPH (1991) 197–206
6. Lischinski, D., Tampieri, F., Greenberg, D. P.: Combining Hierarchical Radiosity and Discontinuity Meshing. SIGGRAPH (1993)
7. Sillion, F.: Clustering and Volume Scattering for Hierarchical Radiosity calculations. *Fifth Eurographics Workshop on Rendering*, Darmstadt, June 1994 (in these proceedings).
8. Smits, B. E., Arvo, J. R., Salesin, D. H.: An Importance-Driven Radiosity Algorithm. SIGGRAPH (1992) 273–282
9. Teller, S. J., Hanrahan, P. M.: Global Visibility Algorithm for Illumination Computations. SIGGRAPH (1993) 239–246



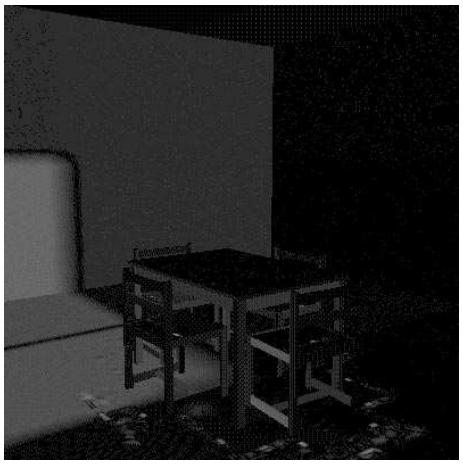
1. The Original Algorithm



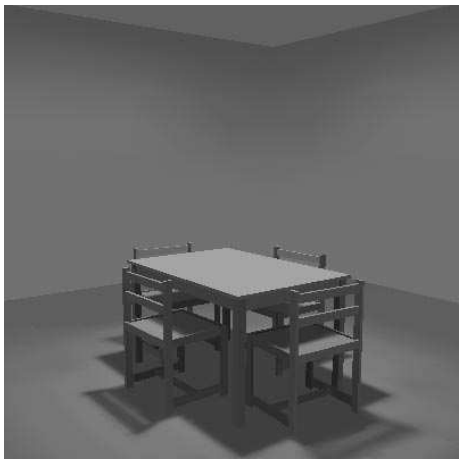
2. The Grid Produced



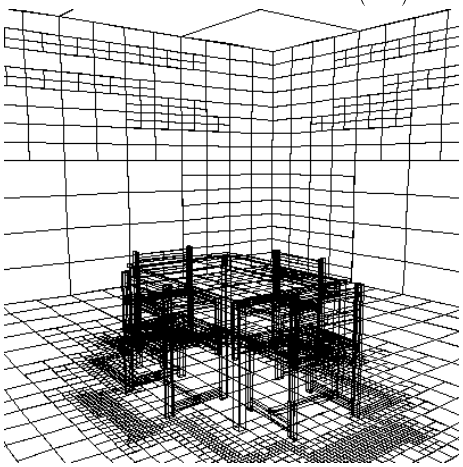
3. With Lazy Linking



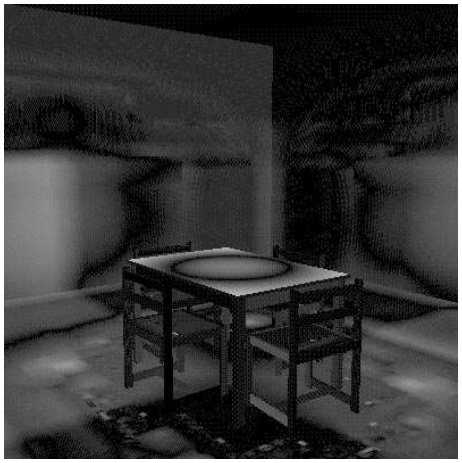
4. Diff. Between 1. and 3. (×8)



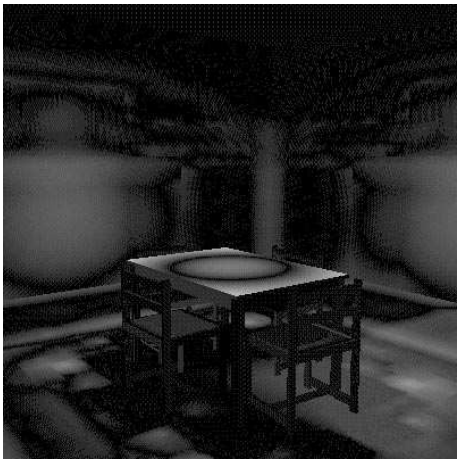
5. With Link Reduction



6. The Grid Produced



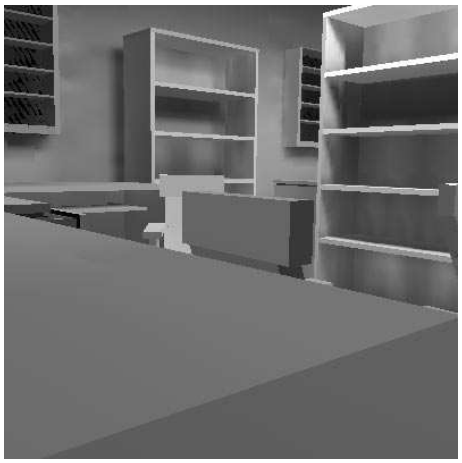
7. Diff. Between 1. and 5. (×8)



8. Diff. Between 3. and 5. (×8)

9. Full Office

10. East Room



11. West Room

12. Hevea

2.7.3 Wavelet Radiosity on Arbitrary planar surfaces (EGWR 2000)

Auteurs : Nicolas H , François C et Laurent A
Conférence : 11e *Eurographics Workshop on Rendering*, Brno, République tchèque.
Date : juin 2000

Wavelet Radiosity on Arbitrary Planar Surfaces

Nicolas Holzschuch[1], François Cuny[2] and Laurent Alonso[1]
ISA research team
LORIA[3]
Campus Scientifique, BP 239
54506 Vandœuvre-les-Nancy CEDEX, France

Abstract. Wavelet radiosity is, by its nature, restricted to parallelograms or triangles. This paper presents an innovative technique enabling wavelet radiosity computations on planar surfaces of arbitrary shape, including concave contours or contours with holes. This technique replaces the need for triangulating such complicated shapes, greatly reducing the complexity of the wavelet radiosity algorithm and the computation time. It also gives a better approximation of the radiosity function, resulting in better visual results. Our technique works by separating the radiosity function from the surface geometry, extending the radiosity function defined on the original shape onto a simpler domain – a parallelogram – better behaved for hierarchical refinement and wavelet computations.

1 Introduction

Wavelet radiosity [12] is one of the most interesting technique for global illumination simulation. Recent research [7] has shown that higher order multi-wavelets ($\mathcal{M}_2$ and $\mathcal{M}_3$) are providing a very powerful tool for radiosity computations. Multi-wavelets can approximate the radiosity function efficiently with a small number of coefficients. As a consequence, they give a solution of better quality in a shorter time.

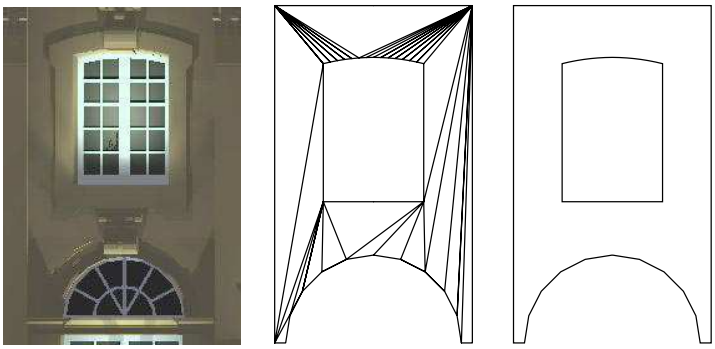
Multi-wavelets are defined only on parallelograms and triangles. This causes problems for radiosity computations on scenes coming from real world applications, such as architectural scenes, or CAD scenes. In such scenes, planar surfaces have a fairly complicated shape (see figure 1 and 12(a)). To do wavelet radiosity computations on such scenes, we have to tessellate these planar shapes into triangles and parallelograms, which results in a large number of input primitives (see figure 1(b)). Furthermore, this decomposition is purely geometrical and was not based on the illumination, yet it will influence our approximation of the radiosity function. In some cases, this geometric decomposition results in a poor illumination solution (see figure 2(a) and 11(a)).

In the present paper, we separate the radiosity function from the surface geometry. This enables us to exploit the strong approximating power of multi-wavelets for radiosity computations, with planar surfaces of arbitrary shape – including concave contours, contours with holes or disjoint contours. Our algorithm results in a better approximation of the radiosity function (see figure 2(b) and 11(b)) with a smaller number of input primitives, faster convergence and lower memory costs.

[1] INRIA Lorraine.

[2] Institut National Polytechnique de Lorraine.

[3] UMR n° 7503 LORIA, a joint research laboratory between CNRS, Institut National Polytechnique de Lorraine, INRIA, Université Henri Poincaré and Université Nancy 2.

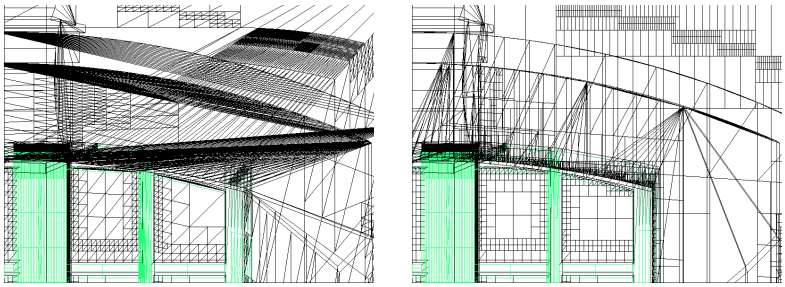


(a) Detail of figure 12(a)

(b) Tessellated: 32 triangles

(c) Our algorithm: 1 surface

Fig. 1. Planar surfaces can have a fairly complicated shape



(a) Tessellated

(b) Our Algorithm

Fig. 2. Wavelet radiosity on arbitrary planar surfaces (see also figure 11)

Our algorithm extends the radiosity function defined on the original shape onto a simpler domain, better behaved for hierarchical refinement and wavelet computations. This extension of the radiosity function is defined to be easily and efficiently approximated by multi-wavelets. The wavelet radiosity algorithm is modified to work with this abstract representation of the radiosity function.

Our paper is organised as follows: in section 2, we will review previous work on radiosity with planar surfaces of complicated shape. Section 3 is a detailed explanation of our algorithm and of the modifications we brought to the wavelet radiosity algorithm. Section 4 presents the experiments we have conducted with our algorithm on different test scenes. Finally, section 5 presents our conclusions.

2 Previous work

The wavelet radiosity method was introduced by [12]. It is an extension of the radiosity method [11] and especially of the hierarchical radiosity method [13]. It allows the use of higher order basis functions in hierarchical radiosity.

In theory, higher order wavelets are a very powerful tool to approximate rapidly varying functions with little coefficients. In practice, they have several drawbacks, especially in terms of memory costs. In the early implementations of wavelets bases in the radiosity algorithm, these negative points were overcoming the positive theoretical advantages [19]. Recent research [7] has shown that using new implementation methods [2, 3, 7, 18, 21] we can actually exploit the power of higher order wavelets, and that their positive points are now largely overcoming the practical problems. They provide a better approximation of the radiosity function, with a small number of coefficients, resulting in faster convergence and smaller memory costs.

On the other hand, higher order wavelets, and especially multi-wavelets ($\mathcal{M}_2$ and $\mathcal{M}_3$) are defined as the tensor products of one-dimensional wavelets. As a consequence, they are defined over a square. The definition can easily be extended on parallelograms or triangles, but higher order wavelets are not designed to describe the radiosity function over complex surfaces.

Such complex surfaces can occur in the scenes on which we do global illumination simulations. Especially, scenes constructed using CAD tools such as CSG geometry or extrusion frequently contain complex planar surfaces, with curved boundaries or holes in them.

The simplest solution to do radiosity computations on such surfaces is to tessellate them into triangles, and to do radiosity computations on the result of the tessellation. This method has several negative consequences on the radiosity algorithm:

- It increases the number of input surfaces and the algorithmic complexity of the radiosity algorithm is linked to the square of the number of input surfaces.
- The tessellation is made before the radiosity computations and it influences these computations. It can prevent us from reaching a good illumination solution.
- The tessellation does not allow a hierarchical treatment over the original surface, only over each triangle created by the tessellation. We can not fully exploit the capabilities of hierarchical radiosity, and especially of wavelet radiosity.
- By artificially subdividing an input surface into several smaller surfaces, we are creating discontinuities. These discontinuities will have to be treated at some point in the algorithm.
- Tessellation can create poorly shaped triangles (see figure 1(b)), or slivers. These slivers can cause Z-buffer artifacts when we visualise the radiosity solution, and are harder to detect in visibility tests (*e.g.* ray-casting).

Some of these problems can be removed by using clustering [10, 16, 17]. In clustering, neighbouring patches are grouped together, into a *cluster*. The cluster receives radiosity and distributes it to the patches that it contains. On the other hand, current clustering strategies are behaving poorly in scenes with many small patches located close to each other [14]. It would probably be more efficient to apply clustering to the original planar surfaces instead of applying it to the result of the tessellation.

A better grouping strategy is face-clustering [20]. In face-clustering, neighbouring patches are grouped together according to their coplanarity. Yet even face-clustering depends on the geometry created by the tessellation. Furthermore, it would not allow us to exploit the strong approximating power of multi-wavelets.

- if the original planar shape is polygonal:
  - compute its convex hull (in linear time) using the chain of points [9].
  - compute the minimal enclosing parallelograms of the convex hull (in linear time) using Schwarz *et al.* [15].
  - if the previous algorithm gives several enclosing parallelograms, select the one that has angles closer to $\frac{\pi}{2}$.
- if the original shape is a curve, or contains curves:
  - approximate the curve by a polygon
  - compute the enclosing parallelogram of the polygon
  - compute the extrema of the curve in the directions of the parallelogram.
  - if needed, extend the parallelogram to include these extrema.

Fig. 3. Our algorithm for finding an enclosing parallelogram.

Bouatouch *et al.* [5] designed a method for discontinuity meshing and multi-wavelets. In effect, they are doing multi-wavelets computations over a non-square domain. However, their algorithm requires several expensive computations of push-pull coefficients. Our algorithm avoids these computations.

Baum *et al.* [1] designed a method for radiosity computations with arbitrary planar polygons, including polygons with holes. Their method ensures that the triangles produced are well-shaped, and suited for radiosity computations. Since it is designed for non-hierarchical radiosity, it is done in a preliminary step, before all radiosity computations. Our method, designed for wavelet radiosity, acts during the global illumination simulation, and adapts the refinement to the radiosity.

3 The Extended Domain Algorithm

In this section, we present our algorithm for wavelet radiosity computations on planar surfaces of arbitrary shape. Our algorithm separates the radiosity function from the surface geometry; we introduce a simple domain that will be used for radiosity computations. The radiosity function on the original surface is inferred from the radiosity on the simple domain.

Section 3.1 explains how we select an extended domain for our computations. In section 3.2, we describe how we extend the definition of the radiosity function over this domain. The extended domain is then used in a wavelet radiosity algorithm like an ordinary patch, with some specific adjustments. These adjustments are described in section 3.3.

3.1 Selection of an extended domain

The first step of our algorithm is the choice of an extended domain, which we will use for wavelet radiosity computations. This extended domain must obey two rules:

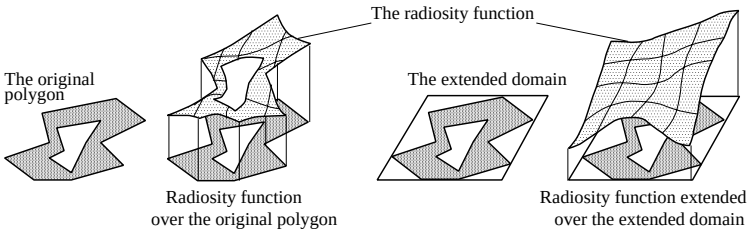


Fig. 4. Extending the radiosity function over the extended domain.

- it must enclose the original shape,
- it must be well suited for wavelet radiosity computations.

Since multi-wavelets ($\mathcal{M}_2$, $\mathcal{M}_3$...) are defined as tensor products of one-dimensional wavelets, the second rule implies that the extended domain must be a parallelogram. Moreover, if this parallelogram is closer to a rectangle, there will be less distortions in the wavelet bases, resulting in a better approximation of the radiosity function. So we want the angles of the parallelogram to be close to $\frac{\pi}{2}$.

Since only radiosity computations made on the original shape are of interest, we also want the enclosing parallelogram to be as close as possible from the original shape.

Basically, any parallelogram satisfying these criterions could be used with our algorithm. The algorithm used in our implementation is described in figure 3. The key point is that this algorithm runs in linear time with respect to the number of vertices: the convex hull of a chain of points in 2D can be computed in linear time [9], and Schwarz's algorithm for the enclosing parallelogram is also linear [15].

3.2 Extending the radiosity function over this domain

Once we have an extended domain, we need to define the radiosity function over this domain. This extension of the radiosity function must obey two rules (see figure 4):

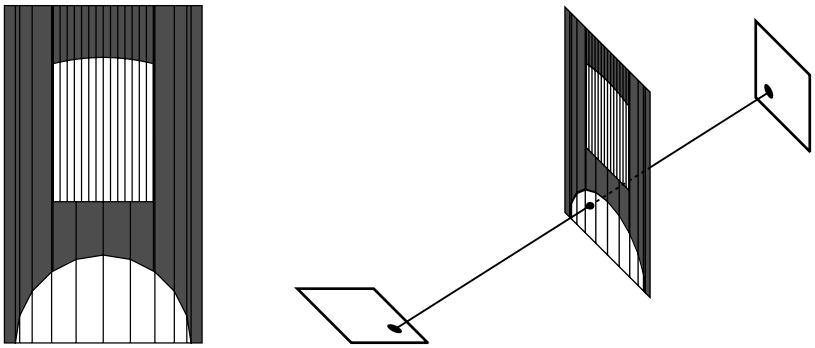
- it must be equal to the original radiosity function over the original domain
- it must be as simple and as smooth as possible, to be efficiently approximated by multi-wavelets.

The second point is crucial: we have to compute the radiosity function over the entire domain. Because of the hierarchical nature of wavelets, during the push-pull step radiosity values computed at one point of the domain can influence other points of the domain. So our extension of the radiosity function must be computed with the same precision regardless of whether we are on the original surface or not.

Since the discontinuities of the radiosity function and its derivatives only come from visibility discontinuities, we do not want to introduce more visibility discontinuities in our extension. We define an *extended visibility* function V': the visibility between a point Q in space and a point P on the extended domain is defined as the visibility between Q and P', where P' is defined as the closest point from P on the original planar surface:

$$V'(Q,P) = V(Q,P')$$

Of course, if P is already on the original planar surface, P' is equal to P. In that case, the extended visibility function is equal to the standard visibility function.



(a) Trapezoidal map for the surface in figure 1(c)

(b) Using the trapezoidal map for visibility queries

Fig. 5. Trapezoidal map of an arrangement of line segments

The radiosity function on the extended domain is then defined as the radiosity function, as computed by the wavelet radiosity algorithm, using this extended visibility function in the radiosity kernel.

3.3 Using the extended domain in the wavelet radiosity algorithm

In this section, we describe our adaptation of the wavelet radiosity algorithm to work with our extended domains. We use a standard wavelet radiosity algorithm [7, 21]. The core of the algorithm is left unchanged (refinement oracle, link storage). We will review here the points that require some special attention:

- reception and push-pull
- visibility
- emission
- refinement

Reception and Push-Pull. The wavelet radiosity algorithm is a hierarchical algorithm. During the push-pull step, radiosity values computed at one point of the patch can influence the representation of radiosity for the entire patch. Hence, we want the same precision for all radiosity computations over the entire patch.

For reception, the extended domain is therefore treated just like an ordinary patch. All parts of the extended domain are receiving radiosity, with the same precision, regardless of whether or not they belong to the original planar surface.

Similarly, we are doing the push-pull step over the entire extended domain, without any reference to the original surface. Since our extended domain is by design a parallelogram, instead of an ordinary polygon, we do not have to compute any expensive push-pull coefficients.

Visibility. For all the visibility computations, only the original planar surface can act as an occluder. The extended domain is never used in visibility computations.

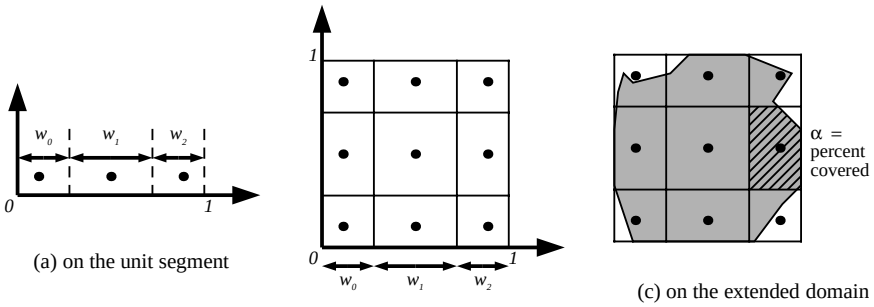


Fig. 6. The weights of the quadrature points can be seen as the area of a *zone of influence*.

To detect if the original surface is actually occluding an interaction, we compute the trapezoidal map of an arrangement of line segments [4, 8] over the segments of the contour of the original surface (see figure 5). For each trapeze, we store its status – whether it is inside or outside of the original surface.

Using randomized algorithms, trapezoidal maps can be constructed in time $O(n \log n)$, where n is the number of vertices. Once constructed, they can be queried in $O(\log n)$ time. Since the construction algorithm is randomised, we shuffle the segments of the original surface before building the trapezoidal map.

Visibility queries in our radiosity algorithm are visibility queries between two points, either two quadrature points [7] or the closest point on the original surface from a quadrature point (see section 3.2). We compute the intersection between the ray joining these quadrature points and the supporting plane of the original surface, check whether the intersection point is inside the bounding box of the extended domain, then check whether it is inside the extended domain itself, then query the trapezoidal map to check if it is inside or outside the original surface.

Emission. During the reception, the entire extended domain has received illumination. The radiosity received over parts of the extended domain that are not included in the original surface does not exist in reality, and it should not be sent back into the scene. Otherwise, there would be an artificial creation of energy, violating the principle of conservation of energy.

Because of the hierarchical nature of the wavelet radiosity algorithm, it would be difficult to compute the exact part of this radiosity function that really exists. Instead, we act on the weights of the quadrature points.

In the wavelet radiosity algorithm, all the transfer coefficients between an emitter and a receiver are computed using quadratures. Quadratures allow the evaluation of a complex integral by sampling the function being integrated at the quadrature points, and multiplying the values by quadrature weights. Most implementations use Legendre-Gauss quadratures.

Since the weights are positive and their sum is equal to 1, you can visualise them as being the length of a *zone of influence* for the corresponding quadrature point (see figure 6(a) for the one dimension case). The same applies in two dimensions: the weights of the quadrature points can be seen as the area of a zone of influence (see

for each interaction $s \rightarrow r$:
  for each quadrature point q_i on the emitter s
    A_i = area of influence of q_i
    $\alpha_i \leftarrow$ percentage of A_i that is inside the original emitter
    q_i' = nearest point from q_i on the emitter
    for each quadrature point p_j on the receiver r
      p_j' = nearest point on the receiver
      $V(q_i', p_j')$ = visibility between q_i' and p_j'
      $G(q_i, p_j)$ = radiosity kernel between q_i and p_j
      $B_r += \alpha_i w_i w_j B_s(q_i) V(q_i', p_j') G(q_i, p_j)$
    end for
  end for

Fig. 7. Pseudo-code for wavelet radiosity emission using the extended domain.

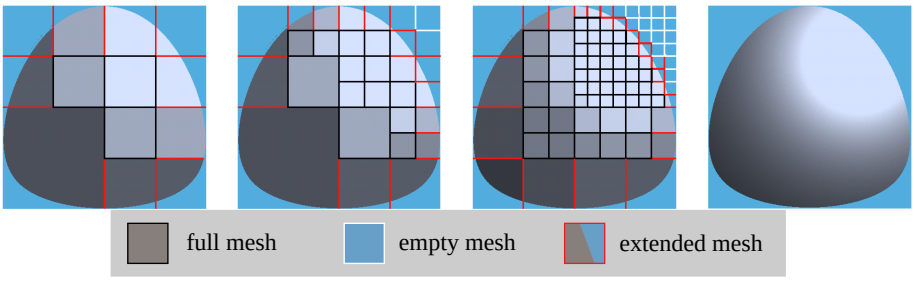


Fig. 8. Refinement of the extended domain

figure 6(b)); the weight of quadrature point $p_{i,j}$ is $w_i w_j$. Please note that these zones of influence are not equal to the Voronoï diagram of the quadrature points.

We suggest an extension to the Gaussian quadrature to take into account the fact that the extended domain is not entirely covered by the actual emitter: the weight of a quadrature point is multiplied by the proportion of its area of influence that is actually covered by the emitter. For example, on figure 6(c), the weight of the quadrature point in the hashed area should be $w_1 w_2$. Since the fraction of its area of influence covered by the emitter is α, the weight used in the computation will be $\alpha w_1 w_2$.

Our method allows for a quick treatment of low precision interactions, and for high precision interactions, it tends toward the exact value. The more we refine an interaction, the more precision we get on the radiosity on the emitter. We also get the exact value if the zone of influence is entirely full or entirely empty.

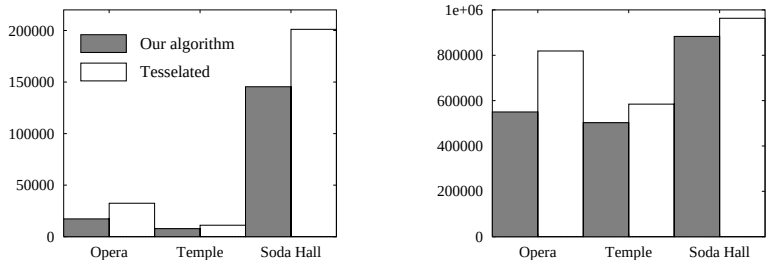
In some cases, it can happen that the quadrature point falls outside the original emitter. We use these quadrature points anyway.

Figure 7 shows the pseudo-code for radiosity emission using the extended domain.

Refinement. As with the original wavelet radiosity algorithm, the extended domain can be subdivided if the interaction needs to be subdivided. The refinement oracle deals with the extended domain as it would deal with any other patch. Because of the hierarchical representation of the radiosity function in the wavelet radiosity algorithm, we must have the same precision on the radiosity function over the entire domain. The push-pull step can make parts of the domain that are not inside the original surface influence our representation of the radiosity function over the entire domain.

Name	# initial surfaces	after tessellation	ratio
Opera	17272	32429	1.88
Temple	7778	11087	1.43
Soda Hall	145454	201098	1.38

Table 1. Description of our test scenes



(a) Before refinement

(b) After refinement

Fig. 9. Number of patches in our test scenes

If the extended domain is refined, we deal with each part of the subdivided extended domain as we would deal with the original extended domain. Two special cases can appear (see figure 8):

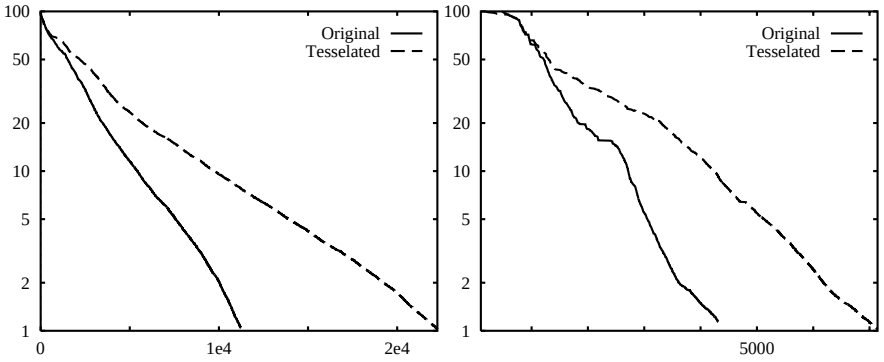
- if the result of the subdivision does not intersect at all with the original planar surface, it is empty. Therefore it cannot play a role in the emission of radiosity, but we keep computing the radiosity function over this patch.
- if the result of the subdivision is totally included inside the original planar surface. In that case, we are back to the standard wavelet radiosity algorithm on parallelograms.

4 Experiments

We have tested our algorithm for wavelet radiosity on arbitrary planar surfaces on various test scenes (see figure 12 for images of our test scenes, and table 1 and figure 9(a) for their description). We were interested in a comparison between our algorithm and the standard wavelet radiosity algorithm, acting on parallelograms and triangles. All the computations were conducted on the same computer, a SGI Origin 2000, using a parallel version [6] of our wavelet radiosity algorithm [7, 21].

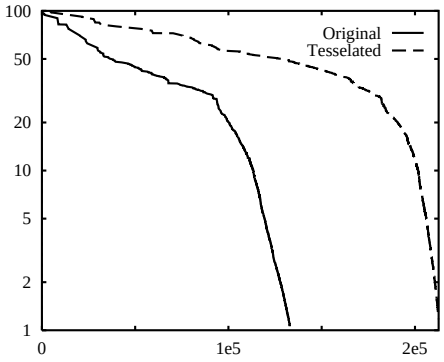
In all these test scenes, the number of surfaces after tesselation is less than twice the number of surfaces in the original scene. Much less than what could be expected from figure 1. Most of the initial surfaces in the scenes are parallelograms or triangles, and don't require tesselation.

The first result is that our algorithm gives better visual quality than doing wavelet radiosity computations on a tessellated surface (see figure 2 and 11). Our separation of the radiosity function from the surface geometry results in a better approximation of the radiosity function.



(a) Opera

(b) Temple



(c) Soda Hall

Fig. 10. Convergence rate (un-shot energy over initial energy) as a function of computation time (in seconds).

Beyond this important result, we were interested in a comparison of computation time and memory costs for both algorithms.

Obviously, our algorithm reduces the number of patches, and therefore the memory cost of the initial scene (see table 1 and figure 9(a)). According to our computations, it also reduces the number of patches in the final scene, although not in the same proportions (see figure 9(b)).

The later result was to be expected: the wavelet radiosity algorithm will refine the original scene a lot, resulting in numerous sub-patches. The number of patches in the scene after the radiosity computations is mainly linked to the complexity of the radiosity function itself, and not to the complexity of the scene. However, it appears that our algorithm results in more efficient refinement, since we reach convergence with a smaller number of patches. In some scenes, we can reach convergence with 30 % less patches.

The fact that our algorithm allows for more efficient refinement also appears in the

computation times (see figure 10). In our experiments, we measure the energy initially present in the scene and the energy that hasn't yet been propagated in the scene. The ratio of these two measures tells us how far we are from complete convergence. Figure 10 displays this ratio as a function of the computation time, both for our algorithm and for the wavelet radiosity algorithm operating on a tessellated version of the scene. Our algorithm ensures a faster convergence on all our test scenes. The speedup is of about 30 %, which shows that acting on the original planar surface instead of the tessellated surface gives more efficient refinement.

5 Conclusion

In conclusion, we have presented a method to separate the radiosity function from the surface geometry. This method removes the need to tessellate complex planar surfaces, resulting in a more efficient global illumination simulation, with better visual quality. Our method results in faster convergence, with smaller memory costs.

In our future work, we want to extend this algorithm to discontinuity meshing. Discontinuity meshing introduces a geometric model of the discontinuities of the radiosity function and its derivatives, the *discontinuity mesh*. The discontinuity mesh provides optimal meshing for radiosity computations near the discontinuities. The discontinuity mesh is a complicated structure, and it can influence radiosity computations away from the discontinuities, for example because of triangulation. We want to use our algorithm to smoothly integrate the discontinuity mesh in the natural subdivision for multi-wavelet radiosity, removing the need to tesselate the discontinuity mesh.

We also want to explore a combination of our algorithm with clustering techniques. First, our algorithm could be used to group together neighbouring coplanar patches in a natural way. This would help the clustering strategy [14] and give a more accurate result. Second, we would like to integrate our algorithm with face-clustering, bringing multi-wavelets into face-clusters.

Finally, our separation of the radiosity function from the surface geometry could also be used to compute radiosity using multi-wavelets on curved surfaces. There are several parametric surfaces for which the limits of the parametric space are not square. We suggest using our algorithm to enclose these limits into a square limit, making it easier for multi-wavelets.

6 Acknowledgements

Permission to use the Soda Hall model[4] was kindly given by Prof. Carlo Sequin.

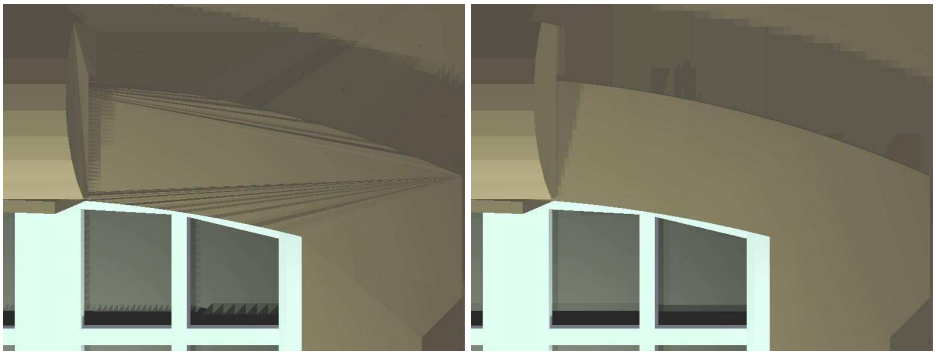
Jean-Claude Paul has started and motivated all this research. The authors would like to thank him for his kind direction, support and encouragements.

References

1. D. R. Baum, S. Mann, K. P. Smith, and J. M. Winget. Making Radiosity Usable: Automatic Preprocessing and Meshing Techniques for the Generation of Accurate Radiosity Solutions. *Computer Graphics (ACM SIGGRAPH '91 Proceedings)*, 25(4):51–60, July 1991.
2. P. Bekaert and Y. Willems. Error Control for Radiosity. In *Rendering Techniques '96 (Proceedings of the Seventh Eurographics Workshop on Rendering)*, pages 153–164, New York, NY, 1996. Springer-Verlag/Wien.

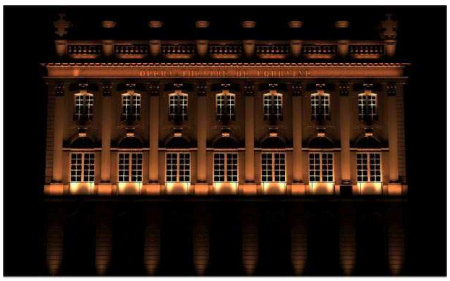
[4]The Soda Hall model is available on the web, at `http://www.cs.berkeley.edu/~kofler`.

3. P. Bekaert and Y. D. Willems. Hirad: A Hierarchical Higher Order Radiosity Implementation. In *Proceedings of the Twelfth Spring Conference on Computer Graphics (SCCG '96)*, Bratislava, Slovakia, June 1996. Comenius University Press.
4. J.-D. Boissonnat and M. Yvinec. *Algorithmic Geometry*. Cambridge University Press, 1998.
5. K. Bouatouch and S. N. Pattanaik. Discontinuity Meshing and Hierarchical Multiwavelet Radiosity. In W. A. Davis and P. Prusinkiewicz, editors, *Proceedings of Graphics Interface '95*, pages 109–115, San Francisco, CA, May 1995. Morgan Kaufmann.
6. X. Cavin, L. Alonso, and J.-C. Paul. Parallel Wavelet Radiosity. In *Second Eurographics Workshop on Parallel Graphics and Visualisation*, pages 61–75, Rennes, France, Sept. 1998.
7. F. Cuny, L. Alonso, and N. Holzschuch. A novel approach makes higher order wavelets really efficient for radiosity. *Computer Graphics Forum (Eurographics 2000 Proceedings)*, 19(3), Sept. 2000. To appear. Available from http://www.loria.fr/~holzschu/Publications/paper20.pdf.
8. O. Devillers, M. Teillaud, and M. Yvinec. Dynamic location in an arrangement of line segments in the plane. *Algorithms Review*, 2(3):89–103, 1992.
9. H. Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Nov. 1987.
10. S. Gibson and R. J. Hubbold. Efficient hierarchical refinement and clustering for radiosity in complex environments. *Computer Graphics Forum*, 15(5):297–310, Dec. 1996.
11. C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Battaile. Modelling the Interaction of Light Between Diffuse Surfaces. *Computer Graphics (ACM SIGGRAPH '84 Proceedings)*, 18(3):212–222, July 1984.
12. S. J. Gortler, P. Schroder, M. F. Cohen, and P. Hanrahan. Wavelet Radiosity. In *Computer Graphics Proceedings, Annual Conference Series, 1993 (ACM SIGGRAPH '93 Proceedings)*, pages 221–230, 1993.
13. P. Hanrahan, D. Salzman, and L. Aupperle. A Rapid Hierarchical Radiosity Algorithm. *Computer Graphics (ACM SIGGRAPH '91 Proceedings)*, 25(4):197–206, July 1991.
14. J. M. Hasenfratz, C. Damez, F. Sillion, and G. Drettakis. A practical analysis of clustering strategies for hierarchical radiosity. *Computer Graphics Forum (Eurographics '99 Proceedings)*, 18(3):C–221–C–232, Sept. 1999.
15. C. Schwarz, J. Teich, A. Vainshtein, E. Welzl, and B. L. Evans. Minimal enclosing parallelogram with application. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages C34–C35, 1995.
16. F. Sillion. A Unified Hierarchical Algorithm for Global Illumination with Scattering Volumes and Object Clusters. *IEEE Transactions on Visualization and Computer Graphics*, 1(3), Sept. 1995.
17. B. Smits, J. Arvo, and D. Greenberg. A Clustering Algorithm for Radiosity in Complex Environments. In *Computer Graphics Proceedings, Annual Conference Series, 1994 (ACM SIGGRAPH '94 Proceedings)*, pages 435–442, 1994.
18. M. Stamminger, H. Schirmacher, P. Slusallek, and H.-P. Seidel. Getting rid of links in hierarchical radiosity. *Computer Graphics Journal (Proc. Eurographics '98)*, 17(3):C165–C174, Sept. 1998.
19. A. Willmott and P. Heckbert. An empirical comparison of progressive and wavelet radiosity. In J. Dorsey and P. Slusallek, editors, *Rendering Techniques '97 (Proceedings of the Eighth Eurographics Workshop on Rendering)*, pages 175–186, New York, NY, 1997. Springer Wien. ISBN 3-211-83001-4.
20. A. Willmott, P. Heckbert, and M. Garland. Face cluster radiosity. In *Rendering Techniques '99*, pages 293–304, New York, NY, 1999. Springer Wien.
21. C. Winkler. *Expérimentation d'algorithmes de calcul de radiosité à base d'ondelettes*. Thèse d'université, Institut National Polytechnique de Lorraine, 1998.

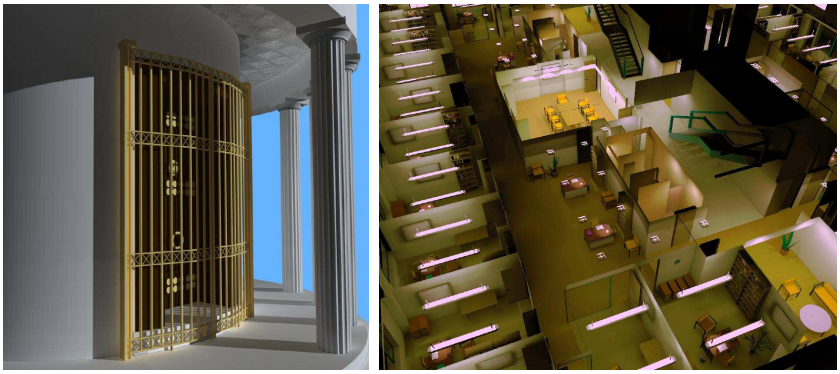


(a) Tessellated (b) Our Algorithm

Fig. 11. Using our algorithm for wavelet radiosity on arbitrary planar surfaces (see also figure 2)



(a) Opera



(b) Temple (c) Soda Hall

Fig. 12. Our test scenes

2.7.4 A novel approach makes higher order wavelets really efficient for radiosity (EG 2000)

Auteurs : François C , Laurent A et Nicolas H
Conférence : Eurographics 2000, Interlaken, Suisse. Cet article a également été publié dans *Computer Graphics Forum*, vol. 19, n° 3.
Date : septembre 2000

EUROGRAPHICS 2000 / M. Gross and F.R.A. Hopgood
(Guest Editors)

Volume 19 (2000), Number 3

A novel approach makes higher order wavelets really efficient for radiosity

François Cuny[†], Laurent Alonso[‡] and Nicolas Holzschuch[‡]

ISA research team
LORIA[§]
Campus Scientifique, BP 239
54506 Vandœuvre-les-Nancy CEDEX, France

Abstract

Since wavelets were introduced in the radiosity algorithm[5], surprisingly little research has been devoted to higher order wavelets and their use in radiosity algorithms. A previous study[13] has shown that wavelet radiosity, and especially higher order wavelet radiosity was not bringing significant improvements over hierarchical radiosity and was having a very important extra memory cost, thus prohibiting any effective computation. In this paper, we present a new implementation of wavelets in the radiosity algorithm, that is substantially different from previous implementations in several key areas (refinement oracle, link storage, resolution algorithm). We show that, with this implementation, higher order wavelets are actually bringing an improvement over standard hierarchical radiosity and lower order wavelets.

1. Introduction

Global illumination simulation is essential for realistic rendering of virtual scenes. In global illumination, we take the geometric definition of a virtual scene, we simulate the propagation of light throughout the scene, modelling its visual and physical effects, such as shadows and reflections. Global illumination simulation has applications in all the areas where a realistic rendering is interesting, such as architecture, archeology, urban planning and computer-aided design.

The radiosity method is one of the methods used in global illumination simulation. In the radiosity method, we model the exchanges of energy between the objects of the scene in order to compute the radiant energy per unit area (or *radiosity*) on all the surfaces of all the objects in the scene. The radiosity can be used directly to display the objects of the scene and the quality of the simulation is directly linked to the precision we have on the radiosity function.

The radiosity function is usually computed using finite element methods. The most efficient of these methods are hierarchical and use a multi-scale representation of the radiosity function[6] to reduce the algorithmic complexity of the computations. Hierarchical methods have been extended with wavelets[5]. The simplest wavelet base is piecewise constant (Haar wavelets), but many other wavelet bases can be used in radiosity computations.

In theory, higher order wavelets are providing a more compact representation of complex functions. Hence they use less memory and give a smoother representation of the function, that looks better on display. Higher order wavelets should be the ideal choice for radiosity computations.

In practice, the memory required to store the interactions between objects grows with the *fourth* power of the order of the wavelet base, prohibiting any real computation with complex wavelets. Furthermore, Haar wavelets allow many simplifications and optimisations that exploit their great simpleness. If these optimisations are kept with higher order wavelets, they can inhibit some of their properties. In one

[†] Institut National Polytechnique de Lorraine.
[‡] INRIA Lorraine.
[§] UMR n° 7503 LORIA, a joint research laboratory between CNRS, Institut National Polytechnique de Lorraine, INRIA, Université Henri Poincaré and Université Nancy 2.

experimental study[13] the practical problems of higher order wavelets were largely overcoming their theoretical benefits.

However, these practical problems are not inherent to higher order wavelets themselves, only to their implementation in the radiosity method. In this paper, we present a new approach to higher order wavelets, that is substantially different from previous implementations in several key areas, such as refinement oracle, link storage and resolution algorithm. Our approach has been developed by taking a complete look at higher order wavelets and at the way they should integrate with the radiosity method. With this implementation, we show that the theoretical advantages of higher order wavelets are overcoming the practical problems that have been encountered before. Higher order wavelets are now providing a better approximation of the radiosity function, with faster convergence to the solution. They also require *less* memory for storage.

Our paper is organised as follows: in section 2, we review the previous research on wavelet radiosity and higher order wavelets. Then in section 3, we present our implementation, concentrating on the areas where it is substantially different from previous implementations: the refinement oracle, not storing the interactions and the consequences it has on the resolution algorithm.

The main result that we present in this paper is the experimental study we have conducted on higher order wavelets with our implementation. Section 4 is devoted to this experimentation and its results, namely that higher order wavelets are providing a faster convergence, a solution of better quality and require less memory for their computations. Finally, section 5 presents our conclusions and future areas of research.

2. Previous work

In this section we review the basis of the wavelet radiosity algorithm (section 2.1), then we present the implementation details of previous implementations for key areas of the algorithm (section 2.2): the refinement oracle, the visibility estimation and the memory problem. This review will help for the presentation of our own implementation of these areas, in section 3.

2.1. The wavelet radiosity algorithm

In the radiosity method, we try to solve the global illumination equation, restricted to diffuse surfaces with no participating media:

$$B(x) = E(x) + \rho(x) \int_S B(y) K(x,y) dy \qquad (1)$$

Eq. 1 expresses the fact that the radiosity at a given point x in the scene, $B(x)$, is equal to the radiosity emitted by x alone, $E(x)$, plus the radiosity reflected by x, coming from all the other objects in the scene. $K(x,y)$ is the kernel of the equation, and expresses the part of radiosity emitted by point y that reaches x.

To compute the radiosity function, we use finite element methods. The function we want to compute, $B(x)$, is first projected onto a finite set of basis functions ϕ_i:

$$\tilde{B}(x) = \sum_i \alpha_i \phi_i(x) \qquad (2)$$

Our goal is to compute the best approximation of the radiosity function, given the set of basis functions ϕ_i. We must also find the optimal set of basis functions. A possibility is to use wavelets. Wavelets are mathematical functions that provide a multi-resolution analysis. They allow a multi-scale representation of the radiosity function on every object. This multi-scale representation can be used in the resolution algorithm[6, 5], allowing us to switch between different representations of the radiosity function, depending on the degree of precision required. This multi-scale resolution results in a great reduction of the complexity of the algorithm[6].

There are two broad classes of resolution algorithm: *gathering* and *shooting*. In gathering, each patch updates its own radiosity function using the energy sent by all the other patches, whereas in shooting each patch sends energy into the scene, and all the other patches update their own radiosity. In both cases, the energy is carried along *links*, that are established by the wavelet radiosity algorithm, and used to store the information related to the interaction. A key element of the wavelet radiosity algorithm is the *refinement oracle*, that tells which levels of the different multi-scale representation of radiosity should interact.

Finally, before each energy propagation, we must update the multi-scale representation of radiosity, so that each level contains a representation of all the energy that has been received by the object at all the other levels. This is done during the *push-pull* phase.

2.2. Details of previous implementations

2.2.1. Refinement oracles

The refinement oracle is one of the most important parts in hierarchical radiosity algorithms. Since it tells at which level the interaction should be established, it has a strong influence on both the quality of the radiosity solution and the time spent doing the computations. A poor refinement oracle will give poor results, or will spend a lot of time doing unnecessary computations.

In theory, the decision whether or not to refine a given interaction could only be taken with the full knowledge of the complete solution. However, the refinement oracle must take the decision using only the information that is locally available: the energy to be sent, and the geometric configuration of the sender and the receiver.

Given two patches in the scene, let us consider their interaction: patch s, with its current approximation of the radiosity function $\tilde{B}_s(y)$, is sending light toward patch r. Using a combination of eq. 1 and eq. 2, we can express the contribution of patch s to the radiosity of patch r:

$$B_{s\to r}(x) = \rho \sum_i \alpha_i \int_s \phi_i(y) K(x,y) dy \qquad (3)$$

For the interaction between the two patches we will use the relationship coefficients, C_{ij}:

$$\begin{aligned} B_{s\to r}(x) &= \sum_j \beta_j \phi_j(x) \\ \beta_j &= \int_r B_{s\to r}(x)\phi_j(x) \\ \beta_j &= \rho \sum_i \alpha_i \int_r \int_s \phi_i(y)\phi_j(x)K(x,y)dydx \\ \beta_j &= \rho \sum_i \alpha_i C_{ij} \end{aligned}$$

These C_{ij} coefficients express the relationship between the basis functions $\phi_j(x)$ and $\phi_i(y)$. Computing the C_{ij} requires the computation of a complex integral, which cannot be computed analytically and must be approximated, usually using quadratures.

In most current implementations, refinement oracle estimate the error on this approximation of the C_{ij}. This error is then multiplied by the energy of the sender, to avoid refining interactions that are not carrying significant energy. There are several ways to estimate the error on the C_{ij} coefficients: pure heuristics[6], sampling the C_{ij} at several sample points[5] and a conservative method giving an upper-bound on the propagation of the energy[10, 8].

A recurrent problem with current refinement oracles is that they concentrate on the C_{ij} coefficients. This provides a conservative analysis, but it can be too cautious, especially with higher order basis functions. The C_{ij} coefficients are usually bound with constant functions and hence so is the radiosity function. Such a binding does not take into account the capacity of higher order wavelets to model rapidly varying functions in a compact way. To take this into account, we need to move the radiosity function *inside* the refinement oracle. In section 3.1, we present a refinement oracle that addresses this problem.

2.2.2. Visibility estimations

Discontinuities of the radiosity function and its derivatives are only caused by changes in the visibility between objects[7]. Therefore, great care must be taken when adding visibility information to the radiosity algorithm.

As we have seen, we use a quadrature to compute the C_{ij} coefficients. This quadrature requires several estimates of the kernel function $K(x,y)$ and therefore of the visibility between points x and y. Computing a visibility sample is much costlier than computing a kernel sample without visibility. As a consequence, estimating the visibility between two patches is the most costly operation in wavelet radiosity[9]. Several methods have been developed in order to provide a quick estimate of visibility, sometimes at the expense of reliability.

The easiest method[6, 5] assumes a constant visibility between the patches. The constant is equal to 1 for fully visible patches, 0 for fully invisible patches, and is in $]0,1[$ for partially visible patches. It is estimated by computing several jittered visibility samples between the patches and averaging the results.

Another method computes exact visibility between the corners of the patches, and interpolates between these values for points located between the corners, using barycentric coordinates.

Shadow masks[16, 11] have also been used in wavelet radiosity computations. In theory, shadow masks allow the decoupling of visibility from radiosity transport, and therefore a better compression of the radiosity transport operator, thus reducing the memory cost.

All these methods attempt to approximate visibility by computing less visibility samples than kernel samples, in order to reduce the cost of visibility in wavelet radiosity. According to an experimental study of wavelet radiosity conducted by Willmott[13, 14] the result is a poor approximation of the radiosity function, especially near shadow boundaries.

Another method is to compute exactly one visibility sample for each kernel sample. It has been used at least by Gershbein[4], although it is not explicitely stated in his paper. According to our own experience, as well as Willmott extended study[14], this method gives better visual results. Furthermore, it gives more numerical precision. On the other hand, it can introduce some artefacts, because the visibility samples are forced to be in a regular pattern.

In our implementation, we used one visibility sample for each kernel sample, because we were looking for numerical accuracy, and because the artefacts are removed by our refinement oracle.

2.2.3. Memory usage

Since the computation of the C_{ij} coefficients can be rather long, they are usually stored once they have been computed, so that they can be reused. The storage is done on the link between s and r.

An important problem with previous wavelet radiosity implementations is the memory required for this storage. If we use wavelet bases of the order m, then we have m one dimensional functions in the wavelet base. For two dimensions, such as the surface of objects in our virtual scene, we have m^2 functions in the base. As a consequence, storing the interaction between two patches requires computing and storing m^4 C_{ij} coefficients.

Hence, the memory usage of wavelet radiosity grows with the fourth power of the wavelet base used. Wavelets of order 3 will have a memory usage almost two orders of magnitude higher than wavelets with 1 vanishing moment. In an experimental study of wavelet radiosity, Willmott[13] showed that this memory usage was effectively prohibiting any serious computation with higher order wavelets.

In 1998, Stamminger[12] showed that it was possible to eliminate completely the storage of the interactions in hierarchical radiosity. His study was only made for hierarchical radiosity, but it could be extended to wavelet radiosity, and it would remove the worst problem of radiosity with higher order wavelets. In section 3.2, we review the consequences of not storing links on the wavelet radiosity algorithm.

3. A novel approach to higher order wavelets in the radiosity algorithm

Since experimental studies conducted with previous implementations of wavelet radiosity have shown that higher order wavelets are behaving more poorly than Haar wavelets, we need to review the key points of our implementation that differ from previous implementations: the refinement oracle and getting rid of interaction storage, along with the consequences it has on the algorithm.

All the elements described in this section have been implemented and tested thanks to our radiosity testbed software, Candela[15].

3.1. The refinement oracle

Instead of estimating the errors on the propagation coefficients, we estimate the error on the propagated energy directly. Our refinement oracle is quite similar to that of Bekaert[2, 3].

To estimate the errors on the radiosity function, we use control points on the receiver. These control points are located so that they provide meaningful information: they are different from quadrature points, and their number depends on the size of the receiver. Some of the control points are located on the boundary of the receiver, in order to ensure continuity with neighbouring patches.

Our refinement oracle is summed up in fig. 1. The radiosity values at the control points $B_{s\to P_i}$ are computed by direct integration of eq. 3 at point $x = P_i$, using a quadrature. To take the norm of the errors at the control points, we can use any norm, such as the L_1 norm, the L_2 norm, the L_∞ norm. We have found that all these norms are giving similar results for refinement.

3.2. Not using links and the consequences

In order to reduce the memory footprint of the radiosity algorithm, we have chosen not to store links, as in Stamminger[12].

for each interaction $s \to r$:
  compute the radiosity function on the receiver: $B_{s\to r}(x)$
  for each control point P_i
    compute the radiosity at this control point directly: $B_{s\to P_i}$
    compare with interpolated value,
    store the difference:
    $\delta_i = |B_{s\to r}(P_i) - B_{s\to P_i}|$
  end for
  take the L_n norm of the differences:
  $\delta_B = \|\delta_i\|_n$
  compare with refinement threshold
end for

Figure 1: *Our refinement oracle*

Not storing links is some kind of a trade-off between memory and time: by not storing links, we are saving memory. However, the information that has not been stored will probably have to be recomputed at some stage in the algorithm, which will cost time.

Not storing links also has consequences on the structure of the algorithm itself. The main consequence is on the choice between gathering and shooting.

Gathering sends energy from all the patches to all the other patches at each iteration. All the links are used during a given iteration.

Shooting sends the unshot energy from one patch to all the other patches. At a given point in time, only the links from the shooting patch to all the patches are being used. The shooting patch is then send to the bottom of the shooting queue, and the links will not be re-used until it gets back to the top of the shooting queue.

Therefore, if we chose not to store links, it makes more sense to use shooting than to use gathering. But the reverse is also true: if you use shooting instead of gathering, it also makes more sense not to store links.

in gathering, the optimal link distribution for one iteration can be computed by refining the link distribution from the previous distribution, because the radiosity gathered at one point can only grow with subsequent iterations.

in shooting, the energy carried along the links is only the unshot energy at the shooting patch. Its distribution changes completely for each use of the patch. As a consequence, the optimal link distribution has no relation with the links computed for previous iterations.

4. Comparison of several wavelet bases

In this section, we present our experimental comparison of different wavelet bases. We start with a description of the experimentation protocol in section 4.1. We then present the results of our experiments in section 4.2. Discussion of these results and comparison with previous studies follows in section 4.3.

4.1. The experimentation protocol

4.1.1. The wavelet bases

We wanted to use our implementation of wavelet radiosity for a comparison of several wavelet bases. We have used the first three multi-wavelets bases : $\mathcal{M}_1$ (Haar), $\mathcal{M}_2$ and $\mathcal{M}_3$.

We use the $\mathcal{M}_n$ multi-wavelets as they were previously defined [1, 5]: the smoothing functions for $\mathcal{M}_n$ are defined by tensorial products of the first n Legendre polynomials.

We have not used flatlets bases ($\mathcal{F}_n$), because although they have n vanishing moments, they are only piecewise constant, and therefore do not provide a better approximation than Haar wavelets with further refinement.

4.1.2. The test scenes

Our tests have been conducted on several test scenes, ranging from simple scenes, such as the blocker (see fig. 2(a)) to moderately complex scenes, such as the class room (see fig. 2(d)). All our test scenes are depicted on fig. 2, with their number of input polygons.

4.1.3. Displaying the results

All the figures in this paper are depicting the exact results of the computations, without any post-processing of any kind: the radiosity function is displayed exactly as it has been computed. Specifically, there has been no attempt to ensure continuity of the radiosity function, except in the refinement oracle. Similarly, we haven't balanced or anchored the computed mesh. So, for example, in fig. 4(c), the continuity of the radiosity function is due only to the refinement oracle depicted in section 3.1.

$\mathcal{M}_3$ wavelets can result in quadrically varying functions, which can not be displayed on our graphics engines. To display these functions, we subdivide each patch into four subpatches, on which we compute four linearly varying functions approximating the quadrically varying radiosity function.

4.1.4. Computing the error

In order to compute the computational error, we have computed a reference solution, using $\mathcal{M}_2$ wavelets, with a very small refinement threshold. Furthermore, the minimal patch area in the reference solution was 16 times smaller than the minimal patch area in the computed solutions. We also checked that with all the wavelet bases, the computed solutions did converge to the reference solution.

We have measured the energetic difference between this reference solution and the computed solutions. In order to have comparable results on all our test scenes, this difference is divided by the total energy of the scene. It is this ratio of the energetic difference over the total energy that we call *global error*. Thus, a global error of 10^{-1} means there is an energetic difference of 10 % between the energetic distributions of the computed solution and the reference solution.

According to our experiments, this measure of global error is consistent, and gives comparable visual results on all the test scenes. For example, a global error of 10^{-1} will always give a poor result (see fig. 3(a)), a global error of 10^{-2} will give a better result, but still with visible artefacts at shadow boundaries (see fig. 3(b)), and a global error of 10^{-3} will always give a correct result (see fig. 3(c)). In our experience, (see fig. 3) the global error must be lower than 5.10^{-3} in order to get visually acceptable results.

As it has been pointed out[12], we have also found that this global error is closely correlated to the refinement threshold on each interaction (the *local error*).

4.1.5. Experimentation details

In all our experiments, we have used the same computer, a SGI Octane working at 225 MHz, with 256 Mb of RAM.

4.2. Results

4.2.1. Visual comparison of our three wavelet bases

The first test to conduct is whether higher order wavelets are giving a better visual impression. In previous tests[13], higher order wavelets were unable to provide a correct approximation of the radiosity function, especially near shadow boundaries. Shadow boundaries are very important because they have a large impact on the visual perception of the scene.

Our first experiment focuses solely on this problem. We have computed direct illumination from an area light source to a planar receiver, with an occluder partially blocking the exchange of light. All wavelet bases were used with the same computation time (66 s).

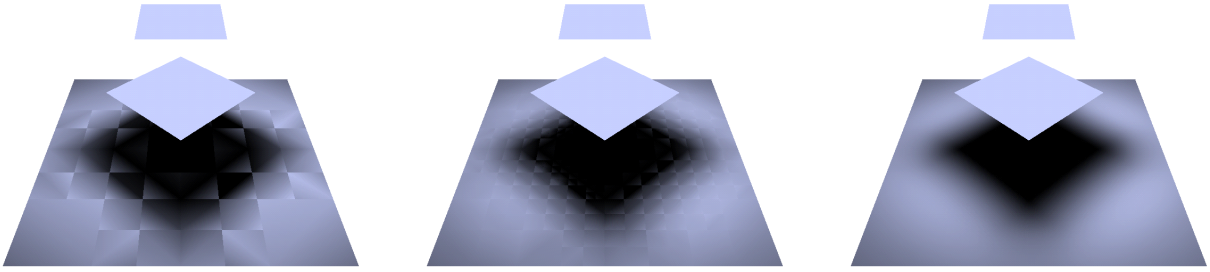
Fig. 4 shows the radiosity function computed for each wavelet base, along with the mesh used for the computation. Two elements appear clearly: higher order wavelets are providing a much more compact representation of the radiosity function, even near shadow boundaries, and the radiosity function computed with $\mathcal{M}_2$ and $\mathcal{M}_3$ wavelets is smoother than the function computed with Haar wavelets.

Haar wavelets are usually not displayed as such, but using some sort of post-processing, such as Gouraud shading. Fig. 5 shows the result of applying Gouraud shading to fig. 4(a). As you can see, although it can hide some of the discontinuities, Gouraud shading can also introduce some new artefacts.

Judging from fig. 4, higher order wavelets are better for radiosity computations than lower order wavelets. This is only a qualitative results and must be confirmed by quantitative studies; that is the object of the coming sections (4.2.2 and 4.2.3).

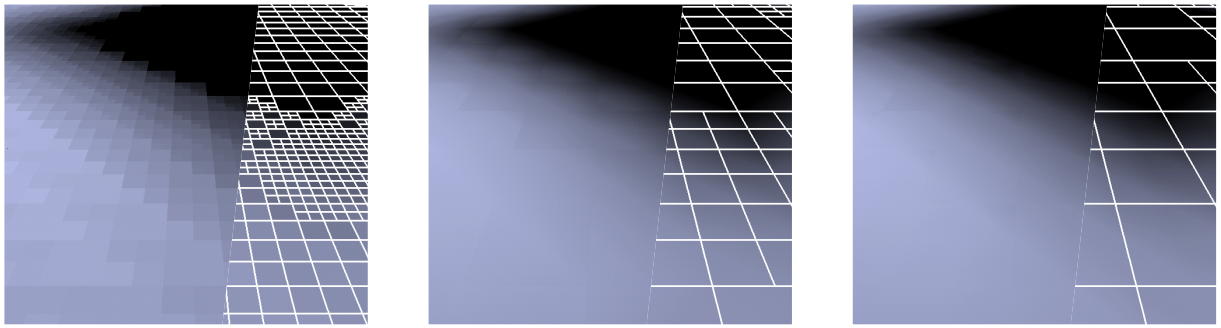
(a) Blocker (3) (b) Tube (5) (c) Dining room (402) (d) Classroom (3153)

Figure 2: *Our test scenes, with their number of input polygons*



(a) 10^{-1} (b) 10^{-2} (c) 10^{-3}

Figure 3: *Visual comparison of results for different values of global error*



(a) Haar (b) $\mathcal{M}_2$ (c) $\mathcal{M}_3$

Figure 4: *Visual comparison of results for our three wavelet bases*

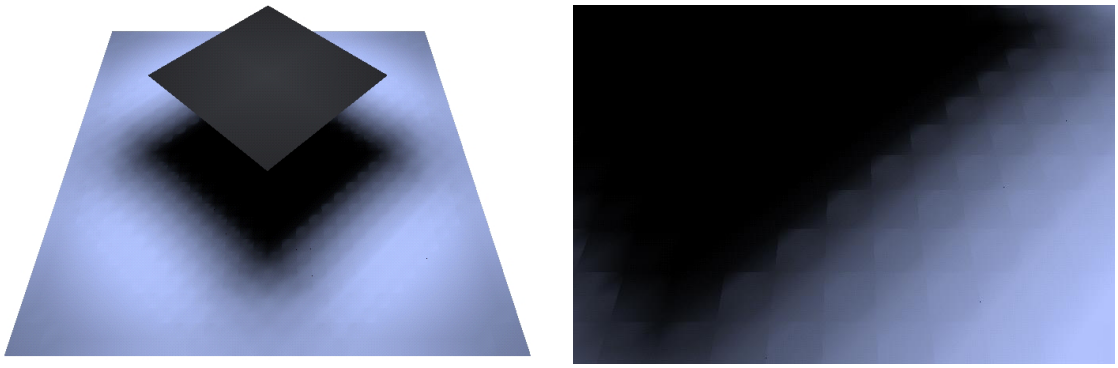
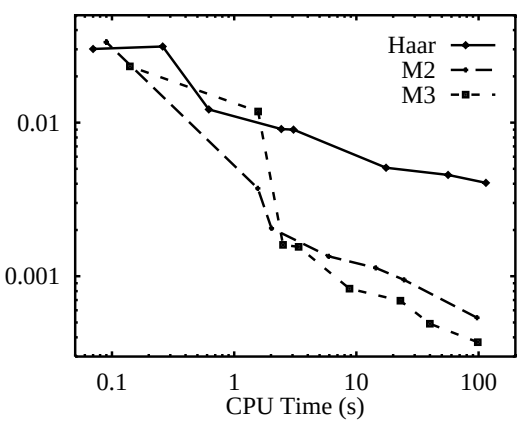
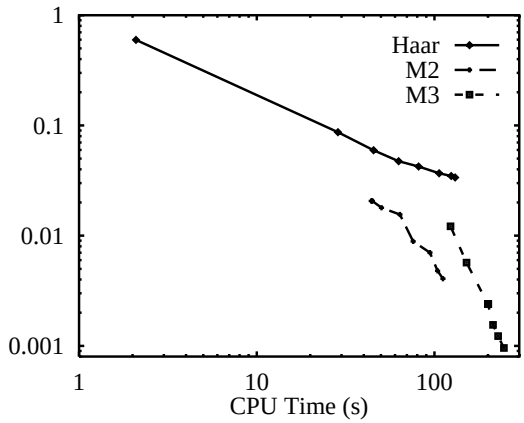


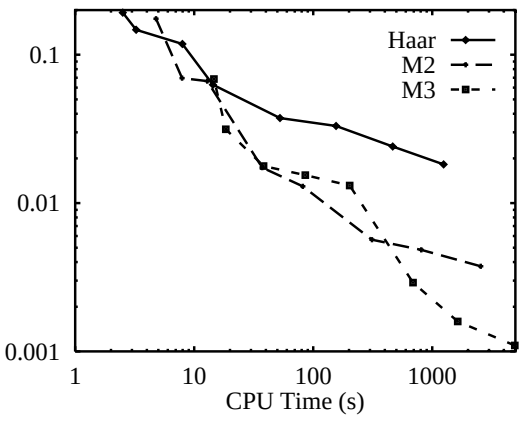
Figure 5: *Applying Gouraud shading to Haar wavelets*



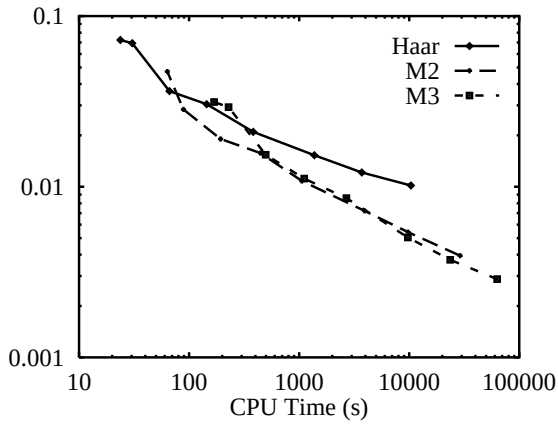
(a) Blocker



(b) Tube



(c) Dining room



(d) Classroom

Figure 6: *Global error with respect to computation time (in s)*

4.2.2. Computation time

Fig. 6 shows the relationship between global error and computation time for our four test scenes and our three wavelet bases.

The most important point that can be extracted from these experimental data is that with our implementation, higher order wavelets are performing *better* than lower order wavelets. They obtain results of higher quality, and they are faster: to get a visually acceptable result on the classroom scene (global error below 5.10^{-3}), $\mathcal{M}_3$ wavelets use 10^4 s (see fig. 6(d)). In the same computation time, Haar wavelets only reach a global error level of 10^{-2}. This test scene is our hardest test scene, with lots of shadow boundaries. It is on such test scenes that higher order wavelets were behaving poorly with previous experimentations[13].

The advantage of higher order wavelets is more significant on high precision computations and on complex scenes. The more precision you need on your computations, the faster they are, compared to lower order wavelets.

On the contrary, for quick approximations, $\mathcal{M}_2$ wavelets are performing better than $\mathcal{M}_3$ wavelets. The same applies to Haar wavelets compared to $\mathcal{M}_2$ wavelets, for very quick and crude approximations.

Each wavelet base has an *area of competence*, where it outperforms all the other wavelet bases: Haar wavelets are the most efficient base for global error above 10^{-1} — which corresponds to a simulation with many artefacts still visible (see fig. 3(a)). $\mathcal{M}_2$ wavelets are better than all the other bases for global error between 10^{-1} and (roughly) 5.10^{-3}, and $\mathcal{M}_3$ wavelets are the best for global error below 5.10^{-3}.

4.2.3. Memory use

The key problem with higher order wavelets in previous studies[13] was their high memory use, that effectively prohibited any real computation. We have computed the memory footprint of our implementation of wavelets for our four test scenes and our three wavelet bases. Fig. 7 shows the memory used by the algorithm as a function of the global error.

As you can see, for high precision computations (global error below 5.10^{-3}), higher order wavelets actually have a *lower* memory use than low order wavelets. The effect is even more obvious on our more complex scenes (see fig. 7(c) and 7(d)).

On the other hand, for low precision computations, this hierarchy is reversed, and Haar and $\mathcal{M}_2$ wavelets have a lower memory use. Once again, each wavelet base has an area of competence, where it outperforms all the other wavelet bases. For very crude approximations, Haar wavelets are the most efficient with respect to memory use, then, for moderately good approximations, $\mathcal{M}_2$ wavelets are the most efficient, until $\mathcal{M}_3$ takes over for really good approximations.

A very impressive result is the way the memory cost of a given wavelet base degrades quickly if we try to bring the global error level below a certain threshold. This effect appears very clearly on fig. 7(c) and 7(d). There seems to be a maximum degree of precision for each wavelet base, and the wavelet base can only conduct global illumination simulations below this degree. Be aware, however, that the degradation is made more impressive on fig. 7 by the fact that we are using a logarithmic scale for global error and a non-logarithmic scale for memory use. Furthermore, the degradation is quite small when it is compared to the total memory used: between 10 % and 20 %. Since the effect appears in a similar way for all the wavelet bases used in the test we think it could be a general effect, and apply to all wavelet bases.

Please note that the fact that higher order wavelets have a lower memory use than lower order wavelets is actually quite logical. Higher order wavelets are providing a more powerful tool for approximating complex functions, with a higher dimensional space for the approximation. Furthermore, they have more vanishing moments, so their representation of a given complex function is more compact and requires less coefficients. Our experiments are therefore bringing practical results in connection with theoretical expectations.

The fact that lower order wavelets are more compact for low precision computations was also to be expected from theory. Low precision computations are, by nature, not taking into account all the complexity of the radiosity function. As a consequence, they provide a very simple function, that is also easy to approximate, especially for simple wavelet bases.

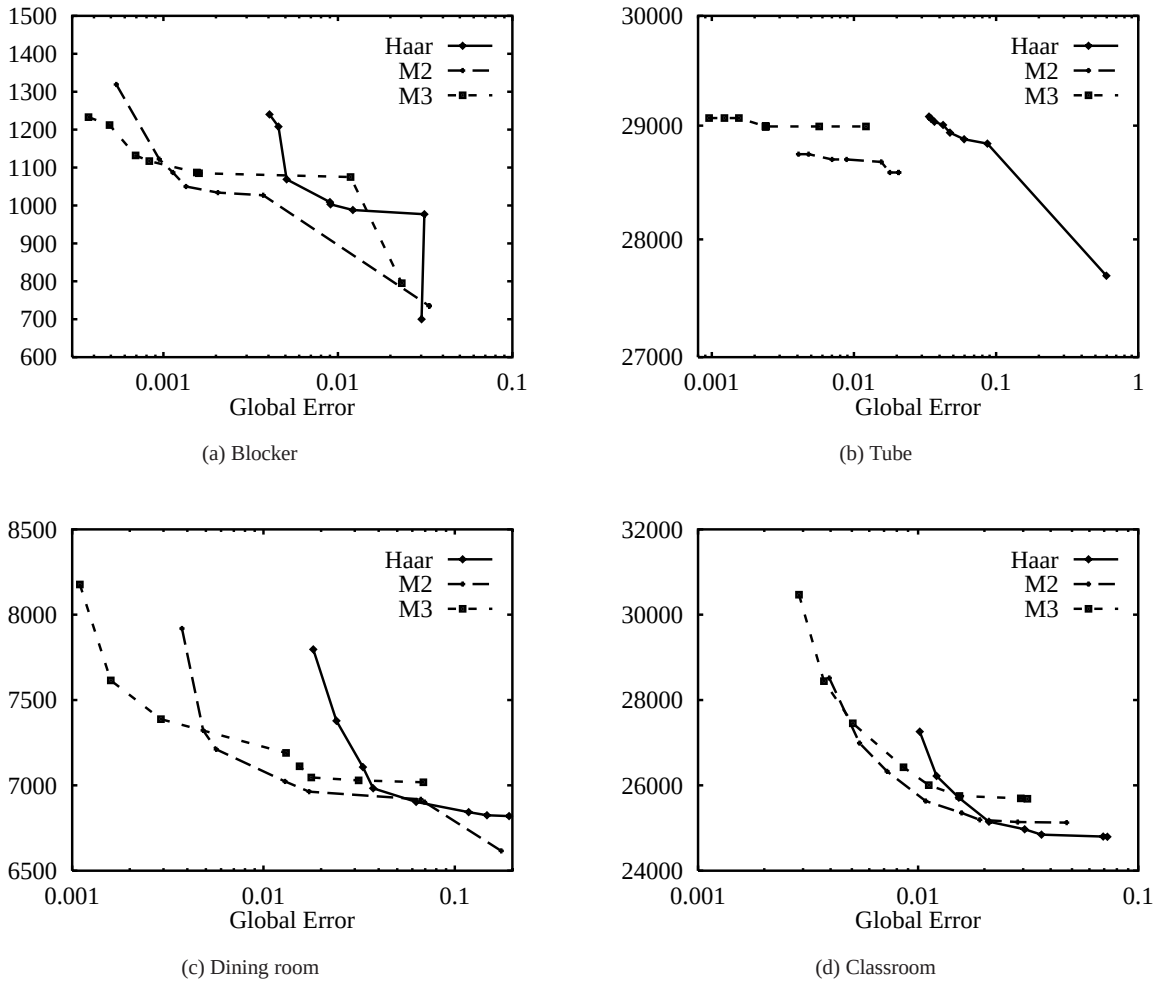
4.3. Discussion and comparison with previous studies

Despite the fact that we are reaching opposite conclusions, we would like to point out that our study is actually consistent with the previous study by Willmott[13, 14].

In Willmott's study, higher order wavelets were carrying a strong memory cost, due to link storage. As a consequence, radiosity computations with higher order wavelets were restricted to low precision computations. According to our experiments, for low precision computations, lower order wavelets are indeed providing a faster approximation, with a lower memory use.

Our study can therefore be seen as an extension of Willmott's study to high precision computations. Such high precision computations were made possible only by getting rid of links[12]. Once you have eliminated link storage, the memory cost of the radiosity algorithm is almost reduced to the cost of mesh storage. The refinement oracle (see section 3.1) ensures that the mesh produced is close to optimal with respect to the radiosity on the surfaces.

Also, by concentrating the oracle on the mesh instead of the interactions, we are able to exploit the power of wavelet bases functions to efficiently approximate functions. This results in a coarser mesh, both at places where the radiosity



(a) Blocker

(b) Tube

(c) Dining room

(d) Classroom

Figure 7: *Memory requirements (in kB) with respect to global error*

function has slow variations, such as an evenly lit wall, and at place with rapid variations, such as shadow boundaries.

5. Conclusion and future work

We have presented an implementation of wavelets bases in the radiosity algorithm. With this implementation, we have conducted experimentations on several wavelet bases. Our experiments show that for high precision computations, higher order wavelets are providing a better approximation of the radiosity function, faster, and with a lower cost in memory. Please note that our implementation is not putting any disadvantage on lower order wavelets; for Haar wavelets, our refinement oracle only uses a few tests and the visibility estimation only requires one visibility test. Similarly, the benefit of not storing links is independant of the wavelet base.

Although in this paper we have only conducted tests on relatively small test scenes (up to 3000 input polygons), our implementation (Candela[15]) enables us to use higher order wavelets on arbitrarily large scenes. Fig. 8* shows a radiosity computation with $\mathcal{M}_2$ wavelets made with our implementation on a scene with 144255 input polygons. The computations took 3 hours, and required approximately 2 Gb of memory on 32 processors of a SGI Origin 2000. The complete solution had approximately 1.5 million patches.

The optimal choice for radiosity computations depends on the degree of precision required. Lower order wavelets are better for low precision computations, and higher order wavelets are better for high precision computations. Each wavelet base corresponds to a certain degree of precision, where it outperforms all the other wavelet bases, both for the computation time and the memory footprint. Although

our computations have been limited to Haar, $\mathcal{M}_2$ and $\mathcal{M}_3$ wavelets, we think that this effect applies to all the other wavelet bases, such as $\mathcal{M}_4$, $\mathcal{M}_5$... and that for even more precise computations, $\mathcal{M}_4$ would outperform $\mathcal{M}_3$, and so on.

However, for moderately precise computations, $\mathcal{M}_2$ wavelets are quite sufficient. The precision level that corresponds, in our experience, to visually acceptable results is at the boundary between the areas of competence of $\mathcal{M}_2$ and $\mathcal{M}_3$, so $\mathcal{M}_2$ wavelets can be used. $\mathcal{M}_2$ wavelets also have a distinct advantage over all the other wavelet bases: they result in linearly varying functions that can be displayed directly on current graphics hardware (using Gouraud shading), as opposed to constant, quadric or cubic functions.

In our future work, we want to explore the possibility to use several different wavelet bases in the resolution process. In this approach, it would be possible to use Haar wavelets for interactions that do not require a lot of precision, such as interactions that do not carry a lot of energy, and $\mathcal{M}_2$, and perhaps $\mathcal{M}_3$, $\mathcal{M}_4$..., wavelets for interactions that require a high precision representation. We think that this approach could be especially interesting with shooting since the first interactions will carry a lot of energy, while later interactions will only carry a small quantity of energy.

We also want to explore the possibility to use higher order wavelets on non-planar objects. Since they have a better ability to model rapidly varying radiosity functions, they seem to be the ideal choice for curved surfaces, such as spheres or cylinders.

6. Acknowledgements

The authors would like to give a very special thank to Jean-Claude Paul. It was his insight that started this work on higher order wavelets, and it was his advice and support that ensured its success.

References

1. B. Alpert, G. Beylkin, R. Coifman, and V. Rokhlin. Wavelet-like bases for the fast solution of second-kind integral equations. *SIAM Journal on Scientific Computing*, 14(1):159–184, January 1993. 5

2. Philippe Bekaert and Yves Willems. Error Control for Radiosity. In *Rendering Techniques '96 (Proceedings of the Seventh Eurographics Workshop on Rendering)*, pages 153–164, New York, NY, 1996. Springer-Verlag/Wien. 4

3. Philippe Bekaert and Yves D. Willems. Hirad: A Hierarchical Higher Order Radiosity Implementation. In *Proceedings of the Twelfth Spring Conference on Computer Graphics (SCCG '96)*, Bratislava, Slovakia, June 1996. Comenius University Press. 4

4. Reid Gershbein. Integration Methods for Galerkin Radiosity Couplings. In P. M. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering) in Dublin, Ireland, June 12-14, 1995)*, pages 264–273, New York, NY, 1995. Springer-Verlag. 3

5. Steven J. Gortler, Peter Schroder, Michael F. Cohen, and Pat Hanrahan. Wavelet Radiosity. In *Computer Graphics Proceedings, Annual Conference Series, 1993 (ACM SIGGRAPH '93 Proceedings)*, pages 221–230, 1993. 1, 2, 3, 5

6. Pat Hanrahan, David Salzman, and Larry Aupperle. A Rapid Hierarchical Radiosity Algorithm. In *Computer Graphics (ACM SIGGRAPH '91 Proceedings)*, volume 25, pages 197–206, July 1991. 1, 2, 3

7. Paul Heckbert. Discontinuity Meshing for Radiosity. In *Third Eurographics Workshop on Rendering*, pages 203–226, Bristol, UK, May 1992. 3

8. Nicholas Holzschuch and Francois. X. Sillion. An exhaustive error-bounding algorithm for hierarchical radiosity. *Computer Graphics Forum*, 17(4):197–218, December 1998. 3

9. Nicolas Holzschuch, Francois Sillion, and George Drettakis. An Efficient Progressive Refinement Strategy for Hierarchical Radiosity. In *Fifth Eurographics Workshop on Rendering*, pages 343–357, Darmstadt, Germany, June 1994. 3

10. Dani Lischinski, Brian Smits, and Donald P. Greenberg. Bounds and Error Estimates for Radiosity. In *Computer Graphics Proceedings, Annual Conference Series, 1994 (ACM SIGGRAPH '94 Proceedings)*, pages 67–74, 1994. 3

11. Philipp Slusallek, Michael Schroder, Marc Stamminger, and Hans-Peter Seidel. Smart Links and Efficient Reconstruction for Wavelet Radiosity. In P. M. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, pages 240–251, New York, NY, 1995. Springer-Verlag. 3

12. M. Stamminger, H. Schirmacher, P. Slusallek, and H.-P. Seidel. Getting rid of links in hierarchical radiosity. *Computer Graphics Journal (Proc. Eurographics '98)*, 17(3):C165–C174, September 1998. 4, 5, 8

13. Andrew Willmott and Paul Heckbert. An empirical comparison of progressive and wavelet radiosity. In Julie Dorsey and Phillip Slusallek, editors, *Rendering Techniques '97 (Proceedings of the Eighth Eurographics Workshop on Rendering)*, pages 175–186, New York, NY, 1997. Springer Wien. ISBN 3-211-83001-4. 1, 2, 3, 4, 5, 8

14. Andrew J. Willmott and Paul S. Heckbert. An empirical comparison of radiosity algorithms. Technical Report CMU-CS-97-115, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, April 1997. Available from http://www.cs.cmu.edu/ radiosity/emprad-tr.html. 3, 8

15. Christophe Winkler. *Expérimentation d'algorithmes de calcul de radiosité à base d'ondelettes*. Thèse d'université, Institut National Polytechnique de Lorraine, 1998. 4, 9

16. Harold R. Zatz. Galerkin Radiosity: A Higher Order Solution Method for Global Illumination. In *Computer Graphics Proceedings, Annual Conference Series, 1993 (ACM SIGGRAPH '93 Proceedings)*, pages 213–220, 1993. 3

Figure 8: *Radiosity computation on a large scene (with $\mathcal{M}_2$ wavelets)*

2.7.5 Combining higher-order wavelets and discontinuity meshing: a compact representation for radiosity (EGSR 2004)

Auteurs : Nicolas H et Laurent A
Conférence : *Eurographics Symposium on Rendering*, Linköping, Suède.
Date : juin 2004

Eurographics Symposium on Rendering (2004)
H. W. Jensen, A. Keller (Editors)

Combining Higher-Order Wavelets and Discontinuity Meshing: a Compact Representation for Radiosity

N. Holzschuch[1] and L. Alonso[2]

[1] ARTIS[†] GRAVIR-IMAG - INRIA [2] ISA/LORIA[‡] - INRIA

Abstract

The radiosity method is used for global illumination simulation in diffuse scenes, or as an intermediate step in other methods. Radiosity computations using Higher-Order wavelets achieve a compact representation of the illumination on many parts of the scene, but are more expensive near discontinuities, such as shadow boundaries. Other methods use a mesh, based on the set of discontinuities of the illumination function. The complexity of this set of discontinuities has so far proven prohibitive for large scenes, mostly because of the difficulty to robustly manage a geometrically complex set of triangles. In this paper, we present a method for computing radiosity that uses higher-order wavelet functions as a basis, and introduces discontinuities only when they simplify the resulting mesh. The result is displayed directly, without post-processing.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Three-Dimensional Graphics and Realism]: I.3.5 [Computational Geometry and Object Modeling]:

1. Introduction

The radiosity method is a finite element method used for simulating light exchanges between diffuse surfaces. As such, it is used either for computing global illumination in diffuse scenes or as an intermediate step in other global illumination methods. Although other rendering methods, such as Bi-Directional Path Tracing or Photon Mapping are highly popular because they account for light exchanges between specular surfaces, many people still use radiosity because it offers the possibility to move the viewpoint in real-time after illumination computations.

However, radiosity methods are difficult to manage. The quality of the output is not always visually correct, and the memory cost of the algorithm can be quite high, since it needs to store a complete representation of the illumination on all objects in the scene. Hierarchical methods are used nowadays to reduce storage costs and computation time, and among them wavelet methods have proven interesting. Using higher-order wavelets as the basis functions, it is possible to approximate smoothly varying illumination with a small number of patches, reducing the memory cost.

A constant problem with basic radiosity methods is their misbehaviour near shadow boundaries. As most of these methods use an axis-aligned hierarchical grid for their finite-element computations, they are missing discontinuities that are not aligned with the grid. Solving this problem requires either using a finer grid size near shadow boundaries or using finite elements aligned with the shadow boundaries, called *discontinuity meshing*. The former method increases the memory cost, while the latter provides good quality reconstruction of illumination but the discontinuities are complex, and managing them in a robust and efficient way is still a research problem.

Since most higher-order wavelets are defined as tensor-products of 1D basis functions, they are only properly defined over parallelogram patches. As a consequence, they are in theory incompatible with discontinuity meshing, which produces complex polygons.

In this paper, we present an algorithm that combines higher-order wavelets with discontinuity meshing. We use

[†] ARTIS is a research project in the GRAVIR/IMAG laboratory, a joint unit of CNRS, INPG, INRIA and UJF

[‡] LORIA is a joint unit of CNRS, INPL, INRIA, UHP and Université Nancy 2.

wavelets, defined on a regular subdivision in places where they provide a good approximation, and we introduce discontinuities only in places where they reduce the complexity of the mesh. This selection of effective discontinuities is done during the refinement process, by the refinement oracle. The mesh produced is still a regular grid, but some of its patches are cut by discontinuities.

We use a fragment program to display quadric wavelets directly. We are displaying the results of our illumination computations immediately, without post-processing or final gather. We are exploiting the fact that higher-order wavelets with the proper refinement oracle result in apparently continuous functions after reconstruction, even in the absence of a specific step to enforce this continuity.

This paper is organised as follows: in the following section, we will review previous work on hierarchical – or wavelet – radiosity and discontinuity meshing, as well as integrating them. Then, in section 3 we will present our algorithm, and in section 4 we will present results and pictures from our experimentations. Finally, we will conclude and expose future research directions.

2. Previous Work

The radiosity method was first introduced for global illumination simulations by Goral *et al.* in 1984 [GTGB84]. It uses a finite element formulation of the rendering equation [KH84] for diffuse scenes, and gets a complete representation of global illumination. The radiosity method was later extended using a hierarchical formulation of the finite element method [HS92, HSA91]. The hierarchical representation limits the complexity of the radiosity algorithm to $O(n)$ instead of $O(n^2)$. This hierarchical formulation was later extended using a wavelet framework [GSCH93, SGCH93].

It is possible to use wavelets of different order (piecewise-constant basis, piecewise-linear basis, piecewise-polynomial basis). Early implementations of higher-order wavelets proved inefficient [WH97], until a complete analysis of the wavelet radiosity algorithm [CAH00] showed that with the right implementation, a good refinement oracle [BW96] and efficient memory management [SSSS98] they were actually more interesting than hierarchical piecewise-constant basis functions for global illumination simulations, with smaller memory costs and shorter computation times.

Piecewise polynomial wavelets are more costly for each patch of the finite element formulation, requiring $(k+1)^2$ coefficients for a wavelet basis made of polynomials of degree k. But they provide a better approximation of the illumination function, resulting in a smaller number of patches. The study by Cuny *et al.* [CAH00] showed that most of the time, the reduction in the number of elements more than compensates for the extra cost for each element, allowing a faster computation of radiosity and a smaller memory cost.

However, many scenes on which we wish to compute global illumination exhibit sharp discontinuities of the illumination functions, for example shadows caused by point light sources or small area light sources, or shadows caused by occluders that are close to the receiver. Regular hierarchical basis of continuous polynomials are unable to model such discontinuities. In the presence of these discontinuities, most radiosity algorithms refine the hierarchy a lot, using very small patches to approximate the radiosity function. The result is that the number of patches used near the discontinuity is roughly independent from the order of the basis function. Since each patch stores $(k+1)^2$ coefficients, wavelet bases of higher-order polynomials end up being more costly at these discontinuities.

Discontinuities of the radiosity function can be computed using geometrical methods [LTG92, Hec92] [DF94, SG94, GS96, DDP02]. An adaptive mesh based on these discontinuities provides a better approximation of the radiosity function [LTG92, Hec92]. Radiosity methods based on the discontinuity mesh have been proposed, either with classical radiosity [LTG92, Hec92, Stu94, DF94] or with hierarchical radiosity [LTG93, DS96, DDP99]. All these methods start with the complete set of discontinuities, triangulating it and refining it as necessary. The entire set of discontinuites is quite large, giving a very complex mesh as a starting point. Managing this mesh proves complicated, and the associated memory cost is not neglictible. [DS96] used a regular mesh for visible areas, but kept the triangulated set of discontinuities for penumbra regions.

Several of the discontinuities in the discontinuity mesh are not visible in the radiosity function. Simplifications of the discontinuity mesh have been suggested [DF94, HWP97, Hed98]. But as they are computing discontinuites before the illumination computations, this selection uses only geometrical tools and has not access to illumination information.

In our algorithm, however, we use a regular subdivision as often as possible, and we only introduce discontinuities if they result in a simpler mesh. Significative discontinuities are thus naturally selected during the hierarchical refinement process.

A single paper has used Discontinuity Meshing together with wavelet radiosity with higher order-basis functions [PB95]. Their study is quite complete, but they used only very simple scenes for their tests: a single patch with a single discontinuity. As a consequence, they could not identify several problems that only occur in larger scenes, such as intersecting discontinuities or the cost of computing push-pull coefficients: their method would not scale to scenes much bigger. They tried to merge wavelets with discontinuities by finding a wavelet-compatible parametrization of the patch that followed the discontinuity. This causes a complex computation of push-pull coefficients for each hierarchical level. Also, building such a parametrization is not al-

ways possible, in the case of intersecting discontinuities. Finally, their approach does not address the problem of managing the set of discontinuities. Our algorithm, by contrast, keeps the same parametrization for all patches in the hierarchy, making it easy to compute push-pull coefficients. We can deal with multiple discontinuities and intersecting discontinuities. Each discontinuity inserted in the hierarchy is treated only at its hierarchical level.

3. Algorithm

In this section, we present our algorithm for merging radiosity using higher-order wavelets with meshing discontinuities. We start with a short summary of the Hierarchical Radiosity algorithm, and how it has been adapted to higher-order wavelets (section 3.1). Then we present our algorithm for merging the wavelet bases with discontinuities (section 3.2). Some finer points of the implementation are explained in section 3.3.

3.1. Wavelet Radiosity

3.1.1. The Hierarchical Radiosity Algorithm

In Wavelet Radiosity, each surface of the scene carries a hierarchical representation of illumination, using the wavelet basis. This representation is computed iteratively, through three essential steps:

- refinement of interactions,
- propagation of energy,
- push-pull.

At the beginning of the algorithm, we select the surface with the largest unshot energy, and indentify all surfaces that are potentially visible from it. We establish interactions between the shooting surface and all these receiving surfaces.

We then *refine* these interactions, in a hierarchical manner. At each point in time, we consider the current multi-scale representation of the interaction, and check whether it is accurate enough, according to the *refinement oracle*. If not, we refine the interaction, by subdividing either the shooting surface or the receiver.

Once we are satisfied with the level of precision on the interaction, we *propagate* the energy by sending the unshot energy of the shooting surface to the receivers, updating the wavelet coefficients on the receivers.

After these steps, the unshot energy of the shooting surface is set to zero, and we pick the surface with the largest unshot energy as the next shooting surface.

After the propagation, the different levels of the hierarchy on each surface have received energy, but there isn't a consistent representation of the energy received at all hierarchical levels. This representation must be reconstructed before we can use the hierarchical representation for shooting or for display. It is done during the *push-pull* step. The *push-pull* step is a recursive procedure, where parent nodes add their energy to their children, and the children's energy is collected in each parent and averaged.

3.1.2. Using Higher-Order Wavelets

Using higher-order wavelets, such as Multi-Wavelets ($\mathcal{M}_2$ and $\mathcal{M}_3$) [Alp93], does not change the algorithm, except in these details:

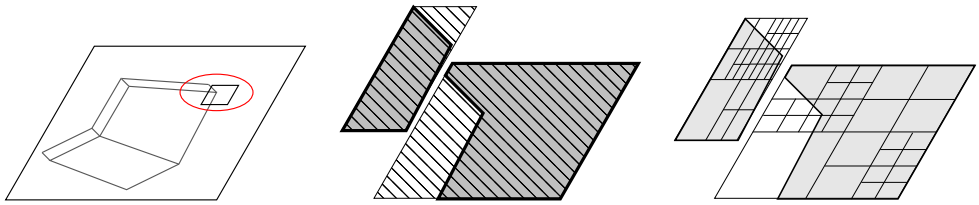
- each patch carries a wavelet representation of the radiosity function. The $\mathcal{M}_n$ basis is made of polynomials of degree $n-1$, so each patch has n^2 basis functions and stores n^2 coefficients.
- The interaction between two patches implies computing the influence that each wavelet coefficient on the shooting patch has on every wavelet coefficient on the receiving patch. Each of these influence coefficients is expressed as an integral, which is approximated using quadratures. As there are n^2 coefficients on each patch, we must evaluate n^4 integrals.
- The push-pull step implies computing the influence that each wavelet coefficient on the parent patch has on every wavelet coefficient on the children patches, and reciprocally. These influences are also expressed as integrals. These integrals only depend on the respective geometry of the parents and children patches in the hierarchy. For a regular subdivision, the push-pull coefficients are therefore constant on the hierarchy, and are pre-computed. For irregular subdivision, the push-pull coefficients must be recomputed at each level, a potentially costly step.
- As 2D wavelets are usually defined as tensor-products of 1D wavelets, they are only defined over a parallelogram. Researchs have shown how to extend this definition for complex planar surfaces [HCA00] and for parametric curved surfaces [ACP*01].
- Given the large number of coefficients for each interaction (n^4, as much as 81 coefficients for polynomials of degree 2), it is important to avoid storing them. Once we have treated an interaction, we delete all its coefficients. This strategy can result in computing the same interaction coefficients twice, but the gain in memory largely offsets the potential loss in time [SSSS98, CAH00].

3.2. Combining Wavelets and Discontinuity Meshing

3.2.1. The algorithm

Our algorithm works as follows:

- For each shooting surface, for each receiving surface, we compute the set of discontinuities on the receiving surface.
- We proceed with the usual refinement of interaction, using the oracle and a regular subdivision.
- When the refinement oracle identifies that the interaction should be subdivided only because of a discontinuity, it



(a) (b) (c)

Figure 1: *A patch cut by a discontinuity (a) results in two children patches. For each children patch, we identify the enclosing parallelogram (b). We conduct standard wavelet radiosity on each parallelogram (c).*

introduces a *discontinuity-based subdivision* instead of a regular subdivision.

- Discontinuity-based subdivision works by:
  - Computing the intersection of the current patch with the discontinuity.
  - For each part of the subdivided patch, identify the smallest parallelogram that encloses it.
  - Apply our radiosity algorithm using a regular subdivision over each parallelogram (see Figure 1).
- Once we are satisfied with the level of refinement for this interaction, we propagate the energy, then erase the discontinuities and the interaction coefficients. Discontinuities that have not been used for subdivision are forgotten.

We want to use the regular subdivision as much as possible for its robustness and simplicity. Our algorithm only introduces discontinuities if they are considered important by the refinement oracle. Smooth transitions that can be properly approximated by the wavelet basis will not be introduced in the hierarchy.

In the following paragraphs, we review each step of this algorithm in detail: refinement oracle, discontinuity-based subdivisions, push-pull over a discontinuity, intersection of discontinuities.

3.2.2. Refinement oracle and selection of discontinuities

We use the refinement oracle described in previous publications [BW96, CAH00]: for each patch, we select testing points, where we compute radiosity directly. The values computed are compared with values obtained using the wavelet basis. If the norm of the differences is above the refinement threshold, the oracle concludes that we should refine.

This oracle works well, especially if the testing points are chosen with a good heuristics. By putting some of the testing points on the boundaries of the patches, we have found that we obtain a representation of radiosity that looks continuous without having to ensure this continuity in post-processing (see [CAH00] and Figure 2 for an example using $\mathcal{M}_3$ wavelets).

In our algorithm, we do two computations of the refinement oracle: one with standard visibility computations, and one assuming full visibility. If their results differ, visibility is the only reason for subdivision and we introduce a discontinuity-based subdivision.

Subdivisions are thus only introduced in the hierarchy if they actually cancel further refinements on at least one side, resulting in a more compact hierarchy. For point light sources, introducing a subdivision generates a coarse mesh on both sides of the subdivision (see Figure 2(a)). For area light sources, introducing subdivisions creates a coarse mesh in fully lit areas and in the umbra, while the penumbra is more refined (see Figure 2(b)).

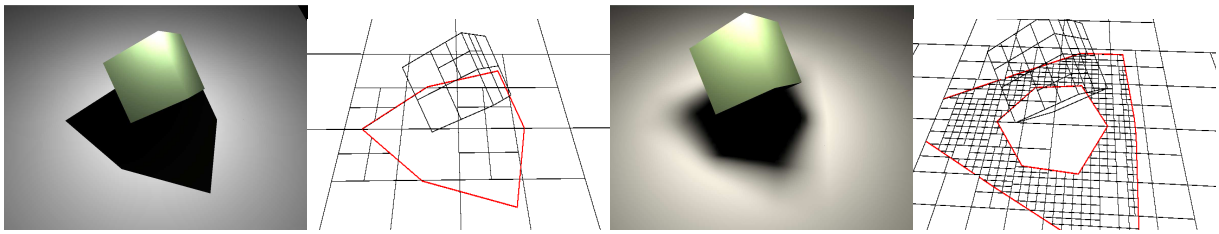
For stability and robustness, a discontinuity is introduced only if the intersection between the discontinuity and the current patch is simple enough. Thus our algorithm only has to manage simple patches and surfaces. For complex occluders casting a combination of simple and complex discontinuities, only the simple discontinuities are introduced in the mesh (see Figure 11).

In our implementation, we have used the following criteria for selecting simple discontinuities: at least one of the patches resulting from the discontinuity-based refinement must be convex, and the number of vertices in each polygon remains below a certain threshold.

3.2.3. Discontinuity-based subdivisions

Once we have selected a patch for discontinuity-based subdivision, we compute the intersection between the patch and the discontinuities, resulting in two separate patches. Most of the time, these sub-patches are neither parallelograms nor triangles. For each of the sub-patches, we build the smallest enclosing parallelogram (see Figure 1). We then use these enclosing parallelograms instead of the patches in the radiosity algorithm as we would use standard patches:

- For *radiosity reception*, the enclosing parallelogram is



(a) Point light source (b) Area light source

Figure 2: *$\mathcal{M}_3$ (quadric) wavelets with discontinuity meshing on simple scenes*

treated as a standard receiver. It is subdivided normally, using regular subdivision.

- For *radiosity emission*, only the actual sub-patch is allowed to emit radiosity; other parts of the enclosing parallelogram are not allowed to emit. Following previous research [HCA00] we do this through the quadrature weights, during the computation of Gaussian quadratures. We see each quadrature weight as the representative of an *area of influence* for the quadrature point (see Figure 3). We modulate the quadrature weight by the percentage of this area of influence that is inside the actual sub-patch.
- For *push-pull*, we use the standard push-pull coefficients since we have a standard subdivision.

3.2.4. Push-Pull Coefficients over a discontinuity

On most steps of the radiosity algorithm, our method uses classical methods. The main difference lies in the push-pull step over the discontinuity.

The enclosing parallelograms of the children patches are overlapping, and we need the push-pull step to compensate for this. Let us assume a patch p has been subdivided into two children patches p_1 and p_2. The children patches p_i are enclosed into parallelograms e_i. Each of the patches have its own set of wavelet basis functions: ϕ_j on p, ϕ^i_j on e_i. The radiosity function is expressed as:

$$\begin{aligned} B_p(x) &= \sum_j \alpha_j \phi_j(x) \\ B_{e_i}(x) &= \sum_j \alpha^i_j \phi^i_j(x) \end{aligned}$$

3.2.4.1. Push Coefficients: For the push step, we need to project B_p on the basis functions of the children e_i. The wavelets coefficients of the projection will be added to the wavelet coefficients on each child e_i. Since, on each patch, wavelets functions form an orthonormal basis, wavelet coefficients are expressed as the scalar product of the radiosity function with the basis functions:

$$\alpha^i_j = \langle B_{e_i} | \phi^i_j \rangle_i$$

where the subscript i on the dot product expresses the fact that the integration takes place on e_i. We are looking for the contribution of B_p to the α^i_j, push^i_j:

$$\begin{aligned} \text{push}^i_j &= \langle B_p | \phi^i_j \rangle_i \\ &= \langle \sum_k \alpha_k \phi_k | \phi^i_j \rangle_i \\ &= \sum_k \alpha_k \langle \phi_k | \phi^i_j \rangle_i \\ &= \sum_k \alpha_k C^i_{kj} \end{aligned}$$

The push coefficients, C^i_{kj}, only depend on the basis functions and on the relative geometry of p and e_i. We have an integral expression for the push coefficients:

$$C^i_{kj} = \langle \phi_k | \phi^i_j \rangle_i = \int_{e_i} \phi_k(x) \phi^i_j(x) dx$$

3.2.4.2. Pull coefficients: For the pull step, we need to combine together the radiosity functions on patches p_i, and express this radiosity on the wavelet basis for patch p. As the e_i patches are overlapping, we restrict the definition of B_{e_i} to its support. We use the characteristic function of e_i, δ_{e_i}, defined as being equal to 1 on e_i and 0 everywhere else.

Combining together the radiosity functions computed on the children gives us:

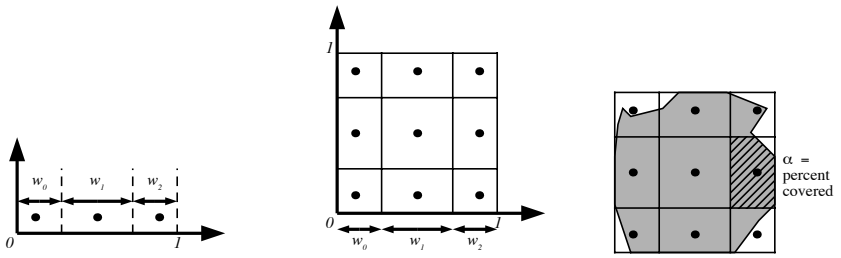
$$B_{e_1}(x)\delta_{e_1}(x) + B_{e_2}(x)\delta_{e_2}(x) = \sum_i \sum_j \alpha^i_j \phi^i_j(x) \delta_{e_i}(x)$$

The pull step projects this combined function on the wavelet basis for p:

$$\begin{aligned} \text{pull}_k &= \sum_i \sum_j \alpha^i_j \langle \phi^i_j \delta_{e_i} | \phi_k \rangle \\ &= \sum_i \sum_j \alpha^i_j D^i_{jk} \end{aligned}$$

The pull coefficients, D^i_{jk} depend on the geometry of the subdivision:

$$D^i_{jk} = \langle \phi^i_j \delta_{e_i} | \phi_k \rangle = \int_p \phi^i_j(x) \delta_{e_i}(x) \phi_k(x) dx$$



(a) on the unit segment (b) on the unit square (c) on the extended domain

Figure 3: *The weights of the quadrature points can be seen as the area of a* zone of influence.

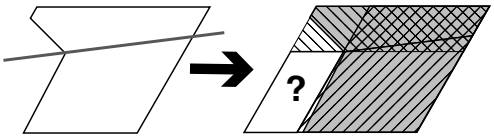


Figure 4: *A non-convex patch cut along a discontinuity can result in two children whose enclosing parallelograms do not cover the enclosing parallelogram of the parent.*

3.2.4.3. Computation of push-pull coefficients For the push-pull step over the discontinuity, we have an integral expression, which we approximate using Gaussian quadratures. As we are integrating a discontinuous function (δ_{e_k}), we might have accuracy problems in the computation. We compensate by using a large number of sampling points. Also, once we have computed the coefficients for one patch, we check that they are consistent with each other, and that there is no creation or destruction of energy during the push-pull step. Should we detect an inconsistency, we recompute them with more precision.

Push-pull coefficients are then stored on the hierarchy. Because these special push-pull coefficients only happen once for each discontinuity-based subdivision, we can afford to spend some time computing them.

3.2.5. Intersection of several discontinuities

The patches resulting from a discontinuity-based subdivision are not necessarily convex. If a non-convex patch is cut by another subdivision, the enclosing parallelograms of the children do not cover the enclosing parallelogram of the parent patch (see Figure 4).

This configuration appears when discontinuities from two light sources intersect each other, or when the umbra and penumbra boundaries touch each other, for an occluder that is in contact with the receiver.

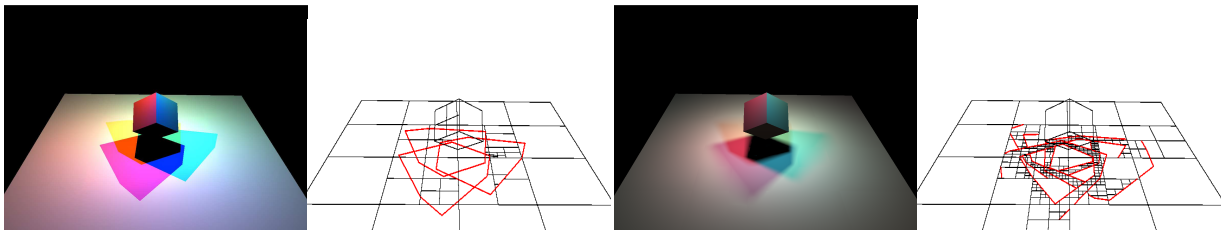
When it appears, it causes the push-pull coefficients to be incomplete in their definition. To account for this, we extend the enclosing parallelograms of the children so that their union covers the enclosing parallelogram of the parent patch. Except for this small point there is no special case in our algorithm for dealing with several light sources and intersecting discontinuities (see Figure 5).

3.3. Implementation details

In this section, we review implementation details of our algorithm. The points described here are not *essential* to our algorithm; others could use different approaches, *e.g.* for computing discontinuities, or for handling visibility queries in radiosity computations. However, the approach we used to solve these problems can be interesting to other researchers.

We have used non-conventional solutions for computing discontinuities, in the refinement oracle, for visibility queries and for displaying results:

Finding Discontinuities: We only need the set of discontinuities for the interaction currently being refined. We compute extremal discontinuities (umbra and penumbra boundaries), using a method based on the `GLU Tesselator` [SWND03]. Our method identifies EV, VE and EEE events, converts these events into 2D polygons corresponding to their intersection with the plane of the receiver, then uses the `GLU Tesselator` to compute the union and the intersection of these 2D polygons. Our algorithm for finding discontinuities is not complete (it can miss some discontinuities) but it is robust and it finds the most important discontinuities.
Umbra and penumbra boundaries are not necessarily linear: on EEE evetns, parts of them can be conic curves. Our algorithm deals with such conics in a straightforward manner.
Once we have computed the umbra and penumbra contours, we have to answer *position queries*: “is this point inside the umbra or not?”. We store the contours in an arrangement of line segments, using trapezoidal maps(see,



(a) Point light sources ($\mathcal{M}_3$ wavelets)

(b) Area light sources ($\mathcal{M}_3$ wavelets)

Figure 5: *Combining together several discontinuities (both scenes have three light sources, red, green and blue, located in a triangle above the cube).*

e.g. [BDS*92, CGA]). This randomized data structure answers our positions query in average time $O(\log n)$, with creation time $O(n \log n)$ and memory cost $O(n)$.

Handling Discontinuites in the Refinement Oracle: The refinement oracle takes sampling points on the receiving patch. Some sampling points can lie on a discontinuity, which makes their exact value unknown. To avoid unnecessary refinement, points lying on a discontinuity can take a different value in the oracle depending on the patch being considered.

Handling Visibility Queries: In our radiosity computations, we need the percentage of the light source that is visible from the receiving points. Previous implementations used a geometric data-structure, the *Backprojection* [DF94] to compute an exact value of this percentage. We are computing it instead using an OpenGL extension, `OcclusionQuery` [ARB], which gives us the percentage of the pixels of the light source that are visible from the receiving point. In our experiments, occlusion queries are more robust than the geometric data structure while having the same speed, and they are much faster than casting rays, while giving more precise results.

Displaying Results: $\mathcal{M}_3$ wavelets give quadrically varying functions; they are displayed using a small fragment program (10 lines of code). Linear interpolation from the graphics hardware (*Gouraud shading*) is not perfect for $\mathcal{M}_2$ wavelets, which are bilinear functions. It is possible to replace this linear interpolation by a small fragment program.

4. Experimentations and Results

4.1. Experimentation protocol

Test scenes: We have used two different test scenes: the Cabin, from Radiance set of test scenes, and Room 523 from the Soda Hall model. For each scene, we used either point light sources or area light sources, giving a total of four test scenes. On all test scenes, we computed direct and indirect illumination. Pictures of the test scenes are available in Figures 7 and 13 (see color plates).

Wavelet Bases: We have tested our algorithm with the first three multi-wavelets bases: $\mathcal{M}_1$ (Haar), $\mathcal{M}_2$ (piecewise-linear) and $\mathcal{M}_3$ (piecewise-quadric). In the pictures, Haar wavelets are displayed after a post-processing step to ensure continuity, $\mathcal{M}_2$ wavelets are displayed using standard linear interpolation from the graphics hardware and $\mathcal{M}_3$ wavelets use a fragment program for the quadrically varying part.

Material: All computations were done on the same computer: a 2.4 GHz Pentium IV, with 1Gb memory and an NVIDIA GeForce FX 5600.

4.2. Visual comparison for point light sources

The first reason to use discontinuity meshing is the quality of the illumination computed. Adapting the mesh to the discontinuities produces a radiosity function that looks pleasing to the eye.

The leftmost columns of Figures 6 and 12 show a side-by-side comparison of the different wavelet bases on a specific detail of the Cabin test scene, with a point light source. All pictures were generated with the same computation time (25 s) to give a fair comparison of the different wavelet bases. Without discontinuity meshing, the most satisfying representation is obtained with $\mathcal{M}_2$ wavelets, but artefacts are clearly visible along the discontinuity line; Haar and $\mathcal{M}_3$ wavelets are visually not acceptable within the prescribed time frame; they would eventually achieve a satisfying result, but for a longer computation time.

With discontinuity meshing, all wavelets bases achieve a visually pleasing result. Our algorithm for merging discontinuities with wavelets thus achieves a visually better result in the same computation time.

We did a similar comparison for Room 523 of the Soda Hall. Figure 8 shows the pictures obtained with the different wavelet bases on a detail of the room. All pictures were

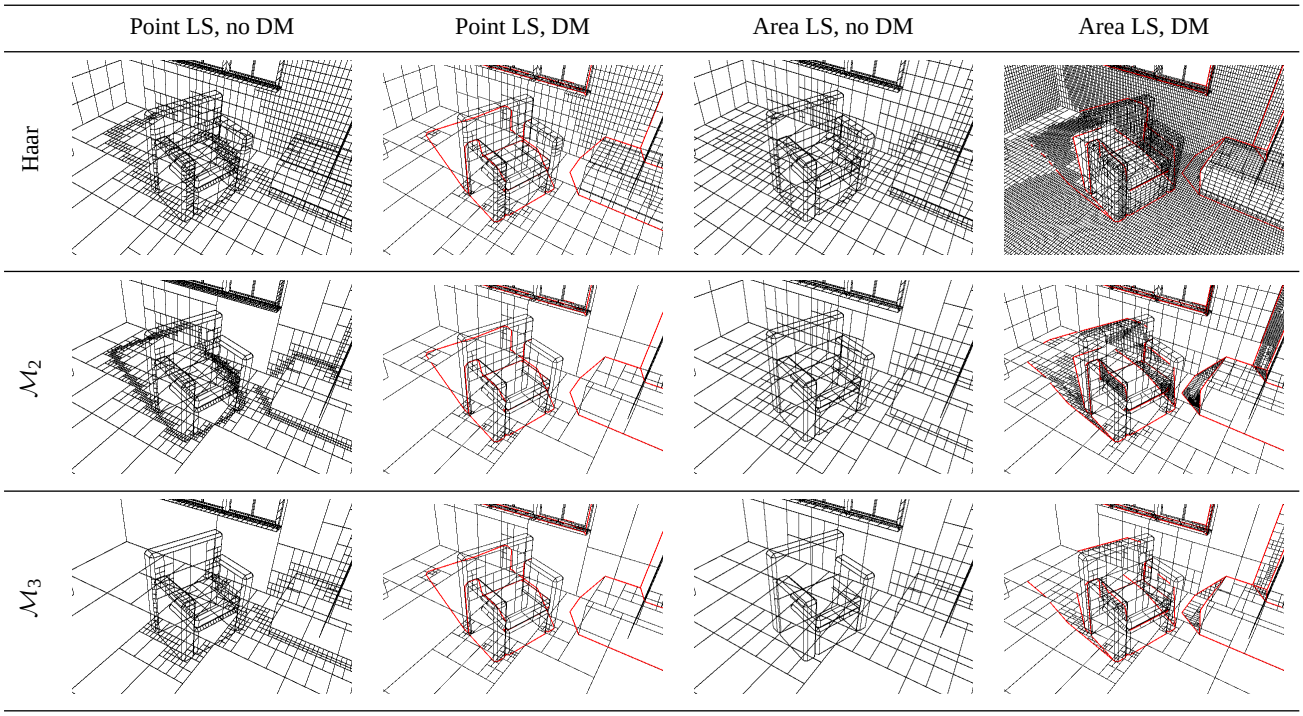


Figure 6: *Wireframe version of simulation for the Cabin test scene. See also the color plates.*

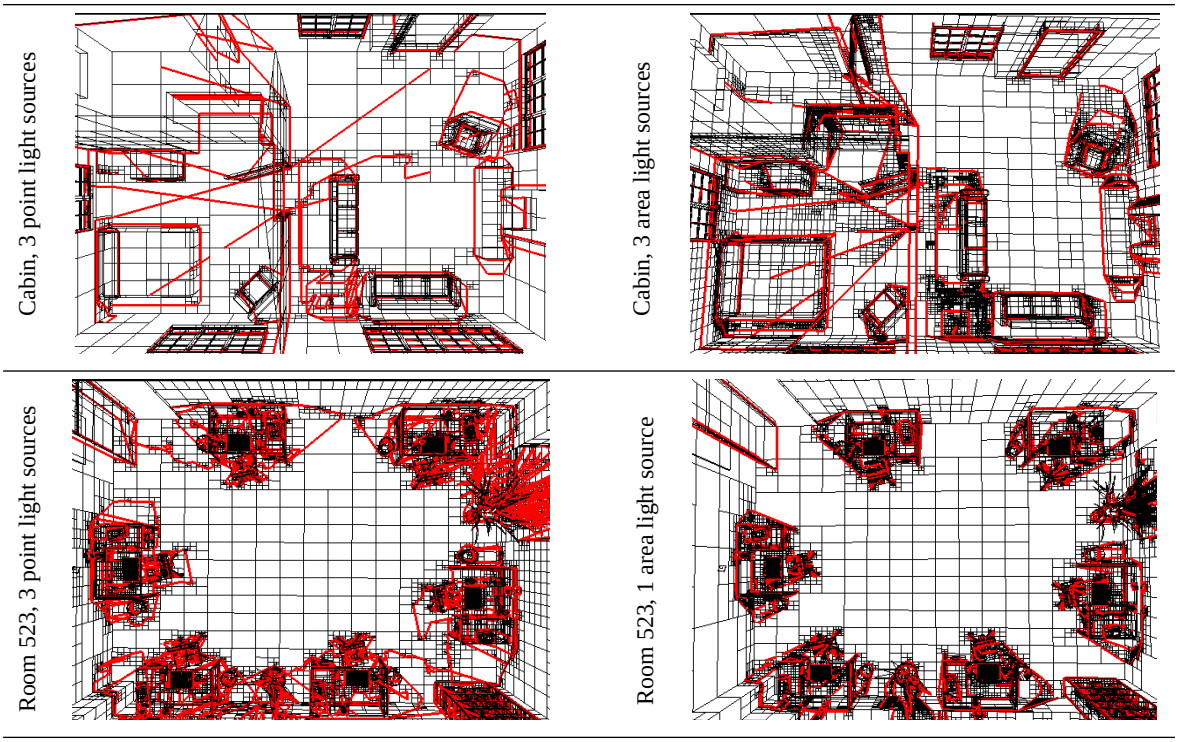
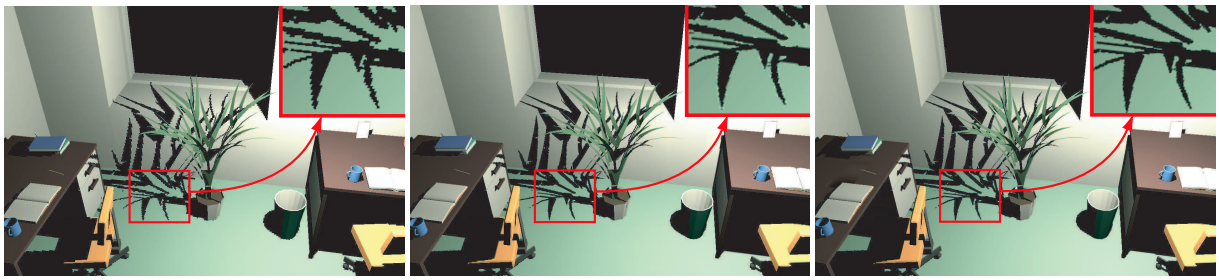


Figure 7: *Wireframe version of our test scenes after simulation with $\mathcal{M}_3$ wavelets. See also the color plates.*

(a) Haar+DM (b) $\mathcal{M}_2$+DM (c) $\mathcal{M}_3$+DM



(d) Haar (e) $\mathcal{M}_2$ (f) $\mathcal{M}_3$

Figure 8: *Visual comparison of the different wavelet bases for a point light source. All pictures used roughly 190 s computation time.*

generated with approximately the same computation time (190 s).

From a distant point of view, all the pictures are of comparable quality. On a large scene like this, with a high number of discontinuities, the time spent estimating the discontinuities and dealing with them becomes equivalent to the time it takes to do regular subdivisions.

However, when we look closely at the shadow boundaries, ringing artefacts and staircase effects become clearly visible (see the full resolution inserts).

We also compared the memory costs for both versions of the program (with discontinuity-based subdivision and without). Figure 9 shows the memory costs for all three wavelet bases, for the pictures on Figures 6, 12 and 8. On a scene with a large number of discontinuities, such as Room 523, our algorithm results in an important gain in memory costs. Each discontinuity introduced replaces a large number of regular patches, resulting in a net gain. On the Cabin test scene, with the prescribed time limit, the refinement was not pushed to the same levels. As a consequence, the memory gain is not as strong.

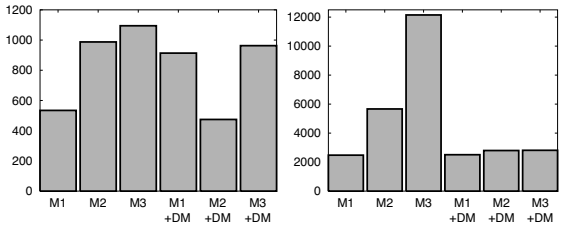
In short, our algorithm for merging discontinuities with higher-order wavelet bases always gives better results than existing algorithms, with a smaller memory cost.

4.3. Visual Comparison for Area Light Sources

The rightmost columns of Figures 6 and 12 show the same comparison of the different wavelet bases for the same detail of the Cabin test scene, this time using an area light source. All pictures were generated with approximately the same computation time (240 s). This time, the benefits of using the discontinuity-based approach appear very clearly. All three wavelet bases greatly outperform the non-discontinuity-based versions. This is because computing the discontinuities speeds-up the visibility computations, the most expensive step in hierarchical radiosity. Within discontinuity-based wavelet methods, $\mathcal{M}_2$ and $\mathcal{M}_3$ wavelets produce the nicest result.

For comparison, Figure 10 shows the memory costs for all wavelet bases on this test scene. Notice that this time, the memory cost is bigger *with* discontinuity meshing than without.

This is a side effect of our comparison method: the time given for the simulation was much too short for the system without discontinuities. It has just computed a crude ver-



(a) Cabin (b) Room 523

Figure 9: *Memory costs (in Kb) for simulations on our test scenes, with point light sources.*

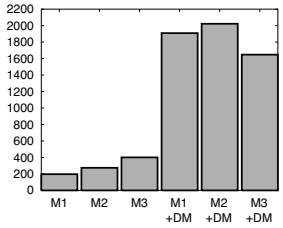


Figure 10: *Memory costs (in Kb) for simulations on the Cabin test scene, with an area light source.*

sion of illumination. If we give it more time for simulation, it eventually computes a nice version of the illumination, at a higher memory cost. Note that the memory cost without discontinuities increases with the order of the wavelet base. This is consistent with previous research [HCA00]: for crude estimates of the illumination, the memory cost increases with the order of the wavelet base, while for high-quality estimates, the memory cost decreases with the order of the wavelet base.

4.4. Selective choice of discontinuities

In places where the radiosity is smoothly varying, our algorithm can choose not to introduce discontinuities, keeping the regular subdivision. This effect appears clearly in the wireframe representations of our test scenes (see the rightmost column of Figures 6 and 12, and Figure 7).

Figure 11 also shows a detail of the Room 523 test scene (with an area light source) where complex discontinuities exist, but were not introduced in the mesh. This behaviour is more likely to occur with area light sources, which produce smoothly varying illumination, than with point light sources, where there is always a C^0 discontinuity at a shadow boundary.

4.5. Influence of the minimal area parameter

An important issue in the practical use of radiosity algorithms is the choice of parameters. A bad choice of the parameters results in a long computation time or a simulation that is not visually pleasing – sometimes both.

The minimal area for patches is one of these parameters. Without discontinuity-based refinement, this minimal area is reached for many patches on all sharp discontinuities (see the wireframe representations of the test scenes, in Figure 6 and additional materials). A variation of this parameter has important consequences on the computation time, on the memory cost and on the quality of the result. Dividing this minimal area by 2 results in twice as many patches being used to represent each sharp discontinuity, potentially doubling the computation times and memory cost.

With discontinuity-based refinement the minimal area is not reached, except in places where discontinuities are too complex. Hence, a variation of this parameter has little consequences on the computation times and memory costs. Our algorithm has almost cancelled the influence of the minimal area parameter. The user of our radiosity system has one main parameter, the maximum value of the error on each interaction. The minimal area still has an effect on the quality of the simulation, computation time and memory cost, but it is a minor effect.

5. Conclusion and Future Directions

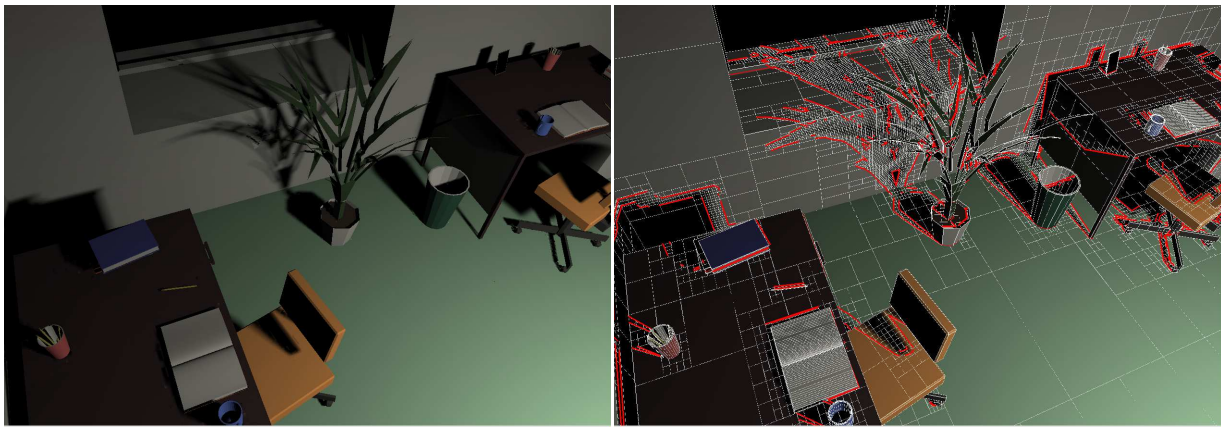
In this paper, we have presented a new algorithm for radiosity computations, that combines higher-order wavelets with discontinuity meshing. Our algorithm uses regular subdivision for wavelets where it is practical, and switches to discontinuity-based subdivision where discontinuities exist. Only discontinuities that are important in the computation of the illumination solution are actually introduced in the mesh. This results in a compact representation of radiosity, with a good compromise between quality and cost.

This representation can be displayed directly on the screen, or it can be used as a starting point for more complete illumination computations, such as Monte-Carlo illumination.

Our algorithm is robust enough to handle complex discontinuities. It automatically discards discontinuities which are not important enough for the radiosity computations, providing a good way to manage the complicated set of discontinuities.

In future work, we want to combine our algorithm with a robust computation of visibility discontinuities (e.g. [DD02]). We also want to combine our work with separate works allowing higher-order wavelet radiosity computations on curved surfaces [ACP*01] and triangular meshes.

In a separate direction of research, although our algorithm



(a) (b)

Figure 11: *Our algorithm only inserts discontinuities that are perceived as useful by the refinement oracle ($\mathcal{M}_3$ wavelets).*

only inserts discontinuities as they are needed in the refinement process, it starts by computing all potential discontinuities for the current interaction, a costly preliminary step. We will explore the possibility to suppress this step, using standard refinement but detecting inside the refinement oracle that subdivision is probably caused by a discontinuity, then only computing and inserting this discontinuity in the mesh. This would reduce the computation cost of our algorithm. In our experiments, almost all the discontinuities caused by indirect lighting are not important enough to justify their insertion in the hierarchy. It is therefore not practical to compute them in advance.

6. Acknowledgements

This work was partly funded by the "Région Rhône-Alpes" under the DEREVE project.

The Soda Hall model was created by the original Berkeley Walkthru team under the direction of Prof. Carlo H. Sequin. For more information on the Soda Hall, see `http://www.cs.berkeley.edu/~sequin/soda/soda.html`

References

[ACP*01] ALONSO L., CUNY F., PETITJEAN S., PAUL J.-C., LAZARD S., WIES E.: The virtual mesh: A geometric abstraction for efficiently computing radiosity. *ACM Transactions on Graphics 20*, 3 (July 2001), 169–201.

[Alp93] ALPERT B. K.: A class of bases in L^2 for the sparse representation of integral operators. *SIAM Journal on Mathematical Analysis 24*, 1 (1993), 246 – 262.

[ARB] ARB_occlusion_query. `http://oss.sgi.com/projects/ogl-sample/registry/ARB/occlusion_query.txt%`.

[BDS*92] BOISSONNAT J.-D., DEVILLERS O., SCHOTT R., TEILLAUD M., YVINEC M.: Applications of random sampling to on-line algorithms in computational geometry. *Discrete and Computational Geometry 8*, 1 (1992), 51–71.

[BW96] BEKAERT P., WILLEMS Y.: Error Control for Radiosity. In *Rendering Techniques '96 (7th Eurographics Workshop on Rendering)* (1996), pp. 153–164.

[CAH00] CUNY F., ALONSO L., HOLZSCHUCH N.: A novel approach makes higher order wavelets really efficient for radiosity. *Computer Graphics Forum (Eurographics 2000) 19*, 3 (2000), C–99–C–108.

[CGA] CGAL, Computational Geometry Algorithms Library. `http://www.cgal.org`.

[DD02] DUGUET F., DRETTAKIS G.: Robust epsilon visibility. *ACM Transactions on Graphics (SIGGRAPH 2002) 21*, 3 (2002).

[DDP99] DURAND F., DRETTAKIS G., PUECH C.: Fast and accurate hierarchical radiosity using global visibility. *ACM Transactions on Graphics 18*, 2 (1999), 128–170.

[DDP02] DURAND F., DRETTAKIS G., PUECH C.: The 3d visibility complex. *ACM Transactions on Graphics 21*, 2 (Apr. 2002).

[DF94] DRETTAKIS G., FIUME E.: A Fast Shadow Algorithm for Area Light Sources Using Backprojection. In *Computer Graphics Proceedings, Annual Conference Series, (ACM SIGGRAPH '94)* (1994), pp. 223–230.

[DS96] DRETTAKIS G., SILLION F.: Accurate Visibility and Meshing Calculations for Hierarchical Radiosity. In *Rendering Techniques '96 (7th Eurographics Workshop on Rendering)* (1996), pp. 269–278.

[GS96] GHALI S., STEWART A. J.: A Complete Treatment of D1 Discontinuities in a Discontinuity Mesh. In *Graphics Interface '96* (May 1996), pp. 122–131.

[GSCH93] GORTLER S. J., SCHRODER P., COHEN M. F., HANRAHAN P.: Wavelet Radiosity. In *Computer Graphics Proceedings, Annual Conference Series, (ACM SIGGRAPH '93)* (1993), pp. 221–230.

[GTGB84] GORAL C. M., TORRANCE K. E., GREENBERG D. P., BATTAILE B.: Modelling the Interaction of Light Between Diffuse Surfaces. *Computer Graphics (ACM SIGGRAPH '84) 18*, 3 (July 1984), 212–222.

[HCA00] HOLZSCHUCH N., CUNY F., ALONSO L.: Wavelet radiosity on arbitrary planar surfaces. In *Rendering Techniques 2000 (11th Eurographics Workshop on Rendering)* (2000), pp. 161–172.

[Hec92] HECKBERT P.: Discontinuity Meshing for Radiosity. In *Third Eurographics Workshop on Rendering* (May 1992), pp. 203–226.

[Hed98] HEDLEY D.: *Discontinuity Meshing for Complex Environments*. PhD thesis, Department of Computer Science, University of Bristol, Bristol, UK, Aug. 1998.

[HS92] HANRAHAN P., SALZMAN D.: A Rapid Hierarchical Radiosity Algorithm for Unoccluded Environments. In *Photorealism in Computer Graphics (Proceedings Eurographics Workshop on Photosimulation, Realism and Physics in Computer Graphics, 1990)* (1992), pp. 151–171.

[HSA91] HANRAHAN P., SALZMAN D., AUPPERLE L.: A Rapid Hierarchical Radiosity Algorithm. *Computer Graphics (ACM SIGGRAPH '91) 25*, 4 (July 1991), 197–206.

[HWP97] HEDLEY D., WORRALL A., PADDON D.: Selective culling of discontinuity lines. In *Rendering Techniques '97 (8th Eurographics Workshop on Rendering)* (1997), pp. 69–80.

[KH84] KAJIYA J. T., HERZEN B. P. V.: Ray Tracing Volume Densities. *Computer Graphics (ACM SIGGRAPH '84) 18*, 3 (July 1984), 165–174.

[LTG92] LISCHINSKI D., TAMPIERI F., GREENBERG D. P.: Discontinuity Meshing for Accurate Radiosity. *IEEE Computer Graphics and Applications 12*, 6 (November 1992), 25–39.

[LTG93] LISCHINSKI D., TAMPIERI F., GREENBERG D. P.: Combining Hierarchical Radiosity and Discontinuity Meshing. In *Computer Graphics Proceedings, Annual Conference Series, (ACM SIGGRAPH '93)* (1993), pp. 199–208.

[PB95] PATTANAIK S. N., BOUATOUCH K.: Discontinuity Meshing and Hierarchical Multiwavelet Radiosity. In *Graphics Interface '95* (May 1995), pp. 109–115.

[SG94] STEWART A. J., GHALI S.: Fast Computation of Shadow Boundaries Using Spatial Coherence and Backprojection. In *Computer Graphics Proceedings, Annual Conference Series 1994 (ACM SIGGRAPH '94)* (1994), pp. 231–238.

[SGCH93] SCHRODER P., GORTLER S. J., COHEN M. F., HANRAHAN P.: Wavelet Projections for Radiosity. In *4th Eurographics Workshop on Rendering* (June 1993), pp. 105–114.

[SSSS98] STAMMINGER M., SCHIRMACHER H., SLUSALLEK P., SEIDEL H.-P.: Getting rid of links in hierarchical radiosity. *Computer Graphics Forum (Eurographics '98) 17*, 3 (September 1998), C165–C174.

[Stu94] STURZLINGER W.: Adaptive Mesh Refinement with Discontinuities for the Radiosity Method. In *Photorealistic Rendering Techniques (5th Eurographics Workshop on Rendering)* (June 1994), pp. 239–248.

[SWND03] SHREINER D., WOO M., NEIDER J., DAVIS T.: *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.4*. Addison Wesley Professional, 2003, ch. 11. Tessellators and Quadrics.

[WH97] WILLMOTT A., HECKBERT P.: An empirical comparison of progressive and wavelet radiosity. In *Rendering Techniques '97 (8th Eurographics Workshop on Rendering)* (1997), pp. 175–186.

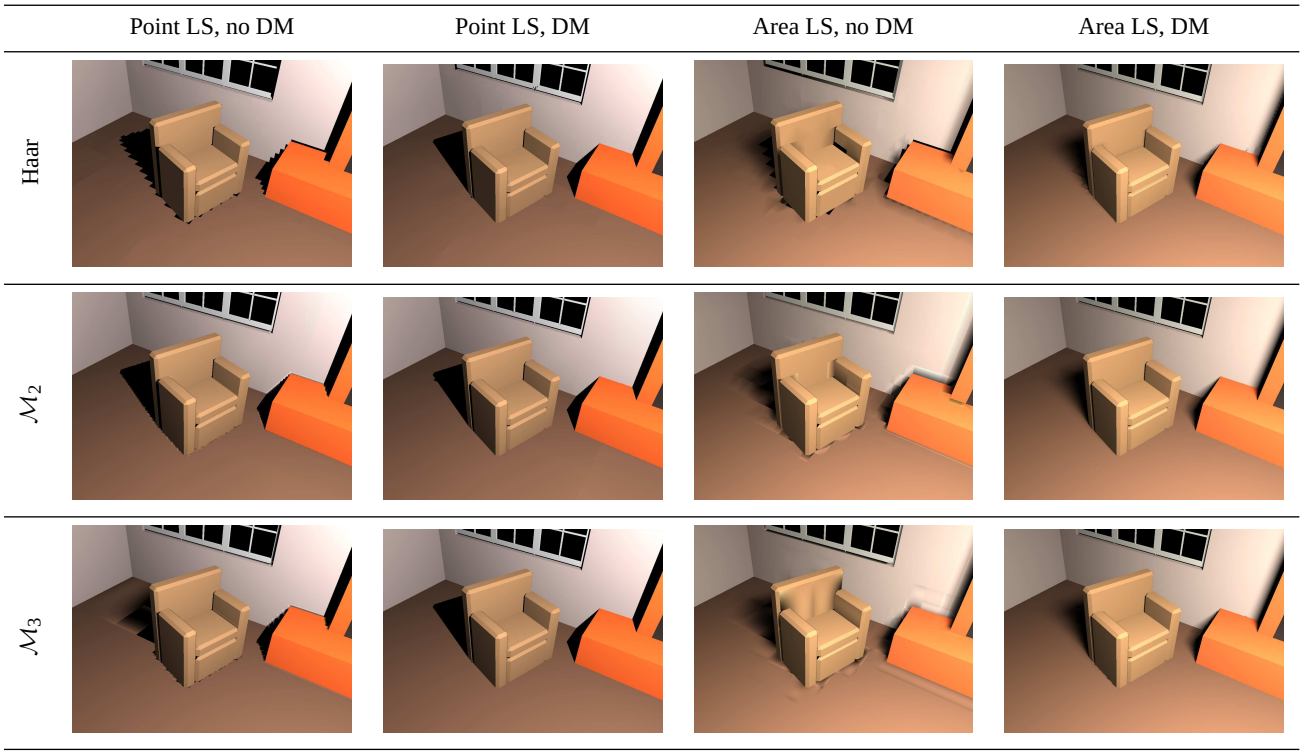


Figure 12: *Visual comparison of results for the Cabin test scene. See also figure 6 for wireframe representation.*

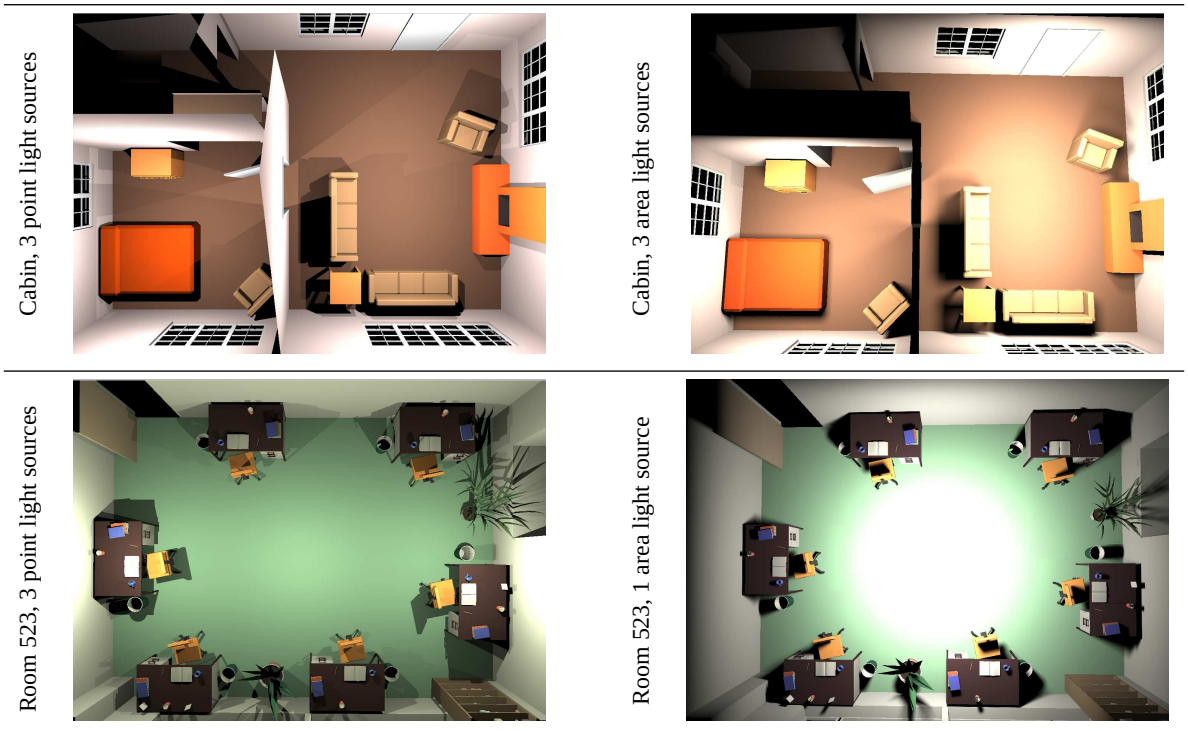


Figure 13: *Our test scenes (all figures with $\mathcal{M}_3$ wavelets). See also figure 7 for wireframe representation.*

2.7.6 Space-time hierarchical radiosity with clustering and higher-order wavelets (CGF 2004)

Auteurs : Cyrille D , Nicolas H François S
Journal : *Computer Graphics Forum*, vol. 23, n° 2.
Date : avril 2004

Volume 23 (2004), number 2 pp. 129–141

COMPUTER GRAPHICS *forum*

Space-Time Hierarchical Radiosity with Clustering and Higher-Order Wavelets

Cyrille Damez[1], Nicolas Holzschuch[2] and François X. Sillion[2]

[1] Max-Planck-Institut für Informatik, Saarbrücken, Germany
[2] ARTIS, GRAVIR/IMAG-INRIA, Grenoble, France

Abstract

We address in this paper the issue of computing diffuse global illumination solutions for animation sequences. The principal difficulties lie in the computational complexity of global illumination, emphasized by the movement of objects and the large number of frames to compute, as well as the potential for creating temporal discontinuities in the illumination, a particularly noticeable artifact. We demonstrate how space-time hierarchical radiosity, i.e. the application to the time dimension of a hierarchical decomposition algorithm, can be effectively used to obtain smooth animations: first by proposing the integration of spatial clustering in a space-time hierarchy; second, by using a higher-order wavelet basis adapted for the temporal dimension. The resulting algorithm is capable of creating time-dependent radiosity solutions efficiently.

Keywords: global illumination, animation, hierarchical radiosity, clustering, wavelets.

ACM CCS: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

1. Introduction

Global illumination techniques have reached the stage where they allow the calculation of high-quality images of three-dimensional scenes, complete with subtle lighting and inter-reflection effects. It is therefore natural to try and use them for the production of animation films, or more generally in all lighting jobs related with special effects, such as combining synthesized elements with live action film footage. Unfortunately, global illumination techniques remain typically expensive to use, even more so in the case of frame-by-frame lighting calculations.

In this paper, we present a fully developed version of the space-time hierarchical radiosity, an algorithm aimed at computing view independent global illumination simulations for animated scenes. It works with scenes containing moving solid objects, whose trajectory are known beforehand and computes a hierarchical radiosity solution for the entire animation, instead of frame-by-frame.

The hierarchical formulation for radiosity is extended by introducing a fourth dimension, time, along with the three spatial dimensions. All four dimensions are treated in the same way, meaning that we can refine an interaction either in time or in space. This results in few computations being done in areas where there is little temporal variation of the illumination, while areas with rapid variation of the illumination will be computed to full precision. The hierarchical formulation guarantees a compact representation of the temporal variations of the radiosity function: as a result, the entire animation is computed much faster than by performing a complete radiosity solution for each frame.

In a way similar to the original Hierarchical Radiosity algorithm [1], the efficiency of the space-time hierarchical radiosity algorithm depends on the depth of the space-time hierarchy built during computations: the deeper the hierarchy, the more efficient the algorithm. This suggests that it should prove especially beneficial for complex scenes.

We have previously presented a preliminary version of the space-time hierarchical radiosity algorithm [2]. We present here a fully developed algorithm. In particular, two major issues are addressed: first, we have modified the algorithm so that it uses linear wavelets for the time dimension, to improve

Submitted September 2003
Revised January 2004
Accepted February 2004

the temporal continuity of the animations produced. Second, we have combined space-time radiosity with clustering, thus enabling the algorithm to work more efficiently and on larger scenes. This allows us to use this algorithm in a range of complexity where its benefits can be fully realized.

The paper is organized as follows. In the next section, we briefly discuss previous work in time-dependent illumination of animated scenes, and review the shortcomings of our preliminary approach. Then, in Section 3, our fully developed algorithm is given in detail. In Section 4, we provide an analysis of the performances of space-time hierarchical radiosity, compared to our previous approach as well as frame-by-frame computations. Finally, in Section 5, we draw our conclusions and trace directions for future work.

2. Background and Motivations

2.1. Global illumination algorithms for animations

Several algorithms to compute global illumination images have been proposed since the pioneering work of Goral *et al.* [3]. As the performance of the said algorithms and the computing power of graphics workstations improved, several propositions have been made to extend these algorithms and reduce the overwhelming cost of computing globally illuminated animations. Two classes of applications can be distinguished:

- Interactive methods, which render new solutions quickly, usually by reusing previous computation results as much as possible. They aim at offering as fast a feedback as possible in response to changes made by the user. Interactive methods have been developed for Hierarchical Radiosity [4,5], Path Tracing [6,7] and Particle Tracing [8,9,10,11].
- Offline methods, where the objects movements are supposed continuous and known *a priori*. They aim at rendering high-quality animations, and therefore should ensure a constant quality. Global Monte Carlo methods [12] and Particle Tracing [13] algorithms have been proposed to compute high-quality global illumination animations.

A study of the current state of the art for both types of animated global illumination algorithms can be found in Damez *et al.* [14]. In this section, we discuss briefly only the methods allowing the computation of higher-quality animations.

Surprisingly, the case of high-quality animations has received little attention when compared to the amount of work devoted to interactive algorithms in the literature. Indeed, most interactive algorithms could be used to compute a movie sequence. However, the quality of the resulting animation may not always be satisfying, as these methods were designed to satisfy real-time constraints instead of animation quality criteria.

In particular, the accumulated errors due to the incremental nature of most interactive algorithm may cause distracting artifacts. The resulting frames quality may seem acceptable when considered separately. Nevertheless, discontinuities in the shading of surfaces may appear between two consecutive frames. The interactive global illumination algorithm proposed by Wald *et al.* [10], though it recomputes a complete global illumination solution for each frame independently, is fast enough to converge to a good quality view-dependent solution within a couple seconds. However, in order to avoid light flickering due to its stochastic nature, it requires to use the same random seeds from one frame to the other, which only ensures temporal continuity of lighting for light paths that do not intersect moving objects.

It also seems natural to try to capitalize on the knowledge of objects movement to enhance the quality of the rendered animation. Therefore, it makes sense to consider high-quality animations rendering as a separate problem, and to develop algorithms specifically designed to solve it. Myszkowski *et al.* [13] extended the density estimation photon-tracing algorithm to the case of animated scenes, allowing the use of photons for several consecutive frames. The decision to extend or contract the segment of time during which a given sample is valid is based on a perception-based Animation Quality Metric. It is used to measure the perceived difference between consecutive frames, and therefore reduce the flickering which results from the stochastic noise. However, to this date, this method is based on a fixed mesh and lack some kind of adaptive refinement scheme. Therefore, the spatial resolution of the solutions computed is limited.

Martin *et al.* [15] proposed a two pass algorithm based on hierarchical radiosity. During the first pass, a coarse hierarchical solution for the complete animation is computed incrementally. Then during the second pass, the resulting mesh and link structure is used to efficiently perform final gathering, assigning to each space-time mesh element a high-resolution texture movie representing the radiosity of this patch during the corresponding interval of time. Since this algorithm efficiently solves the problem of high-quality final gathering for animated scenes, which our approach does not address, both methods can be seen as complementary. In particular, the algorithm of Martin *et al.* does not make use of a cluster hierarchy during the first pass, which limits its application to very simple scenes. However, we show in Section 3.3 how to solve this particular issue. As a consequence, coupling both approaches seems promising.

2.2. Previous work on the space-time hierarchical radiosity algorithm

In order to reduce the cost of diffuse global illumination computations for animations, we introduced in a previous

publication [2] the space-time hierarchical radiosity algorithm. Our preliminary algorithm lacked several key features that would enable its use on scenes with complex geometry or lighting condition. In particular, it did not feature a way to extend the object hierarchy above the surfaces level (an approach that is commonly referred as *clustering* [16,17]).

Moreover, distracting "jumps" in the illumination could appear in scenes where important changes in indirect lighting occur along time. Similar discontinuities can be observed in the spatial dimension for the classical static Hierarchical Radiosity algorithm (*cf.* Figure 1), when the refinement oracle used is based only on a global evaluation of the error. Oracles designed to take into account the distribution of the error on the receiving elements can remove such discontinuities [18,19].

Additionally, the piecewise constant function basis we used did not allow a proper distribution of the approximation errors on the mesh elements. As a consequence, low-intensity light exchanges, such as indirect bounces, were either insufficiently refined, or overly refined when the refinement threshold was reduced.

We demonstrate such discontinuities using a scene for which our preliminary algorithm performed in an obviously unsatisfying manner. This scene is composed of four boxes in a closed room with colored walls (red, blue, green and gray), illuminated by rotating spotlights. As a consequence the lighting in the scene is mostly indirect, and varies in large proportions. The resulting animation presents important lighting discontinuities in time, in particular at the main subdivisions of the animation time interval, as illustrated in Figure 2. Such discontinuities must obviously be reduced in order to make our algorithm of practical use.

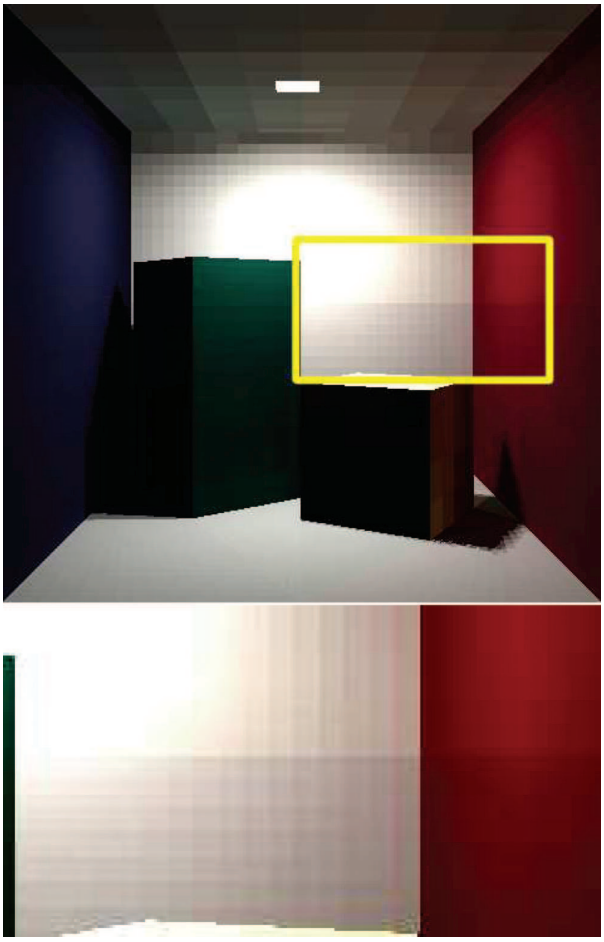


Figure 1: *A radiosity discontinuity is clearly visible between the upper and lower half of the walls.*

3. The Space-Time Hierarchical Radiosity Algorithm

3.1. The space-time radiosity equation

We want to compute the radiosity function $B(p, t)$ for each point p at each time t defined over $(\mathcal{S} \times \mathcal{T})$ where $\mathcal{S}$ is the set of all points on all surfaces of the scene and $\mathcal{T}$ is the time interval over which we want to compute our animation. We define the following functions:

- $r : (\mathcal{S} \times \mathcal{S} \times \mathcal{T}) \to \mathbb{R}$ the distance from point p to point q at time t
- $\theta : (\mathcal{S} \times \mathcal{S} \times \mathcal{T}) \to [0, \pi]$ the angle between the outgoing normal at p and the direction from p to q at time t
- $v : (\mathcal{S} \times \mathcal{S} \times \mathcal{T}) \to \{0, 1\}$ the visibility function between p and q at t
- $\rho : \mathcal{S} \to [0, 1]$ the diffuse reflectance at p
- $E : (\mathcal{S} \times \mathcal{T}) \to \mathbb{R}$ the self-emitted radiosity at p at time t

Using these definitions, for every $X = (p, t) \in (\mathcal{S} \times \mathcal{T})$ the radiosity function satisfies the following equation:

$$B(X) = E(X) + \int_{(\mathcal{S}\times\mathcal{T})} B(Y)\mathcal{K}(X, Y)\,\mathrm{d}Y, \qquad (1)$$

where

$\mathcal{K}$ is defined on $(\mathcal{S} \times \mathcal{T})^2$ as

$$\mathcal{K}((p, t), (q, t')) = \rho(p)k(p, q, t)\delta(t, t') \qquad (2)$$

k is the function defined on $(\mathcal{S} \times \mathcal{S} \times \mathcal{T})$ by

$$k(p, q, t) = \frac{\cos\theta(p, q, t)\cos\theta(q, p, t)}{\pi r(p, q, t)^2} v(p, q, t) \qquad (3)$$

δ is the Dirac distribution equal to 0 when $t \neq t'$

Note that, though equation (1) seems to describe intertemporal light exchanges, such nonphysical transfers are avoided thanks to the Dirac function in equation (2). Note also that

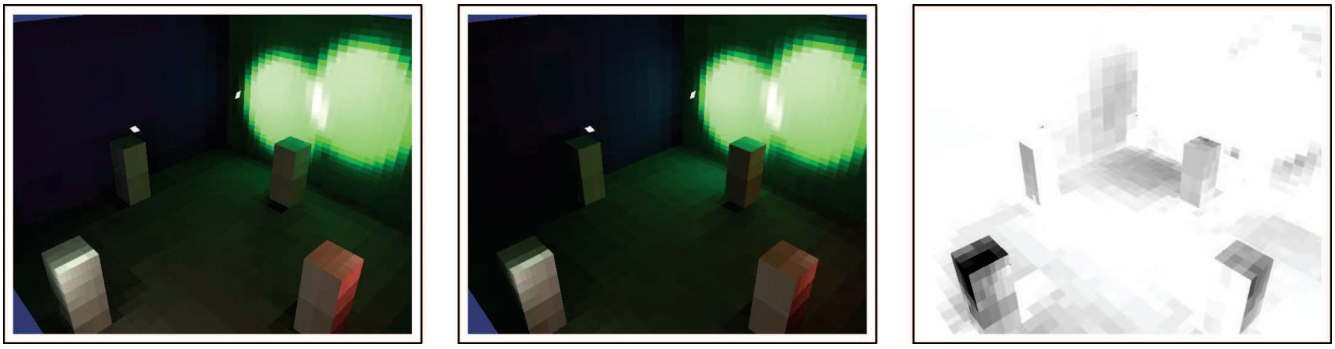


Figure 2: *Example of temporal discontinuities. Visualizations of the scene at $t = 1/2 - \epsilon$, at $t = 1/2 + \epsilon$ and the difference image. The illumination of the entire scene has been modified in a single frame interval.*

equation (1) is formally equivalent to the classical radiosity equation in the static case. Therefore, any algorithm capable of solving the latter can probably be extended in a straightforward manner to solve the former. In particular, we shall see that we can derive a finite element formulation similar to that of standard radiosity [17,20].

3.2. Discretization

Equation (1) is a Fredholm equation of the second kind, and can be discretized by the Galerkin method. We want to compute an approximation $\tilde{B}$ of B in a finite-dimensional function space spanned by an orthogonal basis of functions $(u_i)_{1 \leq i \leq N}$. Therefore, we can express $\tilde{B}$ as a linear combination of the u_i:

$$\tilde{B} = \sum_{j=1}^{N} B_j u_j .$$

The Galerkin condition [17] defines the approximation $\tilde{B}$ so that the residual function

$$r(X) = \tilde{B}(X) - E(X) - \int_{Y \in (\mathcal{S} \times \mathcal{T})} \tilde{B}(Y) \mathcal{K}(X, Y) \, \mathrm{d}Y$$

is orthogonal to all the u_i. In such a case, the coefficients B_j that define $\tilde{B}$ are solutions of the following linear system:

$$(I - M)B = E, \tag{4}$$

where I is the identity matrix, the vector E is defined by

$$\forall i \in [1, N] E_i = \frac{\langle E, u_i \rangle}{\|u_i\|^2}$$

and the matrix coefficients are defined by:

$$\forall (i, j) \in [1, N]^2 M_{i,j} = \frac{\langle \int_Y \mathcal{K}(., Y) u_j(Y) \, \mathrm{d}Y, u_i \rangle}{\|u_i\|^2} \tag{5}$$

The simplest possible choice of a function basis is piecewise constant functions. As discussed in Section 2.2, this choice proves unsatisfying in certain cases, where it causes noticeable temporal discontinuities in indirect lighting. As a consequence, we propose instead functions that are piecewise constant in space, and piecewise *polynomial* in time. To a given element k of our mesh, defined as the cross product of polygon P_k and time interval T_k, correspond L basis functions $u_{Lk+i}(p, t)$ with $0 \leq i < L$, equal to 0 when (p, t) is outside $(P_k \times T_k)$, and to $\Phi^i{}_k(t)$ otherwise. Since the u_i have to form an orthogonal basis, the $\Phi^i{}_k$ are the restriction of the first L Legendre polynomials to the time interval $T_k = [\alpha_k, \beta_k]$, *i.e.*

$$\Phi^0_k(t) = 1$$

$$\Phi^1_k(t) = \sqrt{3} \left(2 \frac{t - \alpha_k}{\beta_k - \alpha_k} - 1 \right)$$

$$\ldots$$

Therefore, the variations of radiosity of each element k in the mesh will be described by L unknown coefficients $B_{Lk}, \ldots, B_{Lk+L-1}$. Furthermore, from equation (5), it can be derived that each pair (k, l) of elements in our mesh corresponds to a $L \times L$ block $\rho_k I_{k,l}$ in the matrix M, where $I_{k,l}$ is the following *interaction matrix*:

$$I_{k,l} = \frac{1}{||P_k||(\beta_k - \alpha_k)} \int_{T_k \cap T_l} G_{k,l}(t) \int_{P_k} \int_{P_l} k(p, q, t) \, \mathrm{d}q \, \mathrm{d}p \, \mathrm{d}t \tag{6}$$

and the matrix $G_{k,l}$ is defined as:

$$G_{k,l}(t) = \begin{pmatrix} \Phi^0_k(t)\Phi^0_l(t) & \cdots & \Phi^0_k(t)\Phi^{L-1}_l(t) \\ \vdots & \ddots & \vdots \\ \Phi^{L-1}_k(t)\Phi^0_l(t) & \cdots & \Phi^{L-1}_k(t)\Phi^{L-1}_l(t) \end{pmatrix}.$$

The interaction matrix extends the traditional notion of form factor used in the classical static radiosity algorithms.

3.3. Hierarchical solution of the discrete equation

We are using piecewise polynomial functions to describe the variations of radiosity in time. Therefore, the resulting algorithm is an extension of the Wavelet Radiosity algorithm [21,22], using Haar basis over the spatial dimension and Alpert's $\mathcal{M}_L$ basis [23] over the time dimension.

Since our mesh elements are defined both by their geometry and their time interval, they can be subdivided either in

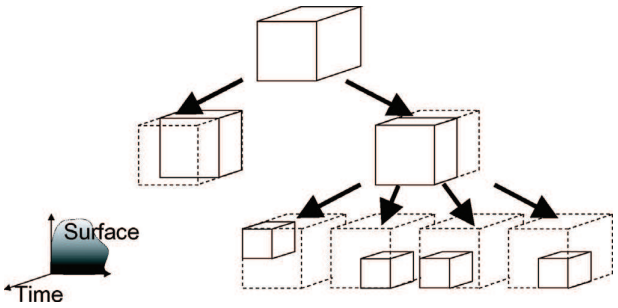


Figure 3: *A simple example of space-time hierarchy: here the root hierarchical element (represented as a space × time 3D volume) has been first subdivided in time. One of the resulting siblings has been subdivided in space.*

space (partitioning their geometry, e.g. using a quadtree subdivision scheme) or in time (subdividing the time range and leaving the geometry unchanged). As illustrated by Figure 3, repeated applications of either subdivision scheme build a data structure that offers a multi-resolution representation of the radiosity function over space and time. As in the original Hierarchical Radiosity algorithm, links joining two hierarchical elements are used to specify at which level of precision the light exchanges should be actually computed.

Consequently, the space-time hierarchical radiosity algorithm, similarly to the original hierarchical algorithm of Hanrahan *et al.* is an iteration composed of the following three steps:

(1) Recursive evaluation of the precision of all links to place them at the appropriate level in the hierarchy. The evaluation of the links is performed by a function named *refinement oracle*. As a side-effect, this recursive process adaptively builds the hierarchical mesh, refining the original mesh elements where more precision is needed. Our oracle is be discussed in Section 3.4.

(2) Gathering the light through the links in the hierarchy. Light exchanges computations are detailed in Section 3.5.

(3) Ensuring the coherence of radiosities between all hierarchical levels. This bidirectional traversal of the hierarchy is referred to as *Push–Pull* and is discussed in Section 3.6.

To fully benefit from the strength of the hierarchical formulation, it is necessary to extend the hierarchy of elements above the initial surfaces level, by hierarchically grouping together surfaces, and eventually groups of surfaces (called *clusters*). At the top of our hierarchy will be one *root cluster*, which will represent all surfaces, during the whole animation. The starting point of the algorithm will be the *root link* joining this cluster to itself, thereby representing all possible interactions between all surfaces in the scene, during the whole animation [16,17].

Hierarchical Radiosity algorithms with clustering for static scenes perform the construction of the cluster hierarchy as a preprocessing step. Such an approach cannot be used in our case, since the resulting spatial hierarchy would preexist the recursive link refinement procedure, preventing any temporal refinement until we reach the surfaces level. We propose instead a new approach, which we name *lazy clustering*. At the beginning of our algorithm, we only build the root cluster, plus one cluster for each different rigid motion in the animation. The rest of the cluster hierarchy is built as a by-product of the link refinement procedure. Therefore, as for surfaces elements in the space-time mesh, we can split clusters that have not been previously refined either in time or in space (see Figure 4):

- Time-refinement of a cluster can be performed by creating two clusters as children of the original one, each one defined over one half of the original time interval. Each surface inside the original cluster must be duplicated and one copy is assigned to each of the two children clusters.
- Space-refinement of a cluster is the act of grouping together some of the surfaces contained in the said cluster, forming new children clusters (which may be space or time split later on during the refinement process), possibly leaving some surfaces as direct children. This can be achieved in applying only one step of any classical top-down recursive clustering method (without the recursion). We chose to adapt Christensen's clustering method [24], which is straightforward to implement and produces a rather well-formed spatial hierarchy [25].

3.4. Space-time refinement oracle

The refinement oracle is the function in charge of evaluating the precision of a given link and decide if it is placed at the appropriate level in the hierarchy. In the case of the space-time hierarchical radiosity algorithm, this function has two goals:

- To decide whether a given link is a precise enough representation of the corresponding light exchange (like in classical Hierarchical Radiosity).
- If it is not precise enough, to determine whether it should be split in space or time.

A simple oracle, based on a comparison of the estimated time-variance and space-variance of the irradiance gathered across the link, always produces the same mesh, regardless of the function basis used. This is obviously not satisfying since piecewise polynomial functions should offer a better approximation of the radiosity than piecewise constant functions at the same subdivision depth.

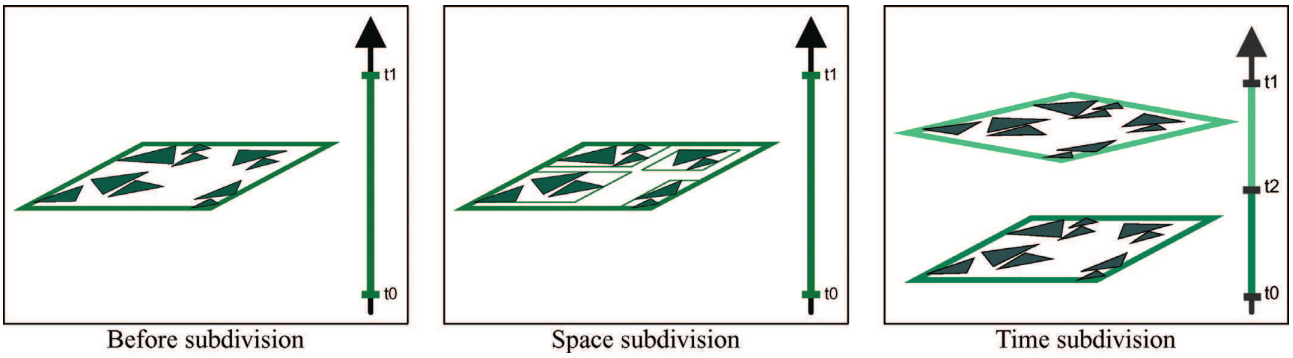


Figure 4: *The cluster in the left-hand image (represented in 2D for simplicity) can be subdivided either spatially (center), or temporally (on the right). Spatial subdivision builds one new hierarchical level of clusters around the surfaces. Temporal subdivision duplicates the surfaces and subdivides the corresponding time interval.*

We propose to extend for space-time radiosity an oracle designed for Wavelet Radiosity in the static case [18,19,26], based on estimates of the error on the propagated energy rather than on estimates of the variation of this energy. We use a grid of control points located on the receiving element, and a set of control points in time. On these control points, at the control times, we estimate the radiosity value using two methods:

(1) by multiplying the emitters' radiosity vector by the interaction matrix corresponding to the link, and then interpolating the radiosity values at the control times,

(2) by direct integration of the radiosity on the emitter, using a quadrature.

The difference between these two values is an indication of the error made when evaluating the interaction at this point for this level of precision. The norm of these differences is used as the error on the current interaction. Refinement will occur if this norm is above the refinement threshold set by the user.

The control points and times must be carefully chosen so that they provide meaningful information. They must be different from the quadrature points and times used for the form factor computations. The number of control points and times must be higher for large receivers so that we do not miss important features. Also, placing control times at the beginning and at the end of the time interval greatly enhances temporal continuity.

Once we have made the decision to refine an interaction, we must choose between refinement in space or in time. We compute two variance estimates for the set of estimated error values on our grid of control points and times:

- An average *spatial variance*: for each fixed control time we compute the variance of error values at each control points, and then take the temporal average.
- An average *temporal variance*: for each fixed control point we compute the variance of error values at each control times, and then take the spatial average.

We refine the interaction in time if the average temporal variance is above the average spatial variance, and in space otherwise.

3.5. Light exchanges computations

Computing the light exchanged between two linked surfaces is straightforward. The product of the link's interaction matrix by the radiosity of the emitter is added to the radiosity of the receiver. The interaction matrix has generally been computed previously during the refinement procedure using simple Gaussian quadratures.

However, interactions involving one or more clusters require a special approach, based on the one described by Sillion [27]. Roughly, anisotropic emission from a cluster is approximated by going down to the surfaces level to estimate the directional radiant intensity exiting the cluster (*Delayed Pull*), and the irradiance gathered by a cluster from a given hierarchical element is distributed to all the surfaces inside the cluster immediately at gathering time according to their orientation (*Immediate Push*). The specificities of the space-time hierarchical radiosity method come from the fact that the position, orientation and radiosity of the objects can change with time.

3.5.1. Emission from a cluster: Delayed pull

In the classical hierarchical radiosity algorithm, the computation of the light emitted from an object involves the computation of the form factor between the sender l and the receiver k. It is very difficult to define what the form factor should be if the sender is an anisotropic cluster. Therefore, we directly compute the irradiance emitted by the cluster to the receiver, by summing the contributions of the N surfaces contained in

the cluster l. At a given time t, point p receives from the N elements i in l the total irradiance:

$$I_{received}(p,t) = \sum_{i=1}^{N} \int_{Q_i} B_i(q,t) g(p,q,t) v(p,q,t) \mathrm{d}q,$$

where the geometric configuration function is defined by:

$$g(x,y,t) = \frac{\mathcal{R}(t)\cos\theta'}{\pi r^2}$$

and the $\mathcal{R}$ function is the receiver factor defined in [27] as $\cos\theta$ if the receiver is a surface and 1 if the receiver is a cluster (the surfaces orientation in the receiving cluster will be taken into account by the Immediate Push mechanism described in Section 3.5.2).

We approximate the received irradiance by projecting it on our function basis: The resulting approximation is a linear combination of our L basis functions:

$$\tilde{I} = \sum_{j=0}^{L-1} \lambda_j u_{Lk+j}.$$

Since the u_i are orthogonal we have:

$$\lambda_j = \frac{1}{||u_{Lk+j}||^2} \langle I_{received}, u_{Lk+j} \rangle = \frac{\sum_i \left[\int_{T_k \cap T_i} \left(\int_{P_k} \int_{Q_i} g(p,q,t) v(p,q,t) \mathrm{d}q \mathrm{d}p \right) B_i(t) \Phi_k^j(t) \mathrm{d}t \right]}{A_k(\beta_k - \alpha_k)}.$$

Computing the above integral is costly as it involves a number of visibility estimations proportional to the number of surfaces in the cluster. Therefore we approximate it by factoring out the visibility, and average it over the sending cluster:

$$\lambda_j = \frac{\sum_i \left[\int_{T_k \cap T_i} \left(\int_{P_k} \int_{Q_i} g(p,q,t) \mathrm{d}q \mathrm{d}p \right) B_i(t) \Phi_k^j(t) \tilde{V}(t) \mathrm{d}t \right]}{A_k(\beta_k - \alpha_k)},$$

where

$$\tilde{V}(t) = \frac{1}{A_k A_l} \int_{P_k} \int_{l} v(x,y,t)\, \mathrm{d}x\, \mathrm{d}y.$$

We compute the α_j and $\tilde{V}(t)$ using a Gaussian quadrature. Since the cost of evaluating the approximate visibility must not depend on the number N of surfaces inside the cluster l we place the quadrature points independently of the surfaces positions inside the cluster's bounding box.

3.5.2. Reception inside a cluster: Immediate push

The reception inside a cluster obeys the *immediate push* principle: the irradiance received at the cluster level is immediately dispatched to all surfaces inside the cluster, where it is multiplied by the cosine of the angle between the normal of the surface and the direction of the incoming radiance. The origin of the incoming radiance is assumed to be the center of the emitter, whether a cluster or a surface.

Since both the cluster and the sender may be moving, the receiver factor is time-dependent. We need to project it on our wavelet basis. Let us assume a cluster k has received an irradiance $I_{received}$. This irradiance is distributed to each surface i in the cluster k according to its orientation:

$$I_i = I_{received}(t) \cos\theta_i(t)$$

I_i is then reprojected on the wavelet basis for the time interval T_i over which the hierarchical element i is defined. The resulting approximate irradiance is then:

$$\tilde{I}_i = \sum_{j=0}^{L-1} \gamma_1 \Phi_i^j$$

and the γ coefficients are:

$$\gamma_j = \frac{1}{||\Phi_i^j||^2} \langle I_i | \Phi_i^j \rangle = \frac{1}{\beta_i - \alpha_i} \int_{T_i \cap T_l} I_{received}(t) \cos\theta_i(t) \Phi_i^j(t)\, \mathrm{d}t.$$

These integrals are once again approximated using a Gaussian quadrature.

Our method contains two successive approximations: we have separately computed the irradiance received at the cluster level, which was time-dependent, projected it onto the function basis, then dispatched it to the surfaces, taking into account the surface movement, and reprojected it on the function basis for the receiving surface. This double approximation is consistent with the clustering approach. If the refinement oracle decides that we can compute an interaction at the cluster level, then this approximation should be sufficient. Spending more computation time to find a better approximation would impair the hierarchical nature of the algorithm and would reduce its performance.

3.6. Push–pull traversal

After the irradiances have been gathered across all links in the scene, a traversal of the complete hierarchy is necessary to maintain coherence between the different hierarchical levels. First, irradiance contributions computed at various level of the hierarchy have to be pushed down to the lowest level of the structure and summed along the way. Here the radiosities of each leaf are computed, and these radiosities are then progressively pulled up the hierarchy and averaged to compute the correct radiosity representation corresponding to each hierarchical level.

In the case of Wavelet Radiosity [21], this process is slightly more complicated than it is for static Hierarchical Radiosity since we need to define how to combine the coefficients describing the radiosity variations, to convert them from one hierarchical level to the other. Remember from Section 3.2 that our multi-resolution basis functions are cross products of the scale functions of the Haar basis over space, and scale functions of the $\mathcal{M}_L$ basis over time. Since we use

a very simple midpoint subdivision scheme when subdividing elements in time, the coefficients that have to be pushed down or pulled up during this traversal can be computed using simple linear transformations, which are independent of the element or the hierarchical level. Both linear transforms are referred to as the *two scale relationship* [22], and are determined by two $L \times L$ matrices P and Q. When $L = 2$ (linear wavelets), those matrices are:

$$P = \begin{pmatrix} 1 & 0 \\ -\frac{\sqrt{3}}{2} & \frac{1}{2} \end{pmatrix} \quad Q = \begin{pmatrix} 1 & 0 \\ \frac{\sqrt{3}}{2} & \frac{1}{2} \end{pmatrix}.$$

When pushing down the total irradiance I (remember that this is a L-dimensional vector) from a given element split in time to each of its two children, the corresponding irradiances I' and I'' to be transmitted to its first and second children are given by the following linear transform:

$$I' = {}^tPI \qquad \text{and} \qquad I'' = {}^tQI.$$

Respectively when pulling up, the average radiosity B of an element can be computed from the radiosities of its two children B' and B'':

$$B = \frac{1}{2}(PB' + QB'')$$

3.7. Practical issues and choices

3.7.1. Choice of a space-time function basis.

Higher-order wavelets have been previously used as function basis for the representation of radiosity [21,22]. However, the algorithms resulting from their straightforward use within the classical Hierarchical Radiosity framework were proved impractical, slower than when using classical piecewise constant function basis and giving poorer results [28]. Further research [26] has later shown that by making use of several recent advances in the field [18,19,26,29] higher-order wavelets were providing a better approximation of the illumination function, requiring less memory and computation time. In particular, the radiosity function produced looks continuous without postprocessing thanks to an adequate refinement oracle.

Unfortunately, to this date, higher-order wavelets in the *spatial* dimension cannot be easily applied to cluster objects. This is due to the fact that it is difficult to provide a function mapping the surfaces contained in a cluster onto the square domain where the wavelet basis is defined. Therefore, in order to be able to use higher-order wavelets for surfaces and clusters for inexpensive approximations, a mechanism to use different approximation order for different hierarchical elements should be defined. Our refinement oracle would automatically adapt to the new function basis. The only point needing change would be the Push–Pull matrices we gave in Section 3.6.

However, this problem does not arise in the *temporal* dimension. Moreover, the lower dimensionality makes the added cost of the use of wavelets lower in the temporal dimension than it is in the spatial dimension. Therefore, we decided to limit our use of wavelets to the description of the temporal variations of radiosity. In our implementation, our function basis was composed of linearly varying functions (the $\mathcal{M}_2$ basis). This choice proved sufficient in practice to significantly reduce the temporal discontinuities (see Section 4).

In order to provide a smooth appearance for patches in our example animations, we applied a simple linear interpolation over the polygons as a postprocess, when traversing the space-time mesh to generate the images. Though it noticeably increases the visual appeal of the results, this postprocess doesn't improve the precision of the solution. Much better reconstruction methods have been proposed for static scenes, such as *final gathering* [30,31], and can be applied here on an image per image basis. Moreover, Martin *et al.* have proposed recently a final gathering acceleration method for animated scenes [15], whose coupling with our approach seems promising.

3.7.2. Memory management issues and refinement ordering.

As our experiments will show in Section 4, the space-time hierarchical radiosity algorithm is quite memory intensive. This is due to the fact that we keep in memory a complete view independent solution describing the variations of radiosity for all surfaces during the whole animation time interval. Though the amount of memory available on graphic workstations is rapidly increasing, it may prove necessary to reduce our algorithm requirements when running on less powerful machines or when computing long animation sequences.

One way of reducing the memory cost is to use hard-disk space to cache parts of the hierarchy that are temporarily not required for computations. For the algorithm to remain efficient when such a caching scheme is used, accesses to the disk should be limited to a small number of large files. The order according to which we traverse the space-time hierarchy during the refinement should be chosen accordingly.

Such a desirable traversal order can be derived if we recall from equation (6) that elements whose time interval do not overlap cannot exchange energy. Since we always subdivide time intervals in two parts of equal length, for every element P that can interact with a given element Q, we know that either $T_P \subset T_Q$ or $T_Q \subset T_P$. Therefore, we can easily determine which element may be needed to compute or refine a certain interaction, by bucketing hierarchical elements and links according to their time interval.

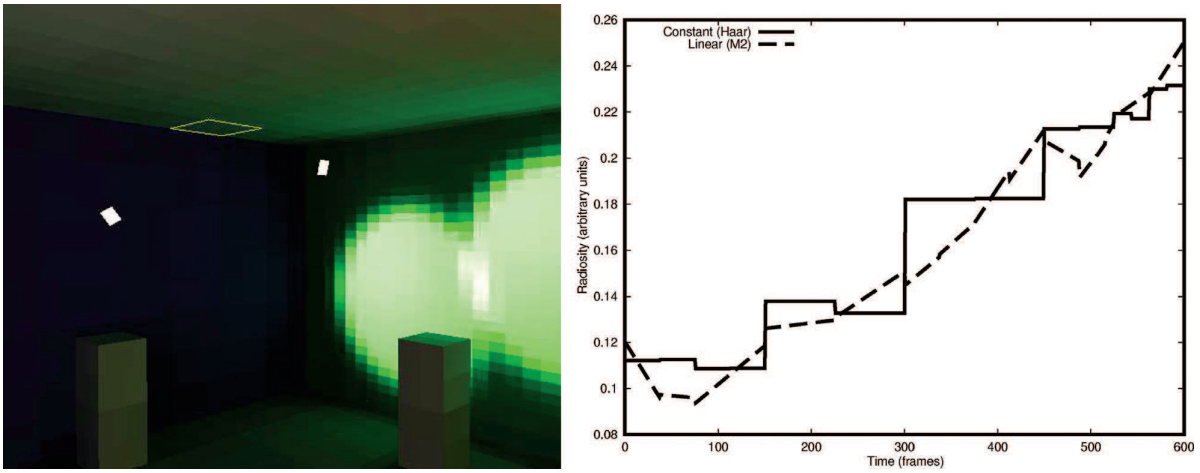


Figure 5: *Variation of the radiosity function at the center of the highlighted element during the animation.*

The refinement can be performed as a traversal of the hierarchy that would correspond to a depth-first-order traversal of the time intervals binary tree. Only the elements whose time interval contain the one currently visited should be kept in memory. Disk access would only take place when moving from one time interval to the other.

We ran an experiment to estimate the corresponding gain in memory that could be expected from such a traversal. For the SPOT scene (see Sections 2.2 and 4.1), we used $2^5 - 1 = 31$ time interval buckets to sort our elements (one for each time interval corresponding to the first five subdivision level). The maximum total cost of the portion of the hierarchy that needs to be kept in memory is 40 MB whereas more than 450 MB are required when we are keeping everything in RAM (*cf.* Table 1). In such a case, file accesses should not reduce excessively the performances of our algorithm: The added cost of reading and writing 31 files each about 15 MB big should be reasonable since an iteration on this scene already requires 10–20 minutes.

4. Experimental Results

In this section, we discuss the performance of our algorithm on several test scenes. In Section 4.1, we demonstrate the improvement on temporal continuity that our use of piecewise linearly varying functions offers when compared to simple "box" functions. In Section 4.2, we demonstrate our use of clustering and we offer comparisons with frame-by-frame Hierarchical Radiosity calculations to show that we obtain good acceleration factors.

4.1. Improvement of temporal continuity

To illustrate the improvement in temporal continuity obtained using the $\mathcal{M}_2$ function basis in the time dimension, we use the test scene from Section 2.2. In this scene, the sweeping movement of spotlights over walls painted in different colors cause important changes in the indirect illumination of the scene. In particular, strong color bleeding effects can be observed moving on the ceiling and the floor of the scene.

As explained in Section 2.2, this scene was purposedly designed as a "worst-case scenario" for the space-time hierarchical radiosity algorithm in order to exhibit strong temporal discontinuities. When using a piecewise constant function basis to describe the variation of radiosity in time, the indirect lighting effects in this scene are extremely discontinuous. For example, the color bleeding patches seem to be updated only every second or so. The amplitude of these discontinuities is shown in the radiosity variation plot of Figure 5. It can be clearly seen that the greatest discontinuity is located at the middle of the animation, then at the first and third quarter. The magnitude of the largest discontinuity is about 40% of the time-average radiosity of this patch, which makes it quite noticeable. Smaller discontinuities can be observed at other even subdivisions of the time interval.

However, the same animation, computed using hierarchical elements linearly varying in time, exhibits a much more coherent indirect illumination. Without paying careful attention, it is difficult to perceive any discontinuity. We can see on Figure 5 that the indirect lighting is obviously more "continuous" than when using a piecewise constant function basis. The largest discontinuity (at $t = 1/4$) has a magnitude of about 7% of the average radiosity of the patch. Figure 6 shows that this strong reduction of discontinuities can be observed on all surfaces of the scene. Table 1 allows the comparison of computation time and memory cost when computing this animation with a frame-by-frame Hierarchical Radiosity algorithm, with our algorithm using the Haar basis, and with our algorithm using the $\mathcal{M}_2$ basis, respectively. All timings have been measured on a single 300Mhz MIPS R12000 processor of an SGI Onyx2 computer.

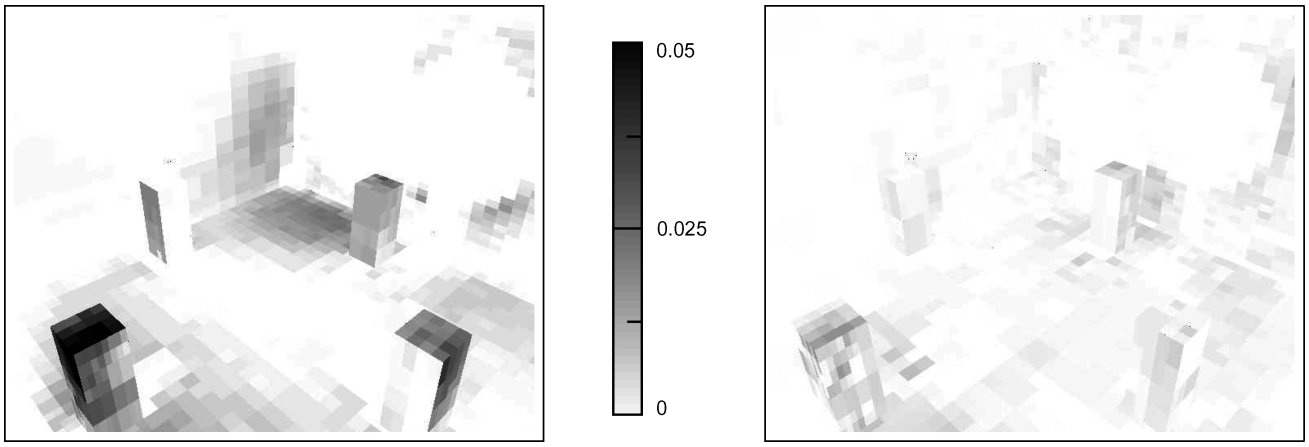


Figure 6: *Comparison of temporal discontinuities at $t = \frac{1}{2}$. Darker colors indicate a higher discontinuity (arbitrary units). Left: hierarchical radiosity, right: $\mathcal{M}_2$ wavelets.*

Table 1: *Performance comparison on the* SPOTS *scene, between frame-by-frame Hierarchical Radiosity, our algorithm using the Haar basis, and our algorithm using the $\mathcal{M}_2$ basis*

	Computation Time			
	Direct Lighting	Indirect Lighting	Total per Image	Memory Used (MB)
Frame-by-Frame HR	1 s × 600 = 600 s	15 s × 600 = 9, 000 s	16.0 s	5
Haar	254 s	1,492 s	2.9 s	587
$\mathcal{M}_2$	271 s	1,172 s	2.4 s	464

The following comments can be made about these results:

- Though this scene is geometrically quite simple, the speedup factor obtained, when compared to a frame-by-frame computation, is about 6. (Note that this acceleration factor is only about 2 if we only take into account the time needed to compute the direct illumination). Our algorithm performance on such scenes where the indirect lighting is dominant and dramatically changing over time is therefore satisfying.
- The memory consumption when using the $\mathcal{M}_2$ basis is 15% lower than when using the Haar basis, in spite of the added storage cost of the second radiosity coefficient and the interaction matrices. The animation has also been computed slightly faster. This is due to the fact that fewer subdivisions in time are needed to obtain a precise enough representation of the variations of radiosity in time, resulting in a faster refinement and a lighter mesh.

4.2. Validation of the Clustering Approach

We have tested our algorithm on scenes composed of several thousands of input polygons (see Figure 7). For such scenes, Hierarchical Radiosity computations without the use of clustering would have been extremely long because of the quadratic cost of the initial linking stage.

The first of our three test animations takes place in a small room with some furniture (a couple desks, chairs, pens, *etc.*). It is lit by four area light sources. The bookshelf, against the wall, falls to the floor. The animation is 4 seconds long, and is composed of 100 frames. The input geometry is composed of 7, 200 polygons. The second animation takes place in a large library hall with several desks separated by rows of bookshelves. This scene is lit by numerous area light sources. A character is moving through the hall. The animation is 20 seconds long and is composed of 500 frames. There are about 35 000 input surfaces. The third animation is somehow similar to the test scene we use in Section 4.1. We replaced the boxes by more complex objects. The resulting scene is composed of approximately 30 000 polygons and is 24 seconds long.

Table 2 summarizes our experimental results for our three test scenes. In this table, we compare the resources necessary to compute the animations when using the space-time hierarchical radiosity algorithm and when performing a hierarchical radiosity with Clustering frame-by-frame providing the same image quality. All timings have been observed on a 300 MHz MIPS R12000.

The more elements the mesh is composed of, the more the hierarchical approach is advantageous. Since it makes it possible to compute more complicated animations, clustering really allows us to benefit fully from the hierarchical nature of

Figure 7: *Sample frames from our test animations.*

Table 2: *Comparative results for the use of clustering: we compare computation time and memory use of our algorithm to the time and memory needed to compute the same animation frame-by-frame with classical Hierarchical Radiosity with Clustering*

	Computation Time		Memory Used	
	STHR	Static HRC	STHR (MB)	Static HRC (MB)
SHELF	3,335 s	184 s × 100 = 18, 400 s	100	16
HALL	33,333 s	1.185 s × 500 = 592, 500 s	842	120
ROBOTS	4,591 s	109 s × 600 = 65, 400 s	475	17

our algorithm. The typical speedup is ranging from 6 to 18.

The memory consumption of our algorithm is quite high, since we keep in memory at the same time a complete view independent global illumination solution for all frames of the animation (we have discussed a possible way to avoid this in Section 3.7.2). However, we can note that the memory cost of our algorithm depends more on the complexity of the illumination than on the number of input polygons. The more complex the input mesh is, the smaller the polygons are on average. Therefore, they are less likely to be subdivided later, and the resulting hierarchy will not be much bigger than if it consisted initially of large unsubdivided surfaces.

5. Conclusion

In this paper, we proposed a new algorithm to compute global illumination in diffuse animated environments. This algorithm is based on the adaptive refinement of a hierarchical mesh defined both over time and space. It can therefore benefit from the *a priori* knowledge of objects movements to factor out a large part of redundant computations.

This technique allows computation of animations with a quality similar to frame-by-frame computation, in a shorter time. Geometrically complex scenes can be dealt with thanks to the definition of a clustering approach extending the space-time mesh. The continuity of indirect lighting is improved by

the simultaneous use of a piecewise-linear wavelet basis in the time dimension and of an adequate space-time refinement oracle.

Promising directions for future research include:

- The derivation of a space-time final gathering approach, adapting the one proposed by Martin, Pueyo and Tost [15].
- The implementation and extensive testing of disk caching schemes such as the one suggested in Section 3.7.2.
- The parallelization of this algorithm. This should be straightforward on a shared memory architecture [32] but will certainly prove more difficult on a cluster of PC.
- Experiments with alternative wavelet basis in the time dimension, for example using higher-order polynomials.
- The extension of our algorithm to nondiffuse scenes, using a unified mesh-based particle shooting approach [9].
- The construction of a refinement criteria using human perception-based animation quality metrics [13].

References

1. P. Hanrahan, D. Salzman and L. Aupperle. A rapid hierarchical radiosity algorithm. In *Computer Graphics (ACM SIGGRAPH '91 Proceedings*), vol. 25, pp. 197–206. 1991.

2. C. Damez and F. Sillion. Space-time hierarchical radiosity. In *Proceedings of the 10th Eurographics Workshop on Rendering*, pp. 235–246. 1999.

3. C. M. Goral, K. E. Torrance, D. P. Greenberg and B. Battaile. Modelling the interaction of light between diffuse surfaces. In *Computer Graphics (ACM SIGGRAPH '84 Proceedings*), vol. 18, pp. 212–222. 1984.

4. E. Shaw. Hierarchical radiosity for dynamic environments. *Computer Graphics Forum*, 16(2):107–118, 1997.

5. G. Drettakis and F. Sillion. Interactive update of global illumination using a line-space hierarchy. In *Computer Graphics (ACM SIGGRAPH '97 Proceedings*), pp. 57–64. 1997.

6. B. Walter, G. Drettakis and S. Parker. Interactive rendering using the render cache. In *Proceedings of the 10th Eurographics Workshop on Rendering*, pp. 235–246. 1999.

7. P. Tole, F. Pellaccini, B. Walter and D. P. Greenberg. Interactive global illumination in dynamic scenes. *ACM Transactions on Graphics (SIGGRAPH '02 Proceedings*), vol. 21(3), pp. 537–546. 2002.

8. A. Keller. Instant radiosity. In *Computer Graphics (ACM SIGGRAPH '97 Proceedings*), pp. 49–56. 1997.

9. X. Granier and G. Drettakis. Incremental updates for rapid glossy global illumination. In *Computer Graphics Forum (Proceedings of Eurographics 2001*), vol. 20, pp. 268–277. 2001.

10. I. Wald, T. Kollig, C. Benthin, A. Keller and P. Slusallek. Interactive global illumination. In *Proceedings of the 13th Eurographics Workshop on Rendering*. 2002.

11. K. Dmitriev, S. Brabec, K. Myszkowski and H.-P. Seidel. Interactive global illumination using selective photon tracing. In *Proceedings of the 13th Eurographics Workshop on Rendering*. 2002.

12. G. Besuievsky and X.Pueyo. Animating radiosity environments through the multi-frame lighting method. *Journal of Visualization and Computer Animation*, 12:93–106, 2001.

13. K. Myszkowski, T. Tawara, H. Akamine and H.-P. Seidel. Perception-guided global illumination solution for animation rendering. In *Computer Graphics (ACM SIGGRAPH '01 Proceedings*), pp. 221–230. 2001.

14. C. Damez, K. Dmitriev and K. Myszkowski. State of the art in global illumination for interactive applications and high-quality animations. *Computer Graphics Forum*, 22(1):55–77, 2003.

15. I. Martín, X. Pueyo and D. Tost. Frame-to-frame coherent animation with two-pass radiosity. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):70–84, 2003.

16. B. Smits, J. Arvo and D. Greenberg. A clustering algorithm for radiosity in complex environments. In *Computer Graphics (ACM SIGGRAPH '94 Proceedings*), pp. 435–442. 1994.

17. F. Sillion. Clustering and volume scattering for hierarchical radiosity calculations. In *Proceedings of the 5th Eurographics Workshop on Rendering*, pp. 105–117. 1994.

18. P. Bekaert and Y. D. Willems. Error control for radiosity. In *Proceedings of the 7th Eurographics Workshop on Rendering*, pp. 153–164. 1996.

19. P. Bekaert and Y. D. Willems. Hirad: A hierarchical higher order radiosity implementation. In *Proceedings of the Twelfth Spring Conference on Computer Graphics (SCCG '96*), Bratislava, Slovakia, Comenius University Press, June 1996.

20. M. F. Cohen and J. R. Wallace. *Radiosity and Realistic Image Synthesis*. Academic Press Professional, Boston, MA, 1993.

21. S. J. Gortler, P. Schroder, M. F. Cohen and P. Hanrahan. Wavelet radiosity. In *Computer Graphics (ACM SIGGRAPH '93 Proceedings*), pp. 221–230. 1993.

22. P. Schroder, S. J. Gortler, M. F. Cohen and P. Hanrahan. Wavelet projections for radiosity. In *Fourth Eurographics Workshop on Rendering*, pp. 105–114. 1993.

23. B. K. Alpert. A class of bases in L^2 for the sparse representation of integral operators. *SIAM Journal on Mathematical Analysis*, 24(1):246–262, 1993.

24. P. H. Christensen, D. Lischinski, E. J. Stollnitz and D. H. Salesin. Clustering for glossy global illumination. *ACM Transactions on Graphics*, 16(1):3–33, 1997.

25. J.-M. Hasenfratz, C. Damez, F. Sillion and G. Drettakis. A practical analysis of clustering strategies for hierarchical radiosity. In *Computer Graphics Forum (Proc. Eurographics '99*), vol. 18, pp. C221–C232. Sept. 1999.

26. F. Cuny, L. Alonso and N. Holzschuch. A novel approach makes higher order wavelets really efficient for radiosity. In *Computer Graphics Forum (Proc. Eurographics 2000*), vol. 19, pp. C99–C108. 2000.

27. F. Sillion. A unified hierarchical algorithm for global illumination with scattering volumes and object clusters. *IEEE Transactions on Visualization and Computer Graphics*, 1(3): 240–254, 1995.

28. A. Willmott and P. Heckbert. An empirical comparison of progressive and wavelet radiosity. In J. Dorsey and P. Slusallek. (eds), *Rendering Techniques '97 (Proceedings of the Eighth Eurographics Workshop on Rendering*), New York, NY, pp. 175–186, Springer, Wien. 1997. ISBN 3-211-83001-4.

29. M. Stamminger, H. Schirmacher, P. Slusallek and H.-P. Seidel. Getting rid of links in hierarchical radiosity. *Computer Graphics Journal (Proc. Eurographics '98*), 17(3), C165–C174, 1998.

30. M. Reichert. *A Two-Pass Radiosity Method to Transmitting and Specularly Reflecting Surfaces*, M.Sc. thesis, Cornell University, 1992.

31. A. Scheel, M. Stamminger and H.-P. Seidel. Grid based final gather for radiosity on complex clustered scenes. *Computer Graphics Forum*, 21(3): 547–556, 2002.

32. F. Sillion and J.-M. Hasenfratz. Efficient parallel refinement for hierarchical radiosity on a DSM computer. In *Proceedings of the Third Eurographics Workshop on Parallel Graphics and Visualisation*. pp. 61–74. 2000. http://www-imagis.imag.fr/Membres/Jean-M..Hasenfratz/PUBLI/EGWPGV00.html.

2.7.7 Accurate detection of symmetries in 3D shapes (TOG 2006)

Auteurs : Aurélien M , Cyril S , Nicolas H et François S
Journal : *ACM Transactions on Graphics*, vol. 25, n° 2.
Date : avril 2006

Accurate Detection of Symmetries in 3D Shapes

AURÉLIEN MARTINET, CYRIL SOLER, NICOLAS HOLZSCHUCH and FRANÇOIS X. SILLION
ARTIS, INRIA Rhône-Alpes

We propose an automatic method for finding symmetries of 3D shapes, that is, isometric transforms which leave a shape globally unchanged. These symmetries are deterministically found through the use of an intermediate quantity: the generalized moments. By examining the extrema and spherical harmonic coefficients of these moments, we recover the parameters of the symmetries of the shape. The computation for large composite models is made efficient by using this information in an incremental algorithm capable of recovering the symmetries of a whole shape using the symmetries of its subparts. Applications of this work range from coherent remeshing of geometry with respect to the symmetries of a shape to geometric compression, intelligent mesh editing, and automatic instantiation.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Curve, surface, solid and object representations*

General Terms: Algorithms

1. INTRODUCTION

Many shapes and geometrical models exhibit *symmetries*: isometric transforms that leave the shape globally unchanged. Using symmetries, one can manipulate models more efficiently through coherent remeshing or intelligent mesh editing programs. Other potential applications include model compression, consistent texture-mapping, model completion, and automatic instantiation.

The symmetries of a model are sometimes made available by the creator of the model and represented explicitly in the file format the model is expressed in. Usually, however, this is not the case, and automatic translations between file formats commonly result in the loss of this information. For scanned models, symmetry information is also missing by nature.

In this article, we present an algorithm that automatically retrieves symmetries in a geometrical model. Our algorithm is independent of the tesselation of the model; in particular, it does not assume that the model has been tesselated in a manner consistent with the symmetries we attempt to identify, and it works well on noisy objects such as scanned models. Our algorithm uses a new tool, the *generalized moment* functions. Rather than computing these functions explicitly, we directly compute their spherical harmonic coefficients, using a fast and accurate technique. The extrema of these functions and their spherical harmonic coefficients enable us to deterministically recover the symmetries of a shape.

For composite shapes, that is, shapes built by assembling simpler structures, we optimize the computation by applying the first algorithm to the subparts, then iteratively building the set of symmetries of

Authors' addresses: A. Martinet, C. Soler, N. Holzschuch, F. X. Sillion, ARTIS, INRIA Rhône-Alpes, Saint Ismier, France; email: Aurelien. Martinet@imag.fr.

the composite shape, taking into account both the relative positions of the subparts and their relative orientations.

We envision many applications for our work, including geometric compression, consistent mesh editing, and automatic instantiation.

This article is organized as follows. In the following section, we review previous work on identifying geometric symmetries on 2D and 3D shapes. Then in Section 3, we present an overview of the symmetry-detection problem and the quantities used in our algorithms. In Section 4, we introduce the generalized moments and our method to compute them efficiently; in Section 5, we present our algorithm for identifying symmetries of a shape. The extension of this algorithm to composite shapes is then presented in Section 6., Finally, in Section 7, we show various applications of our algorithm.

2. RELATED WORK

Early approaches to symmetry detection focused on the 2D problem. Attalah [1985], Wolter et al. [1985] and Highnam [1985] present methods to reduce the 2D-symmetry detection problem to a 1D pattern matching problem for which efficient solution are known [Knuth et al. 1977]. Their algorithms efficiently detect all possible symmetries in a point set but are highly sensitive to noise.

Identifying symmetries for 3D models is much more complex, and little research on this subject has been published. Jiang and Bunke [1991] present a symmetry-detection method, restricted to rotational symmetry, based on a scheme called generate and test, first finding hypothetical symmetry axes, then verifying these assumptions. This method is based on a graph representation of a solid model and uses graph theory. The dependency between this graph representation and the mapping between points makes their method highly dependent on the topology of the mesh and sensitive to small modifications of the object geometry. Brass and Knauer [2004] provide a model for general 3D objects and give an algorithm to test congruence or symmetry for these objects. Their approach is capable of retrieving symmetry groups of an arbitrary shape but is also topology-dependent since it relies on a mapping between points of the model. Starting from an octree representation, Minovic et al. [1993] describe an algorithm based on octree traversal to identify symmetries of a 3D object. Their algorithm relies on PCA to find the candidate axis; PCA, however, fails to identify axes for a large class of objects, including highly symmetric objects such as regular solids.

All these methods try to find strict symmetries for 3D models. As a consequence, they are sensitive to noise and data imperfections. Zabrodsky et al. [1995] define a measure of symmetry for nonperfect models, defined as the minimum amount of work required to transform a shape into a symmetric shape. This method relies on the ability to first establish correspondence between points, a very restrictive precondition.

Sun and Sherrah [1997] use the *Extended Gaussian Image* to identify symmetries by looking at correlations in the Gaussian image. As in Minovic et al. [1993], they rely on PCA to identify potential axes of symmetry, thus possibly failing on highly symmetric objects. More recently, Kazhdan et al. [2004] introduced the *symmetry descriptors*, a collection of spherical functions that describe the measure of a model's rotational and reflective symmetry with respect to every axis passing through the center of mass. Their method provides good results in the shape identification but involves a surface integration for each sampled direction; this surface integration is carried on a voxel grid. Using the symmetry descriptors to identify symmetries requires an accurate sampling in all directions, making their algorithm very costly for an accurate set of results. In contrast, our algorithm only computes a deterministic small number of surface integrals, which are performed on the shape itself, and still provides very accurate results. Effective complexity comparisons will be given in Section 8.

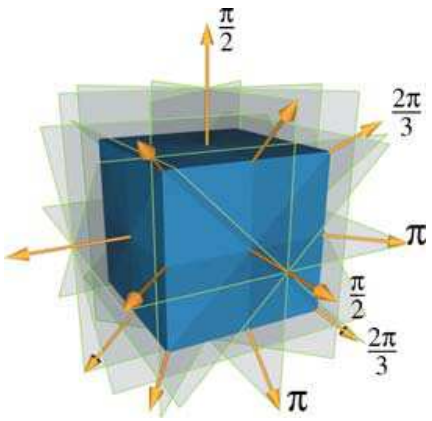


Fig. 1. Mirror symmetries and rotational symmetries found by our algorithm for a cube (for clarity, not all elements are represented).

3. OVERVIEW

Considering a surface $\mathcal{S}$, the *symmetries* of $\mathcal{S}$ are the isometric transforms which map $\mathcal{S}$ onto itself, in any coordinate system centered on its center of gravity. Symmetries of a shape form a group for the law of function composition with identity as its neutral element. For a given shape, the study of such a group relates to the domain of mathematical crystallography [Prince 2004].

The group of the cube, for instance, contains 48 elements (see Figure 1): the identity, eight 3−fold rotations around 4 possible axes, nine 4−fold rotations around 3 possible axes, six 2−fold rotations around 6 possible axes, nine mirror-symmetries, and fifteen other elements obtained by composing rotations and mirror symmetries.

Studying the group of isometries in $\mathbb{R}^3$ shows that, for a given isometry I, there always exists an orthonormal basis $(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z})$ into which the matrix of I takes the following form:

$$I(\lambda,\alpha) = \begin{pmatrix} \lambda & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \quad \text{with} \quad \begin{cases} \alpha \in [0, 2\pi[\\ \lambda = \pm 1 \end{cases}$$

As suggested by the example of the cube, this corresponds to 3 different classes of isometries: rotations, mirror symmetries, and their composition, depending whether λ is positive and/or $\alpha = 0 (\mathrm{mod}\ \pi)$. Finding a symmetry of a shape thus resolves into finding a vector $\boldsymbol{X}$ — which we call the *axis* of the isometry — and an angle α — which we call the *angle* of the isometry — such that $I(\lambda, \alpha)$ maps this shape onto itself.

However, finding all symmetries of a shape is much more difficult than simply checking whether a given transform actually is a symmetry. In particular, the naive approach that would consist of checking as many sampled values of $(\boldsymbol{X}, \lambda, \alpha)$ as possible to find a symmetry is far too costly. We thus need a deterministic method for finding good candidates.

Our approach to finding symmetries is to use intermediate functions, which set of symmetries is a superset of the set of symmetries of the shape itself, but for which computing the symmetries is much easier. By examining these functions, we will derive in Section 5 a deterministic algorithm which finds a finite number of possible candidates for $\boldsymbol{X}$, λ, and α. Because some unwanted triplets of values will appear during the process, these candidates are then checked back on the original shape. Choosing a family of functions which fulfill these requirements is easy. More difficult is the

task of finding such functions for which computing the symmetries can be done both accurately and efficiently.

Inspired by the work on principal component analysis [Minovic et al. 1993], we introduce the *generalized moment* functions of the shape for this purpose. These functions will be the topic of Section 4. These functions, indeed, have the same symmetries as the shape itself plus a small number of extra candidates. Furthermore, we propose an elegant framework based on spherical harmonics to accurately and efficiently find their symmetries.

A second contribution of this article is to extend the proposed algorithm into a constructive algorithm which separately computes the symmetries of subcomponents of an object—using the first method—, and then associates this information to compute symmetries of the whole composite shape. This constructive algorithm proves to be more accurate in some situations and more efficient when it is possible to decompose an object according to its symmetries. It is presented in Section 6.

4. GENERALIZED MOMENTS

In this section, we introduce a new class of functions: the generalized moments of a shape. We then show that these functions have at least the same symmetries as the shape itself and that their own symmetries can be computed in a very efficient way.

4.1 Definition

For a surface $\mathcal{S}$ in a 3-dimensional domain, we define its *generalized moment* of order $2p$ in direction $\boldsymbol{\omega}$ by

$$\mathcal{M}^{2p}(\boldsymbol{\omega}) = \int_{\boldsymbol{s}\in\mathcal{S}} \|\boldsymbol{s} \times \boldsymbol{\omega}\|^{2p} \, \mathrm{d}\boldsymbol{s}. \tag{1}$$

In this definition, $\boldsymbol{s}$ is a vector which links the center of gravity of the shape (placed at the origin) to a point on the surface, and $\mathrm{d}\boldsymbol{s}$ is thus an infinitesimal surface element. $\mathcal{M}^{2p}$ itself is a directional function.

It should be noted that, considering $\mathcal{S}$ to have some thickness dt, the expression $\mathcal{M}^2(\boldsymbol{\omega})dt$ (i.e., the generalized moment of order 2) corresponds to the moment of inertia of the thin shell $\mathcal{S}$ along $\boldsymbol{\omega}$, hence the name of these functions. Furthermore, the choice of an even exponent and a cross-product will lead to very interesting properties.

4.2 Shape Symmetries and Moments

Symmetry properties of a shape translate into symmetry properties of its moment functions. We now introduce a theorem that we will be rely on (see proof in Appendix):

THEOREM 1. *Any symmetry I of a shape $\mathcal{S}$ also is a symmetry of all its $\mathcal{M}^{2p}$ moment functions:*

$$I(\mathcal{S}) = \mathcal{S} \quad \Rightarrow \quad \forall \boldsymbol{\omega} \;\; \mathcal{M}^{2p}(I(\boldsymbol{\omega})) = \mathcal{M}^{2p}(\boldsymbol{\omega}).$$

Furthemore, if $\mathcal{M}^{2p}$ has a symmetry I with axis $\boldsymbol{\omega}$, then the gradient of $\mathcal{M}^{2p}$ is null at $\boldsymbol{\omega}$:

$$\forall \boldsymbol{\omega} \quad \mathcal{M}^{2p}(I(\boldsymbol{\omega})) = \mathcal{M}^{2p}(\boldsymbol{\omega}) \quad \Rightarrow \quad (\boldsymbol{\nabla}\mathcal{M}^{2p})(\boldsymbol{\omega}) = 0.$$

This theorem implies that the axes of the symmetries of a shape are to be found in the intersection of the sets of directions which zero the gradients of each of its moment functions. The properties are not reciprocal, however. Once the directions of the zeros of the gradients of the moment functions have been found, they must be checked on the shape itself to eliminate false positives.

4.3 Efficient Computation

At first sight, looking for the zeros of the gradient of the moment functions requires precise and dense sampling of these functions which would be very costly using their integral form of Equation (1). We thus present an efficient method to compute the generalized even moment functions of a shape, using spherical harmonics. In particular, we can accurately compute the spherical harmonic coefficients of the moment functions without sampling these functions. The search for zeros in the gradient will then be performed efficiently on the spherical harmonic decomposition itself.

Spherical Harmonics. We use real-valued spherical harmonics [Hobson 1931] to represent directional functions. Real spherical harmonics are defined, for integers $l \geq 0$ and $-l \leq m \leq l$, by:

$$Y_l^m(\theta,\varphi) = \begin{cases} \sqrt{2}\, N_l^m P_l^m(cos\theta)\cos(m\varphi) & \text{for } 0 < m \leq l \\ N_l^m P_l^0\ (cos\theta) & \text{for } m = 0 \\ \sqrt{2}\, N_l^m P_l^{-m}(cos\theta)\sin(m\varphi) & \text{for } -l \leq m < 0 \end{cases}$$

where P_l^m are the associated Legendre polynomials; the normalization constants N_l^m are such that the spherical harmonics form an orthonormal set of functions for the scalar product:

$$< f, g >= \int_{\|\omega\|=1} f(\omega) g(\omega)\, \mathrm{d}\omega.$$

This corresponds to choosing:

$$N_l^m = \sqrt{\frac{2l+1}{4\pi}\frac{(l-|m|)!}{(l+|m|)!}}.$$

We will use the following very powerful property of spherical harmonics. Any spherical harmonic of degree l can be expressed in a rotated coordinate system using harmonics of same degree and coefficients depending on the rotation R:

$$Y_l^m \circ R = \sum_{-l \leq m' \leq l} D_l^{m,m'}(R) Y_l^{m'}. \tag{2}$$

Any combination of spherical harmonics of degree less than l can therefore be expressed in a rotated coordinate system, using spherical harmonics of degree less than l, without loss of information. Coefficients $D_l^{m,m'}(R)$ can efficiently be obtained using recurrence formula [Ivanic and Ruedenberg 1996] or directly computed [Ramamoorthi and Hanrahan 2004].

Computation of Moment Functions. As defined by Equation (1), the $2p-$moment function of a shape $\mathcal{S}$ is expressed as:

$$\begin{aligned} \mathcal{M}^{2p}(\omega) &= \int_{\boldsymbol{s}\in\mathcal{S}} \|\boldsymbol{s}\times\omega\|^{2p}\, \mathrm{d}\boldsymbol{s} \\ &= \int_{\boldsymbol{s}\in\mathcal{S}} \|\boldsymbol{s}\|^{2p} \sin^{2p}\beta\, \mathrm{d}\boldsymbol{s} \end{aligned}$$

In this expression, β is the angle between s and ω.

Function $\beta \mapsto \sin^k \beta$ has angular dependence on β only and therefore decomposes into zonal harmonics (i.e., harmonics Y_l^m for which $m = 0$). Performing the calculation shows that, when k is even, the decomposition is finite. Setting $k = 2p$, we obtain :

$$\sin^{2p}\beta = \sum_{l=0}^{p} S_p^l Y_{2l}^0(\beta, .)$$

with:

$$S_p^l = \frac{\sqrt{(4l+1)\pi}}{2^{2l}} \sum_{k=l}^{2l} (-1)^k \frac{2^{2p+1} p!(2k)!(p+k-l)!}{(2(p+k-l)+1)!(k-l)!k!(2l-k)!}. \tag{3}$$

For the sake of completeness, we provide the corresponding derivation and the proof of the finite decomposition in the appendix section of this article.

Let R_s be a rotation which maps z, unit vector along $z-$axis, to s. Using Equation (2) for rotating the Y_{2l}^0 zonal harmonics, we have :

$$\sin^{2p}\beta = \sum_{l=0}^{p} S_p^l \sum_{m=-2l}^{2l} D_{2l}^{0,m}(R_s) Y_{2l}^m(\omega).$$

And finally:

$$\mathcal{M}^{2p}(\omega) = \sum_{l=0}^{p} \sum_{m=-2l}^{2l} C_{2l,m}^{2p} Y_{2l}^m(\omega), \tag{4}$$

using

$$C_{2l,m}^{2p} = S_p^l \int_{s\in\mathcal{S}} \|s\|^{2p} D_{2l}^{0,m}(R_s)\, \mathrm{d}s. \tag{5}$$

Equation (4) says that $\mathcal{M}^{2p}$ decomposes into a finite number of spherical harmonics, and Equation (5) allows us to directly compute the coefficients. The cost of computing $\mathcal{M}^{2p}$ is therefore $(p+1)(2p+1)$ surface integrals (one integral per even order of harmonic, up to order $2p$). This is much cheaper than the alternative method of computing the scalar product of $\mathcal{M}^{2p}$ as defined by Equation (1) with each spherical harmonic basis function: this would indeed require many evaluations of $\mathcal{M}^{2p}$, which itself is defined as a surface integral. Furthermore, numerical accuracy is only a concern when computing the $C_{2k,p}^m$ coefficients, and we can now compute both $\mathcal{M}^{2p}$ and its gradient analytically from Equation (4).

5. FINDING SYMMETRIES OF A SINGLE SHAPE

In this section, we present our algorithm for identifying symmetries of a shape seen as a single entity as opposed to the algorithm presented in the next section where the shape is considered as an aggregation of multiple subparts. For a given shape, we want to determine the axis $\boldsymbol{X}$ and the (λ, α) parameters of the potential isometries, using the generalized moment functions, and check the isometries found against the actual shape.

Central symmetries ($\lambda = -1$ and $\alpha = \pi$) form a specific case since, by construction, $\mathcal{M}^{2p}$ always has a central symmetry. Because central symmetries also do not require an axis, we treat this case directly while checking the other candidate symmetries on the shape itself in Section 5.3.

5.1 Determination of the Axis

As we saw in Section 4.2, the axis of isometries which let a shape globally unchanged also zero the gradient of the generalized even moments of this shape. We thus obtain a superset of them by solving for:

$$\boldsymbol{\nabla}(\mathcal{M}^{2p})(\omega) = \mathbf{0}.$$

In a first step, we estimate a number of vectors which are close to the actual solutions by refining the sphere of directions starting from an icosahedron. In each face, the value of $\|\boldsymbol{\nabla}(\mathcal{M}^{2p})(\omega)\|^2$ is examined

in several directions, and faces are sorted by order of the minimal value found. Only faces with small minimum values are refined recursively. The number of points to look at in each face, as well as the number of faces to keep at each depth level, are constant parameters of the algorithm.

In a second step, we perform a steepest descent minimization on $\|\nabla(\mathcal{M}^{2p})(\omega)\|^2$, starting from each of the candidates found during the first step. For this we need to evaluate the derivatives of $\|\nabla(\mathcal{M}^{2p})\|$ which we do using analytically computed second-order derivatives of the spherical harmonics along with Equation (4). The minimization converges in a few steps because starting positions are by nature very close to actual minima. This method has the double advantage that (1) the derivatives are very efficiently computed and (2) no approximation is contained into the calculation of the direction of the axis beyond the precision of the calculation of the $C^{2p}_{2l,m}$ coefficients.

During this process, multiple instances of the same direction can be found. We filter them out by estimating their relative distance. While nothing in theory prevents the first step from missing the area of attraction of a minimum, it works very well in the present context. Indeed, moment functions are very smooth, and shapes having two isometries with very close—yet different—axis are not common.

Finally, because all moment functions, whatever their order, must have an extremum in the direction of the axis of the symmetries of the shape, we compute such sets of directions for multiple moment functions (e.g., $\mathcal{M}^4$, $\mathcal{M}^6$ and $\mathcal{M}^8$) but keep only those which simultaneously zero the gradient of all these functions, which in practice leaves none or very few false positives to check for.

5.2 Determination of Rotation Parameters

After finding the zero directions for the gradient of the moment functions, we still need to find the parameters of the corresponding isometric transforms. This is done deterministically by studying the spherical harmonic coefficients of the moment functions themselves. We use the following properties.

PROPERTY 1. *A function has a mirror symmetry S_z around the $z = 0$ plane if and only if all its spherical harmonic coefficients for which $l + m$ is even are zero (i.e., it decomposes onto $z-$symmetric harmonics only). In the specific case of the moment functions:*

$$\forall \omega \;\; \mathcal{M}^{2p}(\omega) = \mathcal{M}^{2p}(S_z\omega) \;\; \Leftrightarrow \;\; m \equiv 0 (mod\ 2) \Rightarrow C^{2p}_{2l,m} = 0.$$

PROPERTY 2. *A function has a revolution symmetry around the z axis if and only if it decomposes onto zonal harmonics only, that is,*

$$\forall l \;\; \forall m \quad m \neq 0 \;\; \Rightarrow \;\; C^m_l = 0.$$

PROPERTY 3. *A function is self-similar through a rotation R_α of angle α around z if and only if all its spherical harmonic coefficients C^m_l verify:*

$$\forall l \;\; \forall m \quad C^m_l = \cos(m\alpha)C^m_l - \sin(m\alpha)C^{-m}_l. \tag{6}$$

Property 3 can be adapted to check if the function is self-similar through the composition of a rotation and a symmetry with the same axis (i.e., the case $\lambda = -1$ as defined in Section 3). In this case, the equation to be checked for is:

$$\forall l \;\; \forall m \quad (-1)^{l+m}C^m_l = \cos(m\alpha)C^m_l - \sin(m\alpha)C^{-m}_l. \tag{7}$$

These properties are easily derived from the very expression of the spherical harmonic functions [Hobson 1931].

Before using these properties, the moment function must be expressed in a coordinate system where the z axis coincides with the previously found candidate axis. This is performed using the rotation formula in Equation (2). Then checking for Properties 1 and 2 is trivial provided that some tolerance is accepted on the equalities. Using Property 3 is more subtle; coefficients of the function are first examined by order of decreasing m. For $\lambda = 1$, for instance, when the first nonzero value of C_l^m is found, Equation (6) is solved by:

$$\tan\frac{m\alpha}{2} = \frac{C_l^{-m}}{C_l^m}, \quad \text{that is,} \quad \alpha = \frac{2}{m}\arctan\left(\frac{C_l^{-m}}{C_l^m}\right) + \frac{k\pi}{m},$$

then all the remaining coefficients are checked with the obtained values of α. If the test passes, then α is the angle of an existing rotation symmetry for the moment function. A very similar process is used to search for α when $\lambda = -1$.

The error tolerance used when checking for Properties 1, 2, and 3 can be considered as a way of detecting approximate symmetries on objects. We will show in the results section that symmetries can indeed be detected on noisy data such as scanned models.

5.3 Filtering Results

The condition extracted from Theroem 1 is a necessary condition only. To avoid false positives, the directions and rotation angles obtained from the moment functions must therefore be verified on the shape itself. We do this using a *symmetry measure* inspired by the work of Zabrodsky et al. [1995]. Let $\mathcal{S}$ and $\mathcal{R}$ be two tessellated shapes. Let $V_\mathcal{S}$ and $V_\mathcal{R}$ be the mesh vertices of $\mathcal{S}$ and $\mathcal{R}$. We define the *measure* d_M between $\mathcal{S}$ and $\mathcal{R}$ by:

$$d_M(\mathcal{S},\mathcal{R}) = \max_{p\in V_\mathcal{S}}(\min_{q\in\mathcal{R}} \|p-q\|). \tag{8}$$

The *symmetric measure* $d_A(\mathcal{S})$ of a shape $\mathcal{S}$ with respect to a symmetry A is then defined by:

$$d_A(\mathcal{S}) = \max(d_M(\mathcal{S}, A\mathcal{S}), d_M(A\mathcal{S}, \mathcal{S})).$$

It should be noted that this definition is different from that of the *Hausdorff distance* since, in Equation (8), not all points of $\mathcal{S}$ are considered but only the mesh vertices, whereas all points of $\mathcal{R}$ are used. However, because $\mathcal{S}$ is polyhedral, $d_A(\mathcal{S}) = 0$ still implies that $A\mathcal{S} = \mathcal{S}$.

Computing d_A is costly, but fortunately we only compute it for a few choices of A which are the candidates we found at the previous step of the algorithm. This computation is much cheaper than computing a full symmetry descriptor [Kazhdan et al. 2004] for a sufficient number of directions to reach the precision of our symmetry detection algorithm.

5.4 Results

Complete Example. The whole process is illustrated in Figure 2. Starting from the original object (a), the moment functions of orders 4, 6, and 8 are computed (see, e.g., $\mathcal{M}^8$ in (b)). The gradients of these moments are then computed analytically (c) and used for finding the directions of the minima. The unfiltered set of directions contains 7 directions among which only 3 are common extrema of $\mathcal{M}^4$, $\mathcal{M}^6$, and $\mathcal{M}^8$. This set of 3 directions ($\boldsymbol{D}_1$,$\boldsymbol{D}_2$, and $\boldsymbol{D}_3$) must contain the axes of the symmetries of the shape. After checking the symmetry axis and parameters on the actual shape, $\boldsymbol{D}_1$ is revealed as the axis of a 2-fold symmetry which is the composition of the two remaining mirror symmetries of axes $\boldsymbol{D}_2$ and $\boldsymbol{D}_3$.

The example of the cube, shown in Figure 1, illustrates the extraction of rotations and mirror symmetries. Experiments have shown that our method finds all 48 symmetries whatever the coordinate system the cube is expressed in originaly.

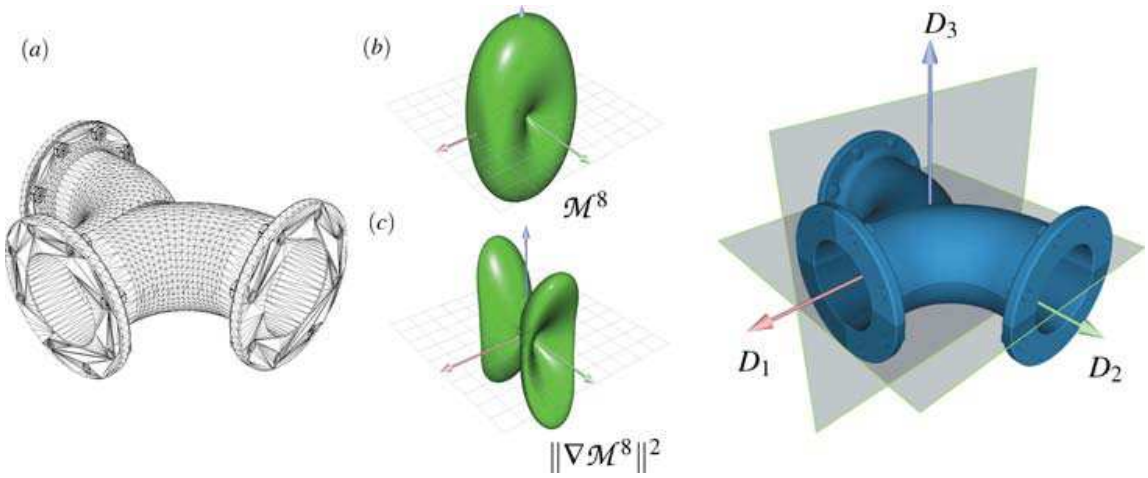


Fig. 2. Extraction of symmetries for a single shape. Starting from the original shape (a), generalized moments (b) and their gradients (c) are computed. The set of their common extrema directions contains the axes of the symmetries of the shape, depicted at right. Here, both mirror symmetries have been found as well as the 2-fold rotational symmetry. Note that the original shape is neither convex nor star-shaped and that the mesh is not consistent with the symmetries of the geometry.

Fig. 3. View of the three 3D models used in the robustness tests presented in Figure 4 shown with their symmetries. For the sake of clarity, we chose models with only one symmetry each.

Robustness Tests. We now study the sensitivity of our method to small perturbations of the 3D model in two different ways.

(1) *Noise.* We randomly perturb each vertex of each polygon independently in the original model by a fraction of the longest length of the model's bounding box.

(2) *Delete.* We randomly delete a small number of polygons in the model.

We use a set of three models to test the robustness of our method. These model as well as their symmetry are shown in Figure 3. For the sake of clarity, we use objects with only one symmetry.

In order to test the robustness of the method, we progressively increase the magnitude of the noise and let the algorithm automatically detect the symmetry. In our robustness tests, we consider shapes as single entities and use the first algorithm presented in Section 5 to detect these symmetries. To evaluate

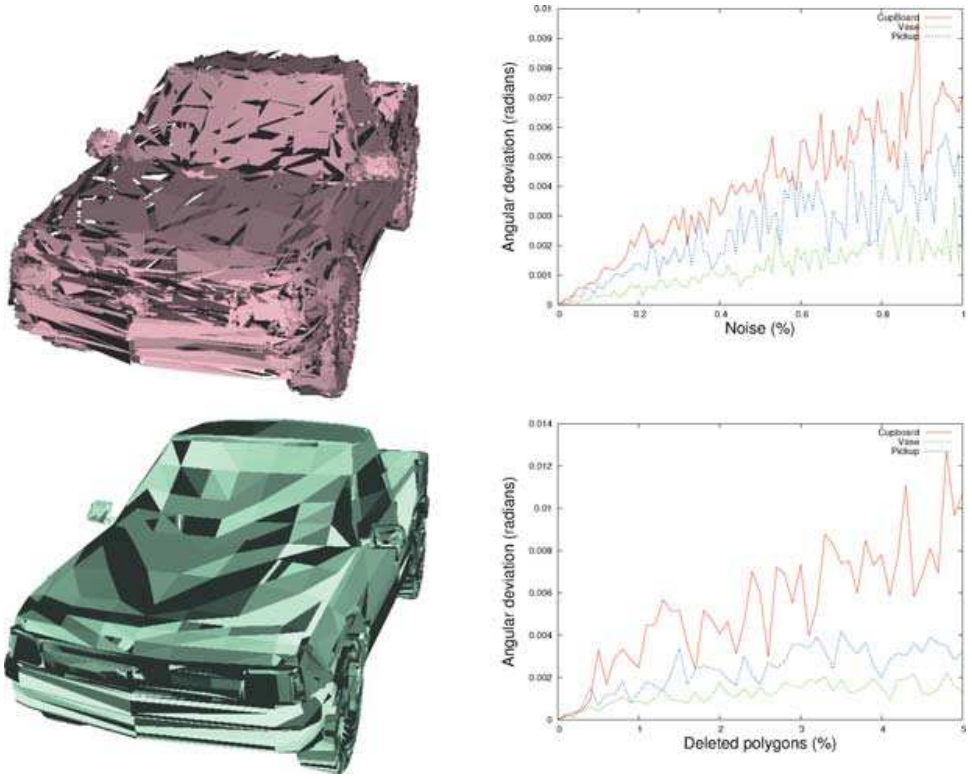


Fig. 4. We test the sensitivity of the method to noise by progressively increasing noise magnitude and letting the algorithm detect the symmetry for each of our three test models. We evaluate the accuracy of the results by computing the angular deviation between the axis found and the axis of the symmetry of the original model. *Top row:* We perturb each vertex of each polygon independently by a fraction of the longest length of the bounding box on each of the three test models. The left figure shows a noisy pick-up model with a noise magnitude of 1% and the right figure shows angular deviation evolution for the three models for a magnitude ranging from 0% to 1%. *Bottom row:* We randomly delete polygons of the models. The left figure shows a noisy pick-up obtained by deleting 5% of the polygons and the right figure shows angular deviation evolution by deleting 0% to 5% of the polygons of the three models. As can be seen from the curve, for small variations of the models, our method has approximatively linear dependency regarding noise and delivers high-quality results even for nonperfect symmetries.

the reliability of the results, we compute the angular deviation between the found axis of symmety and the real one, that is, computed with no noise. In our experiments, noise magnitude varies from 0 to 1% of the longest length of the model's bounding box, and the number of deleted polygons ranges from 0 to 5% of the total number of polygons in the model (see Figure 4).

The results of these experiments show that, for small variations, our method has approximatively linear dependency regarding noise and delivers high-quality results even for nonperfect symmetries. These statistical results can also be used to derive an upper bound on the mean angular error obtained as a function of the noise in the model.

5.4.1 *Application to Scanned Models.* We present in Figure 5 examples of applying the single-shape algorithm to scanned models, retreived from a Web database and used as is (see `http://shapes.aim-at-shape.net`). Our algorithm perfectly detects all the parameters of candidate symmetries for all these

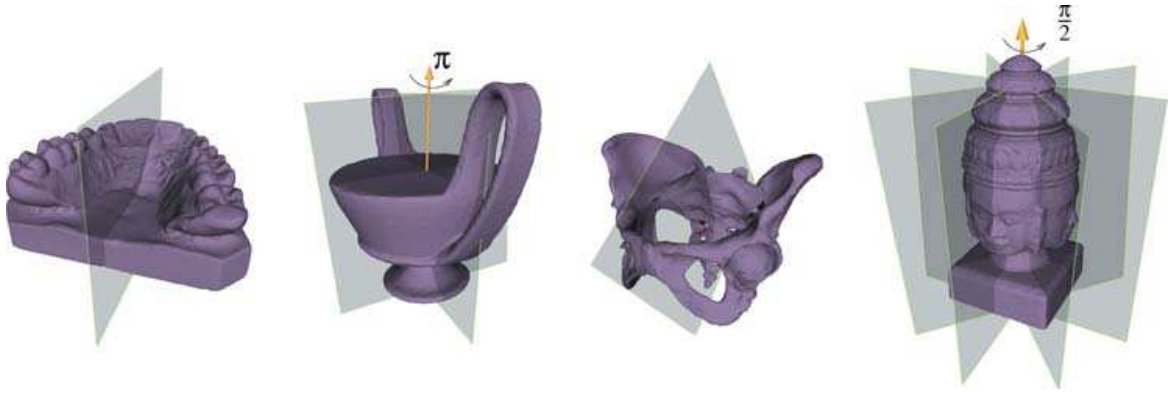


Fig. 5. Our algorithm perfectly detects approximate symmetries of scanned models. Detecting these symmetries requires relaxing the constraints when checking candidate symmetries on the model. Please note that these scanned models are by nature neither axis-aligned nor tesselated according to their symmetries. This illustrates the fact that our algorithm does not depend on the coordinate system nor on the mesh of the objects.

Table I. Computation times (in seconds) for the Four Scanned Models Presented in Figure 5

Model	Teeth	Vase	Pelvis	Angkor statue
# polygons	233, 204	76, 334	50, 000	163, 054
Computing moments*	33.7	11.8	7.26	23.26
Finding parameters	0.4	0.6	0.4	0.7
Checking candidates	9.4	11.1	5	12.2
Total	43.5	23.5	12.66	36.16

*Global computation times for moments of order 2 to 8

shapes. When testing these symmetries, one should allow a large enough symmetry distance error (as defined in Section 5.3) because these models are by nature not perfectly symmetric.

5.5 Discussion

Because the $\mathcal{M}^{2p}$ functions are trigonometric polynomials on the sphere, they have a maximum number of strict extrema depending on p: the larger p is, the more $\mathcal{M}^{2p}$ is able to capture the information of a symmetry, that is, to have an extremum in the direction of its axis. But because all moment functions *must* have a null gradient in this direction (according to Theorem 1), these extrema are bound to become nonstrict extrema for small values of p, and $\mathcal{M}^{2p}$ is forced to be constant on a subdomain of nonnull dimension. Using the cube as an example in which case $\mathcal{M}^2$ is a constant function a trigonometric polynomial of order 2 can simply not have enough strict extrema to represent all 12 distinct directions of the symmetries of the cube.

In all the tests we conducted, however, using moments up to order 10 has never skipped any symmetry on any model. But it would still be interesting to know the exact maximum number of directions permitted by moments of a given order.

6. FINDING SYMMETRIES OF GROUPS OF OBJECTS

In Section 5, we have presented an algorithm for finding the symmetries of single shapes. In this section, we present a *constructive* algorithm which recovers the symmetries of a group of objects—which we call *tiles* to indicate that together they form a larger object—from the symmetries and positions of each separate tile.

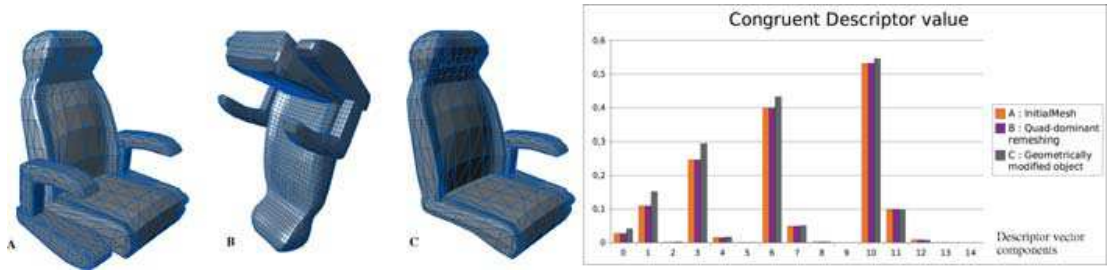


Fig. 6. This figure illustrates the reliability of our congruency descriptor (as defined by Equation (9)). Two identical objects meshed differently and expressed in two different coordinate systems (A and B) have extremely close descriptor vectors, but a slightly different object (C) has a different descriptor. The graphics on the right shows each component of the three descriptors.

The constructive algorithm first computes (if necessary) the symmetries of all separate tiles using the single shape algorithm. Then it detects which tiles are similar up to an isometric transform and finds the transformations between similar tiles. Then it explores all one-to-one mappings between tiles, discarding mappings which do not correspond to a symmetry of the group of tiles as a whole.

Section 6.2 explains how we detect similar tiles and Section 6.3 details the algorithm which both explores tile-to-tile mappings and finds the associated symmetry for the whole set of tiles.

Because it is always possible to apply the algorithm presented in Section 5 to the group of tiles, considering it as a single complex shape, questioning the usefulness of the constructive method is legitimate. For this reason, we will explain in Section 6.5 in which situations the constructive method is preferable to the algorithm for single shapes; but let us first explain the method itself.

6.1 Computing the Symmetries of Each Tile

If not available, the symmetries of each tile are computed using the algorithm presented in Section 5. When assembling known objects together, the economy of this computation can, of course, be performed by simply computing the symmetries of one instance for each class of different tiles.

6.2 Detecting Tiles Congruency

In this subsection, we introduce a shape descriptor suitable for detecting whether two shapes are identical up to an—unknown—isometry. We will use this tool for classifying tiles before trying to find a mapping of a composite object onto itself.

Let $\mathcal{S}$ be a shape and $C^{2p}_{2l,m}$ the spherical harmonic coefficients of its generalized even moment functions $\mathcal{M}^{2p}$ up to an order p. Our shape descriptor is defined as the $p(p+1)/2$-vector obtained by packing together the frequency energy of the spherical harmonic decomposition of all moments of $\mathcal{S}$ up to order p:

$$D_{2p} = \left[d^0_0, d^2_0, d^2_2, \ldots, d^{2p}_0, d^{2p}_2 \ldots d^{2p}_{2p}\right] \tag{9}$$

with

$$d^{2k}_{2l} = \sum_{-2l \le m \le 2l} \left(C^{2k}_{2l,m}\right)^2 \tag{10}$$

(See Figure 6). It has been shown by Kazhdan et al. [2003] that d^k_l, as defined in Equation (10), does not depend on the coordinate system the spherical harmonic decomposition is expressed in. This means that each d^{2p}_{2l}, and therefore D_{2p} itself, is not modified by isometric transforms of the shape. Mirror

Table II. Percentage of Tiles Matched by our Shape Descriptor That Are Effectively Identical For Our Test Scenes

Max order	39,557 Polygons 851 Tiles	182,224 Polygons 480 Tiles	515,977 Polygons 5,700 Tiles
2	92.1%	43.9%	92.3%
4	100%	78.0%	100%
6	100%	92.2%	100%
8	100%	100%	100%

Fig. 7. Scenes used for testing the object congruency descriptor. In each scene, the descriptor has been used to detect objects with similar geometry (but possibly different meshes) up to a rigid transform. Objects found to be congruent are displayed with the same color.

symmetries do not affect d_{2l}^{2p} either since they only change the sign of the coefficient for some harmonics in a coordinate system aligned with the axis.

Two tiles A and B are considered to be similar up to an isometric transform, at a precision ε, when:

$$\|D_{2p}(A) - D_{2p}(B)\| < \varepsilon.$$

Theoretically, this shape descriptor can produce false positives, that is, tiles that are not congruent but have the same descriptor, but it can not produce false negatives because of its deterministic nature. Our experiments have shown that using moments up to order 6 produces a sufficiently discriminant shape descriptor on all test scenes. This is illustrated in Table II where we present the average precision value, that is, the percentage of matched tiles that are actually identical up to an isometric transform, for a set of architectural scenes (Figure 7).

By definition, congruent tiles should have the same set of symmetries, possibly expressed in different coordinate systems. Since we know the symmetries of each of the tiles, we introduce this constraint, thereby increasing the discriminating power of our shape descriptor as shown in Table III.

6.3 Algorithm for Assembled Objects

6.3.1 *Overview.* Once we have determined all classes of congruent tiles, the algorithm examines all the one-to-one mappings of the set of all tiles onto itself which map each tile onto a similar tile. For each one-to-one mapping found, it determines the isometric transforms which are simultaneously compatible with each tile and its symmetries.

The algorithm works recursively: at the beginning of each recursion step, we have extracted two subsets of tiles, $\mathcal{H}_1$ and $\mathcal{H}_2$, of the composite shape $\mathcal{S}$, and we have computed the set of all possible isometric transforms that globally transform $\mathcal{H}_1$ into $\mathcal{H}_2$. Then, taking two new similar tiles, $\mathcal{S}_1 \in \mathcal{S} \setminus \mathcal{H}_1$

Table III. Percentage of Tiles Matched By Our Shape Descriptor That Are Effectively Identical Using the Added Constraint That Identical Tiles Must Have the Same Set of Symmetries Up to a Rigid Transform

Max order	39,557 Polygons 851 Tiles	182,224 Polygons 480 Tiles	515,977 Polygons 5,700 Tiles
2	95.6%	73.4%	97%
4	100%	96.0%	100%
6	100%	100%	100%
8	100%	100%	100%

and $\mathcal{S}_2 \in \mathcal{S} \setminus \mathcal{H}_2$, we restrict the set of isometric transforms to the isometric transforms that also map $\mathcal{S}_1$ onto $\mathcal{S}_2$ (but not necessarily $\mathcal{S}_2$ onto $\mathcal{S}_1$). Because these tiles have symmetries, this usually leaves multiple possibilities.

Note that the global symmetries found must always be applied with respect to the center of mass $\boldsymbol{g}$ of $\mathcal{S}$, according to the definition of a symmetry of $\mathcal{S}$.

At the end of the recursion step, we have the set of isometric transforms that map $\mathcal{H}_1 \cup \{\mathcal{S}_1\}$ onto $\mathcal{H}_2 \cup \{\mathcal{S}_2\}$.

Each recursion step narrows the choice of symmetries for $\mathcal{S}$. The recursion stops when either this set is reduced to identity transform or when we have used all the component tiles in the model. In the latter case, the isometric transforms found are the symmetries of the composite shape. The recursion is initiated by taking for $\mathcal{H}_1$ and $\mathcal{H}_2$ two similar tiles, that is, two tiles of the same class.

In the following paragraphs, we review the individual steps of the algorithm: finding all the isometric transforms which map tile $\mathcal{S}_1$ onto similar tile $\mathcal{S}_2$ and reducing the set of compatible symmetries of $\mathcal{S}$. We then illustrate the algorithm in a step-by-step example.

6.3.2 *Finding All the Isometries Which Transform a Tile onto a Similar Tile.* At each step of our algorithm, we examine pairs of similar tiles, $\mathcal{S}_1$ and $\mathcal{S}_2$, and we have to find all the isometries which map $\mathcal{S}_1$ onto $\mathcal{S}_2$.

If $\boldsymbol{g}_i$ is the center of mass of tile $\mathcal{S}_i$ and $\boldsymbol{g}$ is the center of mass of the composite shape $\mathcal{S}$, this condition implies that the isometries we are looking for transform vector $\boldsymbol{g}_1 - \boldsymbol{g}$ into $\boldsymbol{g}_2 - \boldsymbol{g}$. In order to generate the set of all isometric transforms that map $\mathcal{S}_1$ onto $\mathcal{S}_2$, we use the following property.

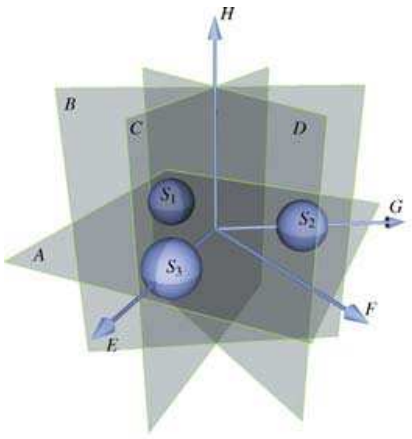
PROPERTY 4. *If J is an isometry that maps $\mathcal{S}_1$ onto a similar tile $\mathcal{S}_2$, then all the isometries K which map $\mathcal{S}_1$ onto $\mathcal{S}_2$ are of the following form:*

$$K = JT^{-1}AT \quad \textit{with} \quad A \in G_{\mathcal{S}_1} \quad \textit{such that} \quad A(\boldsymbol{g}_1 - \boldsymbol{g}) = \boldsymbol{g}_1 - \boldsymbol{g}, \tag{11}$$

where $G_{\mathcal{S}_1}$ is the group of symmetries of $\mathcal{S}_1$, and T is the translation of vector $\boldsymbol{g} - \boldsymbol{g}_1$ (refer to the Appendix for proof of this property).

This property states that, once we know a single *seed* isometric transform which maps $\mathcal{S}_1$ onto $\mathcal{S}_2$, we can generate all such transforms by using the elements of $G_{\mathcal{S}_1}$ in Equation (11).

6.3.3 *Finding a Seed Transform.* We need to find a seed transform J that maps $\mathcal{S}_1$ onto $\mathcal{S}_2$. For each tile, we extract a minimum set of independent vectors that correspond to extremas of their generalized even moment functions. The number of vectors needed depends on the symmetries of the tile. J is then defined as any isometric transform that maps the first set of vectors onto the second as well as vector $\boldsymbol{g}_1 - \boldsymbol{g}$ onto $\boldsymbol{g}_2 - \boldsymbol{g}$. Most of the time, a single isometric transform is possible at most. When multiple choices exist, the candidate transforms are checked onto the shapes using the distance presented in Section 5.3. This ensures that we find at least one seed transform.



Mapping	Symmetry
$S_1 \to S_1$ $S_2 \to S_2$ $S_3 \to S_3$	mirror-symmetry A
$S_1 \to S_3$ $S_3 \to S_1$ $S_2 \to S_2$	mirror-symmetry B 2-fold symmetry G
$S_1 \to S_2$ $S_2 \to S_1$ $S_3 \to S_3$	mirror-symmetry C 2-fold symmetry E
$S_2 \to S_3$ $S_3 \to S_2$ $S_1 \to S_1$	mirror-symmetry D 2-fold symmetry F
$S_1 \to S_2$ $S_2 \to S_3$ $S_3 \to S_1$	3-fold symmetry H

Fig. 8. Three spheres uniformly distributed on a circle in the z-plane. Etablishing all one-to-one mappings of the set of all tiles onto itself, which map each tile onto a similar tile, are used to detect all the symmetries of the shape. Note that the 3−fold symmetry H is detected and is associated to a circular permutation mapping.

6.3.4 *Ensuring Compatibility with Previous Isometries.* During the recursion, we need to store the current set of compatible isometries we have found. We do this by storing a minimal set of linearly independent vectors along with their expected images by these isometries. For example, if we have to store a symmetry of revolution, we store only one vector, the axis of the symmetry, and its image (itself). For mirror symmetries, rotations, and central symmetries, we store three independent vectors, along with their images by this isometric transform. For instance, in the case of a rotation of angle π around axis $\boldsymbol{X}$, we have:

$$\boldsymbol{X} \mapsto \boldsymbol{X} \quad \boldsymbol{Y} \mapsto -\boldsymbol{Y} \quad \boldsymbol{Z} \mapsto -\boldsymbol{Z}. \tag{12}$$

By examining all the one-to-one mappings of the set of all tiles onto itself, which map each tile onto a similar tile, we are able to detect all symmetries of the set of tiles (see Figure 8). Note in this example that the 3−fold symmetry H is detected and is associated to a circular permutation mapping.

6.4 Step-By-Step Example

Figure 9 presents a very simple example of a shape (a pair of pliers) composed of 3 tiles, $\mathcal{S}_1, \mathcal{S}_2$ (the handles), and $\mathcal{R}$ (the head). Two of the tiles are similar up to an isometric transform, $\mathcal{S}_1$ and $\mathcal{S}_2$. Figure 9 also displays the centers of mass, $\boldsymbol{g}_1$, and $\boldsymbol{g}_2$ of tiles $\mathcal{S}_1$ and $\mathcal{S}_2$ (which are not in the plane $z = 0$), and the center of mass $\boldsymbol{g}$ of the whole shape. In the coordinate systems centered on their respective centers of mass, $\mathcal{S}_1$ and $\mathcal{S}_2$ have a mirror symmetry of axis $\boldsymbol{Z}$, and $\mathcal{R}$ has a rotation symmetry around axis $\boldsymbol{X}$ of angle π.

Our constructive algorithm starts by selecting tile $\mathcal{R}$ and a similar tile (here, the only possible choice is $\mathcal{R}$).

Step 1. The algorithm explores the possibilities to transform $\mathcal{R}$ into itself. Two possibilities exist (a) the identity transform, and (b) the rotation around $\boldsymbol{X}$ of angle π, deduced from (a) by Property 4.
At this point, the algorithm branches, and either tries to map $\mathcal{S}_1$ to itself (branch 1) or to $\mathcal{S}_2$ (branch 2).
Branch 1, Step 1. The algorithm tries to match $\mathcal{S}_1$ to itself. The only compatible transform is the identity transform.

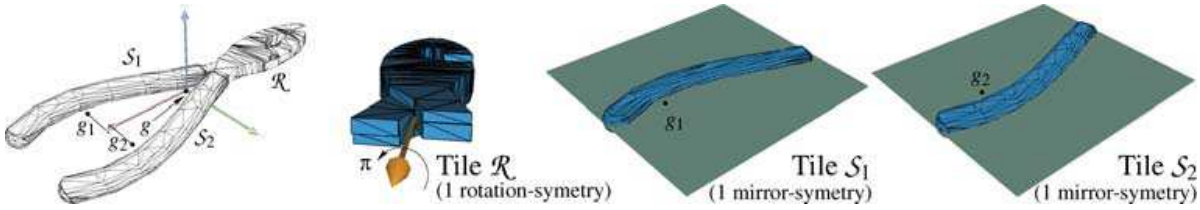


Fig. 9. Illustration of the constructive algorithm on a very simple example: from the symmetries of each of the 3 parts of the object, the symmetries of the whole object are recovered. Please note that no symmetry was ommitted in this Figure. In particular, tile $\mathcal{R}$ has only a rotational symmetry but no mirror symmetry. See text of Section 6.4 for a detailed explanation.

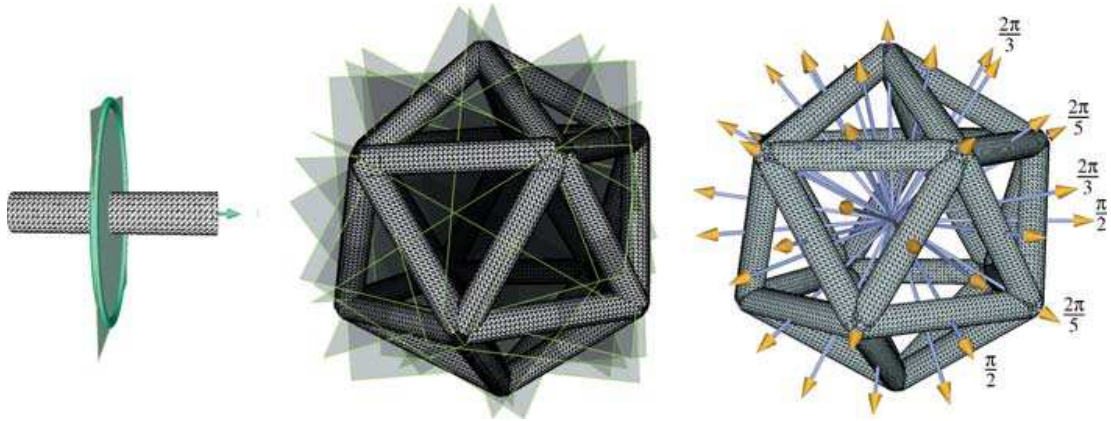


Fig. 10. A complex model which has the same group of symmetries as the icosahedron. The constructive algorithm successfully retrieves all 15 planes of mirror symmetries (*center*) and all 31 distinct axes of rotational symmetries (*right*) using the rotational and mirror symmetry of each tile (at *left*). The presence of 3−fold and 5−fold symmetries proves that our algorithm also detects symmetries which map a set of similar tiles onto itself through a complex permutation.

Branch 1, Step 2. The algorithm then tries to map $\mathcal{S}_2$ to itself. Once again, the only possible transform is the identity transform, and the recursion stops because all the tiles in the model have been used.
Branch 2, Step 1. The algorithm tries to match $\mathcal{S}_1$ to $\mathcal{S}_2$. The only compatible transform is the rotation around $\boldsymbol{X}$ of angle π.
Branch 2, Step 2. The algorithm then tries to match $\mathcal{S}_2$ to $\mathcal{S}_1$. Once again, the only compatible transform is the rotation around $\boldsymbol{X}$ of angle π, and the recursion stops because all the tiles in the model have been used.

Two symmetries have been found that map the shape onto itself, the identity transform and the rotation around $\boldsymbol{X}$ of angle π. Note that, although our algorithm can potentially create lots of branching, we prune branches that result in empty sets of transforms and, in practice, we only explore a small number of branches.

6.5 Application Scenarios

In order to illustrate the efficiency of the constructive algorithm, we show in this section various situations where this method is a valuable alternative to the single-shape algorithm.

6.5.1 *Application to an Agregation of Many Objects.* Figure 10 presents a complex model which has the same group of symmetries as an icosahedron. The constructive algorithm retrieves all the 31 distinct axis of rotational symmetries (Figure 10, *right*) as well as the 15 axis of plannar symmetries (Figure 10,

Table IV. Comparison of the costs of the single-shape algorithm presented in Section 5 to the cost of the constructive algorithm to find all 46 symmetries of the icosahedron shape displayed on Figure 10 at equivalent precision. Because the object is close to a sphere and because it has many symmetries, the constructive algorithm performs much better

Method	Single shape (order 10)	Constructive (order 4)
Moments calculation	500 sec	30×0.5 sec
Symmetry verification	46×55 sec	$30 \times 2 \times 1.5$ sec
Tile congruency	N/A	2 sec
Tile mappings	N/A	10 sec
Total	**50mn 30 sec**	**1mn 57 sec**

middle) of the shape, using the symmetries of each tile (Figure 10, *left*), which are 1 revolution symmetry and 1 mirror symmetry.

Conversely, directly applying the first algorithm on such a shape shows that $\mathcal{M}^2$ to $\mathcal{M}^8$ are extremely close to constant functions, making the extraction of directions an inaccurate process. The single-shape algorithm still correctly finds all the axis if using moments up to order 10, but this has some impact on computation times. Furthermore, the single-shape algorithm requires checking all of the symmetries found on the model which is a significant part of its computation time. This is not the case for the constructive algorithm because it relies on its knowledge of the symmetries of the tiles only. Because many symmetries exist for this model, the total computation time of the single-shape algorithm is therefore much higher. This is summarized in Table IV where we compare the computation times for both methods at equivalent precision (i.e., 10^{-4} radians).

6.5.2 *Finding Symmetries Inside Noncoherent Geometry.* There exist common situations where 3D scenes do not come as a set of closed separate objects but as an incoherent list of polygons. This happens, for instance, when retrieving geometric data from a Web site, mostly because a list of polygons constitutes a practical common denominator to all possible formats.

In such a case, applying the single-shape algorithm would certainly give the symmetries of the whole scene but if we are able to partition the set of polygons into adequate groups, that is, tiles to which we apply the constructive algorithm, we may be able to extract symmetric objects from the scene as well as the set of symmetries for the whole scene more rapidely as illustrated in Figure 10.

The gain in using the constructive algorithm to recover symmetries in the scene resides in the fact that, once tile symmetries have been computed, grouping them together and testing for symmetries in composed objects only adds a negligible cost which is not the case when we try to apply the single-shape algorithm to many possible groups of polygons or even to the entire scene itself.

The various issues in the decomposition of a raw list of polygons into intelligent tiles are beyond the scope of this article. In our case, tiles only need to be consistent with the symmetries. We propose the following heuristic to achieve this correctly for most scenes:

We define tiles as maximal sets of edge-connected polygons. To obtain them, we insert all vertices of the model into a KDTree and use this KDTree to efficiently recover which polygons share vertices up to a given precision and share an edge. By propagating connectivity information between neighboring polygons, we then build classes of edge-connected polygons, which we define to be our tiles. Figure 11 gives examples of such tiles for objects collected from the Web as a raw list of polygons.

Our simple heuristic approach of making tiles produced very good results on all scenes we tested and suffices for a proof of concept of the constructive algorithm. This is illustrated in Figure 11 where a lamp object and a chess game are shown along with their global symmetries. These symmetries were

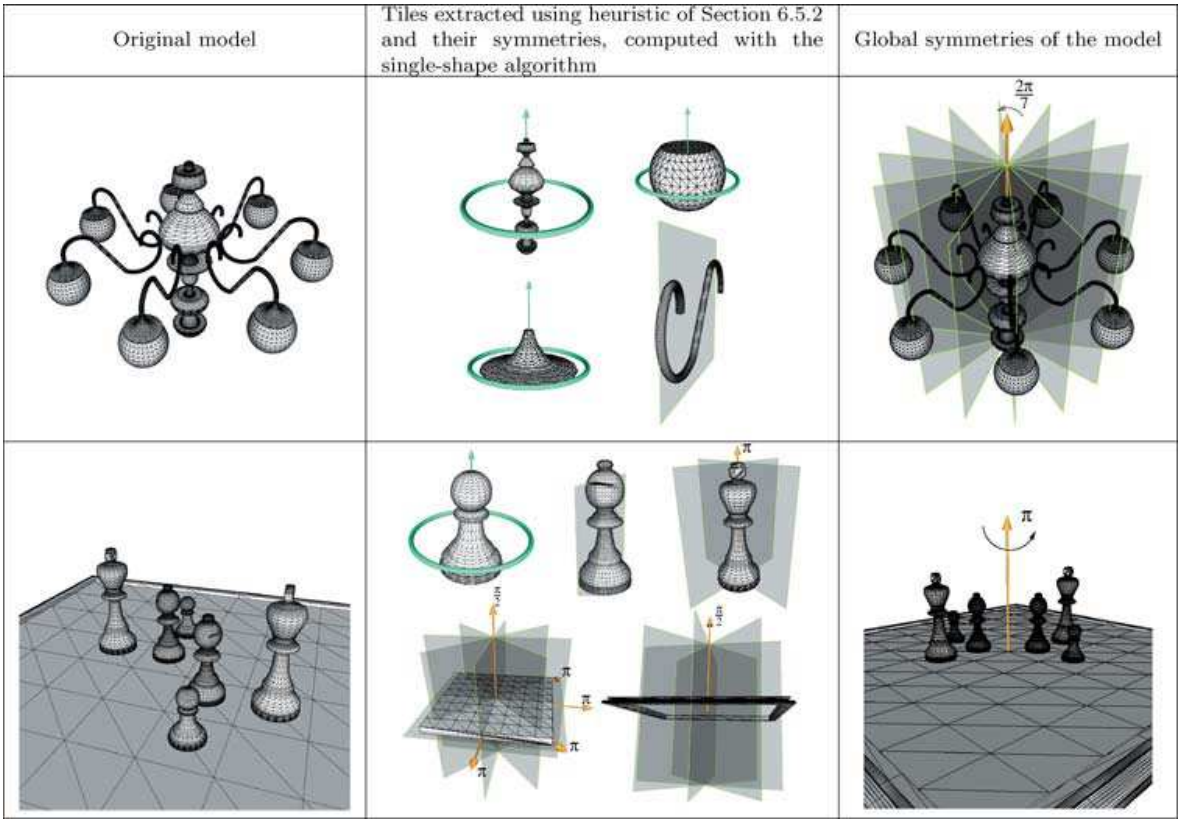


Fig. 11. Two models taken from the Web. From the raw list of polygons (left) our heuristic for scene partitionning extracts tiles before the single-shape algorithm computes the symmetries for each of them (center). Using this information, the constructive algorithm computes the symmetries of the whole model (right). *Top row*: A lamp object which has seven mirror symmetries and a 7−fold rotational symmetry. *Bottom row*: a chess board which is composed of pieces with very different symmetries but reveals to only have a single 2−fold symmetry around a vertical axis (Note: in this last model, once tiles have been identified, chess pieces were moved so as to obtain a model with at least one global symmetry).

computed from the symmetries of each of the subparts. These, in turn, were separately computed using the algorithm presented in Section 5.

Obviously, this application requires that constructed tiles be consistent with symmetries, that is, that it is possible to partition the scene into tiles which will map onto each other through the symmetries of the scene. This may not be easy with scanned models, for instance, nor in perturbated data. In such a case, our simple heuristic should be modified so as to base polygon neighborood relationships on proximity distances between polygons rather than vertex positions only. Doing so, cutting one tile into two parts and remeshing them independently, would have a high probability of producing the same original tile after reconstruction. If not, then the existance of a symmetry inside the model may become questionnable. Suppose, for instance, that the wrench in the step-by-step example (Section 6.4) gets split into tiles that are not exact symmetrical copies of one another, and that these two tiles are too far away to be merged into a single tile. Then the model is by nature not symmetric anymore which will also be the output of the constructive algorithm.

Table V. Computation Times (in seconds) for the Different Steps of our Algorithm, for the Models Shown in this Article

Model	Plier	Lamp	Chessboard
# polygons	1,940	39,550	24,942
# tiles	3	22	8
Computing moments*	0.9	18.2	15
Finding parameters	0.4	1.2	2.0
Checking candidates	2.3	7.4	7.9
Constructive algo.	0.001	0.05	0.01
Total	3.601	26.85	24.91

*Global computation time for moments of order 2 to 8.

6.6 Computation Cost

Computation times (in seconds) for the models shown in this article are given in Table V as well as the complexity of the models. They were measured on a machine equipped with a 2.4GHz processor with 512MB of memory. As expected, the cost of the computation of the moment functions and the cost of the verification of the candidates required by the first algorithm occupy the most important part of the total cost and depend on the model complexity. Conversely, finding the parameters of the symmetries (Section 5.2) as well as applying the constructive algorithm only depends on the number of these symmetries.

Regarding accuracy, both algorithms computed the axes of the symmetries with a maximum error of 10^{-4} radians, independently of shape complexity, in our tests.

7. APPLICATIONS

7.1 Geometry Compression and Instantiation

Our framework can be used for model compression at two different levels. (1) If a model exhibits symmetries, then it can be compressed by storing only the significant part of the model and using the symmetries to recreate the full model. (2) If a model contains multiple instances of the same part, then these parts can be instantiated. (see Figure 12).

Although complex models often do not present symmetries, symmetry-based compression can usually be used on some subparts of the model. The ability to express a model by explicitely storing the significant parts only while instancing the rest of the scene is provided by some recent 3D file formats such as X3D (see Table VI). We thus measure our compression ratios as the size of the X3D files before and after our two compression operations which we detail now.

The scene is first loaded as a raw collection of polygons, before being decomposed into tiles, using the heuristic presented in Section 6.5.2. We then compute symmetries and congruent descriptors for each tile. Computation times shown in Table VI present the average time needed to compute symmetries and congruent descriptors for a single tile. As the process of computing tile properties does not depend on the other tiles, it is an easily parallelizable process. The scene is then first compressed by instancing the tiles. Secondly, when storing each tile, we only store the minimum significant part of its geometry according to its symmetries. This part is extracted using the same algorithm we will present for remeshing a tile according to its symmetries in the next section. Note that compression rates shown on this table are computed using geometry informations only, that is, neither texturing nor material information are taken into account. Compression times shown in Table VI are the times needed to detect all classes of tile congruency.

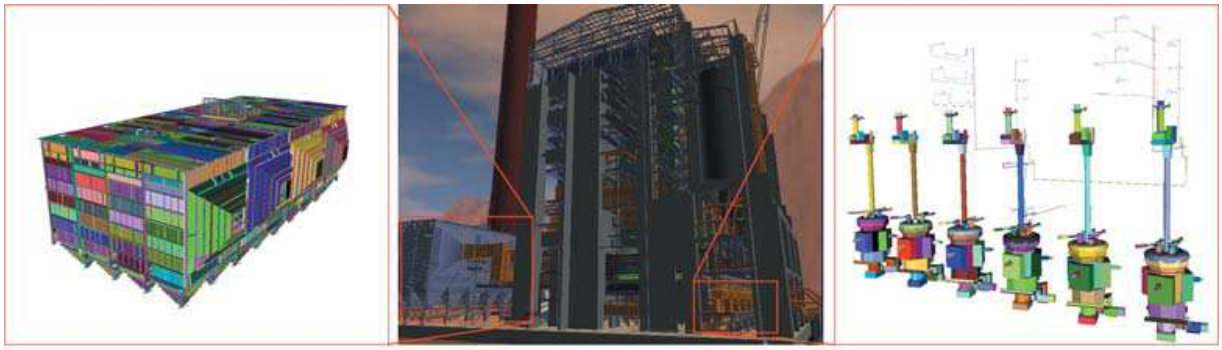


Fig. 12. Detecting symmetries and similarities between tiles created from a raw list of polygons allows us to compress geometric models in two ways: (1) by instancing similar tiles and (2) inside each symmetric tile, by instancing the part of the geometry which permits to reconstruct the whole tile. In such a big model as the powerplant (13 millions triangles), we achieve a compression ratio (ratio of geometry file size in X3D format) of 1:4.5. We show in this figure two subparts of the complete model. For each, we show the tiles computed by our heuristic (see Section 6.5) as well as the obtained compression ratio. The PowerPlant model is a courtesy of The Walkthru Project.

Table VI. Examples of Compression Rates Obtained Using our Symmetry Detection Method Coupled with the Congruency Descriptor. (See text in Section 7.1 for a detailed explanation.)

Model	Room	Plane	Studio	Powerplant
# polygons	39, 557	182, 224	515, 977	12, 748, 510
# tiles	851	480	5, 700	525, 154
av. computing tile properties (secs)	1.45	1.3	1.9	1.1
Compression time (secs)	7.2	9	14.6	311
Compression rate	1 : 2.7	1 : 8.3	1 : 3.5	1 : 4.5

7.2 Mesh Editing

It may be interesting, when an object contains symmetries, to remesh the object with respect to these symmetries. In order to do this, we proceed by first extracting the minimum part of the shape that can be reconstructed through each symmetry independently, then we apply the corresponding symmetry to each of them in order to get as many meshes of the shape which are consistent with each symmetry independently. The final step is to compute the union of all these meshes, merging identical vertices and adding new vertices at edge crossings. While not necessarily optimal, the obtained mesh is consistent with all symmetries of the shape.

Since a coherent remeshing allows for the establishment of a correspondence between model vertices, we have developed a proof-of-concept mesh editing system which allows the user to modify a 3D object under the constraints given by the symmetries of the original object. It appears that, under the constraint of too many symmetries, no vertices can be moved independently of the others, and the geometry is sometimes bound to scale about its center of gravity. Images collected from this program are displayed in Figure 13.

8. DISCUSSION

We discuss here a number of features of our technique as well as differences with existing approaches.

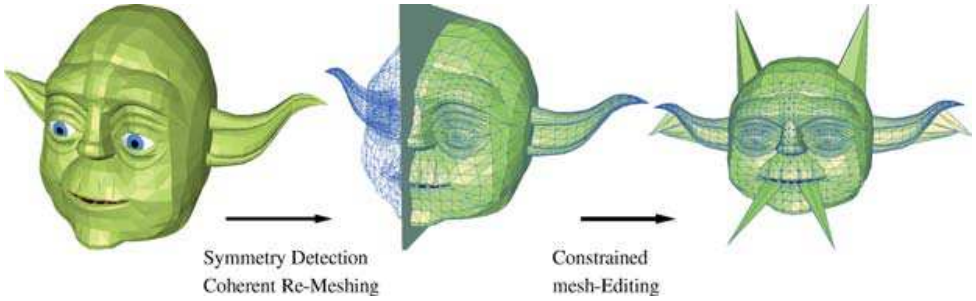


Fig. 13. Starting from an object in arbitrary orientation, we detect symmetries of the shape (in the figure, a planar symmetry) and use it to remesh the objects with respect to these symmetries. Then, a user can easily edit the mesh and modify it while keeping the symmetries of the initial shape.

Using Spherical Harmonics

Generalized moments are a central component of our system. As stated before, we do not compute these functions explicitly but we rather compute their coefficients in a spherical harmonics basis. As for the decomposition itself, any basis could be used. In particular, a well chosen basis of 3D monomials restricted to the unit sphere may also lead to a finite decomposition. Still, using spherical harmonics has many advantages, in particular, because we use the same coefficients computed once for different tasks throughout this article. (1) The expression of moment function as a sum of spherical harmonics provides an accurate detection of the potential axes of symmetries. This detection is made deterministic by finding the zero directions for the gradient of the moment functions. Such a computation is performed analytically from the 2nd order derivatives of the spherical harmonics, and thus does not introduce further approximation. (2) Computing symmetry parameters for the moment functions is made very easy by working on the spherical harmonic coefficients themselves. Since spherical harmonics are orthogonal and easily rotated, finding symmetries on the moment functions translates into simple relationships between the coefficients. (3) The spherical harmonic coefficients provide an effective shape congruency descriptor which we use to detect which tiles are identical up to an unknown isometric transform.

In summary, the use of spherical harmonics provides us a consistent framework throughout the whole process of our symmetry-finding algorithm.

Non Star-Shaped Objects

Whether the direct algorithm presented in Section 5 works for non star-shaped objects is a legitimate question. Our approach never relies on a spherical projection. Indeed, the moment functions, as expressed in Equations (1) and (5) are computed through an integration over the surface itself, possibly covering the same directions multiple times but with different values. Parts of a shape which correspond to a same direction during integration will not contribute the same into the various moment functions because of the varying exponent. By using various orders of moment functions in our symmetry detection process and in the computation of our shape congruency descriptor, we thus capture the geometry of non star-shaped objects as well. Some previous approaches [Kazhdan et al. 2004] achieved this by decomposing the shape into concentric spherical regions before doing a spherical integration which can be assimilated to convoluting the shape with 0-degree functions with concentric spherical support; Our technique is similar, but with another, kind of functions expressed into the form of the even moments. In summary, detecting symmetries on non star-shaped objects has no particular reason to fail which is illustrated by the result in Figure 2.

The second algorithm (for assembled objects) naturally works just as well for non star-shaped objects as illustrated by the examples in Figure 11.

Avoiding Dense Sampling

Previous methods that defined a continuous measure of symmetry([Zabrodsky et al. 1995; Kazhdan et al. 2004]) can theoretically compute both perfect and approximate symmetries. However, detecting symmetries using such methods involves a sampling step of the directions on the sphere, whose density must be adapted to the desired angular precision for the axis of the symmetry.

The work of Kazhdan et al. [2004] leads to impressive results concerning the improvment on the shape matching process. However, relying on this technique to obtain accurate symmetries with high angular precision requires a time-consuming step for the construction of the symmetry descriptors. According to the presented results, the time needed to compute reflective, 2-fold, 3-fold, 4-fold, 5-fold, and axial symmetry information for a spherical function of bandwidth $b = 16$ is 0.59 seconds. As stated in the article [Kazhdan et al. 2004], the number of samples taken on the sphere is $O(b^2)$ (i.e., approximately 10^3 sample directions) which is insufficent to reach a high angular precision equivalent to the one obtained with our method: reaching a precision of 10^{-4} radians would require approximately 10^9 sample directions. This would theoretically increase the computation time to approximately $0.59 \times 10^9/10^3 = 5.9\ 10^5$ seconds, making the method inefficient for this task.

In contrast, our method does not rely on a dense sampling of directions to find symmetries but on the computation of a fixed number of surface integrals which—thanks to the Gauss integration used—provides an extremely accurate approximation of the spherical harmonic coefficients of the moment functions. From there on, no further approximation is introduced in the computation of the directions of the candidate symmetries which lets us achieve an excellent angular precision at a much lower cost.

Furthermore, the cost of our algorithm does not rely on assumptions about the expected results. The method of Kazhdan et al. [2004] indeed computes symmetry descriptors for each kind of searched symmetry. Our method in turn computes all directions of possible symmetries and then checks back on the shape of the obtained candidates.

9. CONCLUSIONS

We have presented an algorithm to automatically retrieve symmetries for geometric shapes and models. Our algorithm efficiently and accurately retrieves all symmetries from a given model, independently of its tesselation.

We use a new tool, the generalized moment functions, to identify candidates for symmetries. The validity of each candidate is checked against the original shape using a geometric measure. Generalized moments are not computed directly: instead, we compute their spherical harmonic coefficients using an integral expression. Having an analytical expression for the generalized moment functions and their gradients, our algorithm finds potential symmetry axes quickly and with good accuracy.

For composite shapes assembled from simpler elements, we have presented an extension of this algorithm that works by first identifying the symmetries of each element, then sets of congruent elements. We then use this information to iteratively build the symmetries of the composite shape. This extension is able to handle complex shapes with better accuracy since it pushes the accuracy issues down to the scale of the tiles.

Future Work

The constructive algorithm presented in Section 6 automatically detects instantiation relationships between tiles into a composite shape.

We are currently developing a constructive instantiation algorithm which iteratively collates similar tiles into instances, checking at each step that the relative orientation of each tile with respect to each already constructed instance is preserved.

This algorithm requires the symmetries of the tiles, and maintaining the symmetries of the instances found so far. For this, we use our shape congruency metric, our algorithm for finding symmetries of single shapes, and our algorithm for finding symmetries on composite shapes.

APPENDIX (PROOFS)

PROOF OF THEOREM 1. Let A be an isometric transform which lets a shape $\mathcal{S}$ be globally unchanged. We have:

$$\begin{aligned}
\forall \boldsymbol{\omega} \quad \mathcal{M}^{2p}(A\boldsymbol{\omega}) &= \int_{\boldsymbol{s}\in\mathcal{S}} \|\boldsymbol{s}\times A\boldsymbol{\omega}\|^{2p}\,\mathrm{d}\boldsymbol{s} \\
&= \int_{\boldsymbol{t}\in A^{-1}\mathcal{S}} \|A\boldsymbol{t}\times A\boldsymbol{\omega}\|^{2p}|\det A|\,\mathrm{d}\boldsymbol{t} \\
&= \int_{\boldsymbol{t}\in A^{-1}\mathcal{S}} \|\boldsymbol{t}\times\boldsymbol{\omega}\|^{2p}\,\mathrm{d}\boldsymbol{t} \\
&= \mathcal{M}^{2p}(\boldsymbol{\omega})
\end{aligned}$$

At line 2, we change variables and integrate over the surface transformed by A^{-1}. At line 3, an isometric transform is a unit transform and so, its determinant is ± 1 and thus vanishes. The cross product is also left unchanged by applying an isometric transform to each of its terms. Line 4: because $A\mathcal{S}=\mathcal{S}$, we also have $\mathcal{S}=A^{-1}S$. The isometric transform A is thus also a symmetry of the $\mathcal{M}^{2p}$ moment functions.

Let A be an isometric transform with axis $\boldsymbol{v}$, and suppose that A is a symmetry of $\mathcal{M}^{2p}$. Let $\boldsymbol{d_v}$ be the direction of steepest descent of function $\mathcal{M}^{2p}$ around direction $\boldsymbol{v}$. Because A is a symmetry of $\mathcal{M}^{2p}$, we have:

$$\boldsymbol{d}_{A\boldsymbol{v}} = A\boldsymbol{d_v} = \boldsymbol{d_v}. \tag{13}$$

If A is a rotation, this is impossible because $\boldsymbol{d_v}\perp\boldsymbol{v}$. Moreover, for all directions ω, we have $\mathcal{M}^{2p}(-\omega)=\mathcal{M}^{2p}(\omega)$ and thus:

$$\boldsymbol{d}_{-\boldsymbol{v}} = -\boldsymbol{d_v}. \tag{14}$$

So, if A is a symmetry, we have $A\boldsymbol{v}=-\boldsymbol{v}$. From Equations (13) and (14), we get $\boldsymbol{d_v}=-\boldsymbol{d_v}$ which is impossible.

In both cases, $\mathcal{M}^{2p}$ can not have a direction of steepest descent in direction $\boldsymbol{v}$. Because $\mathcal{M}^{2p}$ is infinitely derivable, this implies that $\boldsymbol{\nabla}\mathcal{M}^{2p}(\boldsymbol{v})=0$ □

PROOF OF PROPERTY 4. Let $\mathcal{S}$ and $\mathcal{R}$ be two shapes, identical up to an isometric transform. Let J be an isometric transform such that $J\mathcal{S}=\mathcal{R}$. Let T be the translation of vector $-u_{\mathcal{S}}$ with $\boldsymbol{u}_{\mathcal{S}}=\boldsymbol{g}_{\mathcal{S}}-\boldsymbol{g}$ with $\boldsymbol{g}_{\mathcal{S}}$ as the center of mass of $\mathcal{S}$, and $\boldsymbol{g}$ the origin of the coordinate system into which J is applied.

— Let $A\in G_S$ be a symmetry of $\mathcal{S}$ such that $A\boldsymbol{u}_{\mathcal{S}}=\boldsymbol{u}_{\mathcal{S}}$. We have $AT\mathcal{S}=T\mathcal{S}$ (the symmetry A operates in the coordinate system centered on $\boldsymbol{g}_{\mathcal{S}}$). Let $K=JT^{-1}AT$. Then

$$\begin{aligned}
K\mathcal{S} &= JT^{-1}AT\mathcal{S} \\
&= JT^{-1}T\mathcal{S} \\
&= J\mathcal{S} \\
&= \mathcal{R}
\end{aligned}
\qquad\text{and}\qquad
\begin{aligned}
K\mathbf{0} &= JT^{-1}AT\mathbf{0} \\
&= JT^{-1}A(-\boldsymbol{u}_{\mathcal{S}}) \\
&= JT^{-1}(-\boldsymbol{u}_{\mathcal{S}}) \\
&= J\mathbf{0}=0
\end{aligned}$$

By construction K is a rigid transform and conserves distances. It maps the origin onto itself. K is thus an isometric transform. Furthermore, K maps $\mathcal{S}$ to $\mathcal{R}$.

— Let K be an isometric transform such that $K\mathcal{S} = \mathcal{R}$. Let us choose $A = TJ^{-1}KT^{-1}$. This choice leads to $K = JT^{-1}AT$. Moreover:

$$\begin{aligned} AT\mathcal{S} &= TJ^{-1}KT^{-1}T\mathcal{S} \\ &= TJ^{-1}KS \\ &= T\mathcal{S} \end{aligned} \qquad \text{and} \qquad \begin{aligned} A\boldsymbol{u}_{\mathcal{S}} &= TJ^{-1}KT^{-1}\boldsymbol{u}_{\mathcal{S}} \\ &= TJ^{-1}K2\boldsymbol{u}_{\mathcal{S}} \\ &= T2\boldsymbol{u}_{\mathcal{S}} = \boldsymbol{u}_{\mathcal{S}} \end{aligned}$$

and

$$\begin{aligned} A\mathbf{0} &= TJ^{-1}KT^{-1}\mathbf{0} \\ &= TJ^{-1}K\boldsymbol{u}_{\mathcal{S}} \\ &= TJ^{-1}(g_{\mathcal{R}} - \boldsymbol{g}) \\ &= T(-\boldsymbol{u}_{\mathcal{S}}) \\ &= \mathbf{0} \end{aligned}$$

By construction A is affine and conserves distances. It maps $\mathbf{0}$ onto $\mathbf{0}$. A is thus an isometric transform. A is also a symmetry of $\mathcal{S}$ which verifies $A\boldsymbol{u}_{\mathcal{S}} = \boldsymbol{u}_{\mathcal{S}}$.

— The set of isometries which map $\mathcal{S}$ to $\mathcal{R}$ is therefore the set of functions K of the form $K = JT^{-1}AT$, where $A \in G_{\mathcal{S}}$ is a symmetry of $\mathcal{S}$ such that $A(\boldsymbol{g} - \boldsymbol{g}_{\mathcal{S}}) = (\boldsymbol{g} - \boldsymbol{g}_{\mathcal{S}})$.

□

PROOF OF EQUATION 3. We compute the decomposition of function $\theta \longmapsto \sin^{2p}\theta$ into zonal spherical harmonics. We prove that this decomposition is finite, and give the values of the coeficients.

By definition [Hobson 1931], we have:

$$\begin{aligned} Y_L^0(\theta,\varphi) &= \sqrt{\frac{2L+1}{4\pi}} P_L(\cos\theta) \\ &= \sqrt{\frac{2L+1}{4\pi}} \frac{(-1)^L}{2^L L!} \frac{d^L}{dx^L}\left[(1-x^2)^L\right](\cos\theta) \end{aligned}$$

where P_k is the Legendre polynomial of order k. Because the set of Legendre polynomials $P_0, P_1, ..., P_n$ is a basis for polynomials of order not greater than n, function $\theta \longmapsto \sin^{2p}\theta = (1-\cos^2\theta)^p$ can be uniquely expressed in terms of $P_L(\cos\theta)$. The decomposition of $\theta \longmapsto \sin^{2p}\theta$ is thus finite and has terms up to Y_{2p}^0 at most.

Let's compute them explicitely:

$$\begin{aligned} \frac{d^L}{dx^L}\left[(1-x^2)^L\right] &= \frac{d^L}{dx^L}\sum_{k=0}^{L}(-1)^{L-k}x^{2L-2k}C_L^k \\ &= (-1)^L\frac{d^L}{dx^L}\sum_{k=0}^{L}(-1)^k x^{2k}C_L^k \\ &= \sum_{L\le 2k\le 2L}(-1)^{L+k}C_L^k 2k(2k-1)...(2k-L+1)x^{2k-L} \\ &= \sum_{L\le 2k\le 2L}(-1)^{L+k}C_L^k\frac{(2k)!}{(2k-L)!}x^{2k-L} \end{aligned}$$

So:

$$Y_L^0(\theta,\varphi) = \sqrt{\frac{2L+1}{4\pi}} \sum_{L\le 2k\le 2L} \frac{(-1)^k}{2^L L!} C_L^k \frac{(2k)!}{(2k-L)!} \cos^{2k-L}\theta$$

The coeficients of the decomposition we are interested in are thus:

$$\int_{\theta=0}^{\pi}\int_{\varphi=0}^{2\pi} Y_L^0(\theta,\varphi)\sin^{2p}\theta\sin\theta d\theta d\varphi = 2\pi\sqrt{\frac{2L+1}{4\pi}} \sum_{L\le 2k\le 2L} \frac{(-1)^k}{2^L L!} C_L^k \frac{(2k)!}{(2k-L)!} I_{2k-L}^p \tag{15}$$

where integrals I_m^p are defined by:

$$I_m^p = \int_{\theta=0}^{\pi} \sin^{2p+1}\theta\cos^m\theta d\theta$$

First, $I_m^p = 0$ for all odd m because the integrand in antisymetric around $x = \pi/2$. Then, if m is even:

$$\begin{aligned} I_m^p &= \underbrace{\left[\frac{1}{2p+2}\sin^{2p+2}\theta\cos^{m-1}\theta\right]_0^{\pi}}_{0} + \frac{m-1}{2p+2}\int_0^{\pi}\sin^{2p+3}\theta\cos^{m-2}\theta d\theta \\ &= \frac{m-1}{2p+2} I_{m-2}^{2p+3} \\ &= \frac{(m-1)(m-3)\ldots 1}{(2p+2)(2p+4)\ldots(2p+m)}\int_o^{\pi}\sin^{2p+m+1}\theta d\theta \end{aligned}$$

Let J_q be the integral defined by

$$J_q = \int_0^{\pi}\sin^{2q+1}\theta d\theta.$$

We have

$$\begin{aligned} J_q &= \underbrace{[-\cos\theta\sin^{2q}\theta]_0^{\pi}}_{0} + 2q\int_0^{\pi}\cos^2\theta\sin^{2q-1}\theta d\theta \\ &= 2qJ_{q-1} - 2qJ_q \end{aligned}$$

Therefore

$$\begin{aligned} J_q &= \frac{2q}{2q+1}J_{q-1} \\ &= \frac{2q(2q-2)\ldots 2}{(2q+1)(2q-1)\ldots 3}J_0 \\ &= \frac{2^{2q+1}(q!)^2}{(2q+1)!} \end{aligned}$$

For m even, we can take $m = 2r$ and $q = p + r$; we get:

$$\begin{aligned} I_{2r}^p &= \frac{(2r)!p!}{2^r r! 2^r (p+r)!}\frac{2^{2p+2r+1}(p+r)!^2}{(2p+2r+1)!} \\ &= \frac{(2r)!p!2^{2p+1}(p+r)!}{r!(2p+2r+1)!} \end{aligned} \tag{16}$$

From Equation (15), we deduce that, for L odd,

$$\int\int Y_L^0(\theta,\varphi)\sin^{2p}\theta\sin\theta d\theta d\varphi = 0.$$

For L even, we set $L = 2l$. Using $r = k - l$ to match Equation (16) in Equation (15), we get:

$$\begin{aligned}
S_p^l &= \int\int Y_{2l}^0(\theta,\varphi)\sin^{2p}\theta\sin\theta d\theta d\varphi \\
&= 2\pi\sqrt{\frac{4l+1}{4\pi}}\sum_{2l\le 2k\le 4l}\frac{(-1)^k}{2^{2l}(2l)!}C_{2l}^k\frac{(2k)!}{(2k-2l)!}\frac{(2k-2l)!p!2^{2p+1}(p+k-l)!}{(k-l)!(2p+2k-2l+1)!} \\
&= \frac{\sqrt{(4l+1)\pi}}{2^{2l}(2l)!}\sum_{l\le k\le 2l}(-1)^k C_{2l}^k\frac{(2k)!p!2^{2p+1}(p+k-l)!}{(k-l)!(2p+2k-2l+1)!} \\
&= \frac{\sqrt{(4l+1)\pi}}{2^{2l}}\sum_{l\le k\le 2l}(-1)^k\frac{(2k)!p!2^{2p+1}(p+k-l)!}{k!(2l-k)!(k-l)!(2p+2k-2l+1)!}
\end{aligned}$$

□

REFERENCES

ATTALAH, M. J. 1985. On symmetry detection. *IEEE Trans. Comput. 34*, 663–666.

BRASS, P. AND KNAUER, C. 2004. Testing congruence and symmetry for general 3-dimensional objects. *Comput. Geom. Theory Appl. 27*, 1, 3–11.

HIGHNAM, P. T. 1985. Optimal algorithms for finding the symmetries of a planar point set. Tech. Rep. CMU-RI-TR-85-13 (Aug). Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

HOBSON, E. W. 1931. *The Theory of Spherical and Ellipsoidal Harmonics*. Cambridge University Press, Cambridge, UK.

IVANIC, J. AND RUEDENBERG, K. 1996. Rotation matrices for real spherical harmonics, direct determination by recursion. *J. Phys. Chem. A. 100*, 6342–6347. (See also *Additions and corrections* in vol. 102, No. 45, 9099-9100).

JIANG, X.-Y. AND BUNKE, H. 1991. Determination of the symmetries of polyhedra and an application to object recognition. In *Proceedings of the International Workshop on Computational Geometry—Methods, Algorithms and Applications (CG '91)*. Lecture Notes in Computer Science, vol. 553. Springer, London, UK, 113–121.

KAZHDAN, M. M., FUNKHOUSER, T. A., AND RUSINKIEWICZ, S. 2003. Rotation invariant spherical harmonic representation of 3D shape descriptors. In *Proceedings of the 2003 Eurographics/ACM Siggraph Symposium on Geometry Processing (SGP '03)*. Eurographics Association, Aire-la-Ville, Switzerland, 167–175.

KAZHDAN, M. M., FUNKHOUSER, T. A., AND RUSINKIEWICZ, S. 2004. Symmetry descriptors and 3D shape matching. In *Proceedings of the 2004 Eurographics/ACM Siggraph Symposium on Geometry Processing (SGP '04)*. Eurographics Association, Aire-la-Ville, Switzerland.

KNUTH, D. E., MORRIS, JR., J. H., AND PRATT, V. R. 1977. Fast pattern matching in strings. *SIAM J. Comput. 6*, 2, 323–350.

MINOVIC, P., ISHIKAWA, S., AND KATO, K. 1993. Symmetry identification of a 3-D object represented by octree. *IEEE Trans. Patt. Analy. Mach. Intell. 15*, 5, 507–514.

PRINCE, E. 2004. *Mathematical Techniques in Crystallography and Materials Science*, 3rd Ed. Springer, Berlin, Germany.

RAMAMOORTHI, R. AND HANRAHAN, P. 2004. A signal-processing framework for reflection. *ACM Trans. Graph. 23*, 4, 1004–1042.

SUN, C. AND SHERRAH, J. 1997. 3D symmetry detection using extended Gaussian image. *IEEE Trans. Patt. Analy. Mach. Intell. 19*, 2 (Feb.), 164–168.

WOLTER, J. D., WOO, T. C., AND VOLZ, R. A. 1985. Optimal algorithms for symmetry detection in two and three dimensions. *Visual Comput. 1*, 37–48.

ZABRODSKY, H., PELEG, S., AND AVNIR, D. 1995. Symmetry as a continuous feature. *IEEE Trans. Patt. Analy. Mach. Intell. 17*, 12, 1154–1166.

Received August 2005; revised December 2005; accepted January 2006

3.

Propriétés de la fonction d'éclairage

Dans les méthodes de simulation de l'éclairage, on cherche à reconstituer une fonction (l'éclairage). Pour ce travail, on ne dispose que d'informations parcellaires, calculées en des points d'échantillonnage. L'efficacité et la précision des calculs dépendent du positionnement de l'échantillonnage, mais la détermination du positionnement optimal suppose une connaissance complète de la fonction d'éclairage.

Il est cependant possible de déterminer certaines caractéristiques de la fonction d'éclairage aux points d'échantillonnage, comme ses dérivées ou la fréquence de ses variations. On peut alors adapter l'échantillonnage en fonction de ces caractéristiques, ce qui permet d'améliorer la qualité de la simulation tout en diminuant le temps de calcul.

Dans ce chapitre, nous avons présenté nos travaux sur les propriétés de la fonction d'éclairage : d'une part la détermination des deux premières dérivées de la radiosité en présence d'obstacles (section 3.1), d'autre part la détermination des fréquences de la fonction d'éclairage (section 3.2). Dans les deux cas, on cherche à déterminer des propriétés locales, en se basant sur l'analyse d'un *local light field*, en tenant compte des obstacles.

Ces informations (dérivée, fréquence) permettent ensuite de guider efficacement la simulation de l'éclairage, que ce soit avec un oracle de raffinement qui contrôle l'erreur commise dans la simulation, ou en adaptant l'échantillonnage dans des algorithmes de *photon tracing*.

3.1 Étude des dérivées de la fonction d'éclairage et applications

3.1.1 Dérivées de la fonction de radiosité

La radiosité en un point $\boldsymbol{x}$ sous l'influence d'une source donnée a une expression intégrale, prenant en compte la fonction d'éclairage de la source :

$$B(\boldsymbol{x}) = \frac{\rho}{\pi} \int_{E(\boldsymbol{x})} B(\boldsymbol{y}) \frac{\cos\theta_1 \cos\theta_2}{r^2} \, \mathrm{d}\boldsymbol{y} \tag{3.1}$$

Où $E(\boldsymbol{x})$ désigne la partie de l'émetteur E qui est visible du point $\boldsymbol{x}$, et inclut donc l'influence des obstacles. La fonction $B(\boldsymbol{x})$ est dérivable, et il est possible de calculer ses dérivéesr [15, 11, 30]. Nous avons fourni l'expression de la dérivée première (le Jacobien ou le gradient) et de la dérivée seconde (le Hessien) de la radiosité au point $\boldsymbol{x}$.

Ces dérivées peuvent être calculées explicitement [15] ; ce calcul réutilise plusieurs quantités qui sont aussi nécessaires pour le calcul de la radiosité. Il est donc possible de calculer simultanément la radiosité et ses dérivées, pour un surcoût modeste (de l'ordre de 30 %) par rapport au calcul de la radiosité seule.

3.1.2 Contrôle de l'erreur lors de la simulation

Mais les dérivées ne donnent qu'une information ponctuelle sur la fonction de radiosité, que l'on peut éventuellement extrapoler par un développement limité, pour obtenir une information locale. Pour des applications pratiques (comme une simulation de l'éclairage dans une maquette virtuelle d'immeuble) il est utile d'avoir une information *globale*, comme l'erreur commise sur l'ensemble de la simulation. Cette information globale peut se déduire d'une information sur l'erreur commise sur chaque interaction entre facettes[1, 2].

Nous avons montré comment utiliser la connaissance des dérivées de la radiosité pour en déduire un encadrement strict de l'erreur commise sur chaque interaction. On peut alors en déduire l'erreur commise sur chaque interaction, et ainsi un oracle de raffinement exhaustif [11, 30] (voir p. 142).

Cet encadrement repose sur deux conjectures, la conjecture d'unimodalité et la conjecture de concavité. Ces deux conjectures étendent et formalisent un résultat connu depuis longtemps, que la radiosité due à une source convexe n'adment qu'un seul maximum[3].

3.2 Étude fréquentielle de la fonction d'éclairage

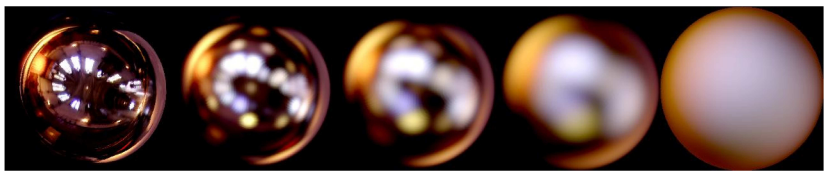


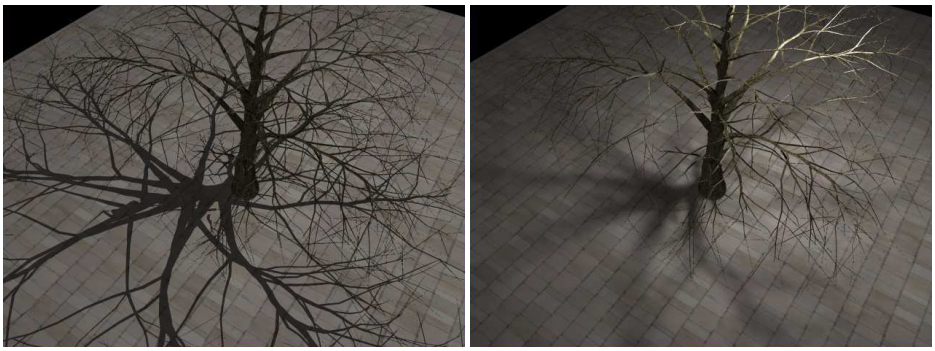
Figure 3.1 – La réflexion est plus ou moins nette en fonction de la BRDF. Ici, la BRDF évolue de parfaitement spéculaire à gauche à diffuse à droite.

La simulation de l'éclairage présente des phénomènes qui sont plus ou moins flous, en fonction des objets et des sources lumineuses. Par exemple, la réflexion sur un objet spéculaire est parfaitement nette, et contient tous les détails de la scène réfléchie, tandis que la réflexion sur un objet diffus est très floue (voir figure 3.1). De la même manière, l'ombre causée par une source ponctuelle est nette, tandis que l'ombre causée par une source surfacique est floue (voir figure 3.2). Enfin, l'éclairage indirect dans une scène est en général plus flou que l'éclairage direct (voir figure 3.3).

Ce côté net ou flou peut se traduire en terme de *contenu fréquentiel* de la fonction d'éclairage : les effets nets (ombres dures, réflexion spéculaires) correspondent à des hautes fréquences, tandis que les effets flous (ombres douces, réflexion diffuse, éclairage indirect) correspondent à des basses fréquences.

Avec François Sillion et Cyril Soler, dans le cadre d'une collaboration avec Frédo Durand et Eric Chan de l'équipe CSAIL du MIT, collaboration financée par une Équipe Associée INRIA, nous avons montré qu'il est possible de prédire ce contenu fréquentiel de l'éclairage en fonction

1. Daniel L , Brian S et Donald P. G . « Bounds and Error Estimates for Radiosity ». Dans *ACM SIGGRAPH '94*, p. 67–74, 1994.
2. James A , Kenneth T et Brian S . « A Framework for the Analysis of Error in Global Illumination Algorithms ». Dans *ACM SIGGRAPH '94*, p. 75–84, 1994.
3. G. D et E. F . « Accurate and Consistent Reconstruction of Illumination Functions Using Structured Sampling ». *Computer Graphics Forum (Eurographics '93)*, 12(3):C273–C284, septembre 1993.



(a) Source ponctuelle (b) Source surfacique

Figure 3.2 – L'ombre causée par une source ponctuelle est nette, tandis que l'ombre causée par une source surfacique est plus floue (images gracieusement fournies par Ulf Assarsson).

(a) Éclairage total (b) Éclairage direct (c) Éclairage indirect

Figure 3.3 – L'éclairage indirect est en général plus flou que l'éclairage direct (images gracieusement fournies par Cyril Soler).

de la scène (sources lumineuses, obstacles, matériaux) [5] (voir p. 164). Nous nous intéressons à la fois au spectre *spatial* et au spectre *angulaire* :

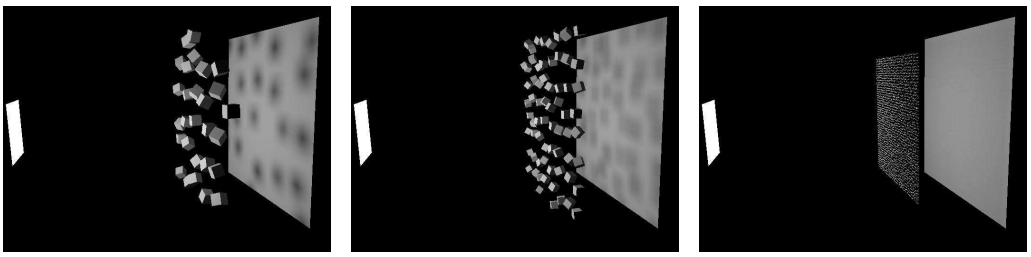
- Nous considérons l'éclairage comme un *local light field*, paramétré par une distance et un angle par rapport à un rayon de référence.
- Au départ de la source lumineuse, le spectre (spatial et angulaire) de ce local light field est connu.
- Chaque étape entre la source lumineuse et le récepteur est vue comme un filtre agissant sur le contenu fréquentiel :
  - Le transport à travers l'espace libre a l'effet d'une affinité orthogonale à l'axe des fréquences angulaires. Cet effet convertit les fréquences spatiales en fréquences angulaires.
  - En présence d'un obstacle, il y a convolution entre le spectre de l'obstacle et celui du local light field, introduisant de nouvelles fréquences spatiales.
  - la réflexion sur un récepteur peut être décomposée en plusieurs phases, avec un filtre particulier associé à chacune. L'effet général est celui d'un filtre passe-bas dans les fréquences angulaires. La fréquence de coupure de ce filtre est liée au caractère spéculaire

ou non de la BRDF. Une BRDF diffuse coupe complètement les fréquences angulaires tandis qu'une BRDF spéculaire les conserve entièrement.

– L'effet combiné de ces différents filtres permet de prédire l'étendue du spectre (spatial et angulaire) en un point donné de la scène. On peut ensuite tirer parti de cette connaissance pour guider les calculs de simulation de l'éclairage, en adaptant l'échantillonnage aux fréquences.

Une des observations les plus intéressantes issues de notre travail est que les fréquences spatiales et angulaires sont liées par l'étape de transport dans l'espace libre. Lorsqu'une BRDF non spéculaire élimine certaines fréquences angulaires, cela a aussi pour conséquence de supprimer des fréquences spatiales. Plus le transport est long, plus les fréquences spatiales sont liées à des fréquences angulaires élevées, et donc plus l'effet de coupure de la BRDF sur les fréquences angulaires se traduit par des fréquences spatiales basses.

Cet effet, confirmé par des études expérimentales, ouvre de nombreuses possibilités dans la simulation de l'éclairage. La capacité à prédire les fréquences maximales en chaque point de la scène permet de guider l'échantillonnage au cours du processus de simulation de l'éclairage, et ce quelle que soit la méthode employée pour les calculs (photon-mapping, radiosité, PRT...). Ce travail devrait être la base de nombreuses études futures et applications pratiques.



(a) Obstacles à basses fréquences : basses fréquences sur le récepteur

(b) Fréquences plus élevées dans les obstacles : fréquences plus élevées sur le récepteur

(c) Obstacles à hautes fréquences : basses fréquences sur le récepteur

Figure 3.4 – Application de notre étude des fréquences de la fonction d'éclairage. Les fréquences spatiales sur le récepteur diffus évoluent de façon non-monotone avec les fréquences des obstacles.

Comme application de notre étude, considérons le spectre de la fonction d'éclairage sur un réflecteur diffus en présence d'obstacles (voir figure 3.4). Ce spectre évolue de façon non-monotone en fonction du spectre des obstacles : dans un premier temps, une augmentation de la fréquence des obstacles se traduit par une augmentation des fréquences spatiales sur le récepteur (voir figure 3.4(b)). En revanche, passé un certain seuil, une augmentation de la fréquence des obstacles se traduit au contraire par une diminution des fréquences spatiales sur le récepteur (voir figure 3.4(c)).

Cet effet, déjà étudié[4], est parfaitement expliqué par notre étude : l'obstacle introduit des fréquences spatiales. Le transport après l'obstacle pousse ces fréquences spatiales dans les fréquences angulaires. La réflexion sur une surface diffuse coupe les fréquences angulaires, et donc les fréquences spatiales qui y sont liées.

4. Francois S et George D . « Feature-Based Control of Visibility Error: A Multiresolution Clustering Algorithm for Global Illumination ». Dans *ACM SIGGRAPH '95*, p. 145–152, 1995.

3.3 Discussion

Dans ce chapitre, nous avons présenté nos travaux sur les propriétés des fonctions d'éclairage. Nous avons montré qu'il est possible de déduire les propriétés locales de l'éclairage en fonction des positions respectives des objets. Ces propriétés peuvent être utilisées pour guider les méthodes de résolution, augmentant ainsi leur efficacité.

Les travaux sur le contenu fréquentiel de la fonction d'éclairage n'ont pas encore livré tout leur potentiel ; nous comptons les poursuivre par de nouvelles recherches.

Chaque réflexion sur une surface non-spéculaire après un transport dans l'espace libre a pour effet de faire baisser le contenu fréquentiel, aussi bien en espace qu'en angle. En conséquence, les effets à haute fréquence vont se produire : soit lors des réflexions spéculaires, soit dans l'éclairage direct, soit lorsque le transport dans l'espace libre a peu d'effet, c'est-à-dire lorsque deux objets sont proches.

Compte-tenu des progrès des cartes graphiques programmables, il est possible de calculer séparément et de façon interactive certains de ces effets à haute fréquence. Les effets à basse fréquence pourraient alors être calculés séparément, avec un échantillonnage plus lâche. Ce calcul en temps-réel des effets d'éclairage à haute fréquence fait l'objet du chapitre suivant.

3.4 Articles

3.4.1 Liste des articles

– Accurate Computation of the Radiosity Gradient for Constant and Linear Emitters (EGWR '95)
– An exhaustive error-bounding algorithm for hierarchical radiosity (CGF '98)
– A Frequency Analysis of Light Transport (Siggraph 2005)

3.4.2 Accurate Computation of the Radiosity Gradient for Constant and Linear Emitters (EGWR '95)

Auteurs : Nicolas H et François S
Conférence : 6e *Eurographics Symposium on Rendering*, Dublin, Irlande.
Date : juin 1995

Accurate Computation of the Radiosity Gradient for Constant and Linear Emitters

Nicolas Holzschuch, François Sillion

iMAGIS/IMAG*

Abstract: Controlling the error incurred in a radiosity calculation is one of the most challenging issues remaining in global illumination research. In this paper we propose a new method to compute the value and the gradient of the radiosity function at any point of a receiver, with arbitrary precision. The knowledge of the gradient provides fundamental informations on the radiosity function and its behaviour. It can specially be used to control the consistency of the discretisation assumptions.

1 Introduction

Computing the effect of a given patch on the radiosity of another patch is easily done assuming the radiosity on both patches are constant. In that case, we can express the influence of the emitter on the receiver with a single number, the form-factor. However, assuming the radiosity on both patches is constant is a strong assumption, and it introduces a specific source of error in the resolution algorithm.

In 1994, Arvo et al. [2] recorded all possible sources of error in global illumination algorithms, and introduced a framework for the analysis of error. Errors can occur at several levels in the resolution process:

- During modeling: our geometry is not exactly that of the scene we want to compute, and the BRDF are not exact either.
- During discretisation: our set of basis functions is not able to represent the real solution, but only an approximated one.
- During computation: we do not compute transfer elements exactly, but only within finite precision.

Lischinski et al. [9] presented an error driven refinement strategy for hierarchical radiosity. They were able to maintain upper and lower bounds on computed radiosity, and to concentrate their work in places where the difference was too large.

However, practical tools are still lacking to measure discretisation error. The problem is to efficiently reconstruct the radiosity function, with only a small number of samples. The best position for sampling points can only be found with total knowledge of the radiosity function.

In practice, at each step, we have to intuit the behaviour of the function from our current set of samples, in order to guess if we should – or not – introduce new sampling points, and where.

* iMAGIS is a joint research project of CNRS/INRIA/INPG/UJF. Postal address: B.P. 53, F-38041 Grenoble Cedex 9, France. E-mail: Nicolas.Holzschuch@imag.fr.

Knowing the radiosity derivatives allows better sampling, and thus reduction of discretisation error. Heckbert [6] and Lischinski et al. [7] predicted an efficient surface mesh using derivatives discontinuities. Drettakis and Fiume [4, 5] used information on the structure of the function to accurately reconstruct the illumination. Vedel and Puech [11] presented a refinement criterion based on gradient values at the nodes.

However, these authors usually resorted to approximated values of the partial derivatives, using several computations of radiosity and finite differences. Computing accurate values for the gradient allows arbitrary precision on our refinement criterion.

Arvo [1] presented a method to compute the irradiance Jacobian in case of partially occluded light sources. His method is presented with constant emitters. This paper introduces a new formulation of the radiosity gradient, valid for arbitrary radiosity functions on the emitter. The derivation is presented in the case of total visibility, i.e. without occluders. However, we shall see that extending the algorithm to the case of partial visibility is easy using Arvo's technique, since the two algorithms are largely independant.

2 Reformulating the Radiosity Equation

We will consider only diffuse surfaces, characterised by their radiosity function $B(x)$, without any assumption about B.

We want to know the value of B at a point x on a given patch A_1, due to the emission of light from another polygon A_2. We will assume a reflectivity of ρ at point x.

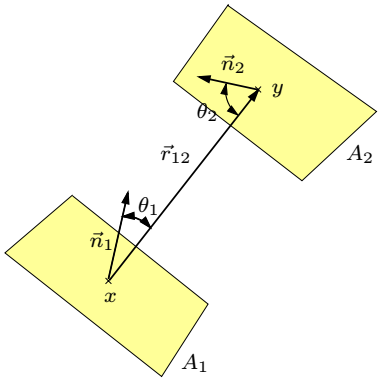


Fig. 1. Geometry of the problem

Our knowledge of radiosity at the receiving point derives from the integral equation:

$$B(x) = \frac{\rho}{\pi} \int_{A_2} \frac{B(y) \cos\theta_1 \cos\theta_2}{\|\vec{r}_{12}\|^2} dA_2 \tag{1}$$

where $\vec{r}_{12}$ is the vector joining point x on the receiver and point y on the emitter. θ_1 is the angle between $\vec{r}_{12}$ and the normal on the receiver, θ_2 the angle between $\vec{r}_{12}$ and the normal on the emitter, and dA_2 the area element on the emitter around point y (see Fig. 1).

Should any occluders be present between point x and emitter A_2, the integral would only be over the part of A_2 visible from x.

We can reformulate Equation 1 as the expression of the flux of a vector field through surface A_2:

$$B(x) = \int_{A_2} \vec{F} \cdot d\vec{A}_2 \tag{2}$$

where $\vec{F}$ is:

$$\vec{F} = -\frac{\rho B(y)(\vec{r}_{12} \cdot \vec{n}_1)\vec{r}_{12}}{\pi \|\vec{r}_{12}\|^4}$$

A classic way to deal with flux integrals as Equation 2 is to transform them into a linear integral using Stoke's theorem[2]:

$$\int_A (\nabla \times \vec{V}) \cdot d\vec{A} = \oint_{\partial A} \vec{V} \cdot d\vec{x} \tag{3}$$

These linear integrals can be easier to compute, and are also easier to estimate if there are no closed forms. However, to use Stoke's theorem (3), we need to express the vector field $\vec{F}$ as the curl of another vector field, $\vec{V}$.

A classic property is that this is equivalent to $\vec{F}$ having a null divergence ($\nabla \cdot \vec{F} = 0$). Basically, the divergence of a vector flux is a quantity that express at each point how much does the flux "radiates away" from this point, while the curl of a vector field "turns around" it at each point. The divergence of a curl is always null ($\nabla \cdot (\nabla \times \vec{V}) = 0$), and if a field has a null divergence, it can be expressed as a curl.

An easy computation shows that the divergence of $\vec{F}$ with respect to point y on surface A_2 is[3]:

$$\nabla \cdot \vec{F} = -\frac{\rho}{\pi} \frac{\vec{r}_{12} \cdot \vec{n}_1}{\|\vec{r}_{12}\|^2} (\nabla(B) \cdot \vec{r}_{12}) \tag{4}$$

and hence is null if the gradient of B on the emitting surface is null. That is to say, if the radiosity of the emitter is constant.

We can always separate $\vec{F}$ in two parts:

$$\vec{F} = \nabla \times (\vec{V}) + \vec{G}$$

Namely:

$$\vec{V} = \rho B(y) \frac{\vec{r}_{12} \times \vec{n}_1}{2\pi \|\vec{r}_{12}\|^2}$$

$$\vec{G} = -\rho \nabla(B) \times \left(\frac{(\vec{r}_{12} \times \vec{n}_1)}{2\pi \|\vec{r}_{12}\|^2} \right)$$

and thus cut Equation 2 in two integrals:

$$B(x) = \oint_{\partial A_2} \vec{V} \cdot d\vec{x}_2 + \int_{A_2} \vec{G} \cdot d\vec{A}_2 \tag{5}$$

Using the properties of cross-products and dot-products, we can rewrite Equation 5 as:

$$\frac{2\pi}{\rho} B(x) = -\vec{n}_1 \cdot \oint_{\partial A_2} B(y) \frac{\vec{r}_{12} \times d\vec{x}_2}{\|\vec{r}_{12}\|^2} + \int_{A_2} \frac{\vec{r}_{12}}{\|\vec{r}_{12}\|^2} \cdot (\vec{n}_1 \times (\nabla(B) \times \vec{n}_2))\, dA_2 \tag{6}$$

[2] ∂A stands for the contour of A, and $\oint$ expresses that this contour is closed.

[3] In this section, all derivative signs (∇, $\nabla\cdot$, $\nabla\times$) are relative to point y on surface A_2.

Note that this rewriting process does not make any assumption whatsoever on $B(y)$. Hence it can be used in any case. An interesting case is when $B(y)$ is constant: then $\vec{G} = \vec{0}$, and the second term is null. Another interesting case is $B(y)$ being linear: then its gradient is constant and can be carried out of the second integral, leaving only a pure geometric factor to compute. Appendix A presents a detailed study of these two cases.

This rewriting process separates the radiosity in two terms, a contour integral that we can generally compute, provided that we know the radiosity on the emitter, and a surface integral, generally harder to compute as an exact term. But, as shown later, having an integral form of this term, we can compute its value with an arbitrary precision.

3 The Radiosity Gradient

An interesting quantity to describe scalar fields, such as $B(x)$ is their gradient. Gradient is the extension of derivation for function of several variables. Basically, $\nabla(B)(x) \cdot \vec{v}$ gives the derivative of function B at point x in the direction of $\vec{v}$.

3.1 Computing the Gradient

The radiosity gradient can be computed from an equation such as Equation 1 or 6:

$$\nabla(B)(x) = \nabla\left(\int_{A_2} \vec{F} \cdot d\vec{A}_2\right) \tag{7}$$

In case the emitter A_2 does not depend on the position of the point x – that is to say, in case there are no occluder between point x and the emitting surface A_2 – this equation is equivalent to:

$$\nabla(B)(x) = \int_{A_2} \nabla\left(\vec{F} \cdot d\vec{A}_2\right)$$

Or, if we use Equation 5:

$$\nabla(B)(x) = \oint_{\partial A_2} \nabla\left(\vec{V} \cdot d\vec{x}_2\right) + \int_{A_2} \nabla(\vec{G} \cdot d\vec{A}_2) \tag{8}$$

If the emitter depends on the position of point x – that is, if there are occluders – the expression of $\nabla(B)(x)$ is the sum of two terms; the first one takes into account the variation of $\vec{F}$, and is exactly the term we are discussing, and the second one takes into account the variation of the emitter. Thus, it is easy to merge a method to compute the gradient with occluders and a constant emitter, as in Arvo [1], and our method to compute the gradient with an arbitrary emitter, but without occluders.

Note that in this section, we are taking a derivative with respect to point x on the receiving surface, not with respect to point y on the emitting surface. So for our derivating operator, the radiosity on the emitting point $B(y)$ can be regarded as constant, as well as its gradient, $\nabla(B)(y)$.

Using the properties of the gradient of a scalar product, starting from Equation 8, we can express the gradient of radiosity at the receiving point:

$$\begin{aligned}\frac{2\pi}{\rho}\nabla(B)(x) = \vec{n}_1 \times \oint_{\partial A_2} B(y)\frac{d\vec{x}_2}{\|\vec{r}_{12}\|^2} + 2\oint_{\partial A_2} B(y)\frac{\vec{n}_1 \cdot \vec{r}_{12}}{\|\vec{r}_{12}\|^4}(\vec{r}_{12} \times d\vec{x}_2) \\ + \int_{A_2} (\vec{n}_1 \times (\nabla(B)(y) \times \vec{n}_2))\frac{dA_2}{\|\vec{r}_{12}\|^2} \\ - 2\int_{A_2} \frac{(\vec{n}_1 \times (\nabla(B)(y) \times \vec{n}_2)) \cdot \vec{r}_{12}}{\|\vec{r}_{12}\|^4}\vec{r}_{12}dA_2\end{aligned} \tag{9}$$

This equation, like the radiosity equation (6) is divided in two parts: a contour integral which usually has a closed form, and a surface integral that we can estimate to any arbitrary precision.

As before, two interesting cases occur: if the gradient on the emitter is null, that is if we assume a constant radiosity on the emitter, all surface integrals vanish. And if the gradient on the emitter is constant, that is if we assume a linear radiosity on the emitter, it can be carried out of the surface integrals, leaving us with purely geometrical factors or vectors to compute. Please refer to Appendix A for a detailed study of these cases.

3.2 Using the gradient

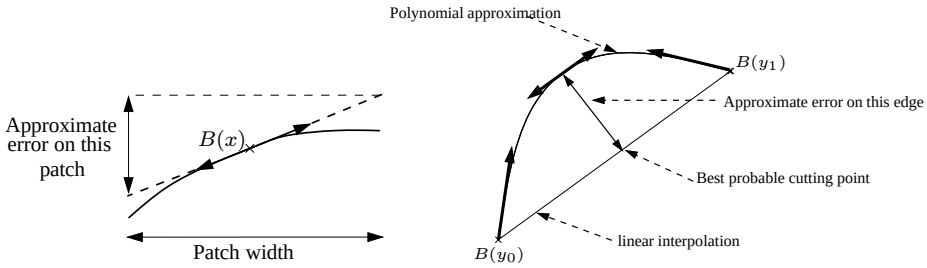
Knowing the gradient at a point gives very valuable information on the function we are studying. As previous authors pointed out, the gradient may be used either to reconstruct the illumination function before display, or to check the consistency of our discretisation hypothesis.

Reconstructing the illumination function If we know the radiosity values and the gradient at our sample points, we can then reconstruct the radiosity function as, e.g. a bicubic spline.

Salesin et al. [10] and Bastos et al. [3] proposed such methods for reconstruction of radiosity using estimates of gradient. Ward and Heckbert [12] computed irradiance gradients to interpolate irradiance on receiving surfaces.

Refinement criterion Many radiosity algorithms assume a constant radiosity over patches. It may seem strange to compute the gradient of radiosity in that case, but in fact the information given by the gradient can also be used there.

Using the derivatives allows precisely to check whether our discretisation hypothesis were correct or not, and if they were not, it also gives a hint on where it would be best to refine in order to minimize the discretisation error.



Constant Radiosity Assumption (a)

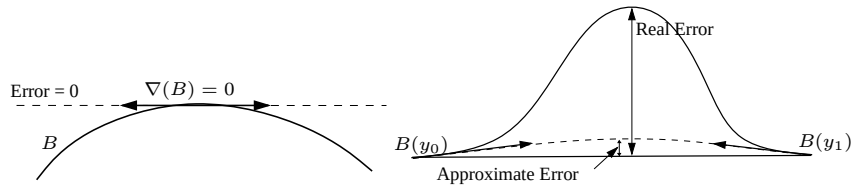
Linear Radiosity Assumption (b)

Fig. 2. Using the gradient to measure discretisation error

If we assume constant radiosity on our patches, the gradient gives a first estimate of how much does the function vary over the patch: $\nabla(B)(x) \cdot \vec{v}$ is approximately the difference between radiosity at point $x + \vec{v}$ and radiosity at point x. The norm of the gradient times the width of the patch gives an approximation of how much does radiosity varies over the patch (see Fig. 2a for an example in 2D). The direction of the gradient gives the best probable direction of refinement.

If we assume linear radiosity on our patches, we can compute a cubic interpolant over the patch using the radiosity and gradient values at each vertices, and then test how much

this cubic interpolant differs from our linear assumption (see Fig. 2b for an example in 2D). We can even compute the difference between linear and cubic interpolant without explicitly computing the interpolants. This criterion also gives the best next sampling point, the position of the maximum difference between the two interpolants.



Constant Radiosity Assumption (a)

Linear Radiosity Assumption (b)

Fig. 3. Sample cases where the proposed refinement criterion fail

Although none of these refinement criterion are foolproof (see Fig. 3 for an example where these two criterion fail to detect an important discretisation error), they provide a way to measure and quantify the discretisation error.

Also, the points where these refinement criterions are more likely to be fooled are basically the extrema of the radiosity functions. We know that a single convex emitter induces only one maximum on the receiver (see, for example, Drettakis [4]).

So, to study the interaction between two patches so as to minimize discretisation error, we would, first, find the theorical position of the maximum of radiosity, then sample it, then refine the receiving patch using our gradient-based criterion.

4 Implementation and First Results

We have implemented the gradient and radiosity formulas described in appendix A, for both constant and linear emitters[4]. Using a `C++` class for vectors, with definitions of cross- and dot-products makes the implementation very straightforward, being a mere recopy of the formulas. The only special attention it needs is avoiding to recompute quantities already computed at previous steps. Most of the quantities needed to express the gradient were also used for the radiosity.

In the color plates, Fig. A shows the radiosity values on a plane, due to a triangular emitter parallel to that plane (see Fig. E for the geometry of the scene). Fig. B shows the norm of the gradient of this radiosity.

Fig. C and D show the same quantities if we assume a linear emitter.

5 Conclusion and Future Work

We have provided a way to compute the gradient of radiosity at the receiving point with any distribution of illumination on the emitter. The gradient can be used in several ways, and specially to compute the discretisation error. It can also be used to find the best next refining point.

Future work will include a complete gradient computation, using the method described by Arvo in [1] to take the possible occluders into account.

[4] Souce code and documentation for this implementation is available at `ftp://safran.imag.fr/pub/holzschu/gradient.tar.gz`.

The ability to compute radiosity gradients for linear emitters is especially interesting when using linear basis functions or linear wavelets. In that case, the discretisation error can be precisely isolated.

Our next step will be a complete implementation of the refinement criterion described in section 3.2, to effectively reduce the discretisation error, within a hierarchical radiosity framework with linear radiosity.

We will then have the possible background for a complete radiosity algorithm with all possible sources of error (visibility, discretisation, computational) recorded and monitored, thus allowing to focus the computing resources at the points where this error is large.

6 Acknowledgements

Color pictures were computed by Myriam Houssay-Holzschuch using the GMT package, developped by Wessel and Smith [13].

The authors would like to thank the anonymous reviewers for useful insights and positive criticism.

References

1. Arvo, J.: The Irradiance Jacobian for Partially Occluded Polyhedral Sources. SIGGRAPH (1994) 343–350
2. Arvo, J., Torrance, K.,Smits, B.: A Framework for the Analysis of Error in Global Illumination Algorithms. SIGGRAPH (1994) 75–84
3. Bastos, R. M., de Sousa, A. A., Ferreira, F. N., Reconstruction of Illumination Functions using Bicubic Hermite Interpolation. *Fourth Eurographics Workshop on Rendering* (June 1993) 317–326
4. Drettakis, G., Fiume, E.: Concrete Computation of Global Illumination Accurate and Consistent Reconstruction of Illumination Functions Using Structured Sampling. Computer Graphics Forum (Eurographics 1993 Conf. Issue) 273–284
5. Drettakis, G., Fiume, E.: Concrete Computation of Global Illumination Using Structured Sampling. *Third Eurographics Workshop on Rendering* (May 1992) 189–201
6. Heckbert, P. S.: *Simulating Global Illumination Using Adaptative Meshing.* PhD Thesis, University of California, Berkeley, June 1991.
7. Lischinski, D., Tampieri, F., Greenberg, D. P.: Discontinuity Meshing for Accurate Radiosity. *IEEE Computer Graphics and Applications 12,6* (November 1992) 25–39
8. Lischinski, D., Tampieri, F., Greenberg, D. P.: Combining Hierarchical Radiosity and Discontinuity Meshing. SIGGRAPH (1993)
9. Lischinski, D., Smits, B., Greenberg, D. P.: Bounds and Error Estimates for Radiosity. SIGGRAPH (1994) 67–74
10. Salesin, D., Lischinski, D., DeRose, T.: Reconstructing Illumination Functions with Selected Discontinuities. *Third Eurographics Workshop on Rendering* (May 1992) 99–112
11. Vedel, C., Puech, C.: Improved Storage and Reconstruction of Light Intensities on Surfaces. *Third Eurographics Workshop on Rendering* (May 1992) 113–121
12. Ward, G. J., Heckbert, P. S.: Irradiance Gradients. *Third Eurographics Workshop on Rendering* (May 1992) 85–98
13. Wessel, P. and Smith, W. H. F.: Free Software helps Map and Display Data. *EOS Trans. Amer. Geophys. U.*, vol. 72, 441–446, 1991

A Application to Constant and Linear Emitters

A.1 Case of a constant emitter

In the case of a constant emitter the Equations 6 and 9 reduce to:

$$\frac{2\pi}{\rho}B(x) = -\vec{n}_1 \cdot \oint_{\partial A_2} B(y)\frac{\vec{r}_{12} \times d\vec{x}_2}{\|\vec{r}_{12}\|^2} \tag{10}$$

$$-\frac{2\pi}{\rho}\nabla(B)(x) = \vec{n}_1 \times \oint_{\partial A_2} B(y)\frac{d\vec{x}_2}{\|\vec{r}_{12}\|^2} + 2\oint_{\partial A_2} B(y)\frac{\vec{n}_1 \cdot \vec{r}_{12}}{\|\vec{r}_{12}\|^4}(\vec{r}_{12} \times d\vec{x}_2) \tag{11}$$

If A_2 is a polygon, these integrals have a closed form, and yield:

$$\frac{2\pi}{\rho}B(x) = -B_2\vec{n}_1 \cdot \sum_i I_1(i)\,(\vec{r}_i \times \vec{e}_i)$$

$$-\frac{2\pi}{\rho}\nabla(B)(x) = B_2 \sum_i I_1(i)\,(\vec{n}_1 \times \vec{e}_i) + 2B_2 \sum_i (\vec{r}_i \times \vec{e}_i) \cdot \vec{n}_1\,(I_2(i)\vec{r}_i + J_2(i)\vec{e}_i)$$

where the sum extends on all the edges of the polygon, and B_2 is the radiosity of the emitter. $\vec{r}_i$, $\vec{e}_i$, $I_1(i)$, $I_2(i)$ and $J_2(i)$ stand for (see also Fig. 4):

$$\vec{r}_i = \overrightarrow{xE_i}$$

$$\vec{e}_i = \overrightarrow{E_iE_{i+1}}$$

$$I_1(i) = \frac{\gamma_i}{\|\vec{r}_i \times \vec{e}_i\|}$$

$$I_2(i) = \frac{1}{2\|\vec{r}_i \times \vec{e}_i\|^2}\left(\frac{\vec{r}_{i+1} \cdot \vec{e}_i}{\|\vec{r}_{i+1}\|^2} - \frac{\vec{r}_i \cdot \vec{e}_i}{\|\vec{r}_i\|^2} + \|\vec{e}_i\|^2 I_1(i)\right)$$

$$J_2(i) = \frac{1}{2\|\vec{e}_i\|^2}\left(\frac{1}{\|\vec{r}_i\|^2} - \frac{1}{\|\vec{r}_{i+1}\|^2} - 2I_2(i)\vec{r}_i \cdot \vec{e}_i\right)$$

and γ_i is the angle sustended by edge $\vec{e}_i$ from point x.

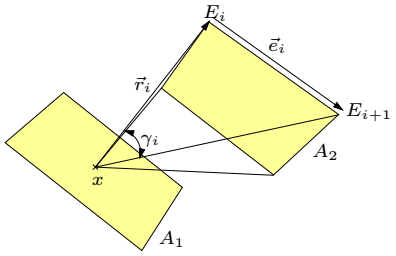


Fig. 4. Geometric Notations Used

Computing B at point x requires roughly 63 multiplications, 6 divisions, 54 additions or substractions, 6 square roots and 3 arc cosines. This equals approximately 300 additions on an SGI Indy computer, with no optimisations and the standard compiler.

Computing $\nabla(B)(x)$ requires roughly 87 multiplications more, 57 additions more and 3 divisions more. Which, with the same material, equals approximately 150 additions.

Although this computationnal cost may depend on implementation details as well as on the computer used (some compilers have very fast implementations of arc cos and square root), computing the gradient along with the radiosity does not over-increase computation time.

A.2 Case of a linear emitter

If the emitter is not constant, the gradient of radiosity on the emitter is not null, and must be used in our computations. However, if we assume the radiosity of the emitter is linear, then its gradient is constant and can be carried out of the integrals. Moreover, this gradient is orthogonal to $\vec{n}_2$, and can be expressed as:

$$\nabla(B)(y) = \vec{n}_2 \times \vec{k}$$

with $\vec{k}$ orthogonal to $\vec{n}_2$. $\vec{k} = \frac{1}{2A_2}((B_2 - B_0)\vec{e}_0 + (B_1 - B_0)\vec{e}_2)$

Using the properties of $\vec{k}$, we can express Equation 6 as:

$$\frac{2\pi}{\rho}B(x) = -\vec{n}_1 \cdot \oint_{\partial A_2} B(y)\frac{\vec{r}_{12} \times d\vec{x}_2}{\|\vec{r}_{12}\|^2} + (\vec{n}_1 \cdot \vec{n}_2)(\vec{k} \cdot (\vec{m} \times \vec{n}_2)) + (\vec{m} \cdot \vec{n}_2)(\vec{n}_2 \cdot (\vec{n}_1 \times \vec{k}))$$

with:

$$\vec{m} = \int_{A_2} \frac{\vec{r}_{12}}{\|\vec{r}_{12}\|^2} dA_2 = \int_{A_2} \nabla(\ln(r_{12}))dA_2$$

Computing the contour integrals does not induce any particular difficulties. However, computing $\vec{m}$ is harder. We can make use of Ostrogradsky's theorem, similar to Stoke's:

$$\int_A \nabla(V) \times d\vec{A} = -\oint_{\partial A} V d\vec{x}$$

to express $\vec{m} \times \vec{n}_2$.

$\vec{m} \cdot \vec{n}_2$ is null if point x is on polygon A_2. If point x is not on polygon A_2, it can be estimated with arbitrary precision.

The formula for $B(x)$ is then:

$$\begin{aligned}\frac{2\pi}{\rho}B(x) = -\vec{n}_1 \cdot \oint_{\partial A_2} B(y)\frac{\vec{r}_{12} \times d\vec{x}_2}{\|\vec{r}_{12}\|^2} - (\vec{n}_1 \cdot \vec{n}_2)\vec{k} \cdot \oint_{\partial A_2} \ln(r_{12})d\vec{x}_2 \\ + (\vec{m} \cdot \vec{n}_2)(\vec{n}_2 \cdot (\vec{n}_1 \times \vec{k}))\end{aligned}$$

If we derive this formula rather than use Equation 9, we find:

$$\begin{aligned}-\frac{2\pi}{\rho}\nabla(B)(x) = \vec{n}_1 \times \oint_{\partial A_2} B(y)\frac{d\vec{x}_2}{\|\vec{r}_{12}\|^2} + 2\oint_{\partial A_2} B(y)\frac{\vec{n}_1 \cdot \vec{r}_{12}}{\|\vec{r}_{12}\|^4}(\vec{r}_{12} \times d\vec{x}_2) \\ - (\vec{n}_1 \cdot \vec{n}_2)\oint_{\partial A_2} \frac{\vec{r}_{12}}{\|\vec{r}_{12}\|^2}(\vec{k} \cdot d\vec{x}_2) \\ + (\vec{n}_2 \cdot (\vec{n}_1 \times \vec{k}))\,(\vec{n}_2 X_1 - 2(\vec{n}_2 \cdot \vec{r}_0)\vec{p})\end{aligned}$$

with:

$$\vec{p} = \int_{A_2} \frac{\vec{r}_{12}}{\|\vec{r}_{12}\|^4} dA_2$$
$$X_1 = \int_{A_2} \frac{dA_2}{\|\vec{r}_{12}\|^2}$$

Computing $\vec{p}$ is exactly like computing $\vec{m}$: we can compute $\vec{p} \times \vec{n}_2$, and we can estimate $\vec{p} \cdot \vec{n}_2$ with arbitrary precision. Then we use:

$$\vec{p} = \vec{n}_2 \times (\vec{p} \times \vec{n}_2) + (\vec{p} \cdot \vec{n}_2)\vec{n}_2$$

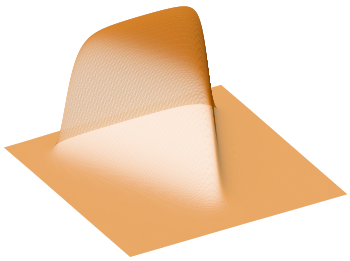
Hence:

$$\begin{aligned}
\frac{2\pi}{\rho} B(x) &= -\vec{n}_1 \cdot \sum_i \left(B_i I_1(i) + \delta B_i J_1(i)\right)(\vec{r}_i \times \vec{e}_i) \\
&\quad - (\vec{n}_1 \cdot \vec{n}_2) \sum_i (\vec{k} \cdot \vec{e}_i) K_1(i) \\
&\quad + (\vec{r}_0 \cdot \vec{n}_2)(\vec{n}_2 \cdot (\vec{n}_1 \times \vec{k})) X_1 \\
-\frac{2\pi}{\rho} \nabla(B)(x) &= \sum_i \left(B_i I_1(i) + \delta B_i J_1(i)\right)(\vec{n}_1 \times \vec{e}_i) \\
&\quad + 2 \sum_i \vec{n}_1 \cdot (\vec{r}_i \times \vec{e}_i) B_i \left(I_2(i)\vec{r}_i + J_2(i)\vec{e}_i\right) \\
&\quad + 2 \sum_i \vec{n}_1 \cdot (\vec{r}_i \times \vec{e}_i) \delta B_i \left(J_2(i)\vec{r}_i + K_2(i)\vec{e}_i\right) \\
&\quad - (\vec{n}_1 \cdot \vec{n}_2) \sum_i (\vec{k} \cdot \vec{e}_i) \left(I_1(i)\vec{r}_1 + J_1(i)\vec{e}_1\right) \\
&\quad - (\vec{n}_2 \cdot (\vec{n}_1 \times \vec{k}))(\vec{n}_2 \cdot \vec{r}_0)\vec{n}_2 \times \sum_i I_2(i)\vec{e}_i \\
&\quad + (\vec{n}_2 \cdot (\vec{n}_1 \times \vec{k})) \left(X_1 - 2(\vec{r}_0 \cdot \vec{n}_2)^2 X_2\right) \vec{n}_2
\end{aligned}$$

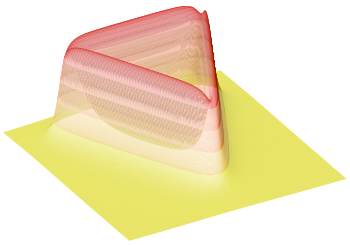
With:

$$\begin{aligned}
\delta B_i &= B_{i+1} - B_i \\
J_1(i) &= \frac{1}{\|\vec{e}_i\|^2} \left[\ln\left(\frac{\|\vec{e}_{i+1}\|}{\|\vec{e}_i\|}\right) + (\vec{r}_i \cdot \vec{e}_i) I_1(i)\right] \\
K_1(i) &= \frac{1}{2\|\vec{e}_i\|^2} \left(\vec{r}_{i+1} \cdot \vec{e}_i \ln(\|\vec{e}_{i+1}\|^2) - \vec{r}_i \cdot \vec{e}_i \ln(\|\vec{e}_i\|^2) + 2\|\vec{r}_i \times \vec{e}_i\|\gamma_i\right) \\
K_2(i) &= \frac{1}{\|\vec{e}_i\|^2} \left(I_1(i) - \|\vec{r}_i\|^2 I_2(i) - 2(\vec{r}_i \cdot \vec{e}_i) J_2(i)\right) \\
X_2 &= \int_{A_2} \frac{dA_2}{\|\vec{r}_{12}\|^4}
\end{aligned}$$

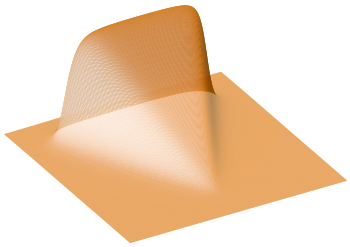
If the the distance between point x and the emitter surface is null, $\vec{m} \cdot \vec{n}_2$ and $\vec{p} \cdot \vec{n}_2$ are both null. If it is not, we prefer to estimate X_1 and X_2. As we know bounds on the values of the function and its derivatives, we make use of a Gaussian quadrature.



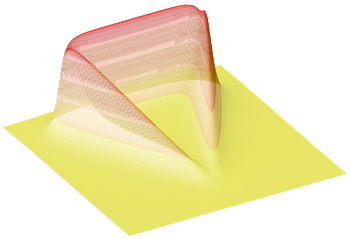
A. Radiosity on the Receiving Plane, due to a Constant Emitter.



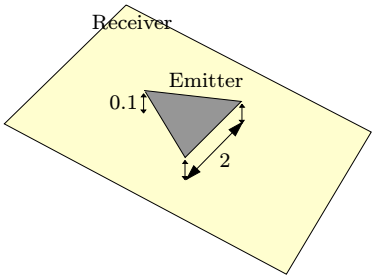
B. Norm of Radiosity Gradient, due to a Constant Emitter.



C. Radiosity on the Receiving Plane, due to a Linear Emitter



D. Norm of the Radiosity Gradient, due to a Linear Emitter



E. Geometry of our Test Scene

3.4.3 An exhaustive error-bounding algorithm for hierarchical radiosity (CGF '98)

Auteurs : Nicolas H et François S
Journal : *Computer Graphics Forum*, vol. 17, n° 4.
Date : décembre 1998

Volume 17 (*1998*), *number 4 pp. 197—218*

An exhaustive error-bounding algorithm for hierarchical radiosity

Nicolas Holzschuch[†]

François X. Sillion

*i*MAGIS[‡]

GRAVIR/IMAG - INRIA

Abstract

This paper presents a complete algorithm for the evaluation and control of error in radiosity calculations. Providing such control is both extremely important for industrial applications and one of the most challenging issues remaining in global illumination research.

In order to control the error, we need to estimate the accuracy of the calculation while computing the energy exchanged between two objects. Having this information for each radiosity interaction allows to allocate more resources to refine interactions with greater potential error, and to avoid spending more time to refine interactions already represented with sufficient accuracy.

Until now, the accuracy of the computed energy exchange could only be approximated using heuristic algorithms. This paper presents the first exhaustive algorithm to compute fully reliable upper and lower bounds on the energy being exchanged in each interaction. This is accomplished by computing first and second derivatives of the radiosity function where appropriate, and making use of two concavity conjectures. These bounds are then used in a refinement criterion for hierarchical radiosity, resulting in a global illumination algorithm with complete control of the error incurred.

Results are presented, demonstrating the possibility to create radiosity solutions with guaranteed precision. We then extend our algorithm to consider linear bounding functions instead of constant functions, thus creating simpler meshes in regions where the function is concave, without loss of precision.

Our experiments show that the computation of radiosity derivatives along with the radiosity values only requires a modest extra cost, with the advantage of a much greater precision.

1. Introduction

Global illumination algorithms now have many applications. One of the most promising fields is in urban and architectural planning, where the use of a global illumination algorithm allows to visualize a future building, and thus to check for misconceptions. For example, it becomes possible to check the ergonomy of the workplace — is there enough light, or too much? — or to ensure that the items in a museum are properly lit.

In such applications, it is vital to be able to quantify the light arriving on each point of the scene, in order to give the user a precise range in which the illumination is guaranteed to fall.

Global illumination algorithms generally have at least a parameter that the user can manipulate, choosing either fast computations or precise results. For Monte-Carlo ray tracing algorithms, this parameter can be the number of rays. For hierarchical radiosity algorithms, it can be the refinement threshold, used to decide whether or not to refine a given

† Current position: Invited Researcher, Department of Computer Science, University of Cape Town, South Africa.

‡ *i*MAGIS is a joint research project between CNRS, INRIA, INPG and Université Joseph Fourier — Grenoble I. Postal address: B.P. 53, F-38041 Grenoble Cedex 9, France. E-mail: `Nicolas.Holzschuch@imag.fr`.

interaction. Until recently, however, we had little knowledge of the total precision of the result computed, or of the relation between the parameters and this precision. Even if it was clear that spending more time on the simulation would produce more precise results, we could not quantify precisely this increase.

In 1994, Lischinski[1] proposed a refinement criterion for hierarchical radiosity such that the error on the energy at each point of the scene could be controlled by the refinement threshold. Their algorithm used upper and lower bounds on the point-to-area form factor for each interaction in order to compute upper and lower bounds for the radiosity at each point in the scene. However, they had no way to compute reliable upper and lower bounds for the point-to-area form-factor on a given interaction, and still resorted to sampling — computing a set of values for the form-factor, and taking the minimum and maximum of these values.

Although Lischinski's method is easy to implement, it is not totally reliable. In this paper, we present a method allowing to compute fully reliable upper and lower bounds for the point-to-area form-factor on any interaction. To achieve this goal, we use our knowledge of the point-to-area form-factor derivatives together with its concavity properties.

These concavity properties of the point-to-area form-factor are described in section 3. They extend the unimodality conjecture proposed by Drettakis[2, 3]. Like the unimodality conjecture, they are only conjectures, and despite their apparent simplicity, we have been unable to find a complete demonstration for them. However, we also have been unable to exhibit a counter-example.

As is explained in appendix B, we can compute exact values for the derivatives of the point-to-area form-factor; either for the first derivative, the gradient vector, or for the second derivative, the Hessian matrix. As we shall also see in appendix B, it is indeed faster to compute an exact value for the form-factor derivative than computing approximate values using several samples. Using our knowledge of the derivatives along with the concavity properties of the point-to-area form-factor, we show in section 4 how to derive bounds for the point-to-area form-factor in any unoccluded interaction. We also show an implementation of the refinement criterion using these bounds.

When dealing with partially occluded interactions we can not use the previous bounds, as the concavity conjectures do not hold in this case. But we can exhibit two emitters that are convex and bound the actual emitter, which we call the minimal and the maximal emitter. Using the previously defined algorithm, we find an upper bound for the maximal emitter, and a lower bound for the minimal emitter. The algorithm for finding these convex emitters is detailed in section 5.

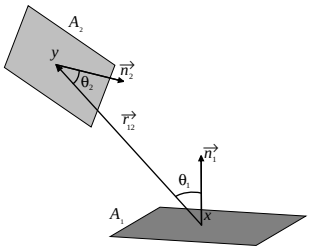


Figure 1: *Geometric notations for the radiosity equation.*

2. Background

The radiosity method was introduced in the field of light transfer in 1984 by Goral[4]. This method uses a simplification in order to solve the global illumination problem: it assumes that all the objects in the scene are ideal diffuse surfaces: their bidirectional reflectance is uniform, and thus does not depend on the outgoing direction.

In this case, the radiosity emitted at a given point x can be expressed as an integral equation:

$$B(x) = E(x) + \rho_d(x) \int_{y \in S} B(y) \frac{\cos\theta_1 \cos\theta_2}{\pi r^2} V(x,y) dy \tag{1}$$

In this equation, S is the set of all points y. r is the distance between point x and point y, θ_1 and θ_2 are the angles between the $\overrightarrow{xy}$ vector and the normals to the surfaces at point x and y respectively (refer to figure 1 for the geometric notations). $\rho_d(x)$ is the diffuse reflectance at point x, and $V(x,y)$ expresses whether point x is visible from point y or not.

In order to solve equation 1, Goral[4] suggested to discretize the scene into a set of patches $[P_i]$, over which a constant radiosity, B_i is assumed.

In this case, the radiosity at point x becomes:

$$B(x) = E(x) + \rho_d(x) \sum_i B_i \int_{y \in P_i} \frac{\cos\theta \cos\theta'}{\pi r^2} V(x,y) dy \tag{2}$$

The purely geometric quantity

$$F_i(x) = \int_{y \in P_i} \frac{\cos\theta \cos\theta'}{\pi r^2} V(x,y) dy$$

is called the point-to-area form-factor at point x from patch i. It only depends on the respective positions of point x and patch i.

Since we assume a constant radiosity value within the patch, we can compute this value as the average of all the point values. This leads to a matrix equation:

$$B_j = E_j + \rho_i \sum_i F_{ji} B_i \tag{3}$$

where the geometric quantity

$$F_{ij} = \frac{1}{A_j} \int_{x \in P_j} \int_{y \in P_i} \frac{\cos\theta \cos\theta'}{\pi r^2} V(x,y) dx dy$$

is called the form-factor. Schröder[5] showed that there is a closed form expression for the form-factor in the case of two fully visible polygonal patches. In the general case, we do not have access to the exact value of the form-factor, but only to approximate values.

Equation 3 can be solved in an iterative manner, using Jacobi or Gauss-Seidel iterative methods (see Cohen[6]). The problem is that in order to compute one full bounce of light across the surfaces in the scene, we have to compute the entire form-factor matrix, which is quadratic with respect to the number of patches.

A significant improvement over the classical radiosity method is hierarchical radiosity. In "standard" radiosity, the discretisation of one object into patches does not depend on the objects with which it interacts. In order to model the interaction between objects that are very close, and exchange lots of energy, we need to subdivide them into many patches, so as to get a precise modelling of the radiosity. On the other hand, an interaction between two objects that are far away could be modelled with fewer patches.

In hierarchical radiosity, introduced in 1990 by Hanrahan[7], each object is subdivided into a *hierarchy* of patches, with each node in the hierarchy carrying the average of the radiosity of its children. Interaction between objects far away from each other are modelled as interactions between nodes at a high level in each hierarchy. On the other hand, interactions between objects close to each other are modelled as interactions between nodes at a lower level in the hierarchy, thereby allowing more precision in the modelling of radiosity. Each interaction between two nodes is modelled by a *link*, a data structure carrying the identity of the sender and the receiver, as well as the form-factor, and possibly other informations on the respective visibility of both patches. This hierarchical radiosity algorithm has later been extended using wavelets (see Gortler[8]).

The most important step in the hierarchical radiosity method is the decision whether or not to refine a given interaction. This decision is deferred to a *refinement criterion*. Early implementations of the hierarchical radiosity method used crude approximations of the form-factor between two patches. It was known that these form-factor estimates were most imprecise when the result of the approximation was large. Hence, interactions were refined as long as the form-factor estimate was above a certain threshold (Hanrahan[7]).

This refinement criterion does not give the user a full control of the precision on the modelling of the radiosity function. In particular, it does not give any *guarantee* that it will refine all problematic interactions, and it can also refine excessively in places where the solution has already attained a correct level of precision (Holzschuch[9]).

Part of these problems can be addressed by using discontinuity meshing, where the patches are first subdivided along the discontinuity lines of the radiosity function and its derivatives (see Heckbert[10], Lischinski[11, 12] and Drettakis[13]). These discontinuity lines can be computed using geometric algorithms. However, as pointed out by Drettakis, these discontinuity lines are not of equal importance. Some of them do not have a noticeable effect on the final radiosity solution. Hence it is not necessary to compute all the discontinuity lines. Deciding which discontinuity lines are relevant is done by a refinement oracle, using heuristic methods like the one described above.

Many of the latest research results have dealt with giving the user a better control of the level of precision in the modelling of radiosity in the hierarchical radiosity method.

In the most promising paper on the subject, Lischinski[1], suggested to compute for each interaction an upper and lower bound for the point-to-patch form-factor between the points of the receiving patch and the emitting patch, namely $F_{\max}$ and $F_{\min}$, as well as an upper and lower bound for the radiosity of the emitting patch, using information already available in the hierarchy. We then know that the radiosity on the receiving patch is between $F_{\max}B_{\max}$ and $F_{\min}B_{\min}$.

Hence, the uncertainty on the radiosity on the receiving patch, due to this particular interaction is:

$$\delta B_{\text{receiver}} = F_{\max}B_{\max} - F_{\min}B_{\min}$$

The inaccuracy on the energy of the receiving patch, due to this particular interaction, is:

$$\delta E_{\text{receiver}} = A_{\text{receiver}} \left(F_{\max}B_{\max} - F_{\min}B_{\min}\right)$$

We can then decide to refine all interactions where this imprecision on the transported energy is above a given threshold. The most difficult part in this algorithm is finding reliable values for the bounds on the form-factor. Lischinski[1] suggested computing exact values for the point-to-area form factor at different sampling points on the receiver, and using the maximum and minimum value at these sampling points as the upper and lower bounds. Although this algorithm does not give totally reliable bounds, it does provide a close approximation, and is quite easy to implement on top of an existing hierarchical radiosity implementation.

In the following sections we show that it is possible to compute reliable upper and lower bounds for the point-to-area form factor. These bounds can then be used in the preceding algorithm, allowing the refinement of all interactions where the inaccuracy on the transported energy is above the threshold.

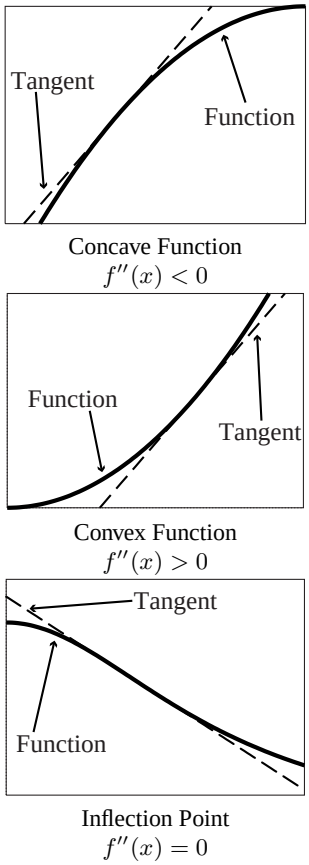


Figure 2: *Concavity for univariate functions.*

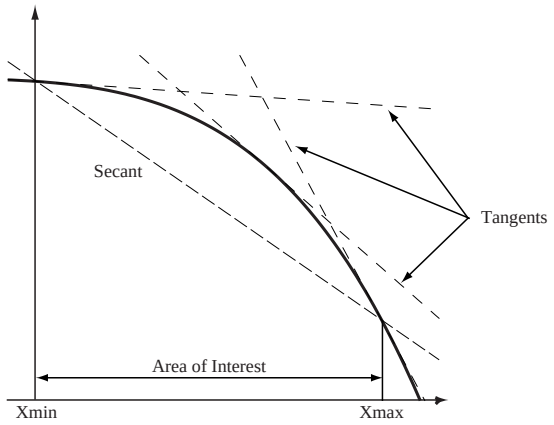


Figure 3: *A function that remains concave across an interval lies above its secant, and below all its tangents on this interval.*

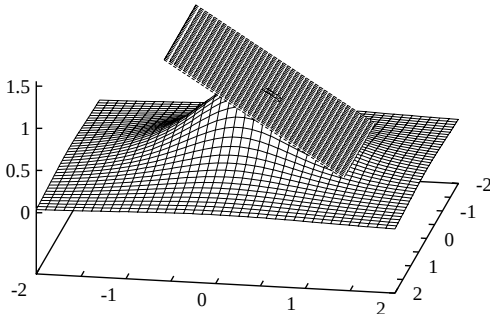


Figure 4: *A point where the function is concave: the function lies* below *the tangent plane.*

3. The Concavity Conjectures

3.1. Definition of Concavity

Univariate functions are said to be concave at a point when they lie entirely below their tangent at that point; conversely, they are said to be convex when they lie above their tangent. When the function crosses its tangent, the point is said to be an inflection point (see figure 2). Classically, the concavity of the function is linked to the sign of its second derivative: if the second derivative is positive, then the function is convex. If it is negative, then the function is concave. It is only when the second derivative changes sign that we have an inflection point.

Concavity is often used to find upper and lower bounds for functions; if a function is concave on an interval, then it is below all its tangents on this interval, and above all its secants (see figure 3). Since concavity allows bounding by affine functions (like tangents) instead of constants, it generally provides bounds that are closer to each other, and hence a "better" range.

This notion of concavity extends naturally to bivariate functions, such as radiosity defined over a surface. A bivariate function is said to be concave at a point when it lies below its tangent plane (see figure 4), convex when it lies above its tangent plane and indefinite when the function crosses the tangent plane (see figure 5). As with univariate functions, concavity can be used to find upper and lower bounds: if a function is concave over a triangular area, then on this area it lies below all its tangent planes, and above the secant plane defined by the three corners of the triangle.

A univariate function usually crosses its tangent at an isolated point, the inflection point. Contrarily, the set of points where a bivariate function crosses its tangent plane is a whole region.

The second derivative of a bivariate function is a 2×2 matrix, called the *Hessian matrix*. As with univariate functions, the concavity of the function is linked to its second derivative: if the Hessian matrix is definite positive, then the function is convex; if the Hessian matrix is definite negative, then the function is concave; if the Hessian matrix is indefinite, then the function is indefinite. The Hessian can be ex-

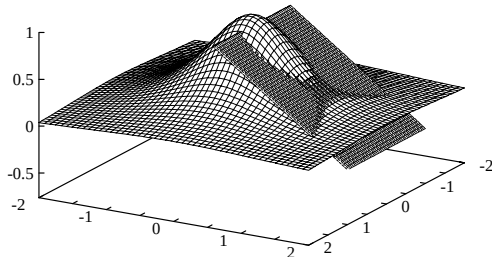


Figure 5: *A point where the concavity is indefinite: the function crosses its tangent plane.*

pressed with respect to the partial derivatives of the function:

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial y^2} & \frac{\partial^2 f}{\partial u \partial v} \\ \frac{\partial^2 f}{\partial u \partial v} & \frac{\partial^2 f}{\partial v^2} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} r & s \\ s & t \end{bmatrix} \qquad (4)$$

The Hessian is definite if $rt - s^2$ is positive. It is definite-positive if $rt - s^2$ is positive and r is positive, definite-negative if $rt - s^2$ is positive and r is negative. If $rt - s^2$ is negative or null, the Hessian is indefinite, and the function crosses its tangent plane.

It must be noted that a function is necessarily concave where it has a local maximum, and convex wherever it has a local minimum. This property is true both for uni- and bivariate functions.

3.2. Concavity of the Point-To-Area Form Factor

3.2.1. Background

Let us single out an interaction between an emitting patch and a receiving patch. We seek an upper and a lower bound for the point-to-area form-factor across the receiver. These upper and lower bounds can then be used by a refinement oracle, as introduced by Lischinski[1].

Using the algorithm described in appendix B, we have access to the form-factor and to its derivatives at any point of the receiver. However, these values are only valid at this specific point. Since we seek a result valid across the whole receiver, we must exhibit a property of the point-to-area form-factor that is valid across the receiver.

A similar approach was used by Drettakis[2, 3]. In the case of a finite convex emitter, with constant radiosity, and of an infinite receiver, Drettakis made the following two conjectures:

Conjecture U1 Radiosity on the receiver has only one maximum.

Conjecture U2 Radiosity on any line on the receiver has only one maximum.

These two conjectures are referred to below as the *unimodality conjectures*.

Figure 6: *The C1 conjecture: the radiosity function has indefinite concavity everywhere, except over a convex area (hatched), where the radiosity function is concave.*

3.2.2. Concavity Conjectures

Like Drettakis, we consider a finite convex emitter, with constant radiosity, and we assume the receiver is an infinite plane. We state the following two conjectures on the concavity of the radiosity on the receiver:

Conjecture C1 The Hessian matrix of the radiosity function is indefinite everywhere, except over a bounded area. On this area, the radiosity function is concave. Furthermore, the area is convex.

Conjecture C2 On any line drawn on the receiver, radiosity is concave over a bounded interval, and convex everywhere else.

Figure 6 illustrates the C1 conjecture: the radiosity function is indefinite everywhere — and crosses its tangent plane — except over a convex region (hatched).

Figure 7 illustrates the C2 conjecture: the radiosity function defined over a given line is convex across $[-\infty, a]$ and across $[b, +\infty]$, and concave across $[a, b]$.

Despite their apparent simplicity, these conjectures have yet escaped demonstration. It is obvious that they are true in the simplest case of a point light source sending light in all directions. However, even for the case of a differential emitter area instead of a point light source, it has not been possible so far to prove the concavity conjectures. Appendix A is a detailed study of the differential emitter area.

3.2.3. Relationship between the conjectures

Our concavity conjectures are actually an extension of the unimodality conjectures: that is, C1 implies U1, and C2 implies U2. Note that we also know that U2 implies U1:

$$\begin{cases} \text{U2} \implies \text{U1} \\ \text{C2} \implies \text{U2} \\ \text{C1} \implies \text{U1} \end{cases}$$

U2 $\implies$ U1:

Proof Assume U1 is false. Then there exists at least two maxima for the radiosity function, M1 and M2. On the line joining M1 and M2 there are two maxima, which is in contradiction with U2. □

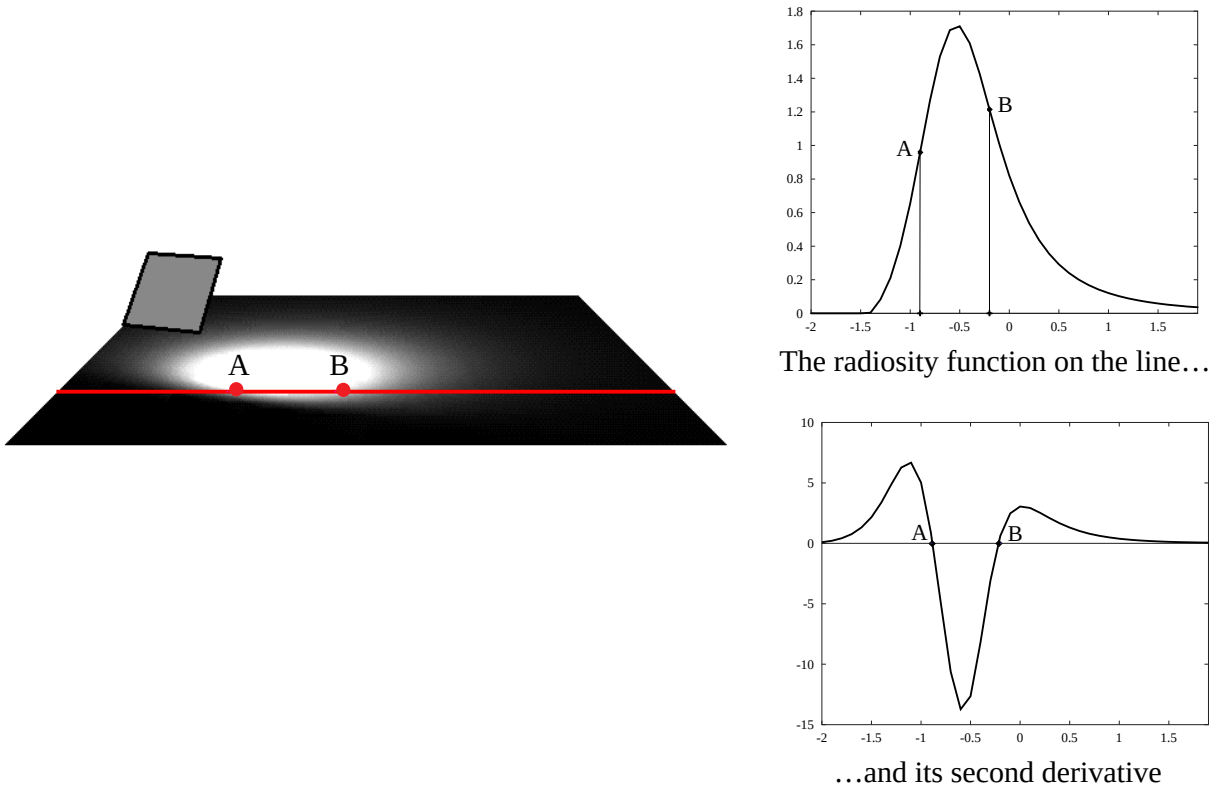


Figure 7: *The C2 conjecture: the radiosity function on a line is concave only over a finite interval, [AB].*

C2 $\Longrightarrow$ U2:

Proof The function is concave on the neighbourhood of each local maximum. If there are two local maxima on a line, there must be a local minimum between them. In the neighbourhood of this local minimum, the function would have to be convex, which is impossible because of C2. □

C1 $\Longrightarrow$ U1:

Proof Assume U1 is false. Then there exists at least two local maxima for the radiosity function. On the neighbourhood of each maximum, the radiosity function is concave. But between the two maxima, there must be a pass-like point, where the concavity is indefinite. This is in contradiction with C1. □

No relationship between C1 and C2: An important point is the *independence* of our two concavity conjectures. C1 does not imply C2, and C2 does not imply C1.

4. Error Control for Unoccluded Interactions

In this section, we describe our algorithm for finding upper and lower bounds for the point-to-area form-factor across the receiver. These values are then used by a refinement oracle like the oracle introduced by Lischinski[1].

4.1. Computing Radiosity Derivatives

Let us call A_2 the emitting patch, A_1 the receiver and x a point on the receiver (see figure 1). In this case, there is an exact formula for the point-to-area form factor (Siegel and Howell[14]):

$$F(x) = -\frac{1}{2\pi}\vec{n}_1 \cdot \oint_{\partial A_2} \frac{\vec{r}_{12}}{\|\vec{r}_{12}\|^2} \times d\vec{\ell}_2 \qquad (5)$$

where the integral is on ∂A_2, the contour of A_2, and $d\vec{\ell}_2$ is the differential element of this contour.

Using this expression of the point-to-area form-factor, it is possible to compute exact formulae for both its first and second derivatives. These formulae for the derivatives are easily implemented, giving access to exact values for the function and its derivatives (see appendix B, and also Arvo[15] or Holzschuch[16, 17]).

If we compute simultaneously the point-to-area form-factor and its derivatives, we can save computation time by reusing some geometric quantities that appear in several formulae. In this case, the overall cost of computing the derivatives is reasonable: there is an increase of 40% for computing the gradient along with the form-factor, and an increase of 100% for computing both the gradient and the Hes-

sian matrix (see appendix B and Holzschuch[16, 17]). This cost must be balanced against what it would require to compute approximate values for the derivatives using several form-factor computations: in this case, the cost increase for the gradient would be of 100%, and that of the Hessian 600%.

In our refinement phase, we compute the values of the point-to-area form-factor and its derivatives at the vertices of the receiving patch. These values can be reused in the radiosity propagation phase to obtain the radiosity values at the vertices.

4.2. Computing Bounds for the Point-to-Area Form-Factor

We show here how our knowledge of the point-to-area form-factor and its derivatives at the vertices of the receiving patch, used jointly with our conjectures, gives us access:

- first, to the *location* of the maximum and the minimum of the point-to-area form-factor,
- second, to an exact value for the minimum,
- third, to an upper bound for the maximum.

4.2.1. The Minimum is at one of the Vertices

An immediate consequence of the unimodality conjectures (U1 and U2) is that the minimum for the point-to-area form-factor is necessarily at one of the vertices of the receiver:

- If the minimum was inside the receiving patch, A1, then there would exist several local maxima for the point-to-area form-factor on the plane supporting A1 — this is in contradiction with U1. Hence, the minimum across A1 must be on the contour of A1.
- The contour of A1 is made of polygonal edges. If on one of these edges the minimum is inside the edge then on the line supporting the edge the form-factor must have two maxima — this is in contradiction with U2.
- Hence, the minimum can only be at one of the vertices of A1.

4.2.2. An exact value for the minimum

Since we chose to compute the point-to-area form-factor at the vertices of the receiving patch, A1, we do have access to the exact value of the minimum across A1: it is the minimum of our computed values for the point-to-area form-factor at the vertices of A1.

4.2.3. Finding the Position of the Maximum

A consequence of U2 is that given a point x, given the point-to-area form-factor $F(x)$ and its gradient at point x, $\nabla F(x)$, for all points p such that $\overrightarrow{xp} \cdot \nabla F(x) < 0$, we have $F(p) < F(x)$.

Otherwise, there would be one local minimum between p and x on the line passing through p and x, and hence two local maxima, which is in contradiction with U2.

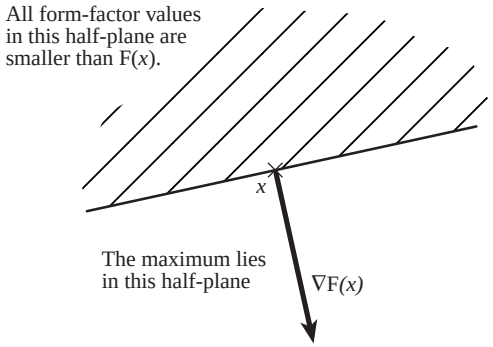


Figure 8: *Knowledge of the gradient helps find the position of the maximum.*

Hence the maximum of the point-to-area form-factor can only be in the half-plane defined by: $\overrightarrow{xp} \cdot \nabla F(x) \geq 0$ (see figure 8.)

This property gives us an algorithm to determine whether the maximum for the point-to-area form-factor across the receiving patch A1 can lie *inside* the patch, or if it must be at one of the vertices (see figure 9):

- For each vertex, there is a half-plane (defined by the form-factor gradient at this vertex) where the form-factor value can be greater than the value at the vertex.
- The intersection of these half-planes is an area where the point-to-area form-factor value can be greater than the value at all the vertices. The intersection of this area with the receiving patch is either empty or not empty.
- If this intersection with the patch is not empty, then there exists an area inside the patch where the maximum can be.
- If this intersection is empty, then the maximum for the form-factor across the patch must be at one of the vertices.

4.2.4. If the Maximum is at one of the Vertices

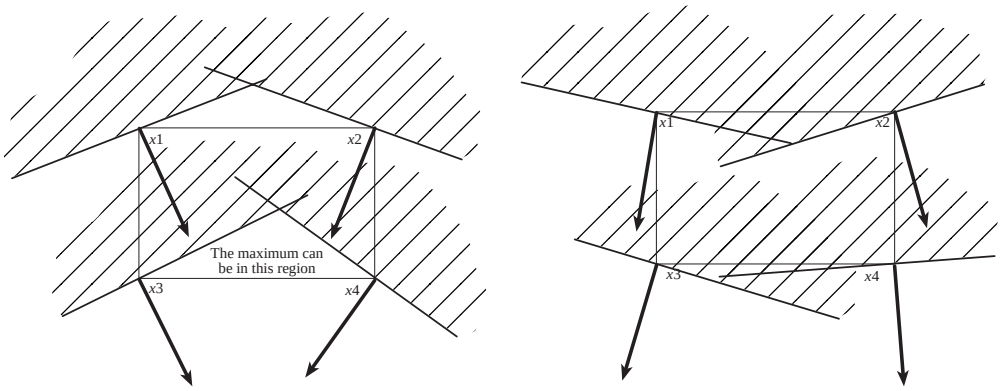
If the above algorithm tells us that the maximum can only be at one vertex of the receiving patch, then we know the exact value of the maximum: it is the value of the point-to-area form-factor at that vertex.

4.2.5. If the Maximum is Inside the Receiving Patch

If the above algorithm tells us there exists an area inside the receiving patch A1 where the maximum can be, then we do not have access to the exact value of the maximum of the point-to-area form-factor across A1.

The only thing we know at this stage is that the value of the maximum must be greater than the values computed at the vertices of A1.

There are three kind of algorithms for finding an upper bound for the point-to-area form-factor across A1:



The maximum can be inside the polygon. The maximum is at one of the vertices.

Figure 9: *Using the gradient to locate the maximum inside or outside the receiving patch.*

Heuristic Algorithms: Compute another sample value for the point-to-area form-factor inside patch A1. The position of the sampling point can be arbitrary or can make use of the information given by the form-factor gradient.

Concavity Algorithms: If the point-to-area form-factor function on the receiving patch is concave, we use the tangent planes to find an upper-bound.

Geometric Algorithms: Using geometric tools, build an emitter that encloses the actual emitter for all the points of the receiving patch, and for which we can find the value of the maximum. This value is an upper-bound.

Heuristic algorithms include gradient descent algorithms, as described by Arvo[15] and Drettakis[2, 3]. Gradient descent algorithms make use of the information provided by the gradient to subdivide the receiving patch until convergence. The gradient can either be approximated (Drettakis[2, 3]) or an exact value (Arvo[15]).

In our implementation, we use concavity algorithms wherever possible, and resort to geometric algorithms if the point-to-area form-factor function is not concave.

4.2.5.1. Concavity Algorithms According to C1, the zone where the point-to-area form-factor function is concave is a convex one. As a consequence, if the form-factor Hessian is definite negative at the vertices of the receiving patch, then it stays definite negative across the receiving patch.

In this case, the form-factor function lies below all its tangent planes at the vertices across the receiving patch. We know these tangent planes since we know the form-factor gradient at the vertices. Finding an upper bound for the point-to-area form-factor is then equivalent to computing the intersection of the tangent planes.

This is mainly a linear programming problem (see, for example, Preparata[18]); the computational complexity of the problem depends on the dimension of the problem — which here is always two since we are dealing with bivariate functions — and on the number of vertices in the receiving patch. Usually, in hierarchical radiosity algorithms, we are restricting ourselves to triangular or quadrangular patches. If this is the case, we can assume the complexity of computing the intersection of the tangent planes is constant.

4.2.5.2. Geometric Algorithms If the form-factor Hessian is not definite negative at all the vertices of the receiving patch, then the point-to-area form-factor function is not concave across the entire receiving patch. It is therefore not possible to use concavity algorithms. In this case, we resort to geometric algorithms: in a plane parallel to the plane of the receiver, we construct an emitter with the following two properties:

- From all the points of the receiver, it is seen as including the original emitter.
- It has two axes of symmetry, so that we can find the maximum form-factor due to the emitter.

The reason for the second item lies in the symmetry principle: if the emitter and the receiver are left unchanged by a planar symmetry, then so is the point-to-area form-factor function on the receiver; thus its maximum can only lie on the intersection of the plane of the symmetry and of the plane of the receiver. If there are two planes that leave the emitter and the receiver un-changed, then the maximum can only be at their intersection (see figure 24, in the color section).

To build this emitter:

- select a plane P parallel to the plane of the receiving patch;
- for each vertex V_i of the receiving patch, build the projection p_i of the original emitter according to this vertex on P (see figure 10);
- this projection is totally equivalent to the original emitter for this particular vertex;

- any convex region enclosing all the p_i projections is seen from all the points of the receiver as enclosing the original emitter; as a consequence, the point-to-area form-factor due to this convex region is greater than the point-to-area form-factor due to the actual emitter;
- building a convex region enclosing the p_i is a standard geometry problem (see, for example, Foley *et al.*[19], Kay[20] or Toth[21]). Constraining this convex region to have two axes of symmetry can either be a consequence of the bounding object used, like ellipses and rectangles, or be a property we add afterward. Since our problem is a two dimensional geometry problem — although we have a set of three dimensional data points — we start by projecting our p_i onto one of the coordinates planes (x, y), (z, x) or (y, z). Once we have built the result in this coordinate plane, we will project it back onto the emitter plane.
- Several algorithms can be used, either giving a faster result, but a greater enclosing emitter, and hence a greater upper-bound, or requiring more time, but giving an enclosing emitter that is closer to the p_i, and hence a smaller upper-bound:
  - Build the convex hull of the p_i, then build a region with two axes of symmetry enclosing the convex hull. This gives the smaller enclosing emitter, but requires more computation time
  - Build a bounding rectangle enclosing the p_i inside the emitter plane, as in Toth[21]. This is one of the fastest possible algorithm. Furthermore, it naturally gives an enclosing emitter with two axes of symmetry, so there is no construction time involved for building the symmetries.
  - Build a bounding ellipse enclosing the p_i inside the emitter plane. This algorithm is slower, but it also gives an enclosing emitter with two axes of symmetry, so there is no construction time involved for building the symmetries.
  - A bounding rectangle using the (x, y, z) axes can give an enclosing emitter much bigger than the p_i, thus inducing a greater upper bound. A simple improvement is to use *slabs*, as suggested in Kay[20]. In this case, in order to build an object with two axes of symmetry, we have to restrict ourselves to two sets of orthogonal slabs. This algorithm requires more computational time than the previous algorithm, but can give a significantly smaller enclosing emitter.
  - If n_e is the number of vertices of the emitter, and n_r the number of vertices of the receiver, the total number of vertices for all the p_i is $n_e n_r$. In this case, the complexity of the convex hull algorithm is $O(n_e n_r \log n_e n_r)$, and the complexity of the other three algorithms is $O(n_e n_r)$.

Figure 11 gives an example of the construction of an enclosing emitter.

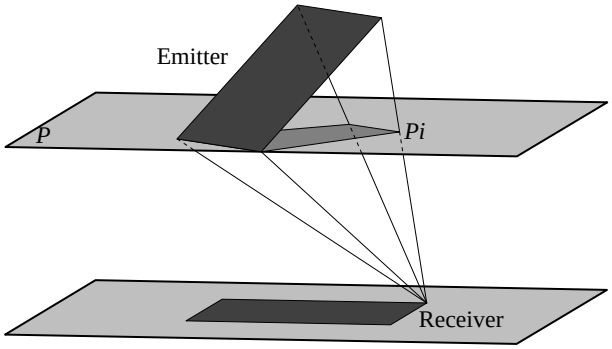


Figure 10: *The projection of the emitter on the plane P from a given vertex.*

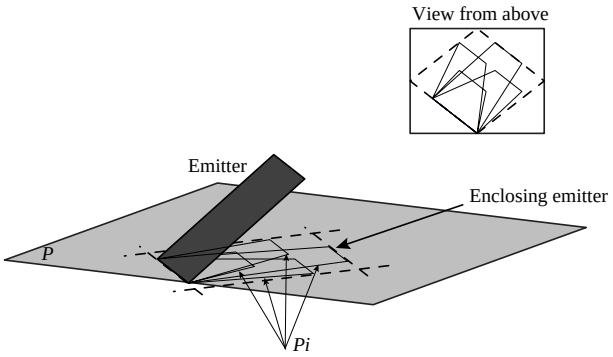


Figure 11: *Building an enclosing emitter in order to find an upper bound.*

4.3. Implementation and Testing

4.3.1. Refinement Criterion

Once we have access, for each iteration, to the minimum and maximum form-factor, it is possible to implement a refinement criterion based on their difference. Following the algorithm suggested by Lischinski[1], we refine every interaction such that:

$$A_{\text{receiver}} \left(B_{\max} F_{\max} - B_{\min} F_{\min}\right) > \varepsilon$$

This means that we refine an interaction whenever the uncertainty on the incoming energy of the receiving patch is above the threshold ε.

4.3.2. Resulting Mesh Simplification

In regions where the Hessian matrix of the form-factor is definite-negative, we know that the form-factor can be bounded between the tangent planes and the secant planes. We can use these bounding planes to find tighter upper and lower bounds for the form-factor.

The form-factor for all the points on the receiving patch

lies below all the tangent planes for the point-to-area form factor, and above the secant plane. Therefore, we can say that our uncertainty on the point-to-area form-factor on the receiver is equal to the maximum of the distance between the secant plane and these tangent planes.

Computing this distance is again a linear programming problem (see, for example, Preparata[18]). The complexity depends on the number of vertices of the receiver, n_r, which is usually three or four. Let us denote by E_{FF} this uncertainty on the form-factor. E_{FF} can be used in our expressions as a replacement for $F_{\max} - F_{\min}$. Using the fact that:

$$(B_{\max}F_{\max} - B_{\min}F_{\min}) = \\ B_{\max}(F_{\max} - F_{\min}) + F_{\min}(B_{\max} - B_{\min})$$

we decide to refine a given interaction if

$$A_{\text{receiver}}\,(B_{\max}E_{FF} + F_{\min}(B_{\max} - B_{\min})) > \varepsilon$$

It must be noted that this new bounding of the form-factor does not introduce any uncertainty. We are still bounding the form-factor by fully reliable functions. However, since these functions are affine instead of constants, they provide much tighter bounds, and we can expect a simpler mesh in the areas where the point-to-area form factor is concave.

Figure 25 (in the color section) shows the result of our refinement criterion on a simple box, with only direct illumination. Notice that the mesh produced is coarser in some areas with respect to the immediately neighbouring areas (the disc-shaped area on the floor, and the drop-shaped areas on the walls). These are the places where the Hessian is definite-negative.

This refinement criterion extends, in some ways, the mesh simplification found in previous work (Holzschuch[9]). The shape of the mesh produced is quite similar between our new algorithm and the algorithm in Holzschuch[9]. However, our new refinement criterion, while keeping low memory costs, also gives fully reliable upper and lower bounds on the radiosity of each patch.

4.3.3. Dealing with Singularities

4.3.4. Relative Complexity of the Algorithm

Our algorithm requires the computation of the first two derivatives of the point-to-area form-factor at the vertices of the receiver. This implies a 100 % increase on the computation time for each vertex (see appendix B). That is to say, computing the point-to-area form-factor and its derivatives costs twice what it would cost to compute the point-to-area form-factor alone.

Since vertices are shared by several patches, this overhead cost is shared by several interactions. On the average, we are only computing one point-to-area form-factor and its derivatives for each patch. Thus, the cost of our algorithm is approximately the cost of computing two point-to-area form-factors for each patch, plus the time needed for the exploitation of the derivatives for computing upper and lower bounds.

Existing heuristic refinement algorithms (see Lischinski[1]) compute one form-factor sample for each of the receiver vertices, plus one sample at the center of the receiving patch. If we assume that the form-factor values at the vertices are shared with the neighbouring patches, we are computing an average of two point-to-area form-factors for each receiver.

Thus, the cost of the heuristic algorithm and the cost of our algorithm are roughly similar. The main overhead of our algorithm when compared with the heuristic algorithm is the time needed for the actual computations for finding the position of the maximum and for finding an upper bound for the maximum, when necessary.

Hence, the relative costs of our refinement criterion are in fact quite small and can be generally regarded as acceptable, especially with respect to the complete control it gives on the error carried by each interaction.

Also, our algorithm allows for a significant mesh simplification (see figure 25, in the color section) which may, depending on the scene considered, induce a smaller computation time for the exhaustive refinement criterion when compared to a heuristic refinement criterion.

5. Error Control for Partially Occluded Interactions

The above algorithm for finding upper and lower bounds only works in the case of unoccluded interactions, and with a convex emitter. This algorithm relies on the concavity and unimodality conjectures, which do not hold if there are occluders between the emitter and the receiver.

However, it is possible to construct, using geometrical tools, a minimal and a maximal emitter that have the following qualities:

- both are convex;
- any point of the minimal emitter is fully visible from the receiver;
- the maximal emitter contains all the points of the emitter that are visible from at least one point of the receiver;

Then at any given point on the receiver,

- the form-factor due to the minimal emitter is lesser or equal to the actual form factor,
- and the form-factor due to the maximal emitter is greater or equal to the actual form-factor.

We apply our previous algorithm to these emitters, and find a lower bound using the minimal emitter, and an upper bound using the maximal emitter.

Figure 26 (in the color section) shows an example of minimal and maximal emitters for a simple configuration with only one occluder: the small red square on the ground is the

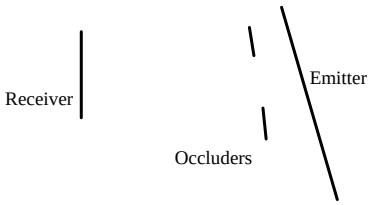


Figure 12: *A single interaction with occluders.*

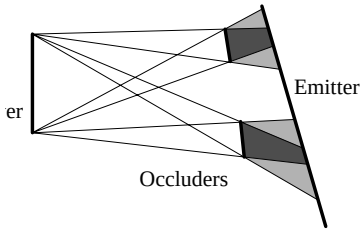


Figure 13: *Computing the "umbra" and "penumbra" volumes using the receiver as a light source.*

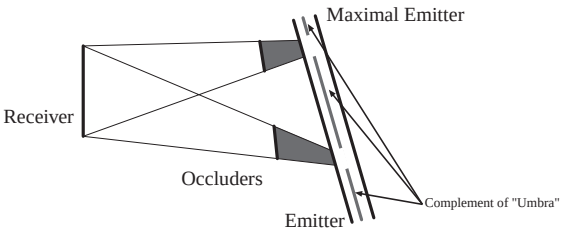


Figure 14: *The maximal emitter can be any convex including the complement of the "umbra" region.*

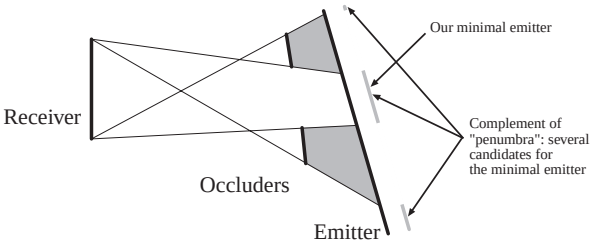


Figure 15: *Several possible candidates for the minimal emitter.*

receiver; the black square with a white border is the occluder, and the bright red area is the minimal emitter — the part of the emitter that is visible from all the points of the receiver. The dark red area is the maximal emitter. The blue line is the contour of the emitter as it is seen from one of the points of the receiver.

5.1. Computing the minimal and maximal emitter

Our definition of minimal and maximal emitter bears a strong resemblance with the definition of umbra and penumbra, except that the roles of the emitter and the receiver are *reversed*.

A similar algorithm has been used by Teller to computer the antipenumbra of an area light source[22], and to solve the visibility problem in a hierarchical radiosity algorithm[23], and by Drettakis[13]. Drettakis[13] used a specific data structure, the *backprojection*, which gives to the program the structure of the projection of the occluders on the emitter plane, from any point on the receiver.

Algorithms used for computing umbra and penumbra can be quickly adapted in order to compute the minimal and maximal emitter for each receiver. Let us consider a single interaction, with one emitter, one receiver, and occluders (see figure 12). We compute the umbra and the penumbra volume using the receiver as a light source (see figure 13). The intersection of these volumes with the emitter plane is a close indication of where the minimal and maximal emitter are.

5.1.1. Computing the maximal emitter using the "umbra" volume

The intersection of the emitter with the "umbra" volume is the set of points on the emitter that are totally invisible from the receiver.

The complement of this intersection is the set of points on the emitter that are visible from at least one point on the receiver.

Since our criterion only works for convex emitters, we have to build a convex emitter that includes this complement. Our basic rule is that we must not under-estimate the point-to-area form-factor, only possibly over-estimate it. Hence, the maximal emitter must be any convex region including the previously computed complement — for example the convex hull of the complement, or the bounding-box of the complement (see figure 14).

5.1.2. Computing the minimal emitter using the "penumbra" volume

Similarly, the intersection of the emitter with the "penumbra" volume is the set of points on the emitter that are at least partially hidden from the receiver.

The complement of this intersection is the set of points on the emitter that are visible all the points of the receiver.

Any convex region that is included in this complement is a suitable candidate for the minimal emitter (see figure 15).

Depending on the position of the occluders, it is possible

to have several candidates for the minimal emitter. Ideally, we would like to pick the candidate that gives the largest estimate for the minimum, since this would give tighter bounds, and hence reduce the number of un-necessary refinements. However, it is impossible to find this without computing the point-to-area form-factor for all the candidates, which would prove very time-consuming. In our implementation, we choose the candidate with the largest area, since it is likely to induce a larger form-factor.

5.2. Implementation and testing

We have implemented our algorithm for finding upper and lower bounds for the point-to-area form-factor using the maximal and minimal emitter.

Figure 27 (in the color section) shows the result of our refinement criterion on a simple scene, with a single occluder. Notice that the algorithm detects the shadow boundaries and refines properly in order to model them. Outside of the shadow, the mesh produced is identical to the mesh produced without occluders.

5.3. Complexity of the Algorithm and Possible Improvements

Our algorithm relies on computation of the umbra and penumbra volumes for all the interactions. This computation can be quite costly, if it is implemented in a naive way.

Previous work by Chin[24] has shown that the use of a BSP-tree can greatly improve the computation of umbra and penumbra volumes. Teller[22] showed that by extending the data structure used to store the interaction between patches to also store the possible occluders for this interaction, the complexity of visibility computations could be greatly reduced. Both these improvements work with our algorithm.

Our algorithm can also be used in a combination with standard discontinuity meshing, as described in Lischinski[11]. A preliminary light-source discontinuity meshing will reduce the complexity of the minimal and maximal emitter computations by providing occlusion information and reducing the number of patches where we have to compute these emitters.

The backprojection algorithm described by Drettakis[13, 3] gives for each patch created during the discontinuity meshing step the geometric structure of the emitter as seen from this patch. Implementing our algorithm on top of a backprojection algorithm should be a straightforward post-processing step.

It has been shown (Lischinski[11] and Drettakis[13, 3]) that the boundary of the umbra volume can include a quadric surface, and hence can be quite complex to model. However, our algorithm does not require a complete computation of the umbra and penumbra volumes for each interaction, but only the computation of a surface included in the umbra volume, and of a surface enclosing the penumbra volume. Two such surfaces can be computed in a straightforward way:

- For each occluder:
  - For each receiver vertex, compute the projection of the occluder onto the emitter supporting plane;
  - The intersection of these projections is the umbra volume for this particular receiver;
  - The convex hull of these projections is the penumbra volume for this receiver.
- The union of the penumbra volumes for all occluders is the penumbra volume for the entire interaction.
- The union of the umbra volumes for all occluders is not equal to the umbra volume for the entire interaction. However, it is included into the actual umbra volume (see Lischinski[11]). Hence, we can use it for building the maximal emitter.

The computation of the projection of the occluders onto the emitter supporting plane, and the computation of the union of these projections can be reused for computing the exact value of the point-to-area form-factor in the radiosity propagation phase.

The only extra cost of our refinement criterion is then the computation of the minimal and maximal emitter knowing the projection of all the occluders on the emitter plane. This is a two-dimensional problem, computing a convex region that contains the complement of the umbra volume, and another convex region that is included into the complement of the penumbra volume. Note that we do not have to explicitely construct the umbra and the penumbra volume, only the two convex regions. We can use several methods for computing these convex regions, as described in section 4.2.5.2. The cost of our algorithm is the cost of finding two convex regions enclosing $n_r n$ polygons, where n is the number of occluders, and n_r is the number of vertices of the receiver.

The heuristic algorithm described by Lischinski[1] uses the same computation of the exact values of the point-to-area form-factor at the vertices of the receiver, which will be reused in the radiosity propagation phase, plus the computation of the point-to-area form-factor at the center of the receiving patch, which implies the projection of the occluders on the emitter supporting plane and the computation of the union of these projections. Hence, the cost of the heuristic algorithm is n projections and the union of n two-dimensional polygons.

6. Conclusions and Future Directions

We have introduced a new and reliable way of computing the maximum and the minimum of the point form-factor on any interaction. These bounds on the form-factor allow a control of the precision of the hierarchical radiosity algorithm,

precision that can be required for certain applications of the algorithm, such as architectural planning.

These bounds have been integrated in a new refinement criterion for hierarchical radiosity. We have also presented another refinement criterion that, while maintaining control on the upper and lower bounds of the energy transported, allows a coarser mesh to be constructed in some places, thus reducing memory and computation costs.

This algorithm is a significant step in error-control for global illumination methods. Although it has been devised and implemented in a hierarchical radiosity framework, nothing in the algorithm prevents the refinement criterion to be implemented with progressive refinement radiosity, as described by Cohen[25].

Knowledge of the error produced in all the parts of the algorithm allows global illumination programs to concentrate their work on parts of the scene where the error is still large, and to skip parts where it can be neglected. Thus, our algorithm can be hoped to accelerate global illumination computations by reducing the amount of unnecessary refinement.

Our algorithm relies on several conjectures: the unimodality conjectures (U1 and U2) and the concavity conjectures (C1), as well as on a knowledge of the radiosity derivatives. Table 1 recalls, for each part of the algorithm, which conjecture and which derivatives are being used.

The concavity and unimodality conjectures assume that radiosity on the emitter is constant, that the receiver is diffuse and that there is full visibility. An extension of our error-control algorithm to cases where radiosity on the emitter is not constant, or to reflectance functions that are not constant would first require a careful study of to what extent do our concavity or unimodality conjectures still hold. For example, it is clear that they cannot hold for whatever distribution of radiosity on the emitter, but only for specific cases. These specific cases, once identified, can be used as a functional basis for radiosity.

We have dealt with the partial visibility problem by computing maximal and minimal emitter, thereby reducing the problem to two full visibility problems. However, it is known that it is possible to compute the radiosity gradient in presence of occluders (see Arvo[15]), and it seems possible to compute the radiosity Hessian in presence of occluders as well (see Holzschuch[17]). In this case, it would be possible to extend our refinement criterion to some partially visible interactions without having to compute the maximum and minimum emitter. Once again, this can be done only in specific configurations where the concavity or unimodality conjectures still hold. This is not the case for generic occluders (see figure 28, in the color section), but only for certain specific, simple occluders (see figure 29 in the color section).

Although the algorithm described in this paper makes use of the U1, U2 and C1 conjectures, and of the form-factor gradient and Hessian, table 1 shows that it is possible to build a simpler algorithm to find upper and lower bounds by using only U1, U2 and the form-factor gradient.

This algorithm would be very similar to the gradient-descent algorithms described by Arvo[15] and Drettakis[2, 13]. The main difference would be the use of geometric tools, as described in section 4.2.5.2 to find an upper bound. These geometric tools will provide a fully reliable upper bound on the receiving patch.

This simpler algorithm would not allow mesh simplification as described in section 4.3.2; also, since this simpler algorithm would only use geometric methods to find upper bounds it can be expected that it will give greater upper bounds, and hence induce more refinement than our current algorithm. On the other hand, this algorithm would not require the computation of the form-factor Hessian, thus saving computation time, and would probably be easier to extend to partial visibility cases, where C1 may not hold.

Future work will include an implementation of this simpler algorithm, and timing and memory costs comparisons between our full algorithm, the simpler algorithm and the heuristic algorithm, as well as error measurements.

7. Acknowledgements

The first author has been funded by an AMN grant from Université Joseph Fourier from 1994 to 1996.

References

1. D. Lischinski, B. Smits, and D. P. Greenberg, “Bounds and Error Estimates for Radiosity”, in *Computer Graphics Proceedings, Annual Conference Series, 1994 (ACM SIGGRAPH ’94 Proceedings)*, pp. 67–74, (1994).

2. G. Drettakis and E. Fiume, “Accurate and Consistent Reconstruction of Illumination Functions Using Structured Sampling”, in *Computer Graphics Forum (Eurographics ’93)*, vol. 12, (Barcelona, Spain), pp. C273–C284, (September 1993).

3. G. Drettakis, “Structured Sampling and Reconstruction of Illumination for Image Synthesis”, CSRI Technical Report 293, Department of Computer Science, University of Toronto, Toronto, Ontario, (January 1994).

4. C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Battaile, “Modelling the Interaction of Light Between Diffuse Surfaces”, in *Computer Graphics (ACM SIGGRAPH ’84 Proceedings)*, vol. 18, pp. 212–222, (July 1984).

5. P. Schröder and P. Hanrahan, “On the Form Factor Between Two Polygons”, in *Computer Graphics Proceedings, Annual Conference Series, 1993 (ACM SIGGRAPH ’93 Proceedings)*, pp. 163–164, (1993).

Parts of the algorithm	Conjectures required	Derivatives required
Position and value of the minimum	U1 and U2	None
Position of the maximum	U2	Gradient
Using tangents to find an upper bound	C1	Hessian
Using geometric algorithms to find an upper bound	None	None
Simplification of the mesh	C1	Hessian

Table 1: *Dependencies for the different parts of the algorithm*

6. M. Cohen and D. P. Greenberg, "The Hemi-Cube: A Radiosity Solution for Complex Environments", in *Computer Graphics (ACM SIGGRAPH '85 Proceedings)*, vol. 19, pp. 31–40, (August 1985).

7. P. Hanrahan, D. Salzman, and L. Aupperle, "A Rapid Hierarchical Radiosity Algorithm", in *Computer Graphics (ACM SIGGRAPH '91 Proceedings)*, vol. 25, pp. 197–206, (July 1991).

8. S. J. Gortler, P. Schröder, M. F. Cohen, and P. Hanrahan, "Wavelet Radiosity", in *Computer Graphics Proceedings, Annual Conference Series, 1993 (ACM SIGGRAPH '93 Proceedings)*, pp. 221–230, (1993).

9. N. Holzschuch, F. Sillion, and G. Drettakis, "An Efficient Progressive Refinement Strategy for Hierarchical Radiosity", in *Fifth Eurographics Workshop on Rendering*, (Darmstadt, Germany), pp. 343–357, (June 1994).

10. P. Heckbert, "Discontinuity Meshing for Radiosity", in *Third Eurographics Workshop on Rendering*, (Bristol, UK), pp. 203–226, (May 1992).

11. D. Lischinski, F. Tampieri, and D. P. Greenberg, "Discontinuity Meshing for Accurate Radiosity", *IEEE Computer Graphics and Applications*, **12**(6), pp. 25–39 (1992).

12. D. Lischinski, F. Tampieri, and D. P. Greenberg, "Combining Hierarchical Radiosity and Discontinuity Meshing", in *Computer Graphics Proceedings, Annual Conference Series, 1993 (ACM SIGGRAPH '93 Proceedings)*, pp. 199–208, (1993).

13. G. Drettakis and E. Fiume, "A Fast Shadow Algorithm for Area Light Sources Using Backprojection", in *Computer Graphics Proceedings, Annual Conference Series, 1994 (ACM SIGGRAPH '94 Proceedings)*, pp. 223–230, (1994).

14. R. Siegel and J. R. Howell, *Thermal Radiation Heat Transfer, 3rd Edition*. New York, NY: Hemisphere Publishing Corporation, (1992).

15. J. Arvo, "The Irradiance Jacobian for Partially Occluded Polyhedral Sources", in *Computer Graphics Proceedings, Annual Conference Series, 1994 (ACM SIGGRAPH '94 Proceedings)*, pp. 343–350, (1994).

16. N. Holzschuch and F. Sillion, "Accurate Computation of the Radiosity Gradient for Constant and Linear Emitters", in *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)* (P. M. Hanrahan and W. Purgathofer, eds.), (New York, NY), pp. 186–195, Springer-Verlag, (1995).

17. N. Holzschuch, *Le Contrôle de l'Erreur dans la Méthode de Radiosité Hierarchique (Error Control in Hierarchical Radiosity)*. Ph.D. thesis, Équipe iMAGIS/IMAG, Université Joseph Fourier, Grenoble, France, (March 5th, 1996).

18. F. P. Preparata and M. I. Shamos, *Computational Geometry – An Introduction*. New York: Springer Verlag, (1985).

19. J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics, Principles and Practice, Second Edition*. Reading, Massachusetts: Addison-Wesley, (1990).

20. T. L. Kay and J. T. Kajiya, "Ray tracing complex scenes", *Computer Graphics*, **20**(4), pp. 269–276 (1986). Proceedings of SIGGRAPH '86 in Dallas (USA).

21. D. L. Toth, "On ray-tracing parametric surfaces", *Computer Graphics*, **19**(3), pp. 171–179 (1985). Proceedings SIGGRAPH '85 in San Francisco (USA).

22. S. J. Teller, "Computing the antipenumbra of an area light source", in *Computer Graphics (ACM SIGGRAPH '92 Proceedings)*, vol. 26, pp. 139–148, (July 1992).

23. S. Teller and P. Hanrahan, "Global Visibility Algorithms for Illumination Computations", in *Computer Graphics Proceedings, Annual Conference Series,*

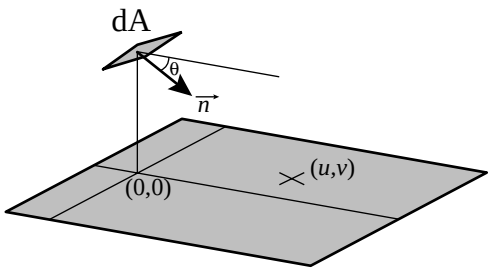


Figure 16: *A differential area emitter and an infinite receiving plane.*

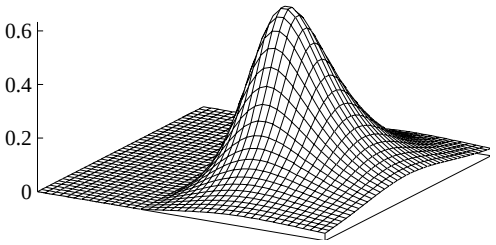


Figure 17: *An example of the point-to-area form-factor function ($\theta = \frac{\pi}{6}$).*

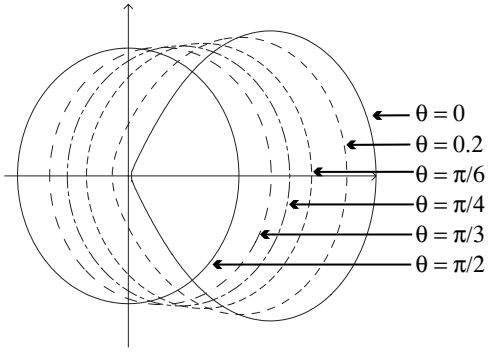


Figure 18: *The areas where the point-to-area form-factor function is concave for different values of θ.*

1993 (ACM SIGGRAPH '93 Proceedings), pp. 239–246, (1993).

24. N. Chin and S. Feiner, "Fast object-precision shadow generation for areal light sources using BSP trees", *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, **25**(2), pp. 21–30 (1992).

25. M. Cohen, S. E. Chen, J. R. Wallace, and D. P. Greenberg, "A Progressive Refinement Approach to Fast Radiosity Image Generation", in *Computer Graphics (ACM SIGGRAPH '88 Proceedings)*, vol. 22, pp. 75–84, (August 1988).

Appendix A: Concavity conjectures: case study of a differential area emitter

Let us consider the case of an infinite receiving plane and a single differential area for the emitter. In this case, due to the symmetries shared by the emitter and the receiver, there is only one parameter: the angle, called θ, between the normal of the emitter and a line parallel to the receiver, (see figure 16).

To express the position of a point on the receiver, we choose a set of axes related to the emitter: the first axes shares the direction of the projection of the normal of the emitter on the receiver, and the second axes is orthogonal to the first. The origin of our coordinate system is the projection of the emitting point. Using this set of coordinates, we have a simple expression for the point-to-area form-factor at any point $M(u, v)$ on the receiver (see figure 17 the aspect of the surface):

$$F(u, v) = \frac{dA}{\pi} \frac{u\cos(\theta) + \sin(\theta)}{(u^2 + v^2 + 1)^2}$$

This value is only for $u\cos(\theta) + \sin(\theta) > 0$. If $u\cos(\theta) + \sin(\theta) \leq 0$, then of course $F(u, v) = 0$.

The C1 concavity conjecture

In this simple case, it is possible to explicitly compute the derivatives of the point-to-area form-factor. An explicit computation of the Hessian shows that it is definite if and only if the expression $S(u, v, \theta)$ is positive, where $S(u, v, \theta)$ is:

$$\begin{aligned} S \;=\; & -3u^4 - 8\tan\theta u^3 - 5\tan^2\theta u^2 - 4u^2v^2 + 3u^2 \\ & +4u\tan\theta - 8\tan\theta uv^2 \\ & -5\tan^2\theta v^2 - v^2 - v^4 + \tan^2\theta \end{aligned}$$

Although it is impossible to find an explicit solution of the equation $S(u, v, \theta) = 0$, it is possible to plot these solutions for different values of θ. Figure 18 shows the contour of the area where $S(u, v, \theta)$ is positive for different values of θ. Outside these areas, $S(u, v, \theta)$ is negative, and hence the Hessian matrix is indefinite. Inside these areas, S is positive, and the form-factor is concave.

An interesting point is the shape of the zones where the point-to-area form factor is concave. When $\theta = \frac{\pi}{2}$, it is of course a disc, due to the symmetries in the scene. When $\theta = 0$, it is a shape like a drop, that tapers to a point in $(0, 0)$. For intermediate values of θ, the zone has an intermediate shape between the drop and the disc, but this shape always appears to be convex.

The C2 concavity conjecture

If we now focus on the radiosity on a specific line $v = au + b$ on the receiving plane, we have, for the form-factor as a

function of u,

$$f(u) = \frac{dA}{\pi} \frac{u\cos(\theta) + \sin(\theta)}{(u^2 + (au+b)^2 + 1)^2}$$

The form-factor is equal to $f(u)$ if $u\cos(\theta)+\sin(\theta) > 0$. If $u\cos(\theta) + \sin(\theta) \leq 0$, then the form-factor is null.

It must be noted that $f(u)$ goes to zero when u goes to $\pm\infty$, and that $f(u)$ is equal to zero only for $u = u_0 = -\tan\theta$.

It is possible to compute the first and the second derivative of $f(u)$. The first derivative, $f'(u)$, is of the sign of a second degree polynomial in u, and the second derivative, $f''(u)$ is of the sign of a third degree polynomial in u. As a consequence, $f'(u)$ can change sign at most twice, and $f''(u)$ at most three times.

Since the function $f(u)$ goes to zero when u goes to $\pm\infty$, it must have one maximum between u_0 and $+\infty$, and one minimum between u_0 and $-\infty$. As a consequence, $f'(u)$ must change sign exactly twice. Let us call u_1 and u_2 the points where the first derivative changes sign ($u_1 < u_0 < u_2$).

$f'(u)$ also goes to zero when u goes to $\pm\infty$. As a consequence, it must have one minimum between u_2 and $+\infty$, and another between $-\infty$ and u_1, and it must have one maximum between u_1 and u_2. So the second derivative changes sign exactly three times. One of the point where the second derivative changes sign is smaller than u_1, which is smaller than u_0, and one of them is greater than u_2, which is greater than u_0.

Then the second derivative changes sign at least once and at most twice on $[u_0, +\infty]$. When u goes to $+\infty$, f is convex, and f'' is positive. So we just proved that f'' can be negative only over a unique bounded segment on $[u_0, +\infty]$.

The form-factor on the line is equal to $f(u)$ for $u > u_0$, and null everywhere else. So the form-factor on a line is concave only over a unique bounded segment. This proves the C2 conjecture for a differential area emitter.

Figure 19 shows an example of such a $f(u)$ function, along with its first and second derivatives. It can be noted that this function is concave over a single segment, and convex everywhere else.

Appendix B: Effective computation of the form-factor derivatives

In this section, we show how it is possible to compute the derivatives of the point-to-area form-factor with little additional computation expense.

In particular, it is shown that the computation of the exact value of the form-factor derivatives is always cheaper than the computation of an approximate value using several form-factor samples. For example, the cost of computing the form-factor gradient is 30 %, while computing an approximate value of the gradient would require two form-factor samples, thus increasing computation time by 100 %

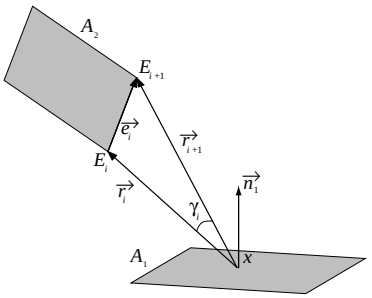


Figure 20: *Notation when the emitter is a polygon.*

$F = 0$
foreach edge $[E_i E_{i+1}]$
$\vec{r_i} = E_i - x$
$\vec{r}_{i+1} = E_{i+1} - x$
$\text{crossprod} = \vec{r_i} \times \vec{r}_{i+1}$
$gamma = \arccos\left(\frac{\vec{r_i}\cdot\vec{r}_{i+1}}{r_i r_{i+1}}\right)$
$I_1 = \frac{gamma}{\|\text{crossprod}\|}$
$\text{mixt} = \vec{n}_1 \cdot \text{crossprod}$
$F- = I_1 \text{mixt}$
$F* = \frac{1}{2\pi}$

Figure 21: *Pseudo-Code for computing the form-factor.*

The Point-to-Area Form-Factor

Let us recall that the point-to-area form-factor from a point x on a patch A_1 to a patch A_2 (see figure 1) can be expressed as a contour integral:

$$F(x) = -\vec{n}_1 \cdot \frac{1}{2\pi} \oint_{\partial A_2} \frac{\vec{r}_{12} \times d\vec{\ell}_2}{\|\vec{r}_{12}\|^2}$$

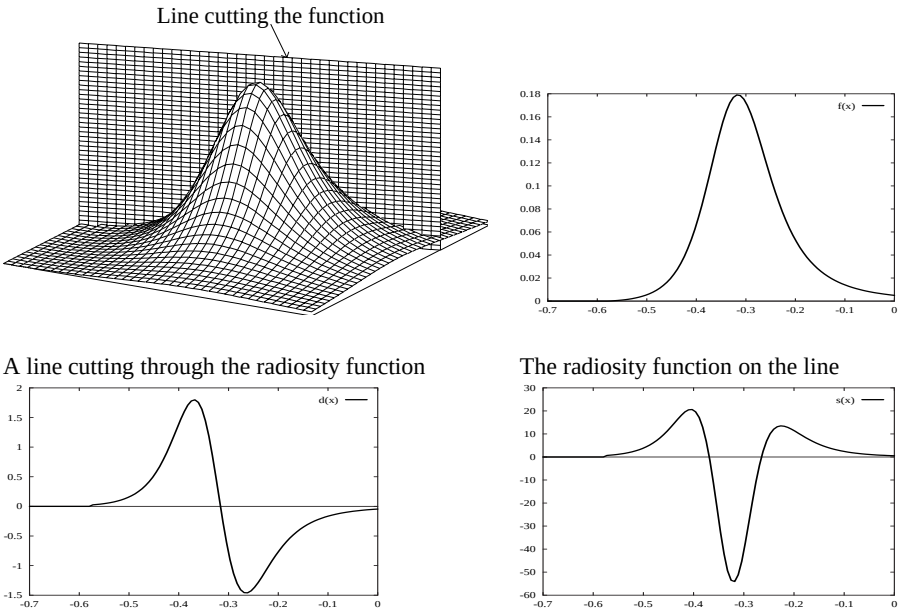
For the explicit derivation of this contour integral from the equation 2, see Siegel and Howell[14].

In the case where the emitter is a polygon, this expression simplifies to a finite sum:

$$F(x) = \frac{1}{2\pi}\vec{n}_1 \cdot \sum_i \vec{\gamma}_i \qquad (6)$$

where $\vec{\gamma}_i$ is the vector of norm γ_i, and of direction the cross-product $\vec{r_i} \times \vec{r}_{i+1}$ (see figure 20).

An example pseudo-code for computing the form-factor using equation 6 can be found in figure 21. This pseudo-code makes use of the standard 3D operations like addition, cross-product and dot product.



A line cutting through the radiosity function

The radiosity function on the line

The first derivative of the radiosity function on the line

The second derivative is negative only over a segment.

Figure 19: *The radiosity on any line on the receiving plane is concave only over a segment.*

Form-Factor Gradient

The point-to-area form-factor gradient can be easily computed by derivation of the previous formula (see Arvo[15], or Holzschuch[17]):

$$\nabla F(x) = -\frac{-1}{2\pi}\sum_i \vec{n}_1 \times \vec{e}_i I_1 + 2\vec{n}_1 \cdot (\vec{r}_i \times \vec{r}_{i+1})\,(\vec{r}_i I_2 + \vec{e}_i J_2)$$

With:

$$I_1 = \frac{\gamma_i}{\|\vec{e}_i \times \vec{r}_i\|}$$

$$I_2 = \frac{1}{2\|e_i \times \vec{r}_i\|^2}\left(\frac{\vec{e}_i \cdot \vec{r}_{i+1}}{r_{i+1}^2} - \frac{\vec{e}_i \cdot \vec{r}_i}{r_i^2} + e_i^2 I_1\right)$$

$$J_2 = \frac{1}{2e_i^2}\left(\frac{1}{r_i^2} - \frac{1}{r_{i+1}^2}\right) - \frac{\vec{e}_i \cdot \vec{r}_i}{e_i^2} I_2$$

The code in figure 21 for computing the form-factor can be extended for computing the gradient. Figure 22 shows the extension of the pseudo-code needed for computing simultaneously the point-to-area form-factor and its gradient (we did not include the part of the code that is exactly identical). As can be seen, most of the costly computations like inverse trigonometric functions have been done for the form-factor, and do not need to be redone for the gradient.

The exact extra cost of computing the gradient depends on the computer and on the compiler used. On an R4000 SGI with the standard cc compiler, it is 30 % (see Holzschuch[16, 17]).

What is fundamental is that it actually costs much less to compute the exact value for the gradient than it would cost to compute two radiosity values, and then to approximate the gradient using these values.

$F = 0$
$\vec{G} = \vec{0}$
foreach edge $[E_i E_{i+1}]$
  .
  .
  .
  $F- = I_1 \mathrm{mixt}$
  $\vec{e}_i = \overrightarrow{E_i E_{i+1}}$
  $I_2 = \frac{\vec{e}_i \cdot \vec{r}_{i+1}}{r_{i+1}^2} - \frac{\vec{e}_i \cdot \vec{r}_i}{r_i^2} + e_i^2 I_1$
  $I_2 / = 2\mathrm{crossprod}^2$
  $J_2 = 0.5\left(\frac{1}{r_i^2} - \frac{1}{r_{i+1}^2}\right) - \vec{e}_i \cdot \vec{r}_i I_2$
  $J_2 / = e_i^2$
  $\vec{G} + = (\vec{n}_1 \times \vec{e}_i) I_1 + 2\mathrm{mixt}(\vec{r}_i I_2 + \vec{e}_i J_2)$
$F* = \frac{1}{2\pi}$
$\vec{G}* = -\frac{1}{2\pi}$

Figure 22: *Pseudo-Code for computing the gradient of the form-factor.*

$F = 0$
$\vec{G} = \vec{0}$
$H = 0$
foreach edge $[E_i E_{i+1}]$
$\vdots$
$\vec{G} += (\vec{n}_1 \times \vec{e}_i) I_1 + 2\text{mixt}(\vec{r}_i I_2 + \vec{e}_i J_2)$
$I_3 = \frac{\vec{e}_i \cdot \vec{r}_{i+1}}{r_{i+1}^4} - \frac{\vec{e}_i \cdot \vec{r}_i}{r_i^4} + e_i^2 I_2$
$I_3 /= 4\text{crossprod}^2$
$J_3 = 0.25 \left(\frac{1}{r_i^4} - \frac{1}{r_{i+1}^4} \right) - \vec{e}_i \cdot \vec{r}_i I_3$
$J_3 /= e_i^2$
$K_3 = I_2 - r_i^2 I_3 - 2\vec{e}_i \cdot \vec{r}_i J_3$
$K_3 /= e_i^2$
$H += -\text{mixt} I_2 I + Q(\vec{r}_i I_2 + \vec{e}_i J_2, \vec{n}_1 \times \vec{e}_i)$
$+2\text{mixt}(Q(\vec{r}_i, \vec{r}_i) I_3 + Q(\vec{e}_i, \vec{e}_i) K_3 + 2 J_3 Q(\vec{e}_i, \vec{r}_i))$
$F* = \frac{1}{2\pi}$
$\vec{G}* = -\frac{1}{2\pi}$
$H* = \frac{1}{\pi}$

Figure 23: *Pseudo-Code for computing the first two derivatives of the form-factor.*

Hessian matrix for the point-to-area form-factor

The point-to-area form factor Hessian matrix can also be computed by derivation of Equation 6 (see Holzschuch[17]):

$$\begin{aligned} H \;=\; & -\frac{1}{\pi} \sum_i Q\left(\vec{n}_1 \times \vec{e}_i, \vec{r}_i I_2 + \vec{e}_i J_2\right) \\ & -\vec{n}_1 \cdot (\vec{r}_i \times \vec{e}_i) I_2 I \\ & +2\vec{n}_1 \cdot (\vec{r}_i \times \vec{e}_i)(Q(\vec{r}_i, \vec{r}_i) I_3 \\ & +Q(\vec{e}_i, \vec{e}_i) K_3 + 2 J_3 Q(\vec{r}_i, \vec{e}_i)) \end{aligned}$$

We use the following notation:

$$\begin{aligned} Q(\vec{a}, \vec{b}) &= \vec{a}^{\mathrm{t}} \vec{b} + \vec{b}^{\mathrm{t}} \vec{a} \\ I_3 &= \frac{1}{4} \frac{1}{\|\vec{e}_i \times \vec{r}_i\|^2} \left(\frac{\vec{e}_i \cdot \vec{r}_{i+1}}{r_{i+1}^4} - \frac{\vec{e}_i \cdot \vec{r}_i}{r_i^4} + 3 e_i^2 I_2 \right) \\ J_3 &= \frac{1}{4 e_i^2} \left(\frac{1}{r_i^4} - \frac{1}{r_{i+1}^4} \right) - \frac{\vec{r}_i \cdot \vec{e}_i}{e_i^2} I_3 \\ K_3 &= \frac{1}{e_i^2} \left(I_2 - r_i^2 I_3 - 2(\vec{r}_i \cdot \vec{e}_i) J_3 \right) \end{aligned}$$

The code for computing the form-factor and the gradient can be extended to compute the second derivative as well. Figure 23 shows the extension of the pseudo-code needed for computing simultaneously the point-to-area form-factor and its first two derivatives (we did not include the part of the code that is exactly identical). Once again, recycling geometric computations previously done reduces the cost of computing the Hessian matrix, even if the cost is still high since matrix operations are quite expensive: a single matrix addition has the same cost as 9 standard additions.

The exact extra cost of computing the Hessian matrix depends on the computer and on the compiler. On a R4000 SGI, with the standard cc compiler, it is 80 % of the cost of the form-factor alone (see Holzschuch[17]), meaning that the overall cost of computing the point-to-area form-factor and its first two derivatives is 2.1 times the cost of computing the form-factor alone. Notice it is much faster to compute the exact value than it would be to compute an approximate Hessian matrix — which would require seven separate form-factor computations.

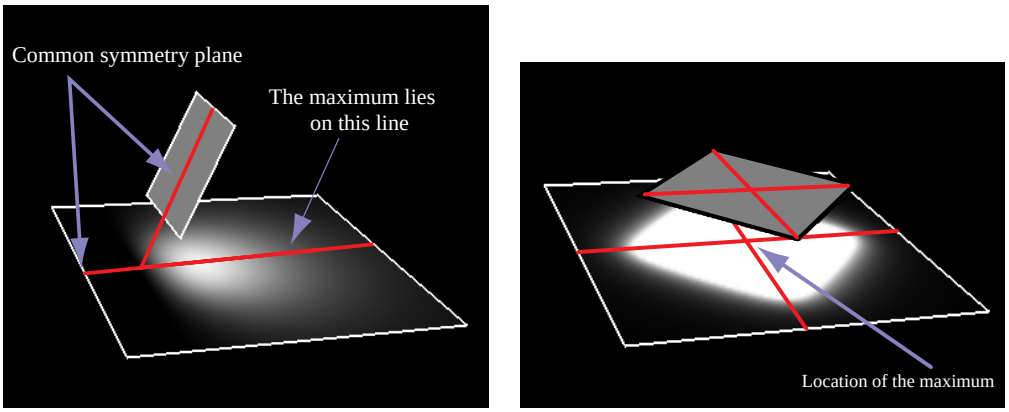


Figure 24: *The symmetries of the scene can help find the location of the maximum.*

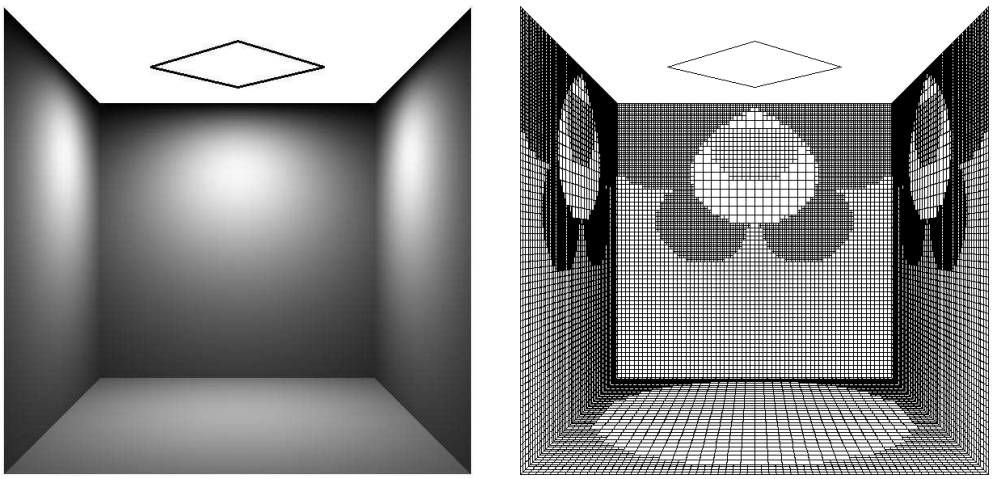


Figure 25: *Direct illumination with our refinement criterion, unoccluded scene.*

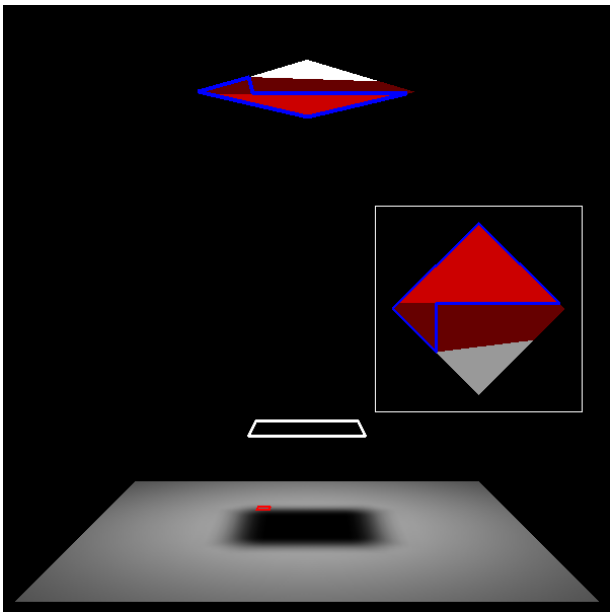


Figure 26: *Minimal and maximal emitter for a simple configuration.*

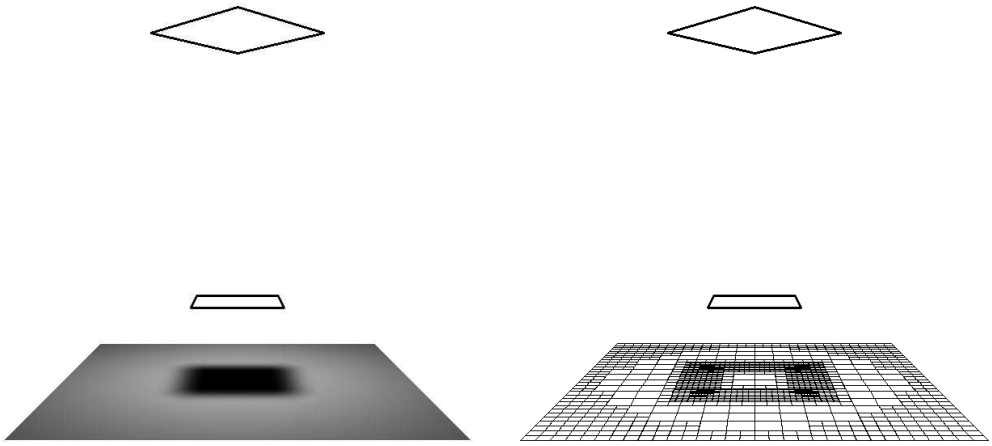


Figure 27: *Direct illumination with our refinement criterion, with one occluder.*

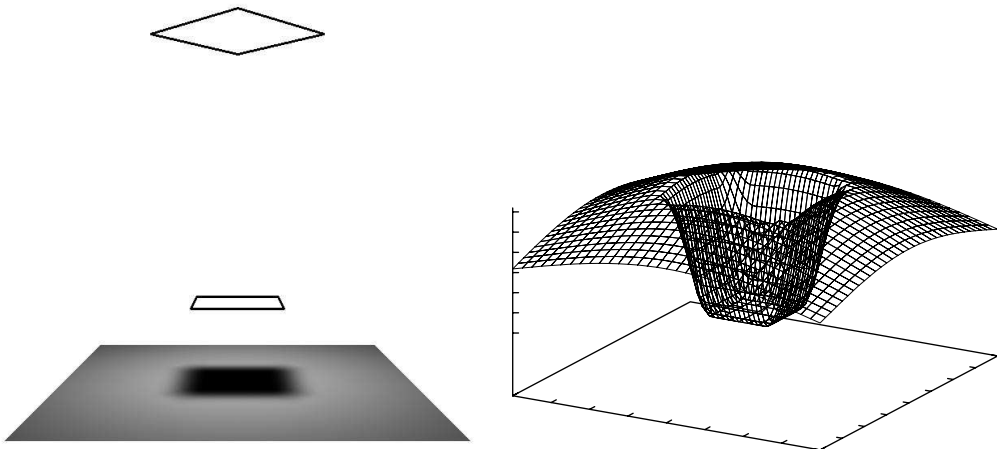


Figure 28: *With generic occluders, the unimodality conjectures do not hold.*

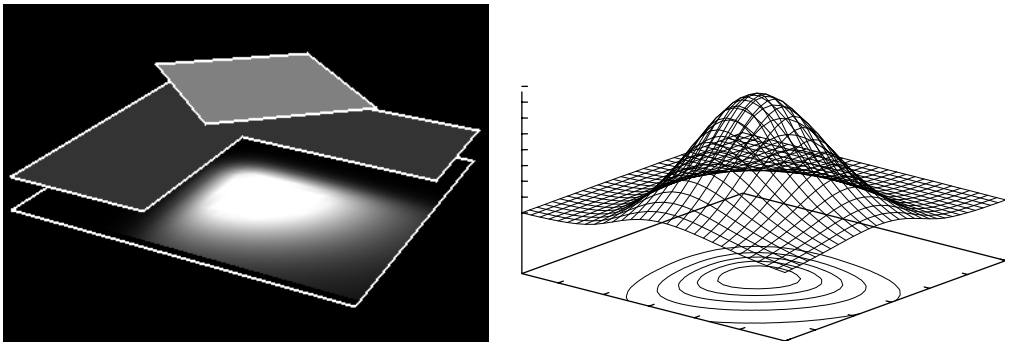


Figure 29: *With certain occluders, the unimodality conjectures still holds.*

3.4.4 A Frequency Analysis of Light Transport (Siggraph 2005)

Auteurs : Frédo D , Nicolas H , Cyril S , Eric C et François S
Conférence : SIGGRAPH 2005. Cet article a été également publié dans *ACM Transactions on Graphics*, vol. 24, n° 3, p. 1115–1126.
Date : août 2005

A Frequency Analysis of Light Transport

Frédo Durand, MIT-CSAIL
Nicolas Holzschuch, Cyril Soler, ARTIS* GRAVIR/IMAG-INRIA
Eric Chan, MIT-CSAIL
François X. Sillion, ARTIS* GRAVIR/IMAG-INRIA

Abstract

We present a signal-processing framework for light transport. We study the frequency content of radiance and how it is altered by phenomena such as shading, occlusion, and transport. This extends previous work that considered either spatial or angular dimensions, and it offers a comprehensive treatment of both space and angle.

We show that occlusion, a multiplication in the primal, amounts in the Fourier domain to a convolution by the spectrum of the blocker. Propagation corresponds to a shear in the space-angle frequency domain, while reflection on curved objects performs a different shear along the angular frequency axis. As shown by previous work, reflection is a convolution in the primal and therefore a multiplication in the Fourier domain. Our work shows how the spatial components of lighting are affected by this angular convolution.

Our framework predicts the characteristics of interactions such as caustics and the disappearance of the shadows of small features. Predictions on the frequency content can then be used to control sampling rates for rendering. Other potential applications include precomputed radiance transfer and inverse rendering.

Keywords: Light transport, Fourier analysis, signal processing

1 Introduction

Light in a scene is transported, occluded, and filtered by its complex interaction with objects. By the time it reaches our eyes, radiance is an intricate function, and simulating or analyzing it is challenging.

Frequency analysis of the radiance function is particularly interesting for many applications, including forward and inverse rendering. The effect of local interactions on the frequency content of radiance has previously been described in a limited context. For instance, it is well-known that diffuse reflection creates smooth (low-frequency) light distributions, while occlusion and hard shadows create discontinuities and high frequencies. However, a full characterization of global light transport in terms of signal processing and frequency analysis presents two major challenges: the domain of light rays is intricate (three dimensions for position and two for direction), and light paths can exhibit an infinite number of bounces (i.e. in terms of signal processing, the system has dense feedback).

To address the above challenges, we focus on the neighborhood of light paths [Shinya et al. 1987]. This restriction to local properties is both a price to pay and a fundamental difficulty with the problem we study: characteristics such as reflectance or presence and size of blockers are *non-stationary*, they vary across the scene.

This paper presents a theoretical framework for characterizing light transport in terms of frequency content. We seek a deep understanding of the frequency content of the radiance function in a scene and how it is affected by phenomena such as occlusion, reflection, and propagation in space (Fig. 1). We first present the two-dimensional case for simplicity of exposition. Then we show that it extends well to 3D because we only consider local neighborhoods of rays, thereby avoiding singularities on the sphere of directions.

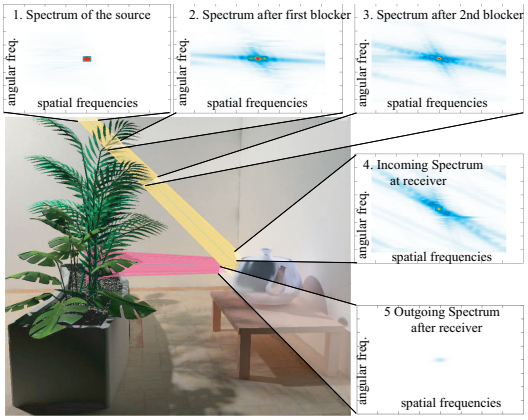


Figure 1: Space-angle frequency spectra of the radiance function measured in a 3D scene. We focus on the neighborhood of a ray path and measure the spectrum of a 4D light field at different steps, which we summarize as 2D plots that include only the radial components of the spatial and angular dimensions. Notice how the blockers result in higher spatial frequency and how transport in free space transfers these spatial frequencies to the angular domain. Aliasing is present in the visualized spectra due to the resolution challenge of manipulating 4D light fields.

Although we perform our derivations in an abstract setting, we keep practical questions in mind. In particular, we strongly believe that a good understanding of frequency creation and attenuation allows for more efficient sampling strategies for stochastic approaches such as Monte-Carlo global illumination. Furthermore, it leads to better sampling rates for light-field rendering, precomputed radiance transfer, and related applications. Finally, our frequency analysis can shed key practical insights on inverse problems and on the field of statistics of natural images by predicting which phenomena can cause certain local-frequency effects.

1.1 Contributions

This paper makes the following contributions:

Frequency study in space and angle. Our framework encompasses both *spatial* and *directional* variations of radiance, while most previous work studied only one of these two components.

Local surface interactions. We describe the frequency effects of local shading, object curvature, and spatially-varying BRDF.

Global light-transport. We provide expressions for the frequency modification due to light transport in free space and occlusion.

Most of the derivations in this paper are carried out in 2D for clarity, but we show that the main characterizations extend to 3D.

*ARTIS is a team of the GRAVIR lab (UMR 5527), a joint unit of CNRS, INPG, INRIA and UJF.

1.2 Related work

Radiance exhibits both spatial and angular variations. A wealth of previous work has studied the frequency content along one of these components, but rarely have both space and angle been addressed. We do not discuss all applications of Fourier analysis, but rather focus on studies of frequency modification in light transport.

Filtering and sampling Heckbert's seminal work on texture antialiasing [1989] derives *local* bandwidth for texture pre-filtering based on a first-order Taylor expansion of the perspective transform. The effect of perspective is also studied in the contexts of holography and light field sampling [Halle 1994; Isaksen et al. 2000; Chai et al. 2000; Stewart et al. 2003], mostly ignoring visibility and specular effects.

Local illumination as a convolution Recently, local illumination has been characterized in terms of convolution and it was shown that the outgoing radiance is band-limited by the BRDF [Ramamoorthi and Hanrahan 2001b; Ramamoorthi and Hanrahan 2004; Basri and Jacobs 2003]. However the lighting is assumed to come from infinity and occlusion is ignored. Frolova et al. [2004] explored spatial lighting variations, but only for convex diffuse objects. We build on these approaches and extend them by adding spatial dimensions as well as other phenomena such as occlusion and transport, at the expense of first-order approximations and a local treatment. Ramamoorthi et al. [2004] have also studied local occlusion in a textured object made of pits such as a sponge. Our treatment of occlusion considers complex blockers at an arbitrary distance of the blocker and receiver.

Wavelets and frequency bases Wavelets and spherical harmonics have been used extensively as basis functions for lighting simulation [Gortler et al. 1993; Keller 2001] or pre-computed radiance transfer [Sloan et al. 2002; Ramamoorthi and Hanrahan 2002]. They are typically used in a data-driven manner and in the context of projection methods, where an oracle helps in the selection of the relevant components based on the local frequency characteristics of radiance. Refinement criteria for multiresolution calculations often implicitly rely on frequency decomposition [Sillion and Drettakis 1995]. In our framework we study the frequency effect of the *equations* of light transport in the spirit of linear systems, and obtain a more explicit characterization of frequency effects. Our results on the required sampling rate can therefore be used with stochastic methods or to analyze the well-posedness of inverse problems.

Ray footprint A number of techniques use notions related to bandwidth in a ray's neighborhood and propagate a footprint for adaptive refinement [Shinya et al. 1987] and texture filtering [Igehy 1999]. Chen and Arvo use perturbation theory to exploit ray coherence [2000]. Authors have also exploited *on-the-fly* the frequency content of the image to make better use of rays [Bolin and Meyer 1998; Myszkowski 1998; Keller 2001]. Our work is complementary and provides a framework for frequency-content prediction.

Illumination differentials have been used to derive error bounds on radiance variations (e.g. gradients [Ward and Heckbert 1992; Annen et al. 2004], Jacobians [Arvo 1994], and Hessians [Holzschuch and Sillion 1998], but only provide local information, which cannot easily be used for sampling control.

Fourier analysis has also been extensively used in optics [Goodman 1996], but in the context of wave optics where phase and interferences are crucial. In contrast, we consider geometric optics and characterize frequency content in the visible spatial frequencies. The varying contrast sensitivity of humans to these spatial frequencies can be exploited for efficient rendering, e.g. [Bolin and Meyer 1995; Ferwerda et al. 1997; Bolin and Meyer 1998; Myszkowski 1998]. Finally we note that the Fourier basis can separate different phenomena and thus facilitate inverse lighting [Ramamoorthi and Hanrahan 2001b; Basri and Jacobs 2003] depth from focus [Pentland 1987] and shape from texture [Malik and Rosenholtz 1997].

ℓ_R	2D light field (2D) around ray R
x	spatial dimension (distance to central ray)
v	directional dimension in 2-plane parameterization
θ	directional dimension in plane-sphere paramerization
$\widehat{f}$	Fourier transform of function f
Ω_X	frequency along dimension X
i	$\sqrt{-1}$
$f \otimes g$	convolution of f by g
d	Transport distance
$V(x,v)$	visibility function of the blockers
$\cos_+(\theta)$	clamped cosine term: $\max(\cos\theta, 0)$
$\mathrm{d}E$	Differential irradiance (after cosine term).
ρ	BRDF

Figure 2: Notations.

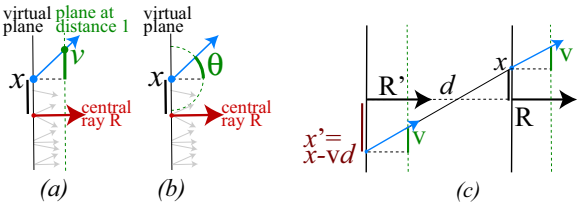


Figure 3: (a-b) The two light field parameterization used in this article. Locally, they are mostly equivalent: we linearize $v = \tan\theta$. (c) Transport in free space: the angular dimension v is not affected but the spatial dimension is reparameterized depending on v.

2 Preliminaries

We want to analyze the radiance function in the neighborhood of a ray along all steps of light propagation. For this, we need a number of definitions and notations, summarized in Fig. 2. Most of the derivations in this paper are carried out in 2D for clarity, but we shall see that our main observations extend naturally to 3D.

2.1 Local light field and frequency content

We consider the 4D (resp. 2D) slice of radiance at a virtual plane orthogonal to a central ray. We focus on the neighborhood of the central ray, and we call radiance in such a 4D (resp. 2D) neighborhood slice a *local light field* (Fig. 3 left). Of the many parameterizations that have been proposed for light fields, we use two distinct ones in this paper, each allowing for a natural expression of some transport phenomena. Both use the same parameter for the spatial coordinates in the virtual plane, x, but they differ slightly in their treatment of directions. For our *two-plane parameterization*, we follow Chai et al. [2000] and use the intersection v with a parallel plane at unit distance, expressed in the local frame of x (Fig. 3-a). In the *plane-sphere* parameterization, we use the angle θ with the central direction (Fig. 3-b) [Camahort et al. 1998]. These two parameterizations are linked by $v = \tan\theta$ and are equivalent around the origin thanks to a linearization of the tangent.

We study the Fourier spectrum of the radiance field ℓ_R, which we denote by $\widehat{\ell_R}$. For the two-plane parameterization, we use the following definition of the Fourier transform:

$$\widehat{\ell_R}(\Omega_x, \Omega_v) = \int_{x=-\infty}^{\infty} \int_{v=-\infty}^{\infty} \ell_R(x,v) \mathrm{e}^{-2\mathrm{i}\pi\Omega_x x} \mathrm{e}^{-2\mathrm{i}\pi\Omega_v v} \,\mathrm{d}x\,\mathrm{d}v \qquad (1)$$

Examples are shown for two simple light sources in Fig. 4, with the spatial dimension along the horizontal axis and the direction along the vertical axis. We discuss the plane-sphere parameterization in Section 4.

One of the motivations for using Fourier analysis is the convolution-multiplication theorem, which states that a convolution

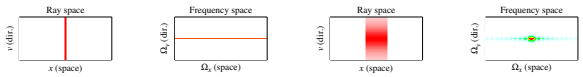


Figure 4: (a) A point light source is a Dirac in space times a constant in angle. (b) Its Fourier transform is a constant in space times a Dirac in angle. (c) A spot light with a finite-size bulb has a smooth falloff in angle. (d) Its Fourier transform is a sinc times a bell curve.

in the primary domain corresponds to a multiplication in the Fourier domain, and vice-versa. As we show in this paper, it affords a compact formulation of frequency modification.

2.2 Overview

When light flows in a scene, phenomena such as transport in free space, occlusion, and shading each modify the local light field in a characteristic fashion. These operations are described (in 2D) in Section 3 as filters operating on the frequency signal $\widehat{\ell}$. In Section 4, we describe the general case of local shading and extend the presentation to 3D in Section 5. Section 6 compares our framework with previous work and shows a simple application.

3 Transport phenomena as linear filters

This section describes the effect on frequency content of successive stages of light transport. All phenomena are illustrated in Fig. 6 for a simple 2D scene, where light is emitted at the source, transported in free space, occluded by obstacles, transported again, and reflected by a surface. At each step, we show a space-direction plot of radiance, in primal and frequency space, as well as space-direction frequency-domain plots obtained in a similar 3D scene (Fig. 9-b). Note the excellent qualitative agreement between the 2D predictions and 3D observations.

3.1 Travel in free space

Travel in free space is a crucial operation because the directional variation also turns into spatial variation. Consider a slide projector: At the source, we have a Dirac in space and the image in the directional domain. At the receiver, the signal of the image is present in a combination of space and angle. When light travels in free space, the value of radiance is unchanged along a ray, and travel in free space is a reparameterization of the local light field (Fig. 3-c). The value of radiance at a point x after transport can be found using:

$$\ell_R(x, v) = \ell_{R'}(x - vd, v), \tag{2}$$

where d is the travel distance. To compute the Fourier transform $\widehat{\ell}_R$, we insert the change of variable $x' = x - vd$ in the integral of Eq. 1:

$$\begin{aligned}\widehat{\ell}_R(\Omega_x, \Omega_v) &= \int_{x'=-\infty}^{\infty}\int_{v=-\infty}^{\infty} \ell_R(x', v) e^{-2i\pi\Omega_x(x'+vd)} e^{-2i\pi\Omega_v v}\,dx'\,dv \\ &= \int_{x'=-\infty}^{\infty}\int_{v=-\infty}^{\infty} \ell_R(x', v) e^{-2i\pi\Omega_x x'} e^{-2i\pi(\Omega_v + d\Omega_x)v}\,dx'\,dv,\end{aligned}$$

This is a shear in the directional dimension (Fig. 6, steps 2 and 4):

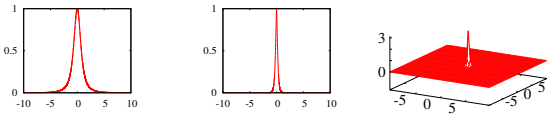
$$\widehat{\ell}_{R'}(\Omega_x,\ \Omega_v) = \widehat{\ell}_R(\Omega_x,\ \Omega_v + d\Omega_x) \tag{3}$$

The longer the travel, the more pronounced the shear.

3.2 Visibility

Occlusion creates high frequencies and discontinuities in the radiance function. Radiance is multiplied by the binary occlusion function of the occluders:

$$\ell_{R'}(x,\ v) = \ell_R(x,\ v)\ V(x, v) \tag{4}$$



(a) $g(v)$ (b) $\widehat{g}(\Omega_v)$ (c) $\widehat{G}(\Omega_u, \Omega_v)$

Figure 5: Unit area in lightfield parameterisation.

where $V(x, v)$ is equal to 1 when there is full visibility and to 0 when the occluders are blocking light transport. At the location of occlusion, V mostly depends on x (Fig. 6, step 3).

According to the multiplication theorem, such a multiplication amounts to a convolution in the frequency domain:

$$\widehat{\ell}_{R'}(\Omega_x,\ \Omega_v) = \widehat{\ell}_R(\Omega_x,\ \Omega_v) \otimes \widehat{V}(\Omega_x, \Omega_v) \tag{5}$$

If the occluders are lying inside a plane orthogonal to the ray, the occlusion function is a constant in angle, and its Fourier transform is a Dirac in the angular dimension. In the general case of non-planar occluders, their spectrum has frequency content in both dimensions, but the angular frequency content is restricted to a wedge with a span proportional to the depth extent of the blockers. Our formulation also handles semi-transparent blockers by using non-binary occlusion functions.

After occlusion, another transport step usually occurs, shearing the spectrum and propagating the occlusion from the spatial dimension to the angular dimension (Fig. 6, step 4).

3.3 Local diffuse shading

We first treat local shading by a planar Lambertian reflector. Curved and glossy reflectors will be treated in greater details in Section 4.

For a diffuse reflector with albedo ρ, the outgoing radiance ℓ_o has no directional variation; it is simply the integral over all directions of the incoming radiance per unit area on the receiver surface:

$$\ell_o(x) = \rho \int_\Omega \ell_i(x,\ v)\,dA_\perp \tag{6}$$

$dA_\perp$, the differential area on the receiver surface is $\cos_+\theta$ times the Jacobian of the lightfield parameterization; with $v = \tan\theta$, we have:

$$dA_\perp = \frac{dv}{(1+v^2)^{\frac{3}{2}}} = g(v)\,dv$$

We introduce $dE(x,\ v) = \ell_i(x,\ v)g(v)$, the differential irradiance at point x from the direction v. Since dE is a product of two functions, its spectrum is the convolution of their spectra:

$$\widehat{dE}(\Omega_x,\ \Omega_v) = \widehat{\ell}_i(\Omega_x,\ \Omega_v) \otimes \widehat{g}(\Omega_v)$$

The reflected radiance ℓ_o is the integral of dE over all directions v; it is therefore the value of $\widehat{dE}$ at $\Omega_v = 0$, that is, $\widehat{\ell_o}(\Omega_t) = \rho\widehat{dE}(\Omega_t,\ 0)$. Putting everything together, we have:

$$\widehat{\ell_o}(\Omega_x) = \rho\left[\widehat{\ell}_i(\Omega_x,\ \Omega_v) \otimes \widehat{g}(\Omega_v)\right]_{\Omega_v=0} \tag{7}$$

$g(v)$ is a bell-like curve (Fig. 5-a); its Fourier transform is:

$$\widehat{g}(\Omega_v) = 4\pi|\Omega_v|K_1(2\pi|\Omega_v|)$$

where K_1 is the first-order modified Bessel function of the second kind. $\widehat{g}$ is highly concentrated on low frequencies (Fig. 5-b); the effect of convolution by $\widehat{g}$ is a very small blur of the spectrum in the angular dimension (Fig. 6, step 5).

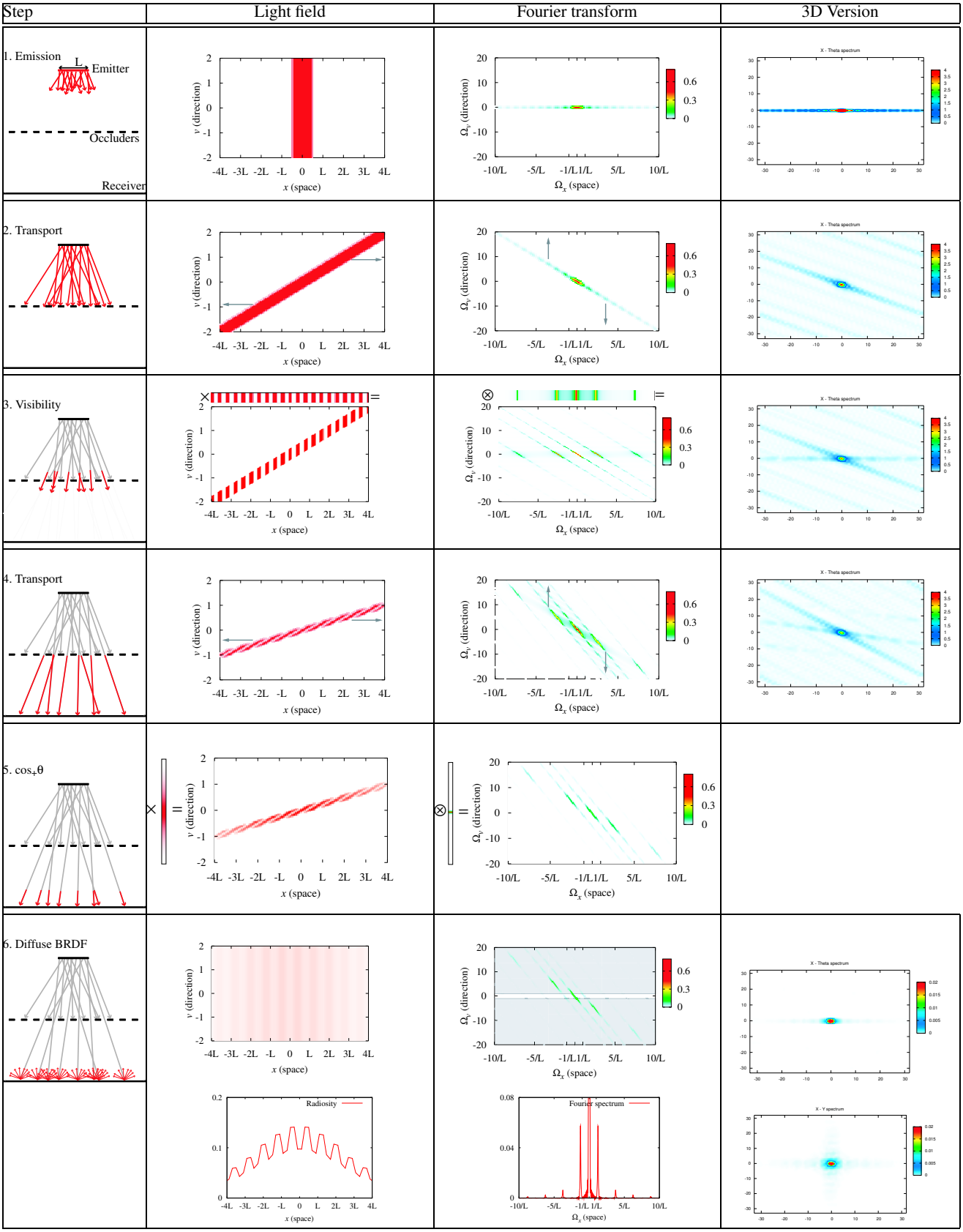
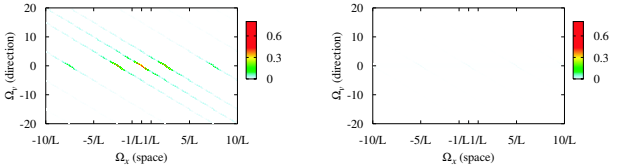


Figure 6: Effects on the spectrum of the various steps of light transport with a diffuse reflector. 2D Fourier transforms for steps 1 to 4 are obtained analytically; step 5 (convolution) is performed numerically. 3D Version spectrums are obtained numerically, *via* a Photon-Mapping algorithm and a FFT of the light field computed.



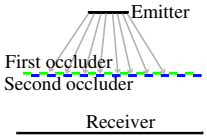


Figure 7: Scene configuration for visibility experiment. Left: spectrum with only one occluder. Right: spectrum with two occluders, computed with full precision and phase.

3.4 Example and discussion

Fig. 6 illustrates the various steps of light transport for a simple scene such as Fig. 9b. The slopes of the transport shears correspond to the travel distance (steps 2 and 4). Visibility increases the spatial frequency content through the convolution by a horizontal kernel in frequency space (step 3). There are only a finite number of blockers in Fig. 6, which explains why their spectrum is not a Dirac comb times a sinc, but a blurry version. The blocker spectrum mostly contains a main central lobe corresponding to the average occlusion and two side lobes corresponding to the blocker main frequency. This results in a replication of the sheared source spectrum on the two sides. The smaller the blocker pattern, the further-away these replicas are in frequency space. The final diffuse integration (step 6) discards all directional frequencies.

The main differences between the 3D and 2D plots of the spectra in Fig. 6 come from aliasing problems that are harder to fix with the 4D light field.Furthermore, in the 3D scene, the position of the blockers is jittered (see Fig. 9), which results in a smoother spectrum.

Feature-based visibility The spectra in Fig. 6 show that the second transport (step 4) pushes the "replicas" to the angular domain. This effect is more pronounced for high-frequency blockers, for which the replicas are farther from the vertical line. Since the final diffuse integration keeps only the spatial line of frequencies (step 5), the main high-frequency lobe of the blockers is eliminated by diffuse shading. This is related to the feature-based approach to visibility [Sillion and Drettakis 1995], where the effect of small occluders on soft shadows is approximated by an average occlusion. However, our finding goes one step further: where the feature-based technique *ignores* high frequencies, we show that, for small-enough blockers, most high-frequencies are effectively removed by integration.

Combining several blockers A difficult scene for visibility is the case of two occluders that individually block half of the light, and together block all the light (Fig. 7). In our framework, if one carries out the computations with full precision, taking phase into account, one gets the correct result: an empty spectrum (Fig. 7, right).

However, for practical applications, it is probably not necessary to compute the full spectrum. Instead, we consider elements of information about the maximal frequency caused by the scene configuration, as we show in Section 6.2. In that case, one can get an overestimation of the frequencies caused by a combination of blockers, but not an underestimation.

4 General case for surface interaction

So far, we have studied only diffuse shading for a central ray normal to a planar receiver (although rays in the neighborood have a non-normal incident angle). We now discuss the general case, taking into account the incidence angle, arbitrary BRDF, receiver curvature as well as spatial albedo variation. Our framework builds upon Ramamoorthi and Hanrahan [2001b] and extends it in several ways, which we discuss in Section 6.1.

In a nutshell, local reflection simply corresponds to a multiplication by the cosine term and a convolution by the BRDF. However, a number of reparameterizations are necessary to take into account the incidence and outgoing angles, as well as the surface curvature. We first treat the special case of rotation-invariant BRDFs such as Phong before addressing more general forms as well as texture and spatially-varying BRDFs. Recall that we study frequency content in ray neighborhoods, which means that for local reflection, we consider an incoming neighborhood and an outgoing neighborhood.

Plane-sphere parameterization Since local reflection mostly involves integrals over the directional dimension, it is more naturally expressed in a parameterization where angles are uniform. This is why we use here a plane-sphere parameterization where the directional component θ is the angle to the central ray (Fig. 3-b). The spatial dimension is unaffected.

In the plane-sphere parameterization, the domain of directions is the S^1 circle, which means that frequency content along this dimension is now a Fourier series, not a transform. Fig. 8 shows the effect of reparameterizing angles on the frequency plane. The frequency distribution is very similar, although the spectrum is blurred by the non-linear reparameterization. For bandwidth analysis, this introduces no significant error. Note that for all local interactions with the surface (and thus in this entire section), there is no limitation to small values of θ, the linearization $v = \tan\theta \approx \theta$ will only be used again after light leaves the surface, for subsequent transport.

4.1 Rotation-invariant BRDFs on curved receivers

Local shading is described by the shading equation

$$\ell_o(x_i, \theta_o) = \int_{\theta_i} \ell(x_i, \theta_i)\, \rho_{x_i}(\theta'_o, \theta'_i)\, \cos_+ \theta'_i \; \mathrm{d}\theta_i \qquad (8)$$

where the primed angles are in the local frame of the normal while the unprimed angles are in the global frame (Fig. 10). For now, we assume that the BRDF ρ does not vary with x_i. Local shading is mostly a directional phenomenon with no spatial interaction: the outgoing radiance at a point is only determined by the incoming radiance at that point. However, the normal varies per point.

As pointed out by Ramamoorthi and Hanrahan [2001b], local reflection combines quantities that are naturally expressed in a global frame (incoming and outgoing radiance) and quantities that live in the local frame defined by the normal at a point (cosine term and BRDF). For this, we need to rotate all quantities at each spatial location to align them with the normal. This means that we rotate (reparameterize) the incoming radiance, perform local shading in the local frame, and rotate (reparameterize) again to obtain the outgoing radiance in a global frame. All steps of the local shading process are illustrated in Fig. 10 and discussed below.

Step 1 & 7: Reparameterization into the tangent frame We first take the central incidence angle θ_0 into account, and reparameterize in the local tangent frame with respect to the central normal direction. This involves a shift by θ_0 in angle and a scale in space

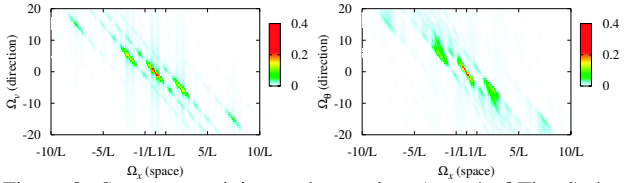
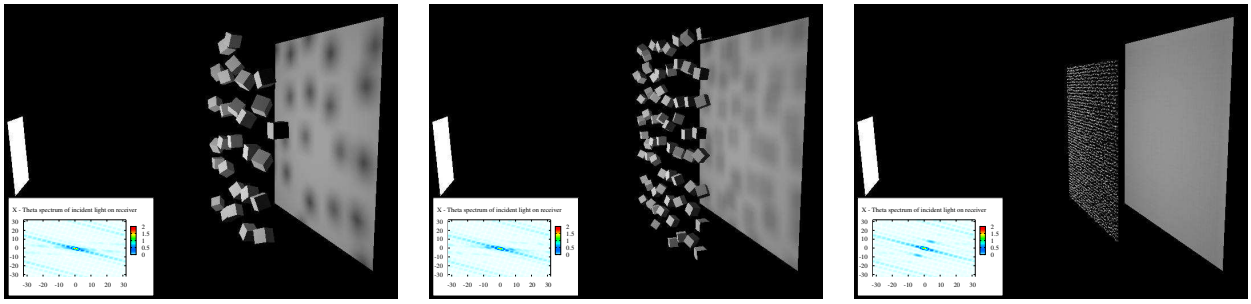


Figure 8: Spectrum arriving at the receiver (step 4 of Fig. 6), before and after the sphere-plane reparameterization. Left: (Ω_x, Ω_v) spectrum. Right: $(\Omega_x, \Omega_\theta)$ spectrum.



(a) (b) (c)

Figure 9: Complex frequency effects in light transport. The three scenes have the same area light and diffuse receiver and differ only by the frequency content of the blockers. (a) Large blockers result in few high frequencies. (b) With smaller (higher frequency) blockers, high frequencies increase on the receiver. (c) For very high-frequency blockers, high frequencies on the receiver nearly disappear.

by $1/\cos\theta_0$. We also flip the directions so that incident rays are pointing up and match the traditional local reflection configuration (Fig. 10, step 1). We omit the full derivation for brevity and provide directly the equations corresponding to step 1 and 7 of Fig. 10:

$$\widehat{\ell_i}(\Omega_x,\ \Omega_\theta) \quad = \quad \mathrm{e}^{-\mathrm{i}\Omega_\theta\theta_0}/|\cos\theta_0|\ \widehat{\ell}(-\Omega_x\cos\theta_0,\ \Omega_\theta) \tag{9}$$

$$\widehat{\ell''}(\Omega_x,\ \Omega_\theta) \quad = \quad \mathrm{e}^{\mathrm{i}\Omega_\theta\theta_1}\ |\cos\theta_1|\ \widehat{\ell_o}(\Omega_x/\cos\theta_1,\ \Omega_\theta) \tag{10}$$

Step 2: Per-point rotation The directional slice corresponding to each point must be shifted to rotate it in the local frame of the normal at that point (Fig. 10 step 2): $\theta_i' = \theta_i - \alpha(x_i)$.

For a smooth surface, we use a first-order Taylor expansion of the angle α of the normal at a point x_i. Given the curvature k, we have $\alpha(x_i) = kx_i$ and the reparameterization is $\theta_i' = \theta_i - k\ x_i$. This is a shear, but now along the directional dimension, in contrast to the transport shear. Similarly, the Fourier transform is sheared along the spatial dimension (Fig. 10 step 2, last row):

$$\widehat{\ell_i'}(\Omega_x', \Omega_\theta') \ = \ \widehat{\ell_i}(\Omega_x' + k\Omega_\theta',\ \ \Omega_\theta') \tag{11}$$

After this reparameterization, our two-dimensional spatio-directional local light field is harder to interpret physically. For each column, it corresponds to the incoming radiance in the frame of the local normal: the frame varies for each point. In a sense, we have unrolled the local surface and warped the space of light ray in the process [Wood et al. 2000]. The direction of the shear depends on the sign of the curvature (concave vs. convex).

Step 3: Cosine term and differential irradiance In the local frame of each point, we compute differential irradiance by multiplying by the spatially-constant clamped cosine function $\cos_+$. This multiplication corresponds in frequency space to a convolution by a Dirac in space times a narrow function in angle:

$$\widehat{\mathrm{d}E'}(\Omega_x, \Omega_\theta) \ = \ \widehat{\ell_i'}(\Omega_x, \Omega_\theta) \ \otimes \ \widehat{\cos_+}(\Omega_\theta)\delta_{\Omega_x=0} \tag{12}$$

Over the full directional domain, the spectrum of $\cos_+$ is:

$$\widehat{\cos_+}(\Omega_\theta) = \cos\left(\pi^2\Omega_\theta\right)\frac{2}{1-(2\pi\Omega_\theta)^2} \tag{13}$$

Most of the energy is centered around zero (Fig. 12-a) and the $1/\Omega_\theta^2$ frequency falloff comes from the derivative discontinuity[1] at $\pi/2$. Equivalent to the two-plane reparameterization (Section 3.3), the cosine term has only a small vertical blurring effect.

[1] A function with a discontinuity in the nth derivative has a spectrum falling off as $1/\Omega^{n+1}$. A Dirac has constant spectrum.

Step 4: Mirror-direction reparameterization Common BRDFs mostly depend on the difference between the mirror reflection and the outgoing direction. This is why we remap the local incoming light using a mirror reflection around the normal (Fig. 10 step 4): $\mathrm{d}E_r'(\theta_r') = \mathrm{d}E'(-\theta_i')$.

$$\widehat{\mathrm{d}E_r'}(\Omega_x,\ \Omega_\theta') = \widehat{\mathrm{d}E'}(\Omega_x,\ -\Omega_\theta') \tag{14}$$

This is equivalent to reparameterizations of surface light fields and BRDFs [Wood et al. 2000; Ramamoorthi and Hanrahan 2002].

Step 5: BRDF convolution In the mirror parameterization, assuming that the BRDF depends only on the angle difference, the shading equation 8 becomes:

$$\ell_o'(x_i, \theta_o') = \int_{\theta_r'} \mathrm{d}E_r'(x_i, \theta_r')\ \ \rho'(\theta_o - \theta_r')\ \ \mathrm{d}\theta_r' \tag{15}$$

Which is a convolution of $\mathrm{d}E_r'$ by ρ' for each x_i: that is, we convolve the 2D function $\mathrm{d}E_r'$ by a spatial Dirac times the directional shift-invariant BRDF ρ' (Fig. 10 step 5). In the Fourier domain, this is a multiplication by a spatial constant times the directional spectrum of the BRDF.

$$\widehat{\ell_o'}(\Omega_x', \Omega_\theta') \ = \ \widehat{\mathrm{d}E_r'}(\Omega_x', \Omega_\theta')\ \ \widehat{\rho'}(\Omega_\theta') \tag{16}$$

Note, however, that our expression of the BRDF is not reciprocal. We address more general forms of BRDF below.

Step 6: Per-point rotation back to tangent frame We now apply the inverse directional shear to go back to the global frame. Because we have applied a mirror transform in step 4, the shear and inverse shear double their effect rather than canceling each other. Since the shear comes from the object curvature, this models the effect of concave and convex mirror and how they deform reflection. In particular, a mirror sphere maps the full 360 degree field to the 180 degree hemisphere, as exploited for light probes.

4.2 Discussion

The important effects due to curvature, cosine term, and the BRDF are summarized in Fig. 10. Local shading is mostly a directional phenomenon, and the spatial component is a double-shear due to curvature (step 2 and 6). The cosine term results, in frequency space, in a convolution by a small directional kernel (step 3) while the BRDF band-limits the signal with a multiplication of the spectrum (step 5). Rougher materials operate a more aggressive low-pass, while in the special case of mirror BRDFs, the BRDF is a Dirac and the signal is unchanged.

Curvature has no effect on the directional bandwidth of the outgoing light field, which means that previous bounds derived in the

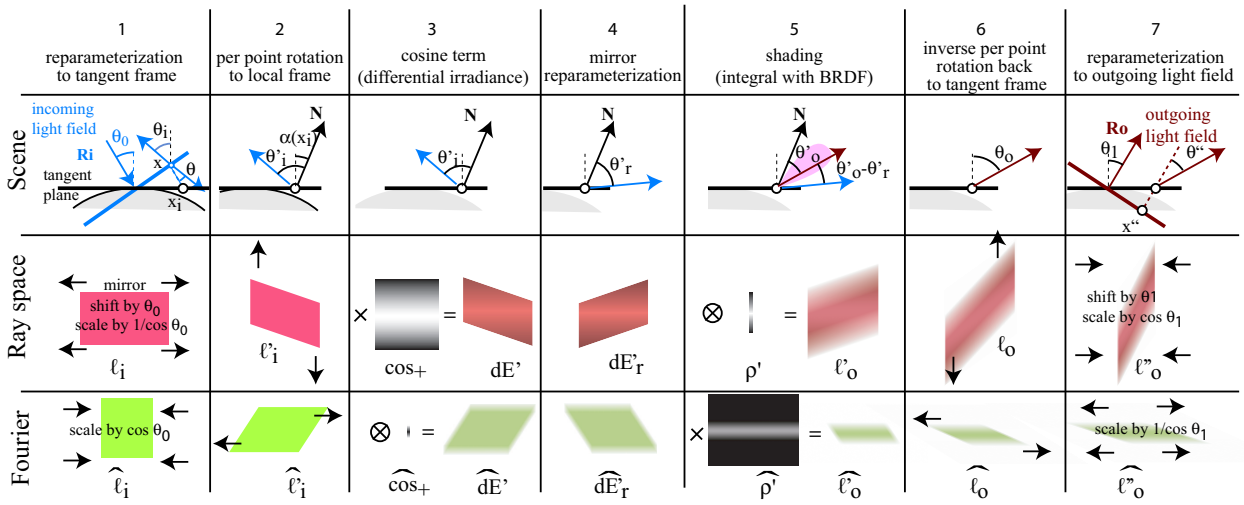


Figure 10: Local shading for a curved receiver with arbitrary BRDF.

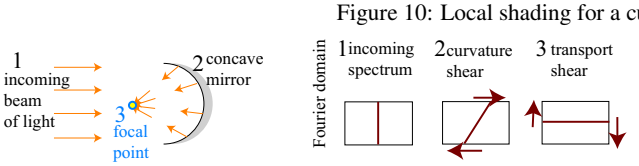
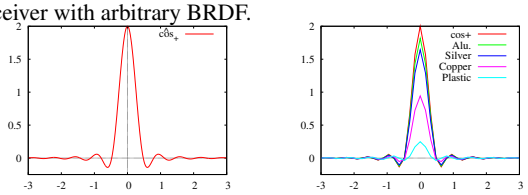


Figure 11: Caustic due to negative curvature. A shear in angle and then in space are combined to result in a transfer from the directional frequencies to the spatial frequencies.



(a) $\widehat{\cos_+}$ (b) $\widehat{\cos_+ f}$

Figure 12: Spectrum of the clamped cosine in the sphere-plane parameterization. Spectrum of $\cos_+ f$ (cosine and Fresnel terms) for different materials.

special case of inifinite lighting [Ramamoorthi and Hanrahan 2004; Basri and Jacobs 2003; Ramamoorthi and Hanrahan 2001a; Ramamoorthi and Hanrahan 2002] are valid for spatially-varying illumination. However, the spatial frequency content is strongly affected by curvature, which has important practical implications.

The effect of the curvature shear is further increased by the spatial scaling back to the tangent frame in step 7, as described by Eq. 10. We stress that this explains the well-known difficulty in sampling specular lighting in situations such as backlighting on the silhouette of a curved object. This is modeled by the effect of the curvature shear, the BRDF bandwidth, and the angular scale due to rotation into the tangent frame.

A case study: simple caustics Caustics are an example of the interaction between spatial and angular aspects of light transport. We illustrate this effect with a simple case similar to a solar oven (Fig 11). A parallel beam of light hits a surface of negative curvature with a mirror (Dirac) BRDF and converges toward a focal point. This is modeled in our framework by an incoming spectrum that has energy only in the angular domain. The shear due to curvature followed by the shear due to transport result in a signal where the energy is concentrated in space: it is a Dirac at the focal point.

4.3 Rotation-varying BRDFs

Not all BRDFs can be simplified into a term that depends only on the difference to the mirror direction. For example, the Fresnel term depends on the incoming angle. We now derive the effect of shading by a BRDF that is factored into separable terms that depend on the incoming angle θ'_i and the difference between the outgoing angle θ'_o and the mirror direction θ'_r [Ramamoorthi and Hanrahan 2002], that is, $\rho(\theta'_i, \theta'_o) = f(\theta'_i)\, \rho'(\theta_o - \theta'_r)$.

Since the term f does not depend on the outgoing angle, it can be applied in the same way as the $\cos_+$ term, using a multiplication that corresponds to a convolution in frequency space; the rest of the shading remains the same with a convolution by ρ'. Combining the multiplication by f with the mirror reparameterization of step 4 and the convolution by ρ' of step 5, we obtain in frequency space a convolution followed by a multiplication:

$$\widehat{\ell'_o}(\Omega'_x, \Omega'_\theta) = \left(\widehat{dE'_r}(\Omega'_x, \Omega'_\theta) \otimes \widehat{f}(-\Omega'_\theta)\delta_{\Omega_x=0}\right) \widehat{\rho'}(\Omega'_\theta) \quad (17)$$

Fig. 12-b shows the spectra of the cosine term $\cos_+$ multiplied by the Fresnel term for typical materials; it contains mostly low frequencies. Other approximations with separable functions depending on θ'_o are equally easy, just reversing the order of the multiplication and convolution. BRDFs are often approximated by sums of separable terms, which can be handled easily in our framework because the Fourier transform is linear.

4.4 Texture mapping

When the result of shading is modulated by a texture $T(x)$, this multiplication corresponds to a convolution in the Fourier domain:

$$\widehat{\ell_T}(\Omega_x, \Omega_\theta) = \widehat{T}(\Omega_x)\delta_{\Omega_\theta=0} \otimes \widehat{\ell_o}(\Omega_x, \Omega_\theta) \quad (18)$$

Since the texture has only spatial components, its spectrum is restricted to the line of spatial frequencies. This means that texture mapping only affects frequencies along the spatial dimension.

4.5 Spatially-varying BRDFs

We now extend our model to include spatially-varying BRDFs and revisit step 5 (shading). For each point, shading is still a convolution

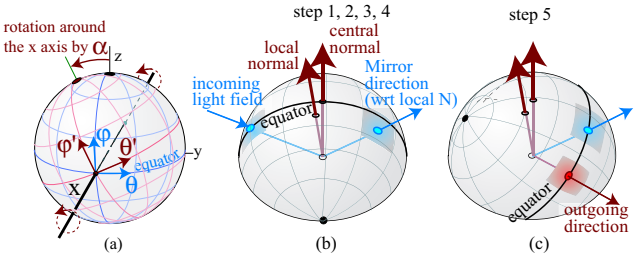


Figure 13: 3D direction parameterizations.

over the directional domain, but the kernel varies spatially.

To model this effect, we exploit the fact that a 2D Fourier transform can be decomposed into two separable 1D transforms, the first one vertically, then horizontally. We consider the intermediate semi-Fourier space $\mathring{\ell}(x, \Omega_\theta)$ that represents for each location x the 1D Fourier transform of the directional variation of incoming light. The full Fourier space is then the 1D Fourier transform of the semi-Fourier transform along the x dimension. We have

$$\mathring{\ell}'_o(x, \Omega_\theta) = \mathrm{d}\mathring{E}'_r(x, \Omega_\theta)\, \mathring{\rho}(x, \Omega_\theta),$$

which is a multiplication in the semi-Fourier domain, and therefore a convolution *along x only* in full Fourier space:

$$\widehat{\ell'}(\Omega_x, \Omega_\theta) = \widehat{\mathrm{d}E'_r}(\Omega_x, \Omega_\theta) \quad \otimes_x \quad \widehat{\rho}(\Omega_x, \Omega_\theta)$$

This means that in order to characterize the effect of spatially-varying BRDFs, we consider the spectrum of $\rho(x, \theta)$. We then take the spectrum of the incoming illumination $\widehat{\ell}$ and convolve it only horizontally along Ω_x, not vertically. We call this a semi-convolution in Ω_x, which we note $\otimes_x$.

In the special case of non-varying BRDFs, the spectrum of $\rho(x, \theta)$ is a Dirac times the directional spectrum of the BRDF. The horizontal convolution is a multiplication. If the spectrum of ρ is separable (texture mapping), then the spatially-varying BRDF case is a multiplication followed by a convolution. The special case of a a spatially-varying combination of BRDFs [Lensch et al. 2001] can be handled more simply as the superposition of multiple BRDFs with weights encoded as textures.

5 Extension to 3D

We now show how our framework extends to 3D scenes.

5.1 Light-field parameterization phenomena

The derivations presented in Section 3 involve a two-plane light-field parameterization and extend directly to 3D. The only notable difference is the calculation of differential irradiance (Eq. 7), where the projected surface area in 3D becomes:

$$\mathrm{d}A_\perp = \frac{\mathrm{d}u\,\mathrm{d}v}{(1 + v^2 + u^2)^2} = G(u, v)\,\mathrm{d}u\,\mathrm{d}v$$

Fig. 5-c presents the spectrum of $G(u, v)$.

5.2 Shading in plane-sphere parameterization

The sphere S^2 of directions is unfortunately hard to parameterize, which prompted many authors to use spherical harmonics as the equivalent of Fourier basis on this domain. In contrast, we have chosen to represent directions using spherical coordinates and to use traditional Fourier analysis, which is permitted by our restriction to local neighborhoods of S^2. This solution enables a more direct extension of our 2D results, and in particular it expresses well the interaction between the spatial and angular components.

Spherical coordinates We use the spherical coordinates θ, φ where θ, in $[-\pi, \pi]$, is the azimuth and φ, in $[-\pi/2, \pi/2]$, the co-latitude. The distortion of this parameterization is $\cos\varphi$, which means that one must remain around the equator to avoid distortion. In this neighborhood, the parameterization is essentially Euclidean, to a first-order approximation.

Local reflection is challenging because it involves four neighborhoods of direction around: the incoming direction, the normal, the mirror direction, and the outgoing direction; in general, we cannot choose a spherical parameterization where they all lie near the equator. Fortunately, we only need to consider two of these neighborhoods at a time (Fig. 4).

For this, we exploit the fact that a rotation around an axis on the equator can be approximated to first order by a Euclidean rotation of the (θ, φ) coordinates: $(\theta', \varphi') = \boldsymbol{R}_\alpha\,(\theta, \varphi)$

For brevity, we omit the comprehensive remapping formulas for 3D shading, but we describe the appropriate parameterization for each step as well as the major differences with the 2D case.

Tangent frame We start with a parameterization where the equator is in the incident plane, as defined by the central ray of the incident light field and the central normal vector (Fig. 13-b). If the light field has been properly rotated, only the x spatial dimension undergoes the scaling by $\cos\theta_0$ (Eq. 9)

Curvature In 2D, we approximated the angle with the local normal linearly by $\alpha(x) = kx$; For a surface, the corresponding linearization of the normal direction (θ_N, φ_N) involves a bilinear form [Do Carmo 1976]:

$$(\theta_N, \varphi_N) = \boldsymbol{M}\,(x, y) \tag{19}$$

If x and y are aligned with the principal directions of the surface, the matrix $\boldsymbol{M}$ is an anisotropic scaling where the scale factors are the two principal curvatures. The corresponding remapping of (x, y, θ, φ) is a shear in 4D, with different amounts in the two principal directions. As with the 2D case, the frequency content is sheared along the spatial dimensions.

Differential irradiance and cosine Step 3 is mostly unchanged. Since we placed the equator along the incident plane, the cosine term depends only on θ to a first approximation. The spectrum is convolved with a small 1D kernel in θ (Fig. 12-a).

Rotationally-symmetric BRDFs The mirror reparameterization of step 4 is unchanged, and the angles remain near the equator since the equator also contains the normals. We express the convolution of the mirrored incoming light field by the BRDF in the neighborhood of the outgoing direction. For this, we rotate our spherical coordinates so that the new equator contains both the mirror direction and the direction of the central outgoing ray (Fig. 13-c). Because all the angles are near the equator, the difference angles between an outgoing ray and a mirrored ray can be approximated by $\theta'_o - \theta'_r$ and $\varphi'_o - \varphi'_r$, and Eq. 16 applies.

Recap of rotations In summary, we first need to rotate the light-field parameterization so that the central incidence plane is along one of the axes before reparameterizing from two-plane to sphere-plane (Fig. 13-b). We then need to rotate between the mirror reparameterization and the BRDF convolution to place the central outgoing direction on the equator (Fig. 13-c). Finally we rotate again to put the outgoing plane defined by the normal and central outgoing direction in the equator (not shown).

5.3 Anisotropies in 3D

Our extension to 3D exploits the low distortion of spherical coordinates near the equator, at the cost of additional reparameterization

	Ray space	Fourier	Spectrum formula
Transport			
Travel	shear	shear	$\widehat{\ell}(\Omega_x,\ \Omega_v + d\Omega_x)$
Visibility	multiplication	convolution	$\widehat{\ell} \otimes \widehat{V}$
Local geometric configuration			
Light incidence	scale spatial	scale spatial	$\frac{e^{-i\Omega_\theta \theta_0}}{\lvert\cos\theta_0\rvert}\ \widehat{\ell}(-\Omega_x \cos\theta_0,\ \Omega_\theta)$
Outgoing angle	scale spatial	scale spatial	$e^{i\Omega_\theta \theta_1}\ \lvert\cos\theta_1\rvert\ \widehat{\ell_o}(\frac{\Omega_x}{\cos\theta_1}, \Omega_\theta)$
Curvature	shear	shear	$\widehat{\ell_i}(\Omega'_x - k\Omega'_\theta\ ,\ \Omega'_\theta)$
Local shading			
Cosine term	multiplication	convolution	$\widehat{\ell'_i} \otimes \widehat{\cos_+}$
BRDF	convolution	multiplication	$\rho'\ \widehat{dE'_r}$
Texture mapping	multiplication	convolution	$\widehat{T} \otimes \widehat{\ell}$
Separable BRDF	multiplication then convolution	convolution then multiplication	$\left(\widehat{dE'_r} \otimes \widehat{f}\right)\ \rho'$
Space-vary. BRDF	semi-convolution (angles only)	semi convolution (spatial only)	$\widehat{dE'}\ \otimes_x\ \widehat{\rho}$

Table 1: Summary of all phenomena

to align the equator with the relevant neighborhoods. Fortunately, these reparameterization act locally like Euclidean rotations along axes that preserve the space-angle separation.

Compared to 2D , the 3D case involves anisotropies both in the directional and spatial components. The *spatial* scale to account for the incident and exitant angle affects only one of the spatial dimensions, along the corresponding plane normal to the tangent frame. Curvature is usually different along the two principal directions. The *directional* cosine term mostly depends on θ, while rotationally-symmetric BRDFs only depend on the spherical distance between mirror and outgoing directions and is more influenced by θ except around the specular peak. These additional anisotropies make the 3D situation more complex, but locally they correspond to linear transforms and preserve the distinction and interaction between spatial and directional effects derived in 2D.

Other local shading effects such as separable BRDFs, texture mapping, and spatially-varying BRDFs can be directly extended from Section 4. While the formulas are complex and are not derived in this paper, the qualititative effects and relevant parameters remain the same as in 2D.

6 Discussion

Table 1 summarizes the building blocks that compose our frequency analysis of light transport. This variety of phenomena can be characterized using simple mathematical operators: scale, shear, convolution and multiplication. Even spatially-varying BRDFs can be handled using a semi-convolution that occurs only along the spatial dimensions.

Some operations such as occlusion are simpler in the original ray space, while others such as shading are more natural in frequency space. Our framework allows us to express them all in a unified way. As discussed above, the 3D case essentially follows the 2D derivation, with additional reparameterizations steps and anisotropies.

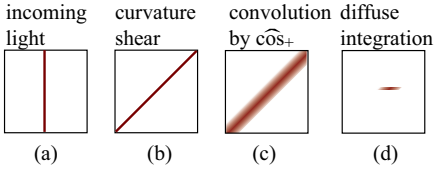


Figure 14: Special case of diffuse shading for an infinite environment map. In this case, the convolution and multiplication are equivalent to a mutltiplication.

In practice, the notion of locality is invoked for three different reasons, whose importance depends on the context and application:
– the use of first-order Taylor series, for example for the curvature or for the $\tan\theta \approx \theta$ remapping,
– the principle of uncertainty, which states that low frequencies cannot be measured on small windows (in which case big neighborhoods are desired), or in other words, that localization is not possible in space and frequency at the same time,
– most real scenes are not stationary, that is, their properties such as the presence and size of blockers vary spatially. Smaller neighborhoods might mean more homogeneous properties and more locally-pertinent conclusions.

We now discuss how our work extends previous frequency characterization, before illustrating how it can be applied, through a proof of concept in the context of ray-tracing.

6.1 Relation to previous work

Light field sampling Our formulation of transport in free space is similar to the derivations of light-field spectra [Isaksen et al. 2000; Chai et al. 2000], and the same relationship between slope and distance is found. Our expression as a transport operator makes it easier to extend these analyzes to arbitrary input signals, and in particular to non-Lambertian objects and occlusion.

Ray footprint Approaches based on ray differentials [Shinya et al. 1987; Igehy 1999; Chen and Arvo 2000] capture the shear transforms due to transport and curvature, and our first-order Taylor expansion for curvature corresponds to the same differentials. The approach by Igehy [1999] only uses 2D derivatives by considering only ray paths that converge to the viewpoint.

Signal processing framework for local reflection Our framework extends Ramamoorthi and Hanrahan's signal processing framework for local reflection [2004] with the following key differences:
– we take into account spatial variation of the incoming light and the curvature of the receiver,
– however, we characterize reflection only for a ray neighborhood,
– they parameterize the outgoing radiance by by α and θ'_o, while we use a more natural outgoing parameterization in the global frame, at the cost of reparameterization,
– as discussed above, our expression of the cosine term is a convolution in the frequency domain. This cleanly separates the computation of incoming irradiance and BRDF convolution, at the cost of additional steps. It also allows us to express the cosine term for BRDFs such as Phong.

On convolution It might come as a surprise that two phenomena that have been expressed by previous work as convolutions in the primary space, soft shadows [Soler and Sillion 1998] and the cosine term [Ramamoorthi and Hanrahan 2004] correspond in our framework to convolutions *in the frequency domain*. We show here that our formulation in fact extends these previous work and that the primary-space convolution is a special case. The key is that they consider functions that do not vary in one of the domains (space resp. direction). The corresponding spectra are therefore restricted to a line, since the Fourier transform of a constant is a Dirac.

Consider the cosine term for infinitely-distant light sources. The lighting varies only in angle, and its spectrum is restricted to the vertical line of directions (Fig. 14(a)). After the curvature shear, it is a 1D function on the line $k\Omega_\theta = \Omega_x$ (Fig. 14(b)), which we convolve with the vertical kernel $\widehat{\cos_+}$. However, for each spatial frequency Ω_x, there is only one non-zero value of the sheared function. As a result, this convolution is a so-called *outer product* of the two one-dimensional functions, and the result is the pairwise product $\widehat{dE'}(\Omega_x, \Omega_\theta) = \ell_i(0, \Omega_x/k)\widehat{\cos_+}(\Omega_x/k - \Omega_\theta)$. (Fig. 14(c)). The diffuse integration then restricts the function to the line $\Omega_\theta = 0$, where $dE'(\Omega_x, 0) = \ell'_i(0, \Omega_x/k)\widehat{\cos_+}(\Omega_x/k)$. The convolution followed by

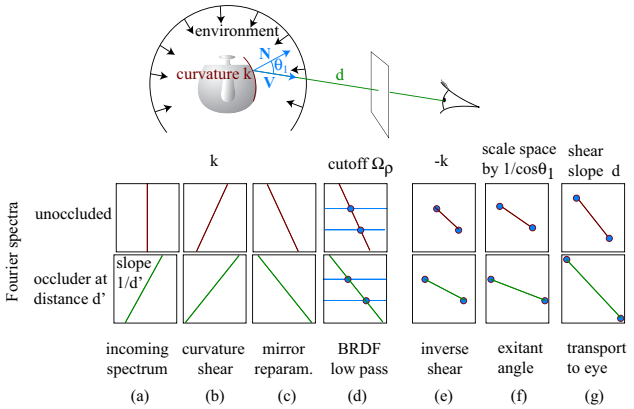


Figure 15: Bandwidth derivation for our adaptive ray tracer.

restriction to the horizontal line turned into a simple product of 1D functions, which corresponds to a convolution in the *primary* space.

The case of soft shadows studied by Soler and Sillion [1998] is similar: the emitter is diffuse and has a spectrum restricted to $\Omega_v = 0$, and the blockers are planar and have the same restrictions. The transport from source to occluders results in slanted lines in frequency space that are convolved together (Fig. 6, step 3). Our framework extends these two cases to arbitrary cases where the spectra are not restricted to lines.

6.2 Sampling rate for glossy rendering

We show how our framework can be used to drive image-space sampling in ray tracing. In particular, we illustrate how our framework can be used to derive sampling rates for algorithms that *do not need to perform computations in the Fourier domain*. While we demonstrate a working implementation, we emphasize that this application is meant only as a proof of concept and that further development is necessary to make it fully general, as we discuss below.

We are motivated by the rendering of glossy materials, which despite effective recent developments [Lawrence et al. 2004] remains computationally expensive. We observe, however, that glossy objects appear blurry, so it should be possible to reduce the image sampling rate. Our framework permits a quantitative expression of the required sampling rate.

Unoccluded environment map We first consider the case of environment-map rendering without occlusion. The incoming light field has only directional content (Fig. 15), and the light incidence angle (Table 1 row 1) has no effect. The shear of curvature (b) results in a line of slope k that gets convolved with the cosine narrow kernel $\widetilde{\cos_+}$, which we neglect. After mirror reparameterization (c), a glossy BRDF band-limits this signal (d), which we approximate by a cutoff at an angular frequency Ω_ρ. This cutoff depends on the BRDF of the object visible in a given region of the image. The upper endpoint of the resulting segment is at coordinate $(-k\Omega_\rho, \Omega_\rho)$. We apply the inverse shear (step e) and the scale by $1/\cos\theta_1 = 1/(\boldsymbol{n}.\boldsymbol{v})$, where $\boldsymbol{n}$ is the normal and $\boldsymbol{v}$ the unit vector towards the viewpoint. We obtain a maximum frequency of $(-\frac{2k}{\cos\theta_1}\Omega_\rho, \Omega_\rho)$ for the light leaving an object in the direction of the viewpoint (Fig. 15 step f). A transport shear with distance d yields a bound of $\left(-\frac{2k}{\cos\theta_1}\Omega_\rho,\ \ \Omega_\rho - d\ \frac{2k}{\cos\theta_1}\Omega_\rho\right)$

A view computation corresponds to the restriction of the function to the directional domain, and if we assume $d >> 1$, we obtain the following approximate bound on the directional bandwidth for a region of the image:

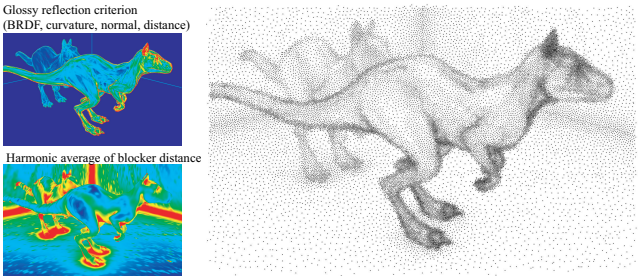


Figure 16: Criteria and sampling pattern used to render Fig. 17. The sampling adapts to curvature, the viewing angle, the BRDF as well as the harmonic average of the distance to potential blockers.

$$B = d\ \frac{2k}{\boldsymbol{n}.\boldsymbol{v}}\Omega_\rho \tag{20}$$

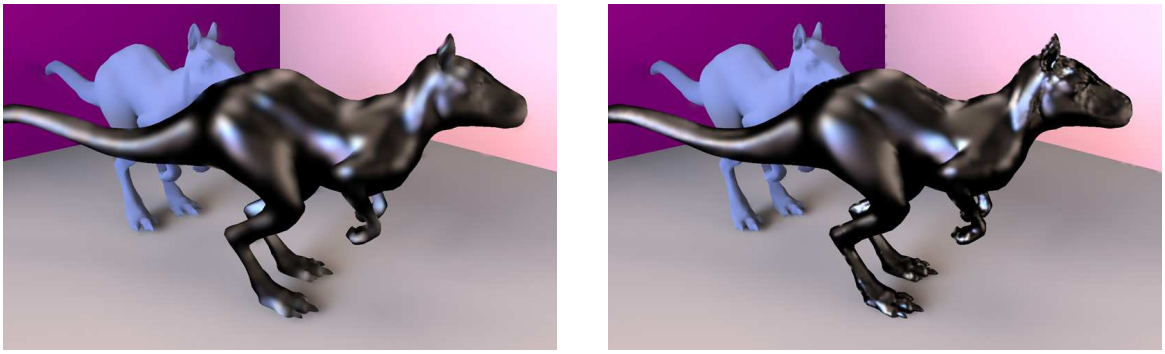
This corresponds to the difficulty in appropriately sampling curved objects at grazing angles, as discussed in Section 4.2. In addition, distant objects are minified and the apparent curvature is increased. In 3D, the curvature and normal angle involve anisotropy, and in practice, we use the absolute value of the larger principal curvature. However, our implementation computes this criterion directly in screen-space with finite-difference approximation to the curvature. As a result, the effect of the normal angle, the distance, and the anisotropies are included for free; see Fig. 16 for a visualization of this criterion. The faceted look is due to the finite-difference curvature and Phong normal interpolation. The BRDF bandwidth for the two dinosaurs and environments were approximated manually based on the BRDF exponent.

Occlusion The above derivation assumes full visibility. We integrate the effect of the blockers using a worst-case assumption on the blocker spectrum, we consider that it has content at all frequency. Based on the effect of the transport shear, we approximate the spectrum due to a blocker at distance d' by a line of slope $1/d'$. Going through the same steps, we obtain an approximate bound of:

$$B' = d\ \frac{1}{\boldsymbol{n}.\boldsymbol{v}}\left(\frac{1}{d'} + 2k\right)\Omega_\rho \tag{21}$$

To evaluate this criterion, we use the harmonic average of the distance to occluders. This information is computed by sampling a hundred rays for a small set of visible points in the image, in practice 20,000. The criteria is reconstructed over the image using the same reconstruction algorithm as for the final image, which we describe shortly. The blocker criterion is shown Fig. 15. It is similar to Ward et al.'s criterion for irradiance caching [Ward et al. 1988], but expressing it in a unified frequency framework allows us to combine it with other bandwidth considerations such as BRDF roughness.

Algorithm and image reconstruction Our proof-of-concept computes visibility using four samples per pixel, but uses aggressively-sparse samples for shading: on average, 0.05 samples per pixel. We use an edge-preserving reconstruction that exploits the higher-resolution depth and normal to reconstruct the shading, in the spirit of McCool's filtering of Monte-Carlo ray tracing outputs [1999] but based on a bilateral filter [Tomasi and Manduchi 1998]. As demonstrated in Fig. 17, this results in a smooth reconstruction where needed and on sharp silhouettes. The spatial width of the bilateral filter is scaled according to the bandwidth prediction. Given a bandwidth map, we use the blue-noise method by Ostromoukhov et al. [2004] to generate a set of image samples (Fig. 17, right). In summary, our algorithm is as follows:



Uniform sampling | Using our bandwidth prediction

Figure 17: Scene rendered without and with adaptive sampling rate based on our prediction of frequency content. Only 20,000 shading samples were used to compute these 800×500 image. Note how our approach better captures the sharp detail in the shiny dinosaur's head and feet. The criteria and sampling are shown in Fig. 16. Images rendered using PBRT [Pharr and Humphreys 2004]

```
Compute visibility at full resolution
        Use finite-differences for curvature criterion
Compute harmonic blocker distance for sparse samples
        Perform bilateral reconstruction
Compute B' based on blocker and curvature
Generate blue noise sampling based on B'
        Compute shading for samples
        Perform bilateral reconstruction
```

Observe how our sampling density is increased in areas of high curvature, grazing angles, and near occluders. The environment map casts particularly soft shadows, and note how the high-frequency detail on the nose of the foreground dinosaur is well captured, especially given that the shading sampling is equivalent to a 200×100 resolution image.

Although these results are encouraging, the approach needs improvement in several areas. The visibility criterion in particular should take into account the light source intensity in a directional neighborhood to better weight the inverse distances. Even so, the simple method outlined above illustrates how knowledge of the modifications of the frequency content through light transport can be exploited to drive rendering algorithms. In particular, similar derivations are promising for precomputed radiance transfer [Sloan et al. 2002] in order to relate spatial and angular sampling.

7 Conclusions and future work

We have presented a comprehensive framework for the description of radiance in frequency space, through operations of light transport. By studying the local light field around the direction of propagation, we can characterize the effect of travel in free space, occlusion, and reflection in terms of frequency content both in space and angle. In addition to the theoretical insight offered by our analysis, we have shown that practical conclusions can be drawn from a frequency analysis, without explicitly computing any Fourier transforms, by driving the sampling density of a ray tracer according to frequency predictions.

Future work On the theory side, we are working on the analysis of additional local shading effects such as refraction, bump-mapping, and local shadowing [Ramamoorthi et al. 2004]. We hope to study the frequency cutoff for micro, meso, and macro-geometry effects [Becker and Max 1993]. The study of participating media is promising given the ability of Fourier analysis to model differential equations. The spectral analysis of light interaction in a full scene is another challenging topic. Finally, the addition of the time dimension is a natural way to tackle effects such as motion blur.

We are excited by the wealth of potential applications encompassed by our framework. In rendering, we believe that many traditional algorithms can be cast in this framework in order to derive theoretical justification, but also to allow extensions to more general cases (such as from diffuse to glossy). Our preliminary study of sampling rates in ray tracing is promising, and we want to develop new algorithms and data structures to predict local bandwidth, especially for occlusion effects. Precomputed radiance transfer is another direct application of our work.

Our analysis extends previous work in inverse rendering [Ramamoorthi and Hanrahan 2001b; Basri and Jacobs 2003] and we are working on applications to inverse rendering with close-range sources, shape from reflection, and depth from defocus.

Acknowledgments

We thank Jaakko Lehtinen, the reviewers of the Artis and MIT teams, as well as the SIGGRAPH reviewers for insightful feedback.

This work was supported by an NSF CAREER award 0447561 "Transient Signal Processing for Realistic Imagery," an NSF CISE Research Infrastructure Award (EIA9802220), an ASEE National Defense Science and Engineering Graduate fellowship, the European Union IST-2001-34744 "RealReflect" project, an INRIA équipe associée, and the MIT-France program.

References

A , T., K , J., D , F., S , H.-P. 2004. Spherical harmonic gradients for mid-range illumination. In *Rendering Techniques 2004 (Proc. EG Symposium on Rendering 2004)*.

A , J. 1994. The irradiance Jacobian for partially occluded polyhedral sources. In *Computer Graphics* Proceedings, Annual Conference Series, ACM SIGGRAPH, 343–350.

B , R., J , D. 2003. Lambertian reflectance and linear subspaces. *IEEE Trans. Pattern Anal. Mach. Intell. 25*, 2.

B , B. G., M , N. L. 1993. Smooth transitions between bump rendering algorithms. In *Computer Graphics* Proceedings, Annual Conference Series, ACM SIGGRAPH, 183–190.

B , M. R., M , G. W. 1995. A frequency based ray tracer. In *Computer Graphics* Proceedings, Annual Conference Series, ACM SIGGRAPH, 409–418.

B , M. R., M , G. W. 1998. A perceptually based adaptive sampling algorithm. In *Computer Graphics* Proceedings, Annual Conference Series, ACM SIGGRAPH, 299–309.

C , E., L , A., F , D. 1998. Uniformly sampled light fields. In *Rendering Techniques '98 (Proc. of EG Workshop on Rendering '98)*, Eurographics, 117–130.

C , J.-X., C , S.-C., S , H.-Y., T , X. 2000. Plenoptic sampling. In *Computer Graphics* Proceedings, Annual Conference Series, ACM SIGGRAPH, 307–318.

C , M., A , J. 2000. Theory and application of specular path perturbation. *ACM Trans. Graph. 19*, 4, 246–278.

D C , M. 1976. *Differential Geometry of Curves and Surfaces*. Prentice Hall.

F , J. A., S , P., P , S. N., G , D. P. 1997. A model of visual masking for computer graphics. In *Computer Graphics* Proceedings, Annual Conference Series, ACM SIGGRAPH, 143–152.

F , D., S , D., B , R. 2004. Accuracy of spherical harmonic approximations for images of Lambertian objects under far and near lighting. In *ECCV 2004, European Conference on Computer Vision*, 574–587.

G , J. W. 1996. *Introduction To Fourier Optics*. McGraw-Hill.

G , S. J., S ¨ , P., C , M. F., H , P. 1993. Wavelet radiosity. In *Computer Graphics* Proceedings, Annual Conference Series, ACM SIGGRAPH, 221–230.

H , M. 1994. Holographic stereograms as discrete imaging systems. In *SPIE Proc. Vol. 2176: Practical Holography VIII*, S. Benton, Ed., SPIE, 73–84.

H , P. 1989. *Fundamentals of Texture Mapping and Image Warping*. Master's thesis, University of California at Berkeley, Computer Science Division.

H , N., S , F. X. 1998. An exhaustive error-bounding algorithm for hierarchical radiosity. *Computer Graphics Forum 17*, 4.

I , H. 1999. Tracing ray differentials. In *Computer Graphics* Proceedings, Annual Conference Series, ACM SIGGRAPH.

I , A., M M , L., G , S. J. 2000. Dynamically reparameterized light fields. In *Computer Graphics* Proceedings, Annual Conference Series, ACM SIGGRAPH, 297–306.

K , A. 2001. Hierarchical monte carlo image synthesis. *Mathematics and Computers in Simulation 55*, 1–3 (Feb.), 79–92.

L , J., R , S., R , R. 2004. Efficient BRDF importance sampling using a factored representation. *ACM Transactions on Graphics (Proc. SIGGRAPH 2004) 23*, 3 (Aug.), 496–505.

L , H. P. A., K , J., G , M., H , W., S - , H.-P. 2001. Image-based reconstruction of spatially varying materials. In *Rendering Techniques '01 (Proc. EG Workshop on Rendering 2001)*, Eurographics, 104–115.

M , J., R , R. 1997. Computing local surface orientation and shape from texture for curved surfaces. *International Journal of Computer Vision 23*, 2, 149–168.

M C , M. D. 1999. Anisotropic diffusion for monte carlo noise reduction. *ACM Transactions on Graphics 18*, 2, 171–194.

M , K. 1998. The visible differences predictor: applications to global illumination problems. In *Rendering Techniques '98 (Proc. EG Workshop on Rendering '98)*, Eurographics.

O , V., D , C., J , P.-M. 2004. Fast hierarchical importance sampling with blue noise properties. *ACM Transactions on Graphics (Proc. SIGGRAPH 2004) 23*, 3 (Aug.), 488–495.

P , A. P. 1987. A new sense for depth of field. *IEEE Transactions on Pattern Analysis and Machine Intelligence 9*, 4 (July).

P , M., H , G. 2004. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann.

R , R., H , P. 2001. An efficient representation for irradiance environment maps. In *Computer Graphics* Proceedings, Annual Conference Series, ACM SIGGRAPH.

R , R., H , P. 2001. A signal-processing framework for inverse rendering. In *Computer Graphics* Proceedings, Annual Conference Series, ACM SIGGRAPH.

R , R., H , P. 2002. Frequency space environment map rendering. *ACM Transactions on Graphics (Proc. SIGGRAPH 2002) 21*, 3, 517–526.

R , R., H , P. 2004. A signal-processing framework for reflection. *ACM Transactions on Graphics 23*, 4.

R , R., K , M., B , P. 2004. A Fourier theory for cast shadows. In *ECCV 2004, European Conference on Computer Vision*, 146–162.

S , M., T , T., N , S. 1987. Principles and applications of pencil tracing. *Computer Graphics (Proc. SIGGRAPH '87) 21*, 4, 45–54.

S , F., D , G. 1995. Feature-based control of visibility error: A multi-resolution clustering algorithm for global illumination. In *Computer Graphics* Proceedings, Annual Conference Series, ACM SIGGRAPH, 145–152.

S , P.-P., K , J., S , J. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. on Graphics 21*, 3, 527–536.

S , C., S , F. X. 1998. Fast calculation of soft shadow textures using convolution. In *Computer Graphics* Proceedings, Annual Conference Series, ACM SIGGRAPH, 321–332.

S , J., Y , J., G , S. J., M M , L. 2003. A new reconstruction filter for undersampled light fields. In *Proc. EG Symposium on Rendering 2003*, Eurographics, 150–156.

T , C., M , R. 1998. Bilateral filtering for gray and color images. In *Proc. IEEE International Conference on Computer Vision*, IEEE, 836–846.

W , G. J., H , P. 1992. Irradiance gradients. In *Proc. of EG Workshop on Rendering '92*, Eurographics, 85–98.

W , G. J., R , F. M., C , R. D. 1988. A ray tracing solution for diffuse interreflection. *Computer Graphics (Proc. SIGGRAPH '88) 22*, 4 (Aug.), 85 – 92.

W , D. N., A , D. I., A , K., C , B., D , T., S , D. H., S , W. 2000. Surface light fields for 3D photography. In *Computer Graphics* Proceedings, Annual Conference Series, ACM SIGGRAPH, 287–296.

4.

Utilisation des cartes graphiques programmables

Nous avons observé que plusieurs effets lumineux, comme les ombres et les reflets spéculaires, sont à la fois essentiels pour la qualité *visuelle* de la simulation et coûteux à calculer dans le cadre d'une simulation globale de l'éclairage. D'un autre coté, les cartes graphiques modernes, programmables, sont capables d'effectuer de nombreux calculs pour chaque pixel de l'image affichée. Il devient alors possible de décharger le CPU d'un certain nombres de calculs en les confiant à la carte graphique.

Dans ce chapitre, nous avons présenté nos travaux sur l'utilisation des cartes graphiques dans la simulation des effets lumineux, tels que les ombres douces et les reflets spéculaires. Ces travaux permettent d'augmenter le réalisme visuel de la simulation de l'éclairage, tout en libérant du temps de calcul pour la simulation d'autres effets.

4.1 Introduction

La simulation de l'éclairage demande beaucoup de ressources de calcul, tant pour le processeur que pour la mémoire. La qualité visuelle du résultat dépend beaucoup de certains effets à haute fréquence, tels que les frontières d'ombre ou les reflets spéculaires. Nos expériences nous ont montré que ces mêmes effets sont aussi les plus coûteux en temps de calcul. Sur certaines scènes, le seul calcul des frontières des ombres causées par l'éclairage direct représente 90 % du temps de calcul de l'éclairage global (direct et indirect).

Cette précision est requise surtout pour l'aspect visuel de la scène et non pour la précision numérique des calculs. Les calculs d'éclairage indirect peuvent utiliser une version simplifiée de l'éclairage direct, avec des frontières d'ombre grossièrement modélisées, sans perte de précision. Plusieurs techniques de simulation de l'éclairage, comme la méthode de radiosité hiérarchique ou le *Photon Mapping*, exploitent d'ailleurs cette propriété.

En reprenant notre étude sur les propriétés fréquentielles de la fonction d'éclairage [5], nous voyons que les effets à haute fréquence se produisent principalement :

– en présence de BRDF spéculaires,
– sur les frontières d'ombre,
– lorsque deux objets sont proches l'un de l'autre.

Une caractéristique commune à tous ces cas est qu'ils ne nécessitent pas une information complète sur l'éclairage dans l'ensemble de la scène : chacun d'eux ne fait intervenir qu'un petit nombre d'objets simultanément. Ainsi, pour déterminer une frontière d'ombre on a seulement besoin de connaître les positions de la source lumineuse, des obstacles et du récepteur. Bien que ces phénomènes ne soient pas purement *locaux*, il ne s'agit pas non plus à proprement parler de phénomènes *globaux*. On pourrait parler de phénomènes *semi-locaux*.

À l'inverse, les phénomènes réellement globaux, qui font intervenir l'éclairage sur l'ensemble de la scène, impliquent une BRDF plutôt diffuse, et sont plutôt des phénomènes à basse fréquence, pour lesquels un échantillonnage spatial faible peut être suffisant.

Nous avons d'un côté un ensemble de phénomènes qui sont importants surtout pour leur effet *visuel*, donc qui n'ont besoin d'être calculés que pour l'image affichée et qui ne font intervenir qu'un petit nombre d'éléments de la scène. De l'autre coté, nous avons des cartes graphiques dont les capacités se sont étendues et qui sont capables d'effectuer des programmes puissants, en parallèle, pour chaque pixel de l'écran. Leurs principales limitations (les calculs sont limités à l'image affichée et chaque programme n'a qu'une petite zone mémoire) correspondent en fait ici à nos besoins.

Il est donc naturel de faire porter la partie coûteuse mais visuellement importante des calculs de simulation de l'éclairage sur les cartes graphiques programmables, tout en gardant le processeur central pour les calculs globaux, qui nécessitent un accès à l'information sur l'ensemble de la scène.

Notons qu'il est également possible d'utiliser les cartes graphiques pour accélérer la simulation de l'éclairage global ; dans le cadre de nos travaux, nous l'avions fait pour accélérer les requêtes de visibilité en utilisant un hémicube [9], pour calculer efficacement les frontières d'ombre et de pénombre [13], pour calculer rapidement le pourcentage de visibilité d'un objet avec des *occlusion queries* [13] et pour l'affichage de fonctions d'éclairage non-linéaires [13]. Ces travaux ne seront pas décrits ici.

4.2 Calcul des ombres douces

Les ombres forment un élément essentiel dans la perception d'une scène virtuelle. Elles fournissent des informations sur les positions relatives des objets, sur leur géométrie, sur leurs reliefs. On distingue généralement les ombres dures, causées par une source ponctuelle, et les ombres douces, causées par une source étendue (voir figure 3.2.

Le calcul d'une ombre dure revient, pour chaque pixel, à résoudre un problème simple de visibilité entre deux points. En revanche, le calcul d'une ombre douce impose de connaître, pour chaque pixel, la *proportion* de la source lumineuse qui est visible du point. Il s'agit donc d'un problème de visibilité point-surface, beaucoup plus complexe. Il existe plusieurs méthodes simples pour calculer des ombres dures en temps-réel[1, 2], tandis que le calcul des ombres douces est encore un problème ouvert.

Dans le cadre du projet CYBER, dirigé par Jean-Marc Hasenfratz, avec l'ingénieur de recherche Marc Lapierre, nous avons effectué un étude exhaustive des algorithmes de calcul des ombres douces en temps-réel [7] (voir p. 184). Cette étude a mis en évidence plusieurs limitations des algorithmes existants :

- Pour tous ces algorithmes, le coût en temps de calcul est directement lié à la taille de la zone de pénombre. Or plus la zone de pénombre est large, et plus elle correspond à des phénomènes à basse fréquence, donc moins visibles. On arrive à ce résultat paradoxal que moins l'ombre douce est visible, et plus elle est coûteuse à calculer.
- Pour calculer la pénombre, il est nécessaire de connaître la portion des obstacles qui masque partiellement la source lumineuse. Cette information étant aussi difficile à calculer que l'ombre douce elle-même, les méthodes temps-réel reposent sur une approximation, en général basée sur les obstacles visibles depuis un point situé au centre de la source. On

1. Lance W . « Casting Curved Shadows on Curved Surfaces ». *Computer Graphics (Proc. of SIGGRAPH '78)*, 12(3):270–274, 1978.
2. Franklin C. C . « Shadow Algorithms for Computer Graphics ». *Computer Graphics (Proc. of SIGGRAPH '77)*, 11(2):242–248, 1977.

ne calcule ensuite que l'ombre douce causée par cette partie des obstacles. Cette approximation limite la qualité des ombres douces, particulièrement si la source lumineuse est très étendue par rapport aux obstacles.

– Plusieurs algorithmes sont basés sur une analyse du modèle géométrique des objets. Le coût des calculs est alors proportionnel à la complexité géométrique des obstacles. L'ombre douce est généralement un phénomène à basse fréquence, dont la complexité visuelle est beaucoup moins grande que la complexité géométrique des obstacles. On a ici un surcroît de travail inutile, lié à la complexité des objets.

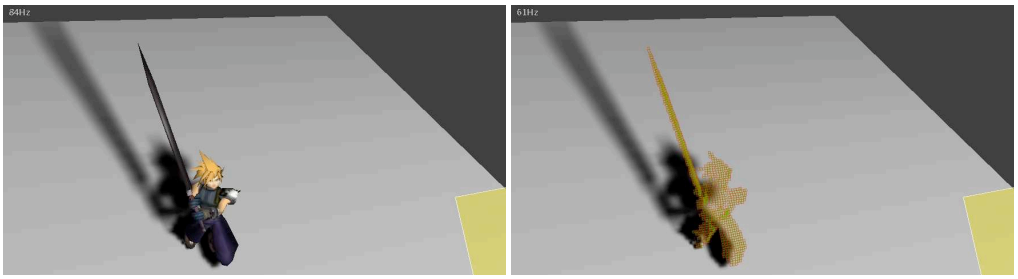


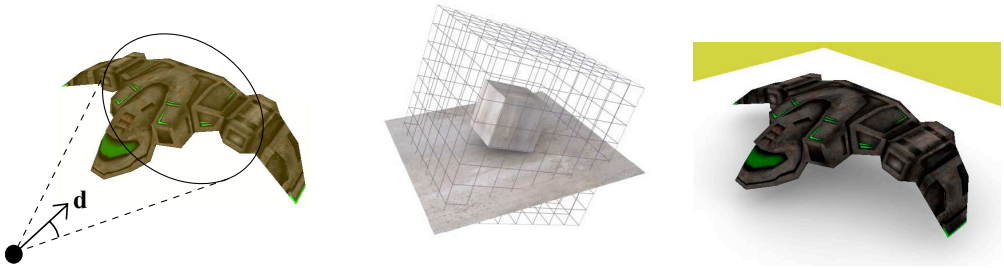
Figure 4.1 – Notre algorithme calcule les ombres douces en temps-réel (à gauche) en remplaçant les obstacles par une version discrétisée (à droite) calculée à partir de la *shadow map*. Ces images sont calculées à 84 Hz.

En se basant sur la connaissance accumulée dans cet état de l'art, ainsi que sur les connaissances issues du projet CYBER, avec l'étudiant en M2R Lionel Atty (co-encadré avec Jean-Marc Hasenfratz), nous avons développé un nouvel algorithme de calcul des ombres douces [2]. Cet algorithme se base sur une discrétisation des obstacles dans une carte de profondeur, et calcule l'ombre douce causée par l'obstacle discrétisé (voir figure 4.1). Bien qu'il ne résolve pas tous les points soulevés par notre étude, cet algorithme a l'avantage d'être très rapide et surtout indépendant de la complexité géométrique des obstacles. De plus, plus la pénombre est large, plus on peut utiliser une discrétisation des obstacles avec un petit nombre de pixels : on arrive ainsi à consacrer moins de temps de calcul aux ombres douces causées par des sources très étendues.

Le point essentiel de notre algorithme est l'utilisation d'une version discrétisée des obstacles pour calculer l'ombre douce. Cette approche est amenée à se multiplier dans les travaux futurs : en remplaçant un modèle géométrique complexe par une version discrète, plus simple et calculée de façon automatique, il serait possible d'accélérer d'autres algorithmes de calcul.

4.3 Précalcul d'occlusion ambiante

L'*occlusion ambiante* est utilisée pour moduler l'éclairage ambiant en fonction des obstacles proches. Elle est définie, soit comme une constante (le pourcentage des directions qui sont bloquées par les obstacles proches), soit comme une fonction plus complexe, par exemple un lobe conique, définissant à la fois le pourcentage d'occlusion et la direction moyenne des obstacles (voir figure 4.2(a)).



(a) L'occlusion ambiante peut être définie comme un cône ($\boldsymbol{d}$, α).

(b) Nous plaçons une grille 3D autour de l'objet. Au centre de chaque cellule, on calcule l'occlusion ambiante.

(c) Lors du rendu, ces valeurs sont utilisées pour l'ombrage des objets voisins. Cette scène est affichée à 800 Hz.

Figure 4.2 – Précalcul d'occlusion ambiante

Pour les scènes animées, des recherches récentes[3, 4] portent sur le stockage d'un champ d'occlusion ambiante, attachée à un objet mobile, et qui influencent les objets voisins. Pour un stockage compact, ces recherches stockent le champ d'occlusion ambiante en le projetant sur un espace de fonctions simples mais adaptées (par exemple des fractions rationnelles de la distance au centre) et en stockant les coefficients dans une carte 2D cubique, indexée par la direction par rapport au centre de l'objet. On a donc un stockage en $O(n^2)$, au prix d'un travail supplémentaire sur les données, à la fois dans le pré-calcul et au moment du rendu.

En collaboration avec Mattias Malmer et Fredrik Malmer (Syndicate Ent. AB) et Ulf Assarsson (Chalmers University of Technology) nous avons montré qu'il était plus rentable de stocker ces données sous forme brute dans une grille 3D, sans pré-traitement [1] (voir p. 224). Au moment du rendu, les données sont affichées directement. Théoriquement, l'inconvénient de cette méthode est que le stockage est en $O(n^3)$. En pratique, l'occlusion ambiante étant un phénomène qui varie très lentement, on utilise de petites valeurs de n. Nous avons trouvé que $n = 32$ convenait pour toutes nos scènes. Pour ces valeurs de n, le coût du stockage brut en $O(n^3)$ est comparable à celui du stockage en $O(n^2)$ (environ 100 Ko par obstacle), à cause du plus grand nombre de coefficients par cellule dans ce dernier.

Outre son intérêt évident sur le plan pratique, ce résultat est également intéressant sur le plan scientifique : pour certains phénomènes, une représentation brute peut être plus intéressante qu'une représentation élaborée. Cet effet est surtout présent sur des phénomènes à basse fréquence, comme l'occlusion ambiante. Il sera intéressant d'étudier l'emploi de tels stockages sous forme de grille 3D pour d'autres phénomènes d'éclairage.

4.4 Calcul des réflexions spéculaires

La réflexion spéculaire sur un objet est un phénomène à haute fréquence, qui participe à l'aspect réaliste de la scène, tout en donnant des informations sur le matériau de l'objet et sur la proximité avec les objets réfléchis. Ces réflexions spéculaires sont généralement simulées en

3. Kun Z , Yaohua H , Steve L , Baining G et Heung-Yeung S . « Precomputed Shadow Fields for Dynamic Scenes ». *ACM Transactions on Graphics (proceedings of Siggraph 2005)*, 24(3), 2005.

4. Janne K et Samuli L . « Ambient Occlusion Fields ». Dans *Symposium on Interactive 3D Graphics and Games*, p. 41–48, 2005.

utilisant une *environment map* de la scène, mais cette technique fait l'hypothèse que la distance entre le réflecteur et l'objet réfléchi est infinie.

(a) Notre algorithme (b) Référence (lancer de rayons) (c) *Environment mapping*

Figure 4.3 – Réflexions spéculaires calculées avec notre algorithme

Avec le doctorant David Roger (co-encadré avec François Sillion), nous avons développé une méthode de calcul des réflexions spéculaires en temps-réel qui fonctionne même lorsqu'il y a contact entre le réflecteur et l'objet réfléchi [3] (voir p. 238). Cette technique calcule (sur la carte graphique) la réflexion des sommets de la scène, puis interpole entre les positions réfléchies des sommets. Le principal avantage de la méthode est sa robustesse, y compris dans des situations difficiles (voir figure 4.3).

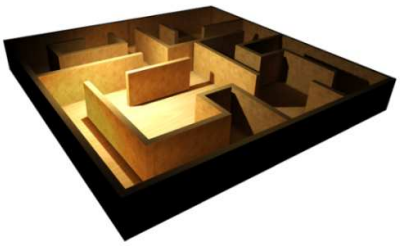
Notre méthode a une complexité linéaire par rapport au nombre d'objets dans la scène et fonctionne en temps réel pour des scènes jusqu'à 20 000 polygones. Le principal inconvénient repose sur l'interpolation linéaire entre les positions réfléchies des sommets, qui est faite par la carte, qui peut produire des erreurs visibles dans la réflexion calculée.

4.5 Éclairage indirect

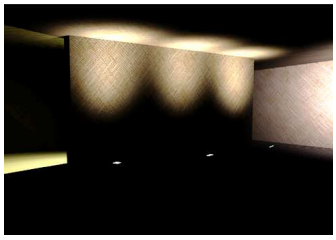
Avec les doctorant Emmanuel Turquin (encadré par François Sillion) et Janne Kontkannen (Helsinki University of Technology), nous nous sommes intéressés à la simulation de l'éclairage indirect en temps réel [12] (voir p. 4.7.6). Notre méthode repose sur trois points importants :

- Un opérateur de transport *global*, qui représente l'éclairage indirect dans l'ensemble de la scène en fonction de l'éclairage direct. Cet opérateur est calculé à partir de l'opérateur de transport *local*, qui représente un rebond de la lumière dans la scène, par multiplications successives.
- Pour pouvoir calculer ces opérateurs de façon efficace, nous les avons exprimé dans une base d'ondelettes spécifique, séparant les dimensions angulaires et spatiales.
- Enfin, nous avons développé une méthode pour calculer directement la projection de l'éclairage direct sur notre base d'ondelettes, afin de pouvoir la donner en entrée à l'opérateur de transport global.

L'essentiel des calculs d'éclairage indirect sont confiés au CPU, tandis que la carte graphique réalise un calcul précis de l'éclairage direct. En combinant les deux résultats, on a accès à l'éclairage global dans la scène de façon interactive (voir figure 4.4).



(a) Scène de test



(b) Éclairage direct

(c) Éclairage indirect calculé par notre algorithme

(d) Éclairage global résultat

Figure 4.4 – Notre algorithme pour le calcul interactif de l'éclairage global. Ces images sont calculées à 15 Hz.

4.6 Discussion

Dans ce chapitre, nous avons présenté nos travaux sur la simulation d'effets lumineux à l'aide des cartes graphiques programmables. Nous avons vu qu'il est possible de simuler certains effets importants pour le réalisme de l'éclairage, comme les ombres douces ou les réflexions spéculaires, à une vitesse compatible avec l'interactivité.

Les cartes graphiques programmables ouvrent de nouvelles directions grâce à leur puissance de calcul, mais cette puissance a ses limites. L'architecture de la carte fait qu'il est plus pratique de l'utiliser pour des calculs *localisés*, ne nécessitant pas un accès global à la scène. Ainsi les cartes graphiques sont bien adaptées pour les calculs d'éclairage en un point, pour les calculs d'ombre (même douce) ou pour les réflexions spéculaires.

De la même manière, nous avons vu que les cartes graphiques pouvaient être utilisée pour les effets lumineux attachés à un objet, et qui ne se portent que sur les objets très proches, comme l'occlusion ambiante.

Ainsi, les phénomènes pour lesquels les cartes graphiques sont les mieux adaptées sont aussi des phénomènes qui sont plutôt à haute fréquence, selon notre analyse fréquentielle du chapitre précédent.

Ces travaux sur l'emploi des cartes graphiques pour la simulation de l'éclairage sont également les travaux les plus prometteurs en termes de collaborations industrielles et de coopérations internationales.

4.7 Articles

4.7.1 Liste des articles

- A survey of real-time soft shadows algorithms (CGF 2003)
- Soft shadow maps: efficient sampling of light source visiblity (CGF 2006)
- Fast Precomputed Ambient Occlusion for Proximity Shadows (JGT 2006)
- Accurate specular reflections in real-time (EG 2006)
- Wavelet radiance transport for interactive indirect lighting (EGSR 2006)

4.7.2 A survey of real-time soft shadows algorithms (CGF 2003)

Auteurs : Jean-Marc H , Marc L , Nicolas H et François X. S .
Journal : *Computer Graphics Forum*, vol. 22, n^o 4 (une version préliminaire a été publiée comme *State-of-the-art-report* à la conférence Eurographics 2003).
Date : décembre 2003.

Volume 22 (2003), Number 4 pp. 753–774

A Survey of Real-time Soft Shadows Algorithms

J.-M. Hasenfratz[†], M. Lapierre[‡], N. Holzschuch[§] and F. Sillion[§]

Artis GRAVIR/IMAG-INRIA[**]

Abstract

Recent advances in GPU technology have produced a shift in focus for real-time rendering applications, whereby improvements in image quality are sought in addition to raw polygon display performance. Rendering effects such as antialiasing, motion blur and shadow casting are becoming commonplace and will likely be considered indispensable in the near future. The last complete and famous survey on shadow algorithms — by Woo et al.[52] *in 1990 — has to be updated in particular in view of recent improvements in graphics hardware, which make new algorithms possible. This paper covers all current methods for real-time shadow rendering, without venturing into slower, high quality techniques based on ray casting or radiosity. Shadows are useful for a variety of reasons: first, they help understand relative object placement in a 3D scene by providing visual cues. Second, they dramatically improve image realism and allow the creation of complex lighting ambiances. Depending on the application, the emphasis is placed on a guaranteed framerate, or on the visual quality of the shadows including penumbra effects or "soft shadows". Obviously no single method can render physically correct soft shadows in real time for any dynamic scene! However our survey aims at providing an exhaustive study allowing a programmer to choose the best compromise for his/her needs. In particular we discuss the advantages, limitations, rendering quality and cost of each algorithm. Recommendations are included based on simple characteristics of the application such as static/moving lights, single or multiple light sources, static/dynamic geometry, geometric complexity, directed or omnidirectional lights, etc. Finally we indicate which methods can efficiently exploit the most recent graphics hardware facilities.*

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism – Color, shading, shadowing, and texture, I.3.1 [Computer Graphics]: Hardware Architecture – Graphics processors, I.3.3 [Computer Graphics]: Picture/Image Generation – Bitmap and framebuffer operations

Keywords: *shadow algorithms, soft shadows, real-time, shadow mapping, shadow volume algorithm.*

1. Introduction

Cast shadows are crucial for the human perception of the 3D world. Probably the first thorough analysis of shadows was Leonardo Da Vinci's[48] (see Figure 1), focusing on paintings and static images. Also of note is the work of Lambert[35] who described the geometry underlying cast shadows (see Figure 1), and more recently the paper from Knill *et al.*[34].

With the emergence of computer graphics technology, researchers have developed experiments to understand the impact of shadows on our perception of a scene. Through different psychophysical experiments they established the important role of shadows in understanding:

- the position and size of the occluder[49, 38, 27, 30, 31];
- the geometry of the occluder[38];
- the geometry of the receiver[38].

Wanger[49] studied the effect of shadow quality on the perception of object relationships, basing his experiments on

† University Pierre Mendès France – Grenoble II

‡ University Joseph Fourier – Grenoble I

§ INRIA

** Artis is a team of the GRAVIR/IMAG laboratory, a joint research unit of CNRS, INPG, INRIA, UJF.

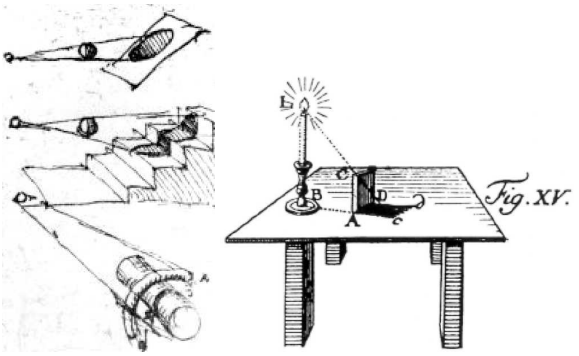


Figure 1: *Left: Study of shadows by Leonardo da Vinci*[48] *— Right: Shadow construction by Lambert*[35].

shadow sharpness. Hubona *et al.*[27] discuss the general role and effectiveness of object shadows in 3D visualization. In their experiments, they put in competition shadows, viewing mode (mono/stereo), number of lights (one/two), and background type (flat plane, "stair-step" plane, room) to measure the impact of shadows.

Kersten *et al.*[30, 31] and Mamassian *et al.*[38] study the relationship between object motion and the perception of relative depth. In fact, they demonstrate that simply adjusting the motion of a shadow is sufficient to induce dramatically different apparent trajectories of the shadow-casting object.

These psychophysical experiments convincingly establish that it is important to take shadows into account to produce images in computer graphics applications. Cast shadows help in our understanding of 3D environments and soft shadows take part in realism of the images.

Since the comprehensive survey of Woo *et al.*[52], progress in computer graphics technology and the development of consumer-grade graphics accelerators have made real-time 3D graphics a reality[3]. However incorporating shadows, and especially realistic soft shadows, in a real-time application, has remained a difficult task (and has generated a great research effort). This paper presents a survey of shadow generation techniques that can create soft shadows in real time. Naturally the very notion of "real-time performance" is difficult to define, suffice it to say that we are concerned with the display of 3D scenes of significant complexity (several tens of thousands of polygons) on consumer-level hardware *ca.* 2003. The paper is organized as follows:

We first review in Section 2 basic notions about shadows: hard and soft shadows, the importance of shadow effects showing problems encountered when working with soft shadows and classical techniques for producing hard shadows in real time. Section 3 then presents existing algorithms for producing soft shadows in real time. Section 4 offers a discussion and classifies these algorithms based on their different abilities and limitations, allowing easier algorithm selection depending on the application's constraints.

2. Basic concepts of hard and soft shadows

2.1. What is a shadow?

Consider a light source L illuminating a scene: *receivers* are objects of the scene that are potentially illuminated by L. A point P of the scene is considered to be in the *umbra* if it can not see any part of L, *i.e.* it does not receive any light directly from the light source.

If P can see a part of the light source, it is in the *penumbra*. The union of the umbra and the penumbra is the shadow, the region of space for which at least one point of the light source is occluded. Objects that hide a point from the light source are called *occluders*.

We distinguish between two types of shadows:

attached shadows, occuring when the normal of the receiver is facing away from the light source;

cast shadows, occuring when a shadow falls on an object whose normal is facing toward the light source.

Self-shadows are a specific case of cast shadows that occur when the shadow of an object is projected onto itself, *i.e.* the occluder and the receiver are the same.

Attached shadows are easy to handle. We shall see later, in Section 4, that some algorithms cannot handle self-shadows.

2.2. Importance of shadow effects

As discussed in the introduction, shadows play an important role in our understanding of 3D geometry:

- Shadows help to **understand relative object position and size** in a scene[49, 38, 27, 30, 31]. For example, without a cast shadow, we are not able to determine the position of an object in space (see Figure 2(a)).
- Shadows can also help us **understanding the geometry of a complex receiver**[38] (see Figure 2(b)).
- Finally, shadows provide useful visual cues that help in **understanding the geometry of a complex occluder**[38] (see Figure 3).

2.3. Hard shadows *vs.* soft shadows

The common-sense notion of shadow is a binary status, *i.e.* a point is either "in shadow" or not. This corresponds to *hard shadows*, as produced by point light sources: indeed, a point light source is either visible or occluded from any receiving point. However, point light sources do not exist in practice and hard shadows give a rather unrealistic feeling to images (see Figure 4(c)). Note that even the sun, probably the most common shadow-creating light source in our daily life, has a significant angular extent and does not create hard shadows. Still, point light sources are easy to model in computer

(a) Shadows provide information about the relative positions of objects. On the left-hand image, we cannot determine the position of the robot, whereas on the other three images we understand that it is more and more distant from the ground.

(b) Shadows provide information about the geometry of the receiver. Left: not enough cues about the ground. Right: shadow reveals ground geometry.

Figure 2: *Shadows play an important role in our understanding of 3D geometry.*

(a) (b) (c)

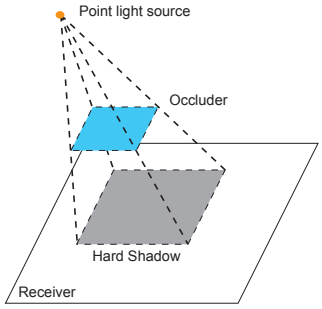
Figure 3: *Shadows provide information about the geometry of the occluder. Here we see that the robot holds nothing in his left hand on Figure 3(a), a ring on Figure 3(b) and a teapot on Figure 3(c).*

graphics and we shall see that several algorithms let us compute hard shadows in real time.

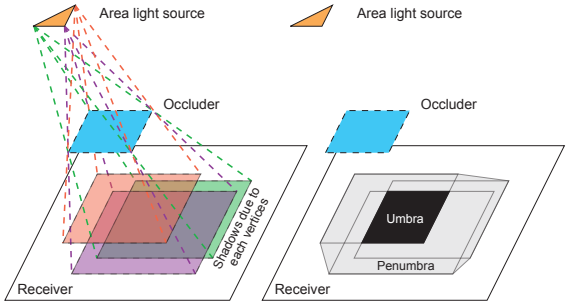
In the more realistic case of a light source with finite extent, a point on the receiver can have a partial view of the light, *i.e.* only a fraction of the light source is visible from that point. We distinguish the *umbra* region (if it exists) in which the light source is totally blocked from the receiver, and the *penumbra* region in which the light source is partially visible. The determination of the umbra and penumbra is a difficult task in general, as it amounts to solving visibility relationships in 3D, a notoriously hard problem. In the case of polygonal objects, the shape of the umbra and penumbra regions is embedded in a discontinuity mesh[13] which can be constructed from the edges and vertices of the light source and the occluders (see Figure 4(b)).

Soft shadows are obviously much more realistic than hard shadows (see Figures 4(c) and 4(d)); in particular the degree of softness (blur) in the shadow varies dramatically with the distances involved between the source, occluder, and receiver. Note also that a hard shadow, with its crisp boundary, could be mistakenly perceived as an object in the scene, while this would hardly happen with a soft shadow.

In computer graphics we can approximate small or distant light source as point sources only when the distance from the light to the occluder is much larger than the distance from the occluder to the receiver, and the resolution of the final image does not allow proper rendering of the penumbra. In all other cases great benefits can be expected from properly representing soft shadows.

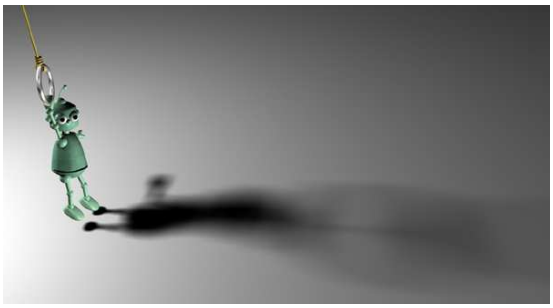


(a) Geometry of hard shadows



(b) Geometry of soft shadows

(c) Illustration of hard shadows



(d) Illustration of soft shadows

Figure 4: *Hard vs. soft shadows.*

2.4. Important issues in computing soft shadows

2.4.1. Composition of multiple shadows

While the creation of a shadow is easily described for a (light source, occluder, receiver) triple, care must be taken to allow for more complex situations.

Shadows from several light sources Shadows produced by multiple light sources are relatively easy to obtain if we know how to deal with a single source (see Figure 5). Due to the linear nature of light transfer we simply sum the contribution of each light (for each wavelength or color band).

Shadows from several objects For point light sources, shadows due to different occluders can be easily combined since the shadow area (where the light source is invisible) is the union of all individual shadows.

With an area light source, combining the shadows of several occluders is more complicated. Recall that the lighting contribution of the light source on the receiver involves a partial visibility function: a major issue is that no simple combination of the partial visibility functions of distinct occluders can yield the partial visibility function of the set of occluders considered together. For instance there may be points in the scene where the light source is not occluded by any object taken separately, but is totally occluded by the set of objects taken together. The correlation between the partial visibility functions of different occluders cannot be predicted easily, but can sometimes be approximated or bounded[45, 5].

As a consequence, the shadow of the union of the objects can be larger than the union of the shadows of the objects (see Figure 6). This effect is quite real, but is not very visible on typical scenes, especially if the objects casting shadows are animated.

2.4.2. Physically exact or fake shadows

Shadows from an extended light source Soft shadows come from spatially extended light sources. To model properly the shadow cast by such light sources, we must take into account all the parts of the occluder that block light coming from the light source. This requires identifying all parts of the object casting shadow that are visible from at least one point of the extended light source, which is algorithmically much more complicated than identifying parts of the occluder that are visible from a single point.

Because this visibility information is much more difficult

Figure 5: *Complex shadow due to multiple light sources. Note the complex interplay of colored lights and shadows in the complementary colors.*

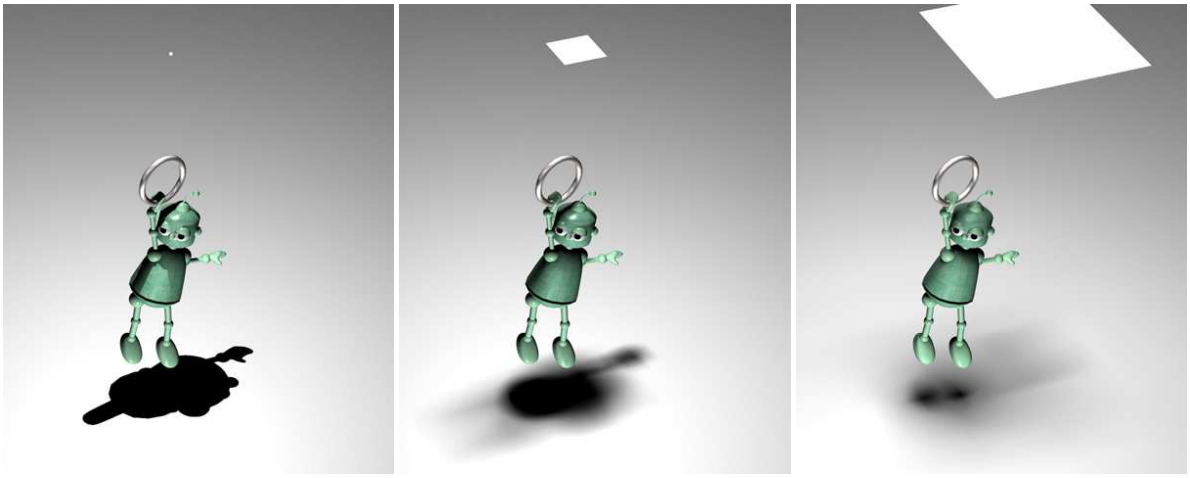


Figure 7: *When the light source is significantly larger than the occluder, the shape of the shadow is very different from the shape computed using a single sample; the sides of the object are playing a part in the shadowing.*

to compute with extended light sources than with point light sources, most real-time soft shadow algorithms compute visibility information from just one point (usually the center of the light source) and then simulate the behavior of the extended light source using this visibility information (computed for a point).

This method produces shadows that are not physically exact, of course, but can be close enough to real shadows for most practical applications. The difference between the approximation and the real shadow is harder to notice if the objects and their shadow are animated — a common occurrence in real-time algorithms.

The difference becomes more noticeable if the difference between the actual extended light source and the point used for the approximation is large, as seen from the object casting shadow. A common example is for a large light source, close enough from the object casting shadow that points of the light source are actually seeing different sides of the object (see Figure 7). In that case, the physically exact shadow is very different from the approximated version.

While large light sources are not frequent in real-time algorithms, the same problem also occurs if the object casting shadow is extended along the axis of the light source, *e.g.* a character with elongated arms whose right arm is pointing toward light source, and whose left arm is close to the receiver.

In such a configuration, if we want to compute a better looking shadow, we can either:

- Use the complete extension of the light source for visibility computations. This is algorithmically too complicated to be used in real-time algorithms.
- Separate the light source into smaller light sources[24, 5]. This removes some of the artefacts, since each light source is treated separately, and is geometrically closer to the

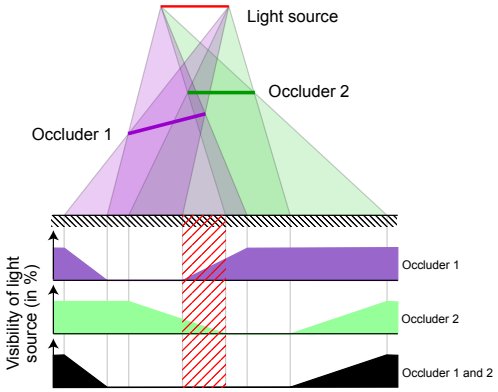


Figure 6: *The shadow of two occluders is not a simple combination of the two individual shadows. Note in particular the highlighted central region which lies in complete shadow (umbra) although the light source is never blocked by a single occluder.*

point sample used to compute the silhouette. The speed of the algorithm is usually divided by the number of light sources.

- Cut the object into slices[45]. We then compute soft shadows separately for each slice, and combine these shadows. By slicing the object, we are removing some of the visibility problems, and we allow lower parts of the object — usually hidden by upper parts — to cast shadow. The speed of the algorithm is divided by the number of slices, and combining the shadows cast by different slices remains a difficult problem.

Approximating the penumbra region When real-time soft shadow algorithms approximate extended light sources using points, they are in fact computing a hard shadow, and extending it to compute a soft shadow.

There are several possible algorithms:

- extend the umbra region outwards, by computing an *outer penumbra* region,
- shrink the umbra region, and complete it with an *inner penumbra* region,
- compute both inner penumbra and outer penumbra.

The first method (outer penumbra only) will always create shadows made of an umbra and a penumbra. Objects will have an umbra, even if the light source is very large with respect to the occluders. This effect is quite noticeable, as it makes the scene appear much darker than anticipated, except for very small light sources.

On the other hand, computing the inner penumbra region can result in light leaks between neighboring objects whose shadows overlap.

Illumination in the umbra region An important question is the illumination in regions that are in the umbra — completely hidden from the light source. There is no light reaching theses regions, so they should appear entirely black, in theory.

However, in practice, some form of ambient lighting is used to avoid completely dark regions and to simulate the fact that light eventually reaches these regions after several reflections.

Real-time shadow methods are usually combined with illumination computations, for instance using the simple OpenGL lighting model. Depending on whether the shadow method operates before or after the illumination phase, ambient lighting will be present or absent. In the latter case the shadow region appears completely dark, an effect that can be noticeable. A solution is to add the ambient shading as a subsequent pass; this extra pass slows down the algorithm, but clever re-use of the Z-buffer on recent graphics hardware make the added cost manageable[40].

Shadows from different objects As shown in Section 2.4.1, in presence of extended light sources, the shadow of the union of several objects is larger than the union of the individual shadows. Furthermore, the boundary of the shadow caused by the combination of several polygonal objects can be a curved line[13].

Since these effects are linked with the fact that the light source is extended, they can not appear in algorithms that use a single point to compute surfaces visible from the light source. All real-time soft shadow algorithms therefore suffer from this approximation.

However, while these effects are both clearly identifiable on still images, they are not as visible in animated scenes. There is currently no way to model these effects with real-time soft shadow algorithms.

2.4.3. Real-time

Our focus in this paper is on real-time applications, therefore we have chosen to ignore all techniques that are based on an expensive pre-process even when they allow later modifications at interactive rates[37]. Given the fast evolution of graphics hardware, it is difficult to draw a hard distinction between real-time and interactive methods, and we consider here that frame rates in excess of 10 fps, for a significant number of polygons, are an absolute requirement for “real-time” applications. Note that stereo viewing usually require double this performance.

For real-time applications, the display refresh rate is often the crucial limiting factor, and must be kept high enough (if not constant) through time. An important feature to be considered in shadowing algorithms is therefore their ability to guarantee a sustained level of performance. This is of course

impossible to do for arbitrary scenes, and a more important property for these algorithms is the ability to parametrically vary the level of performance (typically at the price of greater approximation), which allows an adaptation to the scene's complexity.

2.4.4. Shadows of special objects

Most shadowing algorithms make use of an explicit representation of the object's shapes, either to compute silhouettes of occluders, or to create images and shadow maps. Very complex and volumetric objects such as clouds, hair, grass etc. typically require special treatment.

2.4.5. Constraints on the scene

Shadowing algorithms may place particular constraints on the scene. Examples include the type of object model (techniques that compute a shadow as a texture map typically require a parametric object, if not a polygon), or the necessity/possibility to identify a subset of the scene as occluders or shadow receivers. This latter property is important in adapting the performance of the algorithm to sustain real-time.

2.5. Basic techniques for real-time shadows

In this State of the Art Review, we focus solely on real-time soft shadows algorithms. As a consequence, we will not describe other methods for producing soft shadows, such as radiosity, ray-tracing, Monte-Carlo ray-tracing or photon mapping.

We now describe the two basic techniques for computing shadows from *point light sources*, namely *shadow mapping* and the *shadow volume algorithm*.

2.5.1. Shadow mapping

Method The basic operation for computing shadows is identifying the parts of the scene that are hidden from the light source. Intrisically, it is equivalent to visible surface determination, from the point-of-view of the light source.

The first method to compute shadows[17, 44, 50] starts by computing a view of the scene, from the point-of-view of the light source. We store the z values of this image. This Z-buffer is the *shadow map* (see Figure 8).

The shadow map is then used to render the scene (from the normal point-of-view) in a two pass rendering process:

- a standard Z-buffer technique, for hidden-surface removal.
- for each pixel of the scene, we now have the geometrical position of the object seen in this pixel. If the distance between this object and the light is greater than the distance stored in the shadow map, the object is in shadow. Otherwise, it is illuminated.

Figure 8: *Shadow map for a point light source. Left: view from the camera. Right: depth buffer computed from the light source.*

- The color of the objects is modulated depending on whether they are in shadow or not.

Shadow mapping is implemented in current graphics hardware. It uses an OpenGL extension for the comparison between Z values, `GL_ARB_SHADOW`[†].

Improvements The depth buffer is sampled at a limited precision. If surfaces are too close from each other, sampling problems can occur, with surfaces shadowing themselves. A possible solution[42] is to offset the Z values in the shadow map by a small bias[51].

If the light source has a cut-off angle that is too large, it is not possible to project the scene in a single shadow map without excessive distortion. In that case, we have to replace the light source by a combination of light sources, and use several depth maps, thus slowing down the algorithm.

Shadow mapping can result in large aliasing problems if the light source is far away from the viewer. In that case, individual pixels from the shadow map are visible, resulting in a staircase effect along the shadow boundary. Several methods have been implemented to solve this problem:

- Storing the ID of objects in the shadow map along with their depth[26].
- Using deep shadow maps, storing coverage information for all depths for each pixel[36].
- Using multi-resolution, adaptative shadow maps[18], computing more details in regions with shadow boundaries that are close to the eye.
- Computing the shadow map in perspective space[46], effectively storing more details in parts of the shadow map that are closer to the eye.

The last two methods are directly compatible with existing OpenGL extensions, and therefore require only a small amount of coding to work with modern graphics hardware.

An interesting alternative version of this algorithm is to

† This extension (or the earlier version, `GL_SGIX_SHADOW`, is available on Silicon Graphics Hardware above Infinite Reality 2, on NVidia graphics cards after GeForce3 and on ATI graphics cards after Radeon9500.

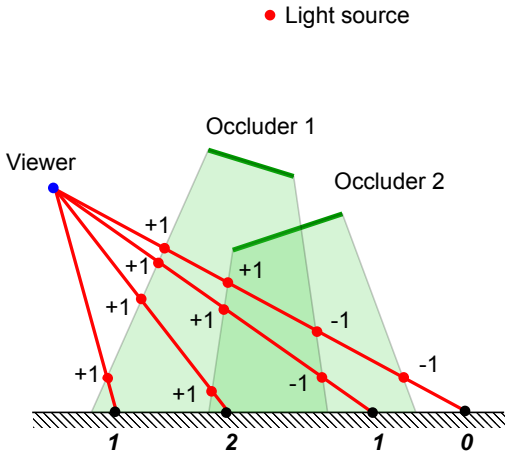


Figure 9: *Shadow volume.*

warp the shadow map into camera space[55] rather than the usual opposite: it has the advantage that we obtain a modulation image that can be mixed with a texture, or blurred to produce antialiased shadows.

Discussion Shadow mapping has many advantages:

- it can be implemented entirely using graphics hardware;
- creating the shadow map is relatively fast, although it still depends on the number and complexity of the occluders;
- it handles self-shadowing.

It also has several drawbacks:

- it is subject to many sampling and aliasing problems;
- it cannot handle omni-directional light sources;
- at least two rendering passes are required (one from the light source and one from the viewpoint);

2.5.2. The Shadow Volume Algorithm

Another way to think about shadow generation is purely geometrical. This method was first described by Crow[12], and first implemented using graphics hardware by Heidmann[23].

Method The algorithm consists in finding the silhouette of occluders along the light direction, then extruding this silhouette along the light direction, thus forming a *shadow volume*. Objects that are inside the shadow volume are in shadow, and objects that are outside are illuminated.

The shadow volume is calculated in two steps:

- the first step consists in finding the silhouette of the occluder as viewed from the light source. The simplest method is to keep edges that are shared by a triangle facing the light and another in the opposite direction. This actually gives a superset of the true silhouette, but it is sufficient for the algorithm.
- then we construct the shadow volume by extruding these edges along the direction of the point light source. For each edge of the silhouette, we build the half-plane subtended by the plane defined by the edge and the light source. All these half-planes define the shadow volume, and knowing if a point is in shadow is then a matter of knowing if it is inside or outside the volume.
- for each pixel in the image rendered, we count the number of faces of the shadow volume that we are crossing between the view point and the object rendered. Front-facing faces of the shadow volume (with respect to the view point) increment the count, back-facing faces decrement the count (see Figure 9). If the total number of faces is positive, then we are inside the shadow volume, and the pixel is rendered using only ambient lighting.

The rendering pass is easily done in hardware using a stencil buffer[23, 32, 15]; faces of the shadow volume are rendered in the stencil buffer with depth test enabled this way: in a first pass, front faces of the shadow volumes are rendered incrementing the stencil buffer; in a second pass, back faces are rendered, decrementing it. Pixels that are in shadow are "captured" between front and back faces of the shadow volume, and have a positive value in the stencil buffer. This way to render volumes is called *zpass*.

Therefore the complete algorithm to obtain a picture using the Shadow Volume method is:

- render the scene with only ambient/emissive lighting;
- calculate and render shadow volumes in the stencil buffer;
- render the scene illuminated with stencil test enabled: only pixels which stencil value is 0 are rendered, others are not updated, keeping their ambient color.

Improvements The cost of the algorithm is directly linked to the number of edges in the shadow volume. Batagelo and Júnior[7] minimize the number of volumes rendered by precalculating in software a modified BSP tree. McCool[39] extracts the silhouette by first computing a shadow map, then extracting the discontinuities of the shadow map, but this method requires reading back the depth buffer from the graphics board to the CPU, which is costly. Brabec and Seidel[10] reports a method to compute the silhouette of the occluders using programmable graphics hardware[14], thus obtaining an almost completely hardware-based implementation of the shadow volume algorithm (he still has to read back a buffer into the CPU for parameter transfer).

Roettger *et al.*[43] suggests an implementation that doesn't require the stencil buffer; he draws the shadow volume in the alpha buffer, replacing increment/decrement with a multiply/divide by 2 operation.

Everitt and Kilgard[15] have described a robust implementation of the shadow volume algorithm. Their method includes capping the shadow volume, setting $w = 0$ for extruded vertices (effectively making infinitely long quads) and setting the far plane at an infinite distance (they prove that this step

only decreases Z-buffer precision by a few percents). Finally, they render the shadow volume using the *zfail* technique; it works by rendering the shadow volume *backwards*:

- we render the scene, storing the Z-buffer;
- in the first pass, we increment the stencil buffer for all back-facing faces, but only if the face is behind an existing object of the scene;
- in the second pass, we decrement the stencil buffer for all front-facing faces, but only if the face is behind an existing object;
- The stencil buffer contains the intersection of the shadow volume and the objects of the scene.

The *zfail* technique was discovered independently by Bilodeau and Songy and by Carmack.

Recent extensions to OpenGL[15, 16, 21] allow the use of shadow volumes using stencil buffer in a single pass, instead of the two passes required so far. They also[15] provide *depth-clamping*, a method in which polygon are not clipped at the near and far distance, but their vertices are projected onto the near and far plane. This provides in effect an infinite view pyramid, making the shadow volume algorithm more robust.

The main problem with the shadow volume algorithm is that it requires drawing large polygons, the faces of the shadow volume. The fillrate of the graphics card is often the bottleneck. Everitt and Kilgard[15, 16] list different solutions to reduce the fillrate, either using software methods or using the graphics hardware, such as scissoring, constraining the shadow volume to a particular fragment.

Discussion The shadow volume algorithm has many advantages:

- it works for omnidirectional light sources;
- it renders eye-view pixel precision shadows;
- it handles self-shadowing.

It also has several drawbacks:

- the computation time depends on the complexity of the occluders;
- it requires the computation of the silhouette of the occluders as a preliminary step;
- at least two rendering passes are required;
- rendering the shadow volume consumes fillrate of the graphics card.

3. Soft shadow algorithms

In this section, we review algorithms that produce soft shadows, either interactively or in real time. As in the previous section, we distinguish two types of algorithms:

- Algorithms that are based on an image-based approach, and build upon the shadow map method described in Section 2.5.1. These algorithms are described in Section 3.1.
- Algorithms that are based on an object-based approach, and build upon the shadow volume method described in Section 2.5.2. These algorithms are described in Section 3.2.

3.1. Image-Based Approaches

In this section, we present soft shadow algorithms based on shadow maps (see Section 2.5.1). There are several methods to compute soft shadows using image-based techniques:

1. Combining several shadow textures taken from point samples on the extended light source[25, 22].
2. Using layered attenuation maps[1], replacing the shadow map with a Layered Depth Image, storing depth information about all objects visible from at least one point of the light source.
3. Using several shadow maps[24, 54], taken from point samples on the light source, and an algorithm to compute the percentage of the light source that is visible.
4. Using a standard shadow map, combined with image analysis techniques to compute soft shadows[9].
5. Convolving a standard shadow map with an image of the light source[45].

The first two methods approximate the light source as a combination of several point samples. As a consequence, the time for computing the shadow textures is multiplied by the number of samples, resulting in significantly slower rendering. On the other hand, these methods actually compute more information than other soft shadow methods, and thus compute more physically accurate shadows. Most of the artefacts listed in Section 2.4.2 will not appear with these two methods.

3.1.1. Combination of several point-based shadow images[25, 22]

The simplest method[22, 25] to compute soft shadows using image based methods is to place sample points regularly on the extended light source. These sample points are used to compute binary occlusion maps, which are combined into an attenuation map, used to modulate the illumination (calculated separately).

Method Herf[25] makes the following assumptions on the geometry of the scene:

- a light source of uniform color,
- subtending a small solid angle with respect to the receiver,
- and with distance from the receiver having small variance.

With these three assumptions, contributions from all sample points placed on the light source will be roughly equal.

The user identifies in advance the object casting shadows, and the objects onto which we are casting shadow. For each object receiving shadow, we are going to compute a texture containing the soft shadow.

Figure 10: *Combining several occlusion maps to compute soft shadows. Left: the occlusion map computed for a single sample. Center: the attenuation map computed using 4 samples. Right: the attenuation map computed using 64 samples.*

Figure 11: *With only a small number of samples on the light source, artefacts are visible. Left: soft shadow computed using 4 samples. Right: soft shadow computed using 1024 samples.*

We start by computing a binary occlusion map for each sample point on the light source. For each sample point on the light source, we render the scene into an auxiliary buffer, using 0 for the receiver, and 1 for any other polygon. These binary occlusion maps are then combined into an attenuation map, where each pixel stores the number of sample points on the light source that are occluded. This attenuation map contains a precise representation of the soft shadow (see Figures 10 and 11).

In the rendering pass, this soft shadow texture is combined with standard textures and illumination, in a standard graphics pipeline.

Discussion The biggest problem for Herf[25] method is rendering the attenuation maps. This requires $N_p N_s$ rendering passes, where N_p is the number of objects receiving shadows, and N_s is the number of samples on the light source. Each pass takes a time proportionnal to the number of polygons in the objects casting shadows. In practice, to make this method run in real time, we have to limit the number of receivers to a single planar receiver.

To speed-up computation of the attenuation map, we can lower the number of polygons in the occluders. We can also lower the number of samples (n) to increase the framerate, but this is done at the expense of image quality, as the attenuation map contains only $n-1$ gray levels. With fewer than 9 samples (3×3), the user sees several hard shadows, instead of a single soft shadow (see Figure 11).

Herf's method is easy to parallelize, since all occlusion maps can be computed separately, and only one computer is needed to combine them. Isard *et al.*[28] reports that a parallel implementation of this algorithm on a 9-node Sepia-2a parallel calculator with high-end graphics cards runs at more than 100 fps for moderately complex scenes.

3.1.2. Layered Attenuation Maps[1]

The Layered Attenuation Maps[1] method is based on a modified layered depth image[29]. It is an extension of the previous method, where we compute a layered attenuation map for the entire scene, instead of a specific shadow map for each object receiving shadow.

Method It starts like the previous method: we place sample points on the area light source, and we use these sample points to compute a modified attenuation map:

- For each sample point, we compute a view of the scene, along the direction of the normal to the light source.
- Theses images are all warped to a central reference, the center of the light source.
- For each pixel of these images:
  - In each view of the scene, we have computed the distance to the light source in the Z-buffer.
  - We can therefore identify the object that is closest to the light source.
  - This object makes the first layer of the layered attenuation map.
  - We count the number of samples seeing this object, which gives us the percentage of occlusion for this object.
  - If other objects are visible for this pixel but further away from the light they make the subsequent layers.
  - For each layer, we store the distance to the light source and the percentage of occlusion.

The computed Layered Attenuation Map contains, for all the objects that are visible from at least one sample point, the distance to the light source and the percentage of sample points seeing this object.

At rendering time, the Layered Attenuation Map is used like a standard attenuation map, with the difference that all the objects visible from the light source are stored in the map:

- First we render the scene, using standard illumination and textures. This first pass eliminates all objects invisible from the viewer.
- Then, for each pixel of the image, we find whether the corresponding point in the scene is in the Layered Attenuation Map or not. If it is, then we modulate the lighting

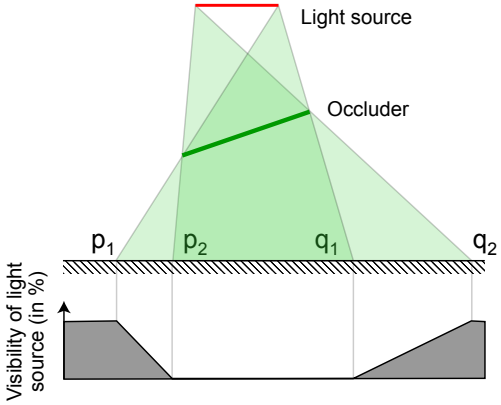


Figure 12: *Percentage of a linear light source that is visible.*

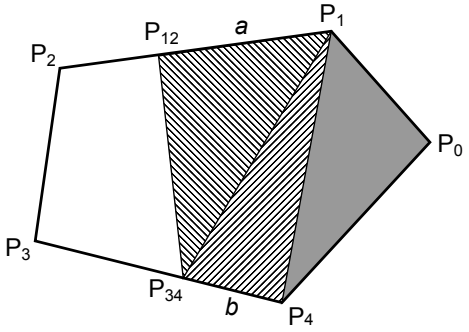


Figure 13: *Using the visibility channel to compute visibility from a polygonal light source. The shadow maps tell us that vertices P_0, P_1 and P_4 are occluded and that vertices P_2 and P_3 are visible. The visibility channel for edge $[P_1P_2]$ tells us that this edge is occluded for a fraction a; similarly, the visibility channel for edge $[P_3P_4]$ tells us that this edge is occluded for a fraction b. The portion of the light that is occluded is the hatched region, whose area can be computed geometrically using a and b.*

value found by the percentage of occlusion stored in the map. If it isn't, then the point is completely hidden from the light source.

Discussion The main advantage of this method, compared to the previous method, is that a single image is used to store the shadowing information for the entire scene, compared to one shadow texture for each shadowed object. Also, we do not have to identify beforehand the objects casting shadows.

The extended memory cost of the Layered Attenuation Map is reasonable: experiments by the authors show that on average, about 4 layers are used in moderately complex scenes.

As with the previous method, the speed and realism are related to the number of samples used on the light source. We are rendering the entire scene N_s times, which precludes real-time rendering for complex scenes.

3.1.3. Quantitative Information in the Shadow Map[24]

Heidrich *et al.*[24] introduced another extension of the shadow map method, where we compute not only a shadow map, but also a visibility channel(see Figure 12), which encodes the percentage of the light source that is visible. Heidrich *et al.*[24]'s method only works for linear light sources, but it was later extended to polygonal area light sources by Ying *et al.*[54].

Method We start by rendering a standard shadow map for each sample point on the linear light source. The number of sample points is very low, usually they are equal to the two end vertices of the linear light source.

In each shadow map, we detect discontinuities using image analysis techniques. Discontinuities in the shadow map happen at shadow boundaries. They are separating an object casting shadow from the object receiving shadow. For each discontinuity, we form a polygon linking the frontmost object (casting shadow) to the back object (receiving shadow). These polygons are then rendered in the point of view of the other sample, using Gouraud shading, with value 0 on the closer points, and 1 on the farthest points.

This gives us a visibility channel, which actually encodes the percentage of the edge linking the two samples that is visible.

The visibility channel is then used in a shadow mapping algorithm. For each pixel in the rendered image, we first check its position in the shadow map for each sample.

- if it is in shadow for all sample points, we assume that it is in shadow, and therefore it is rendered black.
- if it is visible from all sample points, we assume that it is visible, and therefore rendered using standard OpenGL illumination model.
- if it is hidden for some sample point, and visible from another point, we use the visibility channel to modulate the light received by the pixel.

Ying *et al.*[54] extended this algorithm to polygonal area light sources: we generate a shadow map for each vertex of the polygonal light source, and a visibility channel for each edge. We then use this information to compute the percentage of the polygonal light source that is visible from the current pixel.

For each vertex of the light source, we query the shadow map of this vertex. This gives us a boolean information, whether this vertex is occluded or not from the point of view of the object corresponding to the current pixel. If an edge

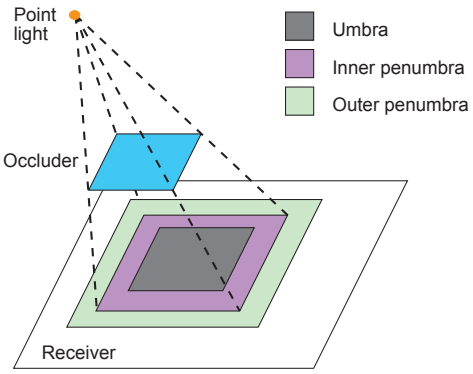


Figure 14: *Extending the shadow of a point light source: for each occluder identified in the shadow map, we compute a penumbra, based on the distance between this occluder and the receiver.*

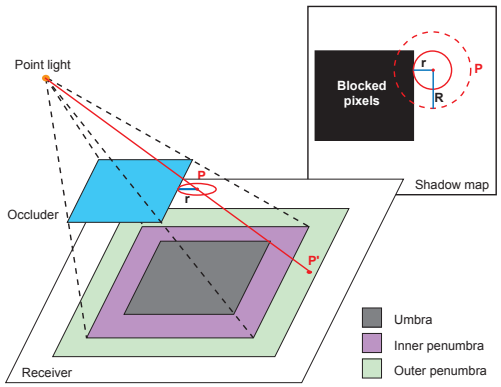


Figure 15: *Extending the shadow of a single sample: For each pixel in the image, we find the corresponding pixel P in the shadow map. Then we find the nearest blocked pixel. P is assumed to be in the penumbra of this blocker, and we compute an attenuation coefficient based on the relative distances betwen light source, occluder and P.*

links an occluded vertex to an non-occluded one, the visibility channel for this edge gives us the percentage of the edge that is occluded (see Figure 13). Computing the visible area of the light source is then a simple 2D problem. This area can be expressed as a linear combination of the area of triangles on the light source. By precomputing the area of these triangles, we are left with a few multiplications and additions to perform at each pixel.

Discussion The strongest point of this algorithm is that it requires a small number of sampling points. Although it can work with just the vertices of the light source used as sampling points, a low number of samples can result in artefacts in moderately complex scenes. These artefacts are avoided by adding a few more samples on the light source.

This method creates fake shadows, but nicely approximated. The shadows are exact when only one edge of the occluder is intersecting the light source, and approximate if there is more than one edge, for example at the intersection of the shadows of two different occluders, or when an occluder blocks part of the light source without blocking any vertex.

The interactivity of the algorithm depends on the time it takes to generate the visibility channels, which itself depends on the complexity of the shadow. On simples scenes (a few occluders) the authors report computation times of 2 to 3 frames per second.

The algorithm requires having a polygonal light source, and organising the samples, so that samples are linked by edges, and for each edge, we know the sample points it links.

3.1.4. Single Sample Soft Shadows[9, 33]

A different image-based method to generate soft shadows was introduced by Parker *et al.*[41] for parallel ray-tracing and later modified to use graphics hardware by Brabec and Seidel[9].

This method is very similar to standard shadow mapping. It starts by computing a standard shadow map, then uses the depth information available in the depth map to extend the shadow region and create a penumbra. In this method, we distinguish between the inner penumbra (the part of the penumbra that is inside the shadow of the point sample) and the outer penumbra (the part of the umbra that is outside the shadow of the point sample, see Figure 14). Parker *et al.*[41] compute only the outer penumbra; Brabec and Seidel[9] compute both the inner and the outer penumbra; Kirsch and Doellner[33] compute only the inner penumbra. In all cases, the penumbra computed goes from 0 to 1, to ensure continuity with areas in shadow and areas that are fully illuminated.

Method In a first pass, we create a single standard shadow map, for a single sample — usually at the center of the light source.

During rendering, as with standard shadow mapping, we identify the position of the current pixel in the shadow map. Then:

- if the current pixel is in shadow, we identify the nearest pixel in the shadow map that is illuminated.
- if the pixel is lit, we identify the nearest pixel in the shadow map that corresponds to an object that is closer to the light source than the current pixel (see Figure 15).

In both cases, we assume that the object found is casting a shadow on the receiver, and that the point we have found is in the penumbra. We then compute an attenuation coefficient based on the relative positions of the receiver, the occluder

and the light source:

$$f = \frac{\text{dist}(\text{Pixel}_{\text{Occluder}}, \text{Pixel}_{\text{Receiver}})}{RSz_{\text{Receiver}}|z_{\text{Receiver}} - z_{\text{Occluder}}|}$$

where R and S are user-defineable parameters. The intensity of the pixel is modulated using[8]:

- $0.5 * (1 + f)$, clamped to $[0.5, 1]$ if the pixel is outside the shadow,
- $0.5 * (1 - f)$, clamped to $[0, 0.5]$ if the pixel is inside the shadow.

For pixels that are far away from the boundary of the shadow, either deep inside the shadow or deep inside the fully lit area, f gets greater than 1, resulting in a modulation coefficient of respectively 0 or 1. On the original shadow boundary, $f = 0$, the two curves meet each other continuously with a modulation coefficient of 0.5. The actual width of the penumbra region depends on the ratio of the distances to the light source of the occluder and the receiver, which is perceptually correct.

The slowest phase of this algorithm is the search of neighbouring pixels in the shadow map, to find the potential occluder. In theory, an object can cast a penumbra than spans the entire scene, if it is close enough to the light source. In practice, we limit the search to a maximal distance to the current pixel of $R_{max} = Rz_{\text{Receiver}}$.

To ensure that an object is correctly identified as being in shadow or illuminated, the information from the depth map is combined with an item buffer, following Hourcade and Nicolas[26].

Discussion The aim of this algorithm is to produce perceptually pleasing, rather than physically exact, soft shadows. The width of the penumbra region depends on the ratio of the respective distances to the light source of the occluder and the receiver. The penumbra region is larger if the occluder is far from the receiver, and smaller if the occluder is close to the receiver.

Of course, the algorithm suffers from several shortcomings. Since the shadow is only determined by a single sample shadow map, it can fail to identify the proper shadowing edge. It works better if the light source is far away from the occluder. The middle of the penumbra region is placed on the boundary of the shadow from the single sample, which is not physically correct.

The strongest point of this algorithm is its speed. Since it only needs to compute a single shadow map, it can achieve framerates of 5 to 20 frames per second, compared with 2 to 3 frames per second for multi-samples image-based methods. The key parameter in this algorithm is R, the search radius. For smaller search values of R, the algorithms works faster, but can miss large penumbras. For larger values of R, the algorithm can identify larger penumbras, but takes longer for each rendering.

A faster version of this algorithm, by Kirsch and Doellner[33], computes both the shadow map and a shadow-width map: for each point in shadow, we precompute the distance to the nearest point that is illuminated. For each pixel, we do a look-up in the shadow map and the shadow-width map. If the point is occluded, we have the depth of the current point (z), the depth of the occluder (z_{occluder}) and the shadow width (w). A 2D function gives us the modulation coefficient:

$$I(z, w) = \begin{cases} 1 & \text{if} \quad z = z_{\text{occluder}} \\ 1 + c_{bias} - c_{scale}\frac{w}{z_{\text{occluder}} - z} & \text{otherwise} \end{cases}$$

The shadow-width map is generated from a binary occlusion map, transformed into the width map by repeated applications of a smoothing filter. This repeated filtering is done using graphics hardware, during rendering. Performances depend mostly on the size of the occlusion map and on the size of the filter; for a shadow map resolution of 512×512 pixels, and a large filter, they attain 20 frames per second. Performance depends linearly on the number of pixels in the occlusion map, thus doubling the size of the occlusion map divides the rendering speed by 4.

3.1.5. Convolution technique[45]

As noted earlier, soft shadows are a consequence of partial visibility of an extended light source. Therefore the calculation and soft shadows is closely related to the calculation of the visible portion of the light source.

Soler and Sillion[45] observe that the percentage of the source area visible from a receiving point can be expressed as a simple convolution for a particular configuration. When the light source, occluder, and receiver all lie in parallel planes, the soft shadow image on the receiver is obtained by convolving an image of the receiver and an image of the light source. While this observation is only mathematically valid in this very restrictive configuration, the authors describe how the same principle can be applied to more general configurations:

First, appropriate imaging geometries are found, even when the objects are non-planar and/or not parallel. More importantly, the authors also describe an error-driven algorithm in which the set of occluders is recursively subdivided according to an appropriate error estimate, and the shadows created by the subsets of occluders are combined to yield the final soft shadow image.

Discussion The convolution technique's main advantages are the visual quality of the soft shadows (not their physical fidelity), and the fact that it operates from images of the source and occluders, therefore once the images are obtained the complexity of the operations is entirely under control. Sampling is implicitly performed when creating a light source image, and the combination of samples is handled

by the convolution operation, allowing very complex light source shapes.

The main limitation of the technique is that the soft shadow is only correct in a restricted configuration, and the proposed subdivision mechanism can only improve the quality when the occluder can be broken down into smaller parts. Therefore the case of elongated polygons in th direction of the light source remains problematic. Furthermore, the subdivision mechanism, when it is effective in terms of quality, involves a significant performance drop.

3.2. Object-Based Approaches

Several methods can be used to compute soft shadows in animated scenes using object-based methods:

1. Combining together several shadow volumes taken from point samples on the light source, in a manner similar to the method described for shadow maps in Section 3.1.1.
2. extending the shadow volume[19, 53, 11] using a specific heuristic (Plateaus[19], Penumbra Maps[53], Smoothies[11]).
3. computing a penumbra volume for each edge of the shadow silhouette[2, 4, 5].

3.2.1. Combining several hard shadows

Method The simplest way to produce soft shadows with the shadow volume algorithm is to take several samples on the light source, compute a hard shadow for each sample and average the pictures produced. It simulates an area light source, and gives us the soft shadow effect.

However, the main problem with this method, as with the equivalent method for shadow maps, is the number of samples it requires to produce a good-looking soft shadow, which precludes any real-time application. Also, it requires the use of an accumulation buffer, which is currently not supported on standard graphics hardware.

An interesting variation has been proposed by Vignaud[47], in which shadow volumes from a light source whose position changes with time are added in the alpha buffer, mixed with older shadow volumes, producing a soft shadow after a few frames where the viewer position does not change.

3.2.2. Soft Planar Shadows Using Plateaus

The first geometric approach to generate soft shadows has been implemented by Haines[19]. It assumes a planar receiver, and generates an attenuation map that represents the soft shadow. The attenuation map is created by converting the edges of the occluders into volumes, and is then applied to the receiver as a modulating texture.

Method The principle of the plateaus method[19] is to generate an attenuation map, representing the soft shadow. The attenuation map is first created using the shadow volume method, thus filling in black the parts of the map that are occluded.

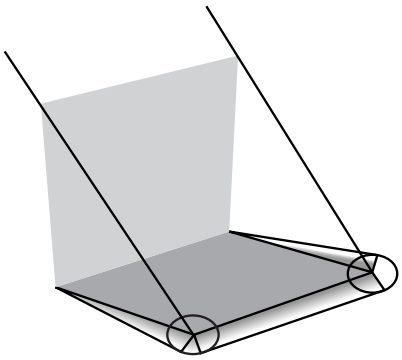


Figure 16: *Extending the shadow volume of an occluder with cones and planes.*

Then, the edges of the silhouette of the objects are transformed into volumes (see Figure 16):

- All the vertices of the silhouette are first turned into cones, with the radius of the cone depending on the distance between the occluder vertex and the ground, thus simulating a spherical light source.
- then edges joining adjacent vertices are turned into surfaces. For continuity, the surface joining two cones is an hyperboloid, unless the two cones have the same radius (that is, if the two original vertices are at the same distance of the ground), in which case the hyperboloid degenerates to a plane.

These shadow volumes are then projected on the receiver and colored using textures: the axis of the cone is black, and the contour is white. This texture is superimposed with the shadow volume texture: Haines' algorithm only computes the outer penumbra.

One important parameter in the algorithm is the way we color the penumbra volume; it can be done using Gouraud shading, values from the Z-buffer or using a 1D texture. The latter gives more control over the algorithm, and allows penumbra to decrease using any function, including sinusoid.

Discussion The first limitation of this method is that it is limited to shadows on planar surfaces. It also assumes a spherical light source. The size of the penumbra only depends on the distance from the receiver to the occluders, not from the distance between the light source and the occluders. Finally, it suffers from the same fillrate bottleneck as the original shadow volume algorithm.

A significant improvement is Wyman and Hansen[53]'s Penumbra Map method: the interpolation step is done using programmable graphics hardware[6, 20, 14], generating a

penumbra map that is applied on the model, along with a shadow map. Using a shadow map to generate the umbra region removes the fill-rate bottleneck and makes the method very robust. Wyman and Hansen report framerate of 10 to 15 frames per second on scenes with more than 10,000 shadow-casting polygons.

The main limitation in both methods[19, 53] is that they only compute the outer penumbra. As a consequence, objects will always have an umbra, even if the light source is very large with respect to the occluders. This effect is clearly noticeable, as it makes the scene appear much darker than anticipated, except for very small light sources.

3.2.3. Smoothies[11]

Chan and Durand[11] present a variation of the shadow volume method that uses only graphics hardware for shadow generation.

Method We start by computing the silhouette of the object. This silhouette is then extended using "smoothies", that are planar surfaces connected to the edges of the occluder and perpendicular to the surface of the occluder.

We also compute a shadow map, which will be used for depth queries. The smoothies are then textured taking into account the distance of each silhouette vertex to the light source, and the distance between the light source and the receiver.

In the rendering step, first we compute the hard shadow using the shadow map, then the texture from the smoothies is projected onto the objects of the scene to create the penumbra.

Discussion As with Haines[19], Wyman and Hansen[53] and Parker[41], this algorithm only computes the outer penumbra. As a consequence, occluders will always project an umbra, even if the light source is very large with respect to the occluders. As mentionned earlier, this makes the scene appear much darker than anticipated, an effect that is clearly noticeable except for very small light sources.

The size of the penumbra depends on the ratio of the distances between the occluder and the light source, and between receiver and light source, which is perceptually correct.

Connection between adjacent edges is still a problem with this algorithm, and artefacts appear clearly except for small light sources.

The shadow region is produced using the shadow map method, which removes the problem with the fill rate bottleneck experienced with all other methods based on the shadow volume algorithm. As with the previous method[53], the strong point of this algorithm is its robustness: the authors have achieved 20 frames per second on scenes with more than 50,000 polygons.

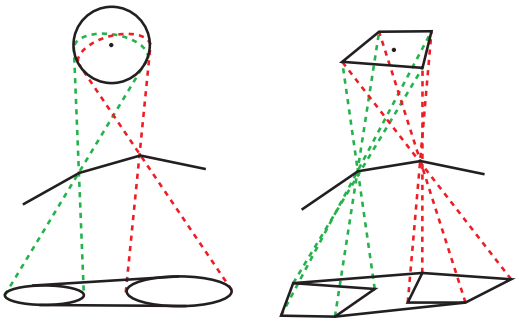


Figure 17: *Computing the penumbra wedge of a silhouette edge: the wedge is a volume based on the silhouette edge and encloses the light source.*

3.2.4. Soft Shadow Volumes[2, 4, 5]

Akenine-Möller and Assarsson[2], Assarsson and Akenine-Möller[4] and Assarsson *et al.*[5] have developed an algorithm to compute soft shadows that builds on the shadow volume method and uses the programmable capability of modern graphics hardware[6, 20, 14] to produce real-time soft shadows.

Method The algorithm starts by computing the silhouette of the object, as seen from a single sample on the light source. For each silhouette edge, we build a *silhouette wedge*, that encloses the penumbra caused by this edge (see Figure 17). The wedge can be larger than the penumbra, that is we err on the safe side.

Then, we render the shadow volume, using the standard method (described in Section 2.5.2) in a visibility buffer. After this first pass, the visibility buffer contains the hard shadow.

In a subsequent pass, this visibility buffer is updated so that it contains the soft shadow values. This is done by rendering the front-facing triangles of each wedge. For each pixel covered by these triangles, we compute the percentage of the light source that is occluded, using fragment programs[20]. For pixels that are covered by the wedge but in the hard shadow (as computed by the previous pass), we compute the percentage of the light source that is visible, and add this value to the visibility buffer. For pixels covered by the wedge but in the illuminated part of the scene, we compute the percentage of the light source that is occluded and substract this value from the visibility buffer (see Figures 18 and 19).

After this second pass, the visibility buffer contains the percentage of visibility for all pixels in the picture. In a third pass, the visibility buffer is combined with the illumination computed using the standard OpenGL lighting model, giving the soft shadowed picture of the scene.

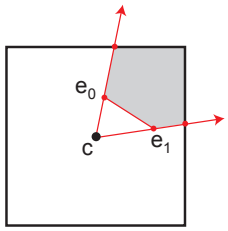


Figure 18: *Computing the area of the light source that is covered by a given edge. The fragment program computes the hatched area for each pixel inside the corresponding wedge.*

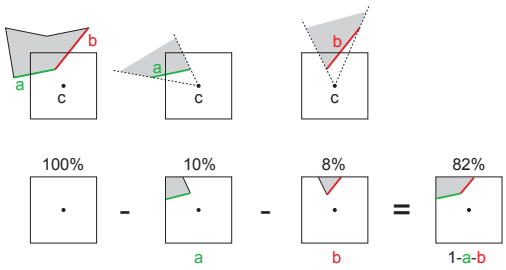


Figure 19: *Combining several connected edges. The portion of the light source that is occluded is equal to the sum of the portions of the light source occluded by the different edges.*

Discussion The complexity of the algorithm depends on the number of edges in the silhouette of the object, and on the number of pixels covered by each penumbra wedge. As a consequence, the easiest optimisation of the algorithm is to compute tighter penumbra wedges[5].

The main advantage of this algorithm is its speed. Using programmable graphics hardware for all complex computations, and tabulating complex functions into pre-computed textures, framerates of 150 frames per second are obtained on simple scenes, 50 frames per second on moderately complex scenes (1,000 shadow-casting polygons, with a large light source), with very convincing shadows. Performance depends mostly on the number of pixels covered by the penumbra wedges, so smaller light sources will result in faster rendering.

It should be noted that although a single sample is used to compute the silhouette of the object, the soft shadow computed by this algorithm is physically exact in simple cases, since visibility is computed on the entire light source. More precisely this happens when the silhouette of the occluder remains the same for all points on the light source, *e.g.* for a convex object that is distant enough from the light source.

The weak point of the algorithm is that it computes the silhouette of the object using only a single sample. It would fail on scenes where the actual silhouette of the object, as seen from the area light source, is very different from the silhouette computed using the single sample. Such scenes include scenes where a large area light source is close to the object (see Figure 7), and scenes where the shadows of several objects are combined together (as in Figure 6). In those circumstances, it is possible to compute a more accurate shadow by splitting the light source into smaller light sources. The authors report that splitting large light sources into 2×2 or 3×3 smaller light sources is usually enough to remove visible artefacts. It should be noted that splitting the light source into n light sources does not cut the speed of the algorithm by n, since the rendering time depends on the number of pixels covered by the penumbra wedges, and smaller light sources have smaller penumbra wedges.

One key to the efficiency of the algorithm is its use of fragment programs[20]. The fragment programs take as input the projections of the extremities of the edge onto the plane of the light source, and give as output the percentage of the light source that is occluded by the edge (see Figure 18). If several edges are projecting onto the light source, their contributions are simply added (see Figure 19) — this addition is done in the framebuffer. The authors have implemented several fragment programs, for spherical light sources, for textured rectangular light sources and for non-textured rectangular light sources.

4. Classification

4.1. Controlling the time

Algorithms used in real time or interactive applications must be able to run at a tuneable framerate, in order to spend less time for rendering at places where there is a lot of computation taking place, and more time when the processor is available.

Ideally, soft shadow methods used in real-time applications should take as input the amount of time available for rendering, and return a soft shadow computed to the best of the algorithm within the prescribed time limit. Since this review focuses on hot research algorithms, this feature has not been implemented in any of the algorithms reviewed here. However, all of these algorithms are tunable in the sense that there is some sort of parameter that the user can tweak, going from soft shadows that are computed very fast, but are possibly wrong, to soft shadows that can take more time to compute but are either more visually pleasing or more physically accurate.

Several of these parameters are available to a various degree in the methods reviewed:

- The easiest form of user control is the use of a different level-of-detail for the geometry of the occluders. Simpler geometry will result in faster rendering, either with image-based methods or with object-based methods. It can be expected that the difference in the shadow will not be noticeable with animated soft shadows.

Method	Time	Quality	Tunable	Light	Scene	Required Hardware
Image-based						
Multi-samples[22, 25]	I	*	Y	Polygon	1 planar receiver	
Distributed Multi-samples[28]	RT	**	Y	Planar		ShadowMap
Single sample[9, 33]	RT	*	Y	Sphere		ShadowMap
Convolution[45]	I	**	Y	Polygon		2D Convol.
Visibility Channel[24, 54]	I	**	Y	Linear, Polygon		2D Convol.
Geometry-based						
Plateaus[19]	I	**	Y	Sphere	1 planar receiver	
Penumbra Map[53]	RT	**	Y	Sphere		Vertex & Frag. Programs
Smoothie[11]	RT	**	Y	Sphere		Vertex & Frag. Programs
Soft Shadow Volumes[2, 4, 5]	RT	***	Y	Sphere, Rect.		Fragment Programs

Table 1: *Comparison of soft shadows algorithms (see Section 4 for details)*

- Another form of user control is to add more samples on the light source[22, 25, 1], or to subdivide large light sources into a set of smaller ones[2, 4, 5, 24, 54]. It should be noted that the order of magnitude for this parameter is variable: 256 to 1024 samples are required for point-based methods[22, 25, 1] to produce shadows without artefacts, while area-based methods[2, 4, 5, 24, 54] just need to cut the light source into 2×2 or 3×3 smaller sources. Either way, the rendering time is usually multiplied by the number of samples or sources.
- All image-based methods are also tuneable by changing the resolution of the buffer.
- Other parameters are method-specific:
  - the single sample soft shadows[9] method is tuneable by changing the search radius;
  - Convolution[45] is tuneable by subdividing the occluders into several layers;
  - Plateaus[19] are tuneable by changing the number of vertices used to discretize the cones and patches;
  - Smoothies[11] are tuneable by changing the maximum width of the smoothies;

4.2. Controlling the aspect

Another important information in chosing a real-time soft shadow algorithm is the aspect of the shadow it produces. Some of the algorithms described in this review can produce a physically exact solution if we allow them a sufficient rendering time. Other methods produce a physically exact solution in simple cases, but are approximate in more complex scenes, and finally a third class of methods produce shadows that are always approximate, but are usually faster to compute.

Physically exact (time permitting): Methods based on point samples on the light source[22, 25, 1] will produce physically exact shadows if the number of samples is sufficient. However, with current hardware, the number of samples compatible with interactive applications gives shadows that are not visually excellent (hence the poor mark these methods receive in table 1).

Physically exact on simple scenes: Methods that compute the percentage of the light source that is visible from the current pixel will give physically exact shadows in places where the assumptions they make on the respective geometry of the light source and the occluders are verified. For example, soft shadow volumes[4, 5] give physically exact shadows for isolated convex objects, provided that the silhouette computed is correct (that the occluder is far away from the light source). Visibility channel[24, 54] gives physically exact shadows for convex occluders and linear light sources[24], and for isolated edges and polygonal light sources[54] . Convolution[45] is physically exact for planar and parallel light source, receiver and occluder.

Always approximate: All methods that restrict themselves to computing only the inner- or the outer-penumbra are intrisically always approximate. They include single-sample soft shadows using shadow-width map[33], plateaus[19] and smoothies[11]. The original implementation of single sample soft shadows[9] computes both the inner- and the outer-penumbra, but gives them always the same width, which is not physically exact.

The second class of methods is probably the more interesting for producing nice looking pictures. While the conditions imposed seem excessively hard, it must be pointed out that they are conditions for which it is *guaranteed* that the shadow is exact in *all* the points of the scene. In most places of a standard scene, these methods will also produce physically exact shadows.

4.3. Number and shape of the light sources

The first cause for the soft shadow is the light source. Each real-time soft shadow method makes an assumption on the light sources, their shapes, their angles of emission and more importantly their number.

Field of emission: All the methods that are based on an image of the scene computed from the light source are restricted with respect to the field of emission of the light source, as a field of emission that is too large will result in distortions in the image. This restriction applies to all image-based algorithms, plus smoothies[11] and volume-based algorithms if the silhouette is computed using discontinuities in the shadow map[39].

On the contrary, volume-based methods can handle omni-directional illumination.

Shape: For extended light sources, the influence of the shape of the light source on a soft shadow is not directly perceptible. Most real-time soft shadow methods use this property by restricting themselves to simple light source shapes, such as spheres or rectangles:

- Single-sample soft shadows[9, 33], plateaus[19] and smoothies[11] assume a spherical light source. Soft shadow volumes[5] also work with a spherical light source.
- Visibility channel[24] was originally restricted to linear light sources.
- Subsequent implementation of the visibility channel works with polygonal light sources[54].
- Other methods place less restriction on the light source. Multi-sample methods[25, 1] can work with any kind of light source. Convolution[45] are also not restricted. However, in both cases, the error in the algorithm is smaller for planar light sources.
- Convolution[45] and soft shadow volumes[4, 5] work with textured rectangles, thus allowing any kind of planar light source. The texture can even be animated[4, 5].

Number: All real-time soft shadow algorithms are assuming a single light source. Usually, computing the shadow from several light sources results in multiplying the rendering time by the number of light sources. However, for all the methods that work for any kind of planar light source[25, 1, 45, 4, 5], it is possible to simulate several co-planar light sources by placing the appropriate texture on a plane. This gives us several soft shadows in a single application of the algorithm. However, it has a cost: since the textured light source is larger, the algorithms will run more slowly.

4.4. Constraints on the scene

The other elements causing shadows are the occluders and the receivers. Most real-time soft shadows methods make some assumptions on the scene, either explicit or implicit.

Receiver: The strongest restriction is when the object receiving shadows is a plane, as with the plateaus method[19]. Multi-sample soft shadow[25, 22] is also restricted to a small number of receivers for interactive rendering. In that case, self-shadowing is not applicable.

Self-shadowing: The convolution[45] method requires that the scene is cut into clusters, within which no self-shadows are computed.

Silhouette: For all the methods that require a silhouette extraction — such as object-based methods — it is implicitly assumed that we can compute a silhouette for all the objects in the scene. In practice, this usually means that the scene is made of closed triangle meshes.

4.5. New generation of GPUs

Most real-time soft shadow methods use the features of the graphics hardware that were available to the authors at the time of writing:

Shadow-map: all image-based methods use the `GL_ARB_SHADOW` extension for shadow maps. This extension (or an earlier version) is available, for example, on Silicon Graphics hardware above the Infinite Reality 2, on NVIDIA graphics cards above the GeForce 3 and on ATI graphics above the Radeon9500.

Imaging subset: along with this extension, some methods also compute convolutions on the shadow map. These convolutions can be computed in hardware if the *Imaging Subset* of the OpenGL specification is present. This is the case on all Silicon Graphics machines and NVIDIA cards.

Programmable GPU: finally, the most recent real-time soft shadow methods use the programming capability introduced in recent graphics hardware. Vertex programs[14] and fragment programs[21] are used for single-sample soft shadows[33], penumbra maps[53], smoothies[11] and soft shadow volumes[4, 5]. In practice, this restricts these algorithms to only the latest generation of graphics hardware, such as the NVIDIA GeForce FX or the ATI Radeon 9500 and above.

Many object-based algorithms suffer from the fact that they need to compute the silhouette of the occluders, a costly step that can only be done on the CPU. Wyman and Hansen[53] report that computing the silhouette of a moderately complex occluder (5000 polygons) uses 10 ms in their implementation. If the next generation of graphics hardware would include the possibility to compute this silhouette entirely on the graphics card[10], object-based algorithms[53, 11, 2, 4, 5] would greatly benefit from the speed-up.

5. Conclusions

In this State of the Art Review, we have described the issues encountered when working with soft shadows. We have presented existing algorithms that produce soft shadows in real time. Two main categories of approaches have been reviewed, based on shadow maps and shadow volumes. Each one has advantages and drawbacks, and none of them can simultaneously solve all the problems we have mentioned. This motivated a discussion and classification of these methods, hopefully allowing easier algorithm selection based on a particular application's constraints.

We have seen that the latest algorithms benefit from the programmability of recent graphics hardware. Two main directions appear attractive to render high-quality soft shadows in real time: by programming graphics hardware, and by taking advantage simultaneously of both image-based and object-based techniques. Distributed rendering, using for instance PC clusters, is another promising avenue although little has been achieved so far. Interactive display speeds can be obtained today even on rather complex scenes. Continuing improvements of graphics technology — in performance and programmability — lets us expect that soft shadows will soon become a common standard in real-time rendering.

Acknowledgments

The "Hugo" robot used in the pictures of this paper was created by Laurence Boissieux.

This work was supported in part by the "ACI Jeunes Chercheurs" *CYBER* of the French Ministry of Research, and by the "Région Rhône-Alpes" through the DEREVE research consortium.

We wish to express our gratitude to the authors of the algorithms described in this review, who have provided us with useful detailed information about their work, and to the anonymous reviewers whose comments and suggestions have significantly improved the paper.

Remark: All the smooth shadows pictures in this paper were computed with distributed ray-tracing, using 1024 samples on the area light sources.

References

1. Maneesh Agrawala, Ravi Ramamoorthi, Alan Heirich, and Laurent Moll. Efficient image-based methods for rendering soft shadows. In *Computer Graphics (SIGGRAPH 2000)*, Annual Conference Series, pages 375–384. ACM SIGGRAPH, 2000. 9, 10, 17, 18
2. Tomas Akenine-Möller and Ulf Assarsson. Approximate soft shadows on arbitrary surfaces using penumbra wedges. In *Rendering Techniques 2002 (13th Eurographics Workshop on Rendering)*, pages 297–306. ACM Press, 2002. 14, 15, 17, 18
3. Tomas Akenine-Möller and Eric Haines. *Real-Time Rendering*. A K Peters Ltd, 2nd edition, 2002. 2
4. Ulf Assarsson and Tomas Akenine-Möller. A geometry-based soft shadow volume algorithm using graphics hardware. *ACM Transactions on Graphics (SIGGRAPH 2003)*, 22(3), 2003. 14, 15, 17, 18
5. Ulf Assarsson, Michael Dougherty, Michael Mounier, and Tomas Akenine-Möller. An optimized soft shadow volume algorithm with real-time performance. In *Graphics Hardware*, 2003. 4, 5, 14, 15, 16, 17, 18
6. ATI. Smartshader^{TM} technology white paper. http://www.ati.com/products/pdf/smartshader.pdf, 2001. 14, 15
7. Harlen Costa Batagelo and Ilaim Costa Júnior. Real-time shadow generation using BSP trees and stencil buffers. In *SIBGRAPI*, volume 12, pages 93–102, October 1999. 8
8. Stefan Brabec. Personnal communication, May 2003. 13
9. Stefan Brabec and Hans-Peter Seidel. Single sample soft shadows using depth maps. In *Graphics Interface*, 2002. 9, 12, 17, 18
10. Stefan Brabec and Hans-Peter Seidel. Shadow volumes on programmable graphics hardware. *Computer Graphics Forum (Eurographics 2003)*, 25(3), September 2003. 8, 18
11. Eric Chan and Fredo Durand. Rendering fake soft shadows with smoothies. In *Rendering Techniques 2003 (14th Eurographics Symposium on Rendering)*. ACM Press, 2003. 14, 15, 17, 18
12. Franklin C. Crow. Shadow algorithms for computer graphics. *Computer Graphics (SIGGRAPH 1977)*, 11(3):242–248, 1977. 8
13. George Drettakis and Eugene Fiume. A fast shadow algorithm for area light sources using backprojection. In *Computer Graphics (SIGGRAPH 1994)*, Annual Conference Series, pages 223–230. ACM SIGGRAPH, 1994. 3, 6
14. Cass Everitt. OpenGL ARB vertex program. http://developer.nvidia.com/docs/IO/8230/GDC2003_OGL_ARBVertexProgram.pdf, 2003. 8, 14, 15, 18
15. Cass Everitt and Mark J. Kilgard. Practical and robust stenciled shadow volumes for hardware-accelerated rendering. http://developer.nvidia.com/object/robust_shadow_volumes.html, 2002. 8, 9
16. Cass Everitt and Mark J. Kilgard. Optimized stencil shadow volumes. http://developer.nvidia.com/docs/IO/8230/GDC2003_ShadowVolumes.pdf, 2003. 9

17. Cass Everitt, Ashu Rege, and Cem Cebenoyan. Hardware shadow mapping. http://developer.nvidia.com/object/hwshadowmap_paper.html. 7

18. Randima Fernando, Sebastian Fernandez, Kavita Bala, and Donald P. Greenberg. Adaptive shadow maps. In *Computer Graphics (SIGGRAPH 2001)*, Annual Conference Series, pages 387–390. ACM SIGGRAPH, 2001. 7

19. Eric Haines. Soft planar shadows using plateaus. *Journal of Graphics Tools*, 6(1):19–27, 2001. 14, 15, 17, 18

20. Evan Hart. ARB Fragment Program: Fragment level programmability in OpenGL. http://www.ati.com/developer/gdc/GDC2003_OGL_ARBFragmentProgram.pdf, 2003. 14, 15, 16

21. Evan Hart. Other New OpenGL Stuff: Important stuff that doesn't fit elsewhere. http://www.ati.com/developer/gdc/GDC2003_OGL_MiscExtensions.pdf, 2003. 9, 18

22. Paul S. Heckbert and Michael Herf. Simulating soft shadows with graphics hardware. Technical Report CMU-CS-97-104, Carnegie Mellon University, January 1997. 9, 17, 18

23. Tim Heidmann. Real shadows, real time. In *Iris Universe*, volume 18, pages 23–31. Silicon Graphics Inc., 1991. 8

24. Wolfgang Heidrich, Stefan Brabec, and Hans-Peter Seidel. Soft shadow maps for linear lights high-quality. In *Rendering Techniques 2000 (11th Eurographics Workshop on Rendering)*, pages 269–280. Springer-Verlag, 2000. 5, 9, 11, 17, 18

25. Michael Herf. Efficient generation of soft shadow textures. Technical Report CMU-CS-97-138, Carnegie Mellon University, 1997. 9, 10, 17, 18

26. J.-C. Hourcade and A. Nicolas. Algorithms for antialiased cast shadows. *Computers & Graphics*, 9(3):259–265, 1985. 7, 13

27. Geoffre S. Hubona, Philip N. Wheeler, Gregory W. Shirah, and Matthew Brandt. The role of object shadows in promoting 3D visualization. *ACM Transactions on Computer-Human Interaction*, 6(3):214–242, 1999. 1, 2

28. M. Isard, M. Shand, and A. Heirich. Distributed rendering of interactive soft shadows. In *4th Eurographics Workshop on Parallel Graphics and Visualization*, pages 71–76. Eurographics Association, 2002. 10, 17

29. Brett Keating and Nelson Max. Shadow penumbras for complex objects by depth-dependent filtering of multi-layer depth images. In *Rendering Techniques 1999 (10th Eurographics Workshop on Rendering)*, pages 205–220. Springer-Verlag, 1999. 10

30. Daniel Kersten, Pascal Mamassian, and David C. Knill. Moving cast shadows and the perception of relative depth. Technical Report n^o 6, Max-Planck-Institut fuer biologische Kybernetik, 1994. 1, 2

31. Daniel Kersten, Pascal Mamassian, and David C. Knill. Moving cast shadows and the perception of relative depth. *Perception*, 26(2):171–192, 1997. 1, 2

32. Mark J. Kilgard. Improving shadows and reflections via the stencil buffer. http://developer.nvidia.com/docs/IO/1348/ATT/stencil.pdf, 1999. 8

33. Florian Kirsch and Juergen Doellner. Real-time soft shadows using a single light sample. *Journal of WSCG (Winter School on Computer Graphics 2003)*, 11(1), 2003. 12, 13, 17, 18

34. David C. Knill, Pascal Mamassian, and Daniel Kersten. Geometry of shadows. *Journal of the Optical Society of America*, 14(12):3216–3232, 1997. 1

35. Johann Heinrich Lambert. *Die freye Perspektive*. 1759. 1, 2

36. Tom Lokovic and Eric Veach. Deep shadow maps. In *Computer Graphics (SIGGRAPH 2000)*, Annual Conference Series, pages 385–392. ACM SIGGRAPH, 2000. 7

37. Céline Loscos and George Drettakis. Interactive high-quality soft shadows in scenes with moving objects. *Computer Graphics Forum (Eurographics 1997)*, 16(3), September 1997. 6

38. Pascal Mamassian, David C. Knill, and Daniel Kersten. The perception of cast shadows. *Trends in Cognitive Sciences*, 2(8):288–295, 1998. 1, 2

39. Michael D. McCool. Shadow volume recontsruction from depth maps. *ACM Transactions on Graphics*, 19(1):1–26, 2000. 8, 18

40. Steve Morein. ATI radeon hyperz technology. In *Graphics Hardware Workshop*, 2000. 6

41. Steven Parker, Peter Shirley, and Brian Smits. Single sample soft shadows. Technical Report UUCS-98-019, Computer Science Department, University of Utah, October 1998. 12, 15

42. William T. Reeves, David H. Salesin, and Robert L. Cook. Rendering antialiased shadows with depth maps. *Computer Graphics (SIGGRAPH 1987)*, 21(4):283–291, 1987. 7

43. Stefan Roettger, Alexander Irion, and Thomas Ertl. Shadow volumes revisited. In *Winter School on Computer Graphics*, 2002. 8

44. Mark Segal, Carl Korobkin, Rolf van Widenfelt, Jim Foran, and Paul Haeberli. Fast shadows and lighting effects using texture mapping. *Computer Graphics (SIGGRAPH 1992)*, 26(2):249–252, July 1992. 7

45. Cyril Soler and François X. Sillion. Fast calculation of soft shadow textures using convolution. In *Computer Graphics (SIGGRAPH 1998)*, Annual Conference Series, pages 321–332. ACM SIGGRAPH, 1998. 4, 6, 9, 13, 17, 18

46. Marc Stamminger and George Drettakis. Perspective shadow maps. *ACM Transactions on Graphics (SIGGRAPH 2002)*, 21(3):557–562, 2002. 7

47. Sylvain Vignaud. Real-time soft shadows on geforce class hardware. http://tfpsly.planet-d.net/english/3d/SoftShadows.html, 2003. 14

48. Leonardo Da Vinci. *Codex Urbinas*. 1490. 1, 2

49. Leonard Wanger. The effect of shadow quality on the perception of spatial relationships in computer generated imagery. *Computer Graphics (Interactive 3D Graphics 1992)*, 25(2):39–42, 1992. 1, 2

50. Lance Williams. Casting curved shadows on curved surfaces. *Computer Graphics (SIGGRAPH 1978)*, 12(3):270–274, 1978. 7

51. Andrew Woo. The shadow depth map revisited. In *Graphics Gems III*, pages 338–342. Academic Press, 1992. 7

52. Andrew Woo, Pierre Poulin, and Alain Fournier. A survey of shadow algorithms. *IEEE Computer Graphics and Applications*, 10(6):13–32, November 1990. 1, 2

53. Chris Wyman and Charles Hansen. Penumbra maps: Approximate soft shadows in real-time. In *Rendering Techniques 2003 (14th Eurographics Symposium on Rendering)*. ACM Press, 2003. 14, 15, 17, 18

54. Zhengming Ying, Min Tang, and Jinxiang Dong. Soft shadow maps for area light by area approximation. In *10th Pacific Conference on Computer Graphics and Applications*, pages 442–443. IEEE, 2002. 9, 11, 17, 18

55. Hansong Zhang. Forward shadow mapping. In *Rendering Techniques 1998 (9th Eurographics Workshop on Rendering)*, pages 131–138. Springer-Verlag, 1998. 8

4.7.3 Soft shadow maps: efficient sampling of light source visiblity (CGF 2006)

Auteurs : Lionel A , Nicolas H , Marc L , Jean-Marc H , Charles H et François X. S .
Journal : *Computer Graphics Forum*, vol. 25, n° 4.
Date : décembre 2006

Volume 0 (1981), Number 0 pp. 1–17

Soft Shadow Maps: Efficient Sampling of Light Source Visibility

Lionel Atty[1], Nicolas Holzschuch[1], Marc Lapierre[2], Jean-Marc Hasenfratz[1,3], Charles Hansen[4] and François X. Sillion[1]

[1] ARTIS/GRAVIR–IMAG INRIA
[2] MOVI/GRAVIR–IMAG INRIA
[3] Université Pierre Mendès-France
[4] School of Computing, University of Utah

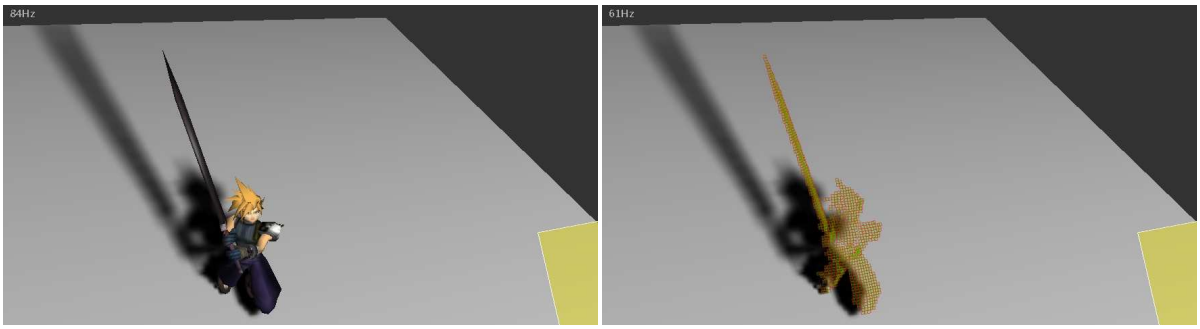


Figure 1: *Our algorithm computes soft shadows in real-time (left) by replacing the occluders with a discretized version (right), using information from the shadow map. This scene runs at 84 fps.*

Abstract

Shadows, particularly soft shadows, play an important role in the visual perception of a scene by providing visual cues about the shape and position of objects. Several recent algorithms produce soft shadows at interactive rates, but they do not scale well with the number of polygons in the scene or only compute the outer penumbra. In this paper, we present a new algorithm for computing interactive soft shadows on the GPU. Our new approach provides both inner- and outer-penumbra, and has a very small computational cost, giving interactive frame-rates for models with hundreds of thousands of polygons.

Our technique is based on a sampled image of the occluders, as in shadow map techniques. These shadow samples are used in a novel manner, computing their effect on a second projective shadow texture using fragment programs. In essence, the fraction of the light source area hidden by each sample is accumulated at each texel position of this Soft Shadow Map*. We include an extensive study of the approximations caused by our algorithm, as well as its computational costs.*

Categories and Subject Descriptors (according to ACM CCS): I.3.1 [Computer Graphics]: Graphics processors I.3.7 [Computer Graphics]: Color, shading, shadowing, and texture

1. Introduction

Shadows add important visual information to computer-generated images. The perception of spatial relationships between objects can be altered or enhanced simply by modifying the shadow shape, orientation, or position [WFG92, Wan92, KMK97]. Soft shadows, in particular, provide robust contact cues by the hardening of the shadow due to proximity resulting in a hard shadow upon contact. The advent of powerful graphics hardware on low-cost computers has led to the emergence of many interactive soft shadow algorithms (for a detailed study of these algorithms, please refer to [HLHS03]).

In this paper, we introduce a novel method based on shadow maps to interactively render soft shadows. Our

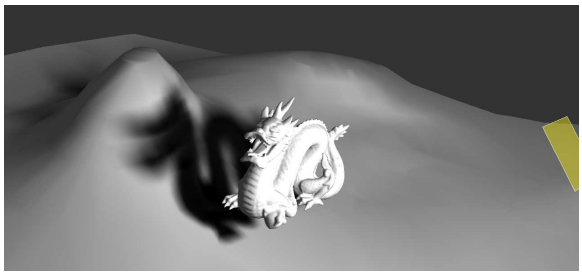


Figure 2: *Applying our algorithm (*200,000 *polygons, occluder map* 256×256*, displayed at 32 fps).*

method interactively computes a projective shadow texture, the *Soft Shadow Map*, that incorporates soft shadows based on light source visibility from receiver objects (see Fig. 2). This texture is then projected onto the scene to provide interactive soft shadows of dynamic objects and dynamic area light sources.

There are several advantages to our technique when compared to existing interactive soft-shadow algorithms: First, it is not necessary to compute silhouette edges. Second, the algorithm is not fill-bound, unlike methods based on shadow volumes. These properties provide better scaling for occluding geometry than other GPU based soft shadow techniques [WH03, CD03, AAM03]. Third, unlike some other shadow map based soft shadow techniques, our algorithm does not dramatically overestimate the umbra region [WH03, CD03]. Fourth, while other methods have relied on an interpolation from the umbra to the non-shadowed region to approximate the penumbra for soft shadows [AHT04, WH03, CD03, BS02], our method computes the visibility of an area light source for receivers in the penumbra regions.

Our algorithm also has some limitations when compared to existing algorithms. First, our algorithm splits scene geometry into occluders and receivers and self shadowing is not accounted for. Also, since our algorithm uses shadow maps to approximate occluder geometry, it inherits the well known issues with aliasing from shadow map techniques. For large area light sources, the soft shadows tend to blur the artifacts but for smaller area light sources, such aliasing is apparent.

We acknowledge that these limitations are important, and they may prevent the use of our algorithm in some cases. However, there are many applications such as video games or immersive environments where the advantages of our algorithm (a very fast framerate, and a convincing soft shadow) outweigh its limitations. We also think that this new algorithm could be the start of promising new research.

In the following section, we review previous work on interactive computation of soft shadows. In Section 3, we present the basis of our algorithm, and in the following section, we provide implementation details. In the next two sections, we conduct an extensive analysis of our algorithm; first, in Section 5, we study the approximations in our soft shadows, then in Section 6 we study the rendering times of our algorithm. Both studies are done first from a theoretical point of view, then experimentally. Finally, in Section 7, we conclude and expose possible future directions for research.

2. Previous Work

Researchers have investigated shadow algorithms for computer-generated images for nearly three decades. The reader is referred to a recent state-of-the art report by Hasenfratz *et al.* [HLHS03], the overview by Woo *et al.* [WPF90] and the book by Akenine-Möller and Haines [AMH02].

The two most common methods for interactively producing shadows are shadow maps [Wil78] and shadow volumes [Cro77]. Both of these techniques have been extended for soft shadows. In the case of shadow volumes, Assarsson and Akenine-Möller [AAM03] used penumbra wedges in a technique based on shadow volumes to produce soft shadows. Their method depends on locating silhouette edges to form the penumbra wedges. While providing good soft shadows without an overestimate of the umbra, the algorithm is fill-limited, particularly when zoomed in on a soft shadow region. Since it is necessary to compute the silhouette edges at every frame, the algorithm also suffers from scalability issues when rendering occluders with large numbers of polygons.

The fill-rate limitation is a well known limitation of shadow-volume based algorithms. Recent publications [CD04, LWGM04] have focused on limiting the fill-rate for shadow-volume algorithms, thus removing this limitation.

On shadow maps, Chan and Durand [CD03] and Wyman and Hansen [WH03] both employed a technique which uses the standard shadow map method for the umbra region and builds a map containing an approximate penumbra region that can be used at run-time to give the appearance, including hard shadows at contact, of soft shadows. While these methods provide interactive rendering, both only compute the outer-penumbra, the part of the penumbra that is outside the hard shadow. In effect, they are overestimating the umbra region, resulting in the incorrect appearance of soft shadows in the case of large area light sources. These methods also depend on computing the silhouette edges in object space for each frame; this requirement limits the scalability for occluders with large numbers of polygons.

Arvo *et al.* [AHT04] used an image-space flood-fill method to produce approximate soft shadows. Their algorithm is image-based, like ours, but works on a detection of shadow boundary pixels, followed by several passes to replace the boundary by a soft shadow, gradually extending the soft shadow at each pass. The main drawback of their method is that the number of passes required is proportional

to the extent of the penumbra region, and the rendering time is proportional to the number of shadow-filling passes.

Guennebaud *et al.* [GBP06] also used the back projection of each pixel in the shadow map to compute the soft shadow. Their method was developed independently of ours, yet is very similar. The main differences between the two methods lie in the order of the computations: we compute the soft shadow in shadow map space, while they compute the soft shadow in screen space, requiring a search in the shadow map.

Brabec and Seidel [BS02] and Kirsch and Doellner [KD03] use a shadow map to compute soft shadows, by searching at each pixel of the shadow map for the nearest boundary pixel, then interpolating between illumination and shadow as a function of the distance between this pixel and the boundary pixel and the distances between the light source, the occluder and the receiver. Their algorithm requires scanning the shadow map to look for boundary pixels, a potentially costly step; in practical implementations they limit the search radius, thus limiting the actual size of the penumbra region.

Soler and Sillion [SS98] compute a soft shadow map as the convolution of two images representing the source and blocker. Their technique is only accurate for planar and parallel objects, although it can be extended using an object hierarchy. Our technique can be seen as an extension of this approach, where the convolution is computed for each sample of an occlusion map, and the results are then combined.

Finally, McCool [McC00] presented an algorithm merging shadow volume and shadow map algorithms by detecting silhouette pixels in the shadow map and computing a shadow volume based on these pixels. Our algorithm is similar in that we are computing a shadow volume for each pixel in the shadow map. However, we never display this shadow volume, thus avoiding fill-rate issues.

3. Algorithm

3.1. Presentation of the algorithm

Our algorithm assumes a rectangular light source and starts by separating potential occluders (such as moving characters) from potential receivers (such as the background in a scene) (Fig. 3(a)). We will compute the *soft shadows* only from the occluders onto the receivers.

Our algorithm computes a *Soft Shadow Map*, (SSM), for each light source: a texture containing the texelwise percentage of occlusion from the light source. This soft shadow map is then projected onto the scene from the position of the light source, to give soft shadows (see Fig. 2).

Our algorithm is an extension of the shadow map algorithm: we start by computing depth buffers of the scene. Unlike the standard shadow map method, we will need two depth buffers: one for the occluders (the *occluder map*) and the other for the receivers.

```
Compute depth map of receivers
Compute depth map of occluders
for all pixels in occluder map
    Retrieve depth of occluder at this pixel
    Compute micro-patch associated with this pixel
    Compute extent of penumbra for this micro-patch
    for all pixels in penumbra extent for micro-patch
        Retrieve receiver depth at this pixel
        Compute percentage of occlusion for this pixel
        Add to current percentage in soft shadow map
    end
end
Project soft shadow map on the scene
```

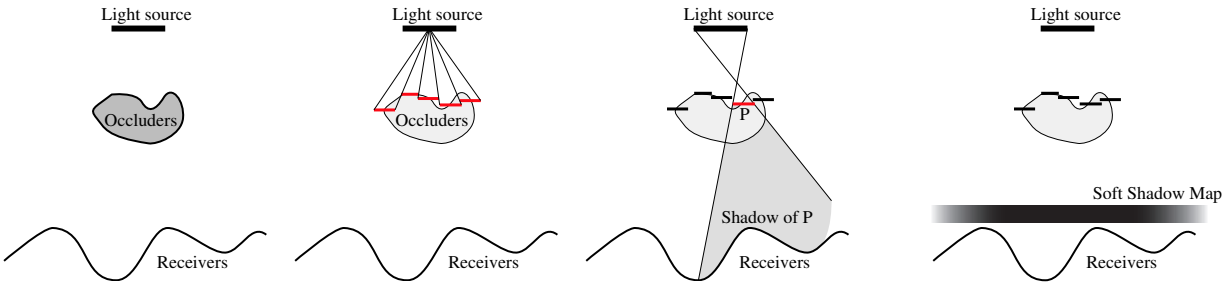
Figure 4: *Our algorithm*

The occluder map depth buffer is used to discretize the set of occluders (see Fig. 3(b)): each pixel in this occluder map is converted into a micro-patch that covers the same image area but is is located in a plane parallel to the light source, at a distance corresponding to the pixel depth. Pixels that are close to the light source are converted into small rectangles and pixels that are far from the light source are converted into larger rectangles. At the end of this step, we have a discrete representation of the occluders.

The receiver map depth buffer will be used to provide the receiver depth, as our algorithm uses the distance between light source and receiver to compute the soft shadow values.

We compute the soft shadow of each of the micro-patches constituting the discrete representation of the occluders (see Fig. 3(c)), and sum them into the soft shadow map (SSM) (see Fig. 3(d)). This step would be potentially costly, but we achieve it in a reasonable amount of time with two key points: 1) the micro-patches are parallel to the light source, so computing their penumbra extent and their percentage of occlusion only requires a small number of operations, and 2) these operations are computed on the graphics card, exploiting the parallelism of the GPU engine. The percentage of occlusion from each micro-patch takes into account the relative distances between the occluders, the receiver and the light source. Our algorithm introduces several approximations on the actual soft shadow. These approximations will be discussed in Section 5.

The pseudo-code for our algorithm is given in Fig. 4. In the following subsections, we will review in detail the individual steps of the algorithm: discretizing the occluders (Section 3.2), computing the penumbra extent for each micro-patch (Section 3.3) and computing the percentage of occlusion for each pixel in the Soft Shadow Map (Section 3.4). Specific implementation details will be given in Section 4.



(a) Scene view (b) Discretizing occluders (c) Soft shadows from one micro-patch (d) Summing the soft shadows

Figure 3: *The main steps of our algorithm*

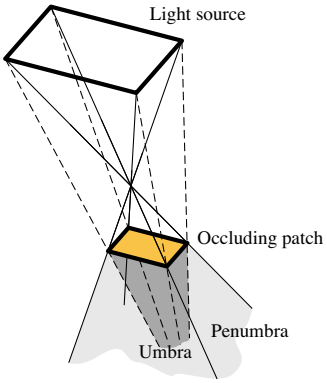


Figure 5: *The penumbra extent of a micro-patch is a rectangular pyramid*

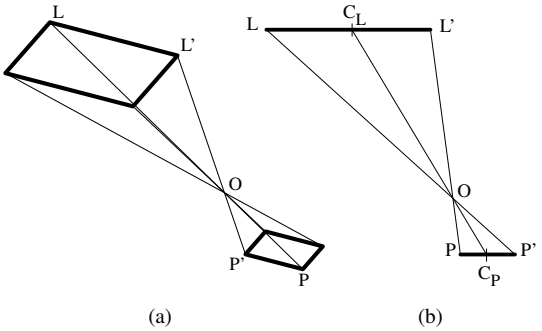


Figure 6: *Finding the apex of the pyramid is reduced to a 2D problem*

3.2. Discretizing the occluders

The first step in our algorithm is a discretization of the occluders. We compute a depth buffer of the occluders, as seen from the light source, then convert each pixel in this *occluder map* into the equivalent polygonal micro-patch that lies in a plane parallel to the light source, at the appropriate depth and occupies the same image plane extent (see Fig. 1).

The occluder map is axis-aligned with the rectangular light source and has the same aspect ratio: all micro-patches created in this step are also axis-aligned with the light source and have the same aspect ratio.

3.3. Computing penumbra extents

Each micro-patch in the discretized occluder is potentially blocking some light between the light source and some portion of the receiver. To reduce the amount of computations, we compute the penumbra extent of the micro-patches, and we only compute occlusion values inside these extents.

Since the micro-patches are parallel, axis-aligned with the light source and have the same aspect ratio, the penumbra extent of each micro-patch is a rectangular pyramid (Fig. 5). Finding the penumbra extent of the light source is equivalent to finding the apex O of the pyramid (Fig. 6(a)). This reduces to a 2D problem, considering parallel edges (LL') and (PP') on both polygons (Fig. 6(b)). Since (LL') and (PP') are parallel lines, we have:

$$\frac{OL}{OP} = \frac{OL'}{OP'} = \frac{LL'}{PP'}$$

This ratio is the same if we consider the center of each line segment:

$$\frac{OC_L}{OC_P} = \frac{LL'}{PP'}$$

Since the micro-patch and the light source have the same aspect ratio, the ratio $r = \frac{LL'}{PP'}$ is the same for both sides of the micro-patch (thus, the penumbra extent of the micro-patch is indeed a pyramid).

We find the apex of the pyramid by applying a scaling to the center of the micro-patch (C_P), with respect to the center

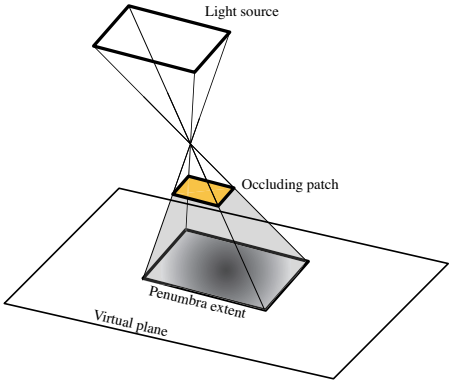


Figure 7: *The intersection between the pyramid and the virtual plane is an axis-aligned rectangle*

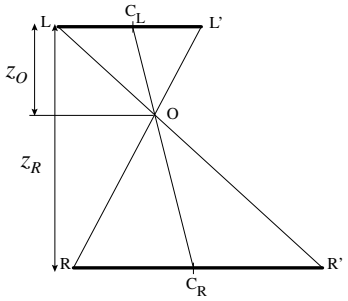


Figure 8: *Computing the position and extent of the penumbra rectangle for each micro-patch.*

of the light source (C_L):

$$\overrightarrow{C_L O} = \frac{r}{1+r}\overrightarrow{C_L C_P}$$

where r is again the ratio $r = \frac{LL'}{PP'}$.

We now use this pyramid to compute occlusion in the soft shadow map (see Fig. 7). We use a virtual plane, parallel to the light source, to represent this map (which will be projected onto the scene). The intersection of the penumbra pyramid with this virtual plane is an axis-aligned rectangle. We only have to compute the percentage of occlusion inside this rectangle.

Computing the position and size of the penumbra rectangle uses the same formulas as for computing the apex of the pyramid (see Fig. 8):

$$\begin{aligned}\overrightarrow{C_L C_R} &= \frac{z_R}{z_O}\overrightarrow{C_L O} \\ RR' &= LL'\frac{z_R - z_O}{z_O}\end{aligned}$$

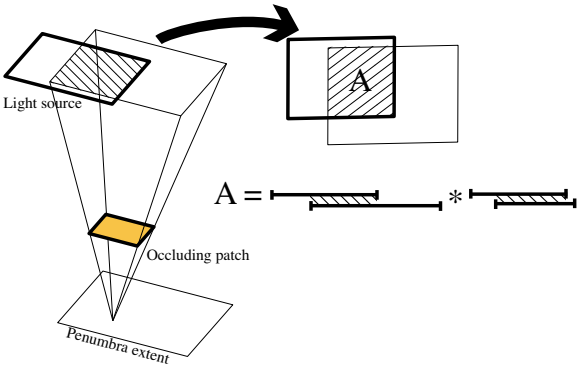


Figure 9: *We reproject the occluding micro-patch onto the light source and compute the percentage of occlusion.*

3.4. Computing the soft shadow map

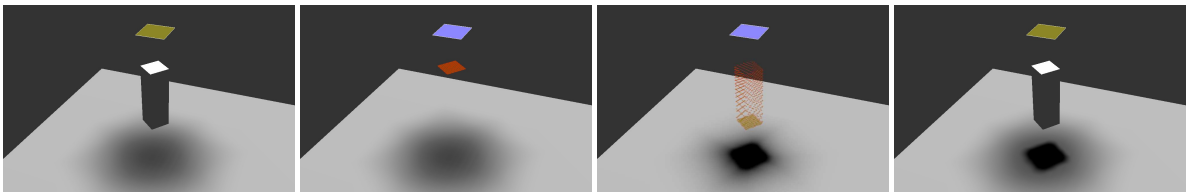
For all the pixels of the SSM lying inside this penumbra extent, we compute the percentage of the light source that is occluded by this micro-patch. This percentage of occlusion depends on the relative positions of the light source, the occluders and the receivers. To compute it, for each pixel on the receiver inside this extent, we project the occluding micro-facet back onto the light source [DF94] (Fig. 9). The result of this projection is an axis-aligned rectangle; we need to compute the intersection between this rectangle and the light source.

Computing this intersection is equivalent to computing the two intersections between the respective intervals on both axes. This part of the computation is done on the GPU, using a fragment program: the penumbra extent is converted into an axis-aligned quad, which we draw in a float buffer. For each pixel inside this quad, the fragment program computes the percentage of occlusion. These percentages are summed using the blending capability of the graphics card (see Section 4.2).

3.5. Two-sided soft-shadow maps

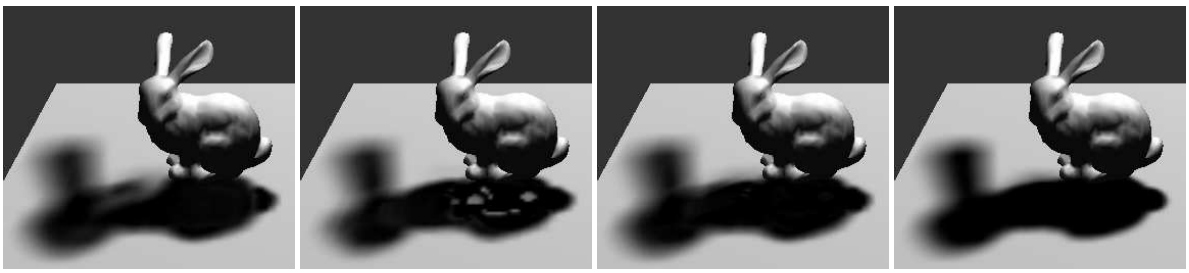
As with many other soft shadow computation algorithms [HLHS03], our algorithm exhibits artifacts because we are computing soft shadows using a single view of the occluder. Shadow effects linked to parts of the occluder that are not directly visible from the light source are not visible. In Fig. 10(a), our algorithm only computes the soft shadow for the front part of the occluder, because the back part of the occluder does not appear in the occluder map. This limitation is frequent in real-time soft-shadow algorithms [HLHS03].

For our algorithm, we have devised an extension that solves this limitation: we compute two occluder maps. In the first, we discretize the closest, front-facing faces of the occluders (see Fig. 10(b)). In the second, we discretize the furthest, back-facing faces of the occluders (see Fig. 10(c)).



(a) Original algorithm (b) Closest, front faces of the occluder discretized with their shadow (c) Furthest, back faces of the occluder discretized with their shadow (d) Combining the two soft shadow maps

Figure 10: *The original algorithm fails for some geometry. The two-pass method gives the correct shadow.*



(a) One pass (148 fps) (b) One pass with bottom patches (142 fps) (c) Two passes (84 fps) (d) Ground truth

Figure 11: *Two-pass shadow computations enhance precision.*

We then compute a soft shadow map for each occluder map, and merge them, using the maximum of each occluder map. The resulting occlusion map has eliminated most artifacts (Fig. 10(d) and 11). Empirically, the cost of the two-pass algorithm is between 1.6 and 1.8 times the cost of the one-pass algorithm. Depending on the size of a model and the quality requirements of a given application, the second pass may be worth this extra cost. For example, for an animated model of less than $100,000$ polygons, the one-pass algorithm renders at approximately 60 fps. Adding the second pass drops the framerate to 35 fps — which is still interactive.

4. Implementation details

4.1. Repartition between CPU and GPU

Our algorithm (see Fig. 4) starts by rendering two depth maps, one for the occluders and one for the receivers; these depth maps are both computed by the GPU. Then, in order to generate the penumbra extents for the micro-patches, the occluders depth map is transferred back to the CPU.

On the CPU, we generate the penumbra extents for the micro-patch associated to each non-empty pixel of the occluders depth map. We then render these penumbra extents, and for each pixel, we execute a small fragment program to compute the percentage of occlusion. Computing the percentage of occlusion at each pixel of the soft shadow map is done on the GPU (see section 4.2).

These contributions from each micro-patch are added together; we use for this the blending ability of the GPU: occlusion percentages are rendered into a floating-point buffer with blending enabled, thus the percentage values for each micro-patch are automatically added to the previously computed percentage values.

4.2. Computing the intersection

For each pixel of the SSM lying inside the penumbra extent of a micro-patch, we compute the percentage of the light source that is occluded by this micro-patch, by projecting the occluding micro-patch back onto the light source (see Fig. 9). We have to compute the intersection of two axis-aligned rectangles, which is the product of the two intersections between the respective intervals on both axes.

We have therefore reduced our intersection problem from a 2D problem to two separate 1D problems. To further optimize the computations, we use the `SAT` instructions in the fragment program assembly language: without loss of generality, we can convert the rectangle corresponding to the light source to $[0,1] \times [0,1]$. Each interval intersection becomes the intersection between one $[a,b]$ interval and $[0,1]$. Exploiting the `SAT` instruction and swizzling, computing the area of the intersection between the projection of the oc-

cluder $[a,b] \times [c,d]$ and the light source $[0,1] \times [0,1]$ only requires three instructions:

```
MOV_SAT rs,{a,b,c,d}
SUB rs, rs, rs.yxwz
MUL result.color, rs.x, rs.z
```

Computing the $[a,b] \times [c,d]$ intervals requires projecting the micro-patch onto the light source and scaling the projection. This uses 8 other instructions: 6 basic operations (`ADD`, `MUL`, `SUB`), one reciprocal (`RCP`) and one texture lookup to get the depth of the receiver. The total length of our fragment program is therefore 11 instructions, including one texture lookup.

4.3. Possible improvements

As it stands, our algorithm makes a very light use of GPU resources: we only execute a very small fragment program, once for each pixel covered by the penumbra extent, and we exploit the blending ability for floating point buffers.

The main bottleneck of our algorithm is that the penumbra extents have to be computed on the CPU. This requires transfering the occluders depth map to the CPU, and looping over the pixels of the occluders depth map on the CPU. It should be possible to remove this step by using the render-to-vertex- buffer function: instead of rendering the occluders depth map, we would directly render the penumbra extents for each micro-patch into a vertex buffer. This vertex buffer would be rendered in a second pass, generating the soft shadow map.

5. Error Analysis and comparison

In this section, we analyze our algorithm, its accuracy and how it compares with the exact soft-shadows. We first study potential sources of error from a theoretical point of view, in Section 5.1, then we conduct an experimental analysis, comparing the soft shadows produced with exact soft shadows, in Section 5.2.

5.1. Theoretical analysis

Our algorithm replaces the occluder with a discretized version. This discretization ensures interactive framerates, but it can also be a source of inaccuracies. From a given point on the receiver, we are separately estimating occlusion from several micro-patches, and adding these occlusion values together. We have identified three potential sources of error in our algorithm:

- We are only computing the shadow of the discretized occluder, not the shadow of the actual occluder. This source of error will be analyzed in Section 5.1.1.
- The reprojections of the micro-patches on the light source may overlap or be disjoined. This cause of error will be analyzed in Section 5.1.2.
- We are adding many small values (the occlusion from each micro-patch) to form a large value (the occlusion from the entire occluder). If the micro-patches are too small, we run into numerical accuracy issues, especially with floating-point numbers expressed on 16 bits. This cause of error will be analyzed in Section 5.1.3.

5.1.1. Discretization error

Our algorithm computes the shadow of the discretized occluder, not the shadow of the actual occluder. The discretized occluder corresponds to the part of the occluder that is visible from the camera used to compute the depth buffers, usually the center of the light source. Although we reproject each micro-patch of the discretized occluder onto the area light source, we are missing the parts of the occluder that are not visible from the shadow map camera but are still visible from some points of the area light source. This is a limitation that is frequent in real-time soft shadow algorithms [HLHS03], especially algorithms relying on the silhouette of the occluder as computed from a single point [WH03, CD03, AAM03].

We also use a discrete representation based on the shadow map, not a continuous representation of the occluder. For each pixel of the shadow map, we are potentially overestimating or underestimating the actual occluder by at most half a pixel.

If the occluder has one or more edges aligned with the edges of the shadow map, these discretization errors are of the same sign over the edge, and add themselves; the worst case scenario is a square aligned with the axis of the shadow map.

For more practical occluders the discretization errors on neighboring micro-patches compensate: some of the micro-patches overestimate the occluder while others underestimate it.

5.1.2. Overlapping reprojections

At any given point on the receiver, the parts of the light source that are occluded by two neighboring micro-patches should be joined exactly for our algorithm to compute the exact percentage of occlusion on the light source. This is typically not the case, and these parts may overlap or there may be a gap between them (Fig. 12). The amount of overlap (or gap) between the occluded parts of the light source depends on the relative positions of the light source, the occluding micro-patches and the receiver

If we consider the 2D equivalent of this problem (Fig. 13), with two patches separated by δh and at a distance z_O from the light source, with the receiver being at a distance z_R from the light source, there is a point P_0 on the receiver where there is no overlap between the occluded parts. As we move away from this point, the overlap increases. For a point at a

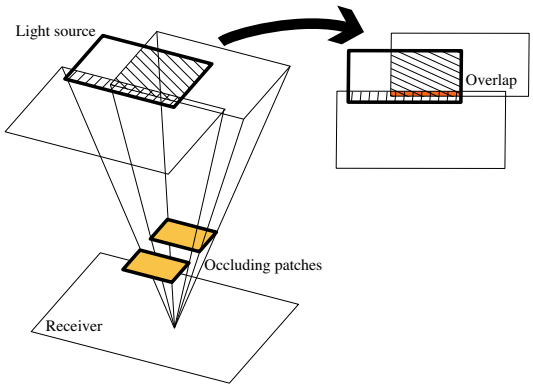


Figure 12: *The reprojection of two neighboring micro-patches may overlap.*

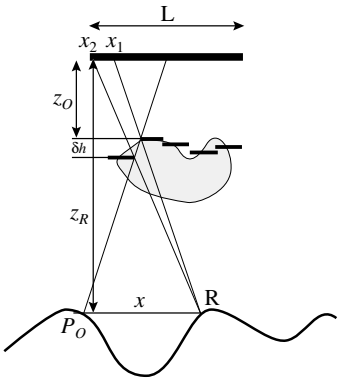


Figure 13: *Computing the extent of overlap or gap between two neighboring micro-patches.*

distance x from P_0, the boundaries of the occluding micro-patches project at abscissa x_1 and x_2; as the occluding micro-patches and the light source lie in parallel planes, we have:

$$\frac{x_1}{x} = \frac{z_O}{z_R - z_O}$$
$$\frac{x_2}{x} = \frac{z_O + \delta h}{z_R - z_O - \delta h}$$

The amount of overlap is therefore:

$$x_2 - x_1 = x\left(\frac{z_O}{z_R - z_O} - \frac{z_O + \delta h}{z_R - z_O - \delta h}\right)$$
$$= -x\frac{z_R \delta h}{(z_R - z_O)(z_R - z_O - \delta h)} \quad (1)$$

x itself is limited, since the occlusion area must fall inside the light source:

$$|x| < \frac{L}{2}\frac{z_R - z_O}{z_O} \quad (2)$$

The amount of overlap is therefore limited by:

$$|x_2 - x_1| < \frac{L}{2}\frac{z_R \delta h}{z_O(z_R - z_O - \delta h)} \quad (3)$$

Equation 3 represents the error our algorithm makes for each pair of micro-patches. The overall error of our algorithm is the sum of the modulus of all these errors, for all the micro-patches projecting on the light source at a given point. This is a conservative estimate, as usually some patches overlap while others present gaps; the actual sum of the occlusion values from all the micro-patches is closer to the real value than what our estimation tells (see Section 5.2).

The theoretical error caused by our algorithm depends on several factors:

Size of the light source: The maximum amount of overlap (Eq. 3) depends directly on the size of the light source. The larger the light source, the larger the error. Our practical experiments confirm this.

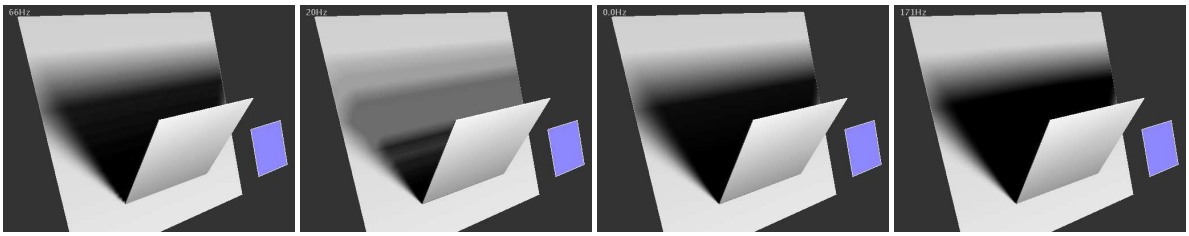
Distance between micro-patches: The maximum amount of overlap (Eq. 3) also depends linearly on δh, the distance in z between neighboring micro-patches. Since δh depends on the discretization of the occluder, the error introduced by our algorithm is related to the resolution of the bitmap: the smaller the resolution of the bitmap, the larger the error. Our practical experiments confirm this, but there is a maximum resolution after which the error does not decrease.

Note that this source of error is related to the *effective* resolution of the bitmap, that is the number of pixels used for discretizing the occluder. If the occluder occupies only a small portion of the bitmap, the effective resolution of the bitmap is much smaller than its actual resolution. Fortunately, the cost of the algorithm is also related to the effective resolution of the bitmap.

Distance to the light source/the receiver: If the occluder touches either the light source or the receiver, the amount of overlap (Eq. 3) goes toward infinity. When the occluder is touching the receiver, the area where the overlap occurs (as defined by equation 2) goes towards 0, thus the error does not appear. When the occluder is touching the receiver, the actual effect depends on the shape of the occluder. In some cases, overlaps and gaps can compensate, resulting in an acceptable shadow.

5.1.3. Floating-point blending accuracy

Our algorithm adds together many small scale occlusion values — the occlusion from each micro-patch — to compute a large scale occlusion value — the occlusion from the complete occluder. This addition is done with the blending ability of the GPU, using blending of floating-point buffers. At the time of writing, blending is only available in hardware for 16-bits floating-point buffers. As a result, we sometimes encounter problems of numerical accuracy.



(a) 128^2 pixels, FP16 blending (66 Hz) (b) 512^2 pixels, FP16 blending (20 Hz) (c) 512^2 pixels, FP32 blending (CPU) (d) Ground truth (CPU)

Figure 14: *Blending with FP16 numbers: if the resolution of the shadow map is too high, numerical issues appear, resulting in wrong shadows. Using higher accuracy for blending removes this issue (here, FP32 blending was done on the CPU).*

Figure 14 shows an example of these problems. Unconventionally, *increasing* the resolution of the shadow map makes these problems *more* likely to appear (for a complete study of floating-point blending accuracy, see appendix A). The best workaround is therefore to use relatively low resolution for the occluder map, such as 128×128 or 256×256. While this may seem a low resolution compared to other shadow map algorithms, our shadow map is focused on the moving occluder (such as a character), not on the entire scene, so 128×128 pixels is usually enough resolution.

We see this is only as a temporary issue that will disappear as soon as hardware FP32 blending becomes available on graphics cards.

5.2. Comparison with ground truth

We ran several tests to experimentally compare the shadows produced by our algorithm with the actual shadows. The reference values were computed using occlusion queries, giving an accurate estimation of the real occlusion of the light source. In this section, we review the practical differences we observed.

5.2.1. Experimentation method

For each image, we computed an error metric as thus: for each pixel in the soft shadow map, we compute the actual occlusion value (using occlusion queries), and the difference with the occlusion value computed using our algorithm. We summed the modulus of the differences, then divided the result by the total number of pixels lying either in the shadow or in the penumbra, averaging the error over the actual soft shadow. We used the number of pixels that are either in shadow or in penumbra and not the total number of pixels in the occluders depth map because the soft shadow can occupy only a small part of the depth map. Dividing by the total number of pixels in the depth map would have underestimated the error.

We have used 3 different scenes (a square plane parallel to the light source, a Buddha model and a Bunny model). These scenes exhibit several interesting features. The Buddha and Bunny are complex models, with folds and creases. The Bunny also has important self-occlusion, and in our scene it is in contact with the ground, providing information on the behavior of our algorithm in that case. The square plane is an illustration of the special case of occluders aligned with the axes of the occluders depth map.

We have tested both the one-pass and the two-pass versions of our algorithm. We selected four separate parameters: the size of the light source, the resolution of the shadow map and moving the occluder, either vertically from the receiver to the light source or laterally with respect to the light source. For each parameter, we plot the variation of the error introduced by our algorithm as a function of the parameter and analyze the results.

5.2.2. Visual comparison with ground truth

Fig. 16 shows a side by side comparison of our algorithm with ground truth. Even though there are slight differences with ground truth, our algorithm exhibits the proper behavior for soft shadows: sharp shadows at places where the object is close to the ground, a large penumbra zone where the object is further away from the receiver. Our algorithm visibly computes both the inner and the outer penumbra of the object.

Looking at the picture of the differences (Fig. 16(d) and 16(g)) between the shadow values computed by our algorithm and the ground truth values, it appears that the differences lie mostly on the silhouette: since our algorithm only computes the soft shadow of the discretized object, as seen from the center of the light source. The actual shape of the soft shadow depends on subtle effects happening at the boundary of the silhouette.

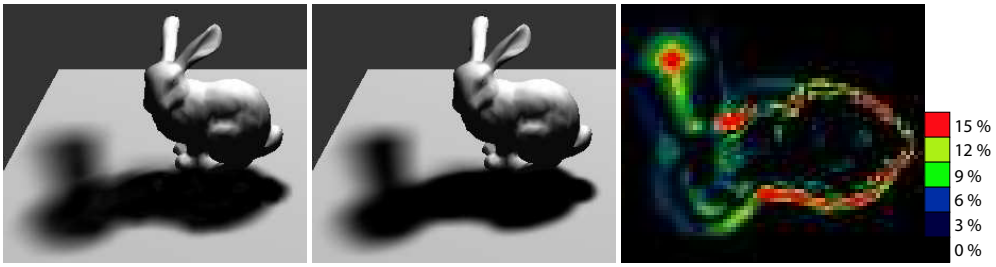
5.2.3. Size of the buffer

Figure 17 shows the average difference between the occlusion values computed with our algorithm and the actual occlusion values for our three test scenes, when changing the resolution of the shadow map. In these figures, the abscissa

(a) Square plane (b) Buddha (c) Bunny

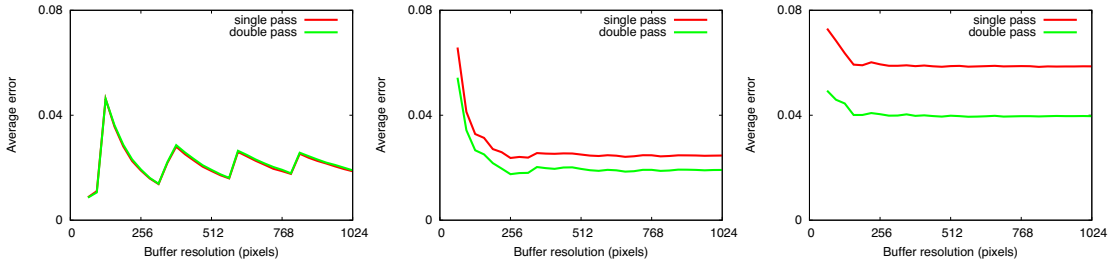
Figure 15: *The test scenes we have used*

(a) Scene view (b) Our algorithm (c) Ground Truth (d) Difference between the occlusion values



(e) Our algorithm (f) Ground Truth (g) Difference between the occlusion values

Figure 16: *Visual comparison of our algorithm with ground truth.*



(a) Square plane (b) Buddha (c) Bunny

Figure 17: *Variation of the error with respect to the resolution of the shadow map*

is the number of pixels for one side of the shadow map, so 128 corresponds to a 128×128 shadow map. For this test, we used non-power of two textures, in order to have enough sampling data. We can make several observations by looking at the data:

Two-pass version: the two-pass version of the algorithm consistently outperforms the single-pass version, always giving more accurate results. The only exception is of course the square plane: since it has no thickness, the single-pass and two-pass version give the same results.

Shadow map Resolution: as expected from the theoretical study (see Section 5.1.2), the error decreases as the resolution of the shadow map increases. What is interesting is that this effect reaches a limit quite rapidly. Roughly, increasing the shadow map resolution above 200 pixels does not bring an improvement in quality. Since the computation costs are related to the size of the shadow map, shadow map sizes of 200×200 pixels are close to optimal.

The fact that the error does not decrease continuously as we increase the resolution of the occluder map is a little surprising at first, but can be explained. It is linked to the silhouette effect. As we have seen in Fig. 16, the error introduced by our algorithm comes from the boundary of the silhouette of the occluder, from parts of the occluder that are not visible from the center of the light source, but visible from other parts of the light source. Increasing the resolution of the shadow map does not solve this problem. The optimal size for the shadow map is related to the size of the light source. As the light source gets larger, we can use smaller bitmaps.

Discretization error: the error curve for the square plane presents many important spikes. Looking at the results, it appears that these spikes correspond to discretization error (see Section 5.1.1). Since the square occluder is aligned with the axis of the shadow map, it magnifies discretization error.

5.2.4. Size of the light source

Figure 18 shows the average difference between the occlusion values computed with our algorithm and the actual occlusion values when we change the size of the light source for our three test scenes. The parameter values range from a point light source (parameter=0.01) to a very large light source, approximately as large as the occluder (parameter=0.2). We used a bitmap of 128×128 pixels for all these tests. We can make several observations by looking at the data:

Point light sources: the beginning of the curves (parameter=0.01) corresponds to a point light source. In that case, the error is quite large. This corresponds to an error of 1, over the entire shadow boundary; as we are computing the shadow of the discretized occluder, we miss the actual shadow boundary, sometimes by as much as half a pixel. The result is a large error, but it occurs only at the shadow boundary.

Light source size: except for the special case of point light sources, the error increases with the size of the light source. This is consistent with our theoretical analysis (see Section 5.1.2).

5.2.5. Occluder moving laterally

Figure 19 shows the average difference between the occlusion values computed with our algorithm and the actual occlusion values, when we move the occluder from left to right under the light source. The parameter corresponds to the position with respect to the center of the light, with 0 meaning that the center of the object is aligned with the center of the light. We used a bitmap of 128×128 for all these tests.

The error is at its minimum when the occluder is roughly under the light source, and increases as the occluder moves laterally. The Buddha and Bunny models are not symmetric, so their curves are slightly asymmetric, and the minimum does not correspond exactly to 0.

5.2.6. Occluder moving vertically

Figure 20 shows the average difference between the occlusion values computed with our algorithm and the actual occlusion values, when we move the occluder vertically. The smallest value of the parameter corresponds to an occluder touching the receiver, and the largest value corresponds to an occluder touching the light source. We used a bitmap of 128×128 for all these tests.

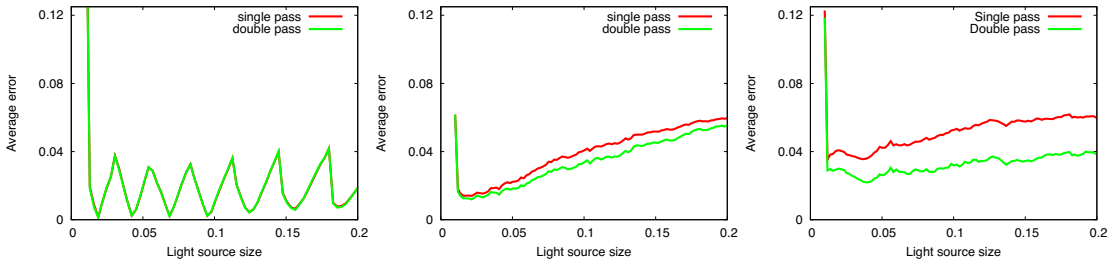
As predicted by the theory, the error increases as the occluder approaches the light source (see Section 5.1.2). For the Bunny, the error becomes quite large when the upper ear touches the light source.

6. Complexity

The main advantages of our algorithm are its rendering speed and its scalability. With a typical setup (a modern PC, an occluder map of 128×128 pixels, a scene between $50,000$ polygons and $300,000$ polygons), we get framerates between 30 and 150 fps. In this section, we study the numerical complexity of our algorithm and its rendering speed. We first conduct a theoretical analysis of the complexity of our algorithm, in Section 6.1, then an experimental analysis, where we test the variation of the rendering speed with respect to several parameters: the size of the shadow map, the number of polygons and the size of the light source (Section 6.2). Finally, in Section 6.3, we compare the complexity of our algorithm with a state-of-the-art algorithm, Soft Shadow Volume [AAM03].

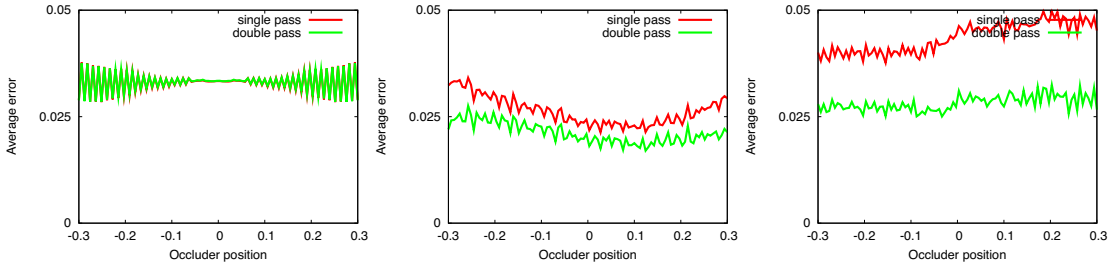
6.1. Theoretical complexity

Our algorithm starts by rendering a shadow map and downloading it into main memory. This preliminary step has a



(a) Square plane (b) Buddha (c) Bunny

Figure 18: *Variation of the error with respect to the size of the light source*



(a) Square plane (b) Buddha (c) Bunny

Figure 19: *Variation of the error with respect to the lateral position of the occluder*

complexity linear with respect to the number of polygons in the scene, and linear with the size of the shadow map, measured in the total number of pixels.

Then, for each pixel of the shadow map corresponding to the occluder, we compute its extent in the occlusion map, and for each pixel of this extent we execute a small fragment program of 11 instructions, including one texture lookup.

The overall complexity of this second step of the algorithm is the number of pixels covered by the occluder, multiplied by the number of pixels covered by the extent for each of them, multiplied by the cost of the fragment program. This second step is executed on the GPU, and benefits from the high-performance and the parallelism of the graphics card.

The worst case situation would be a case where each micro-patch in the shadow map covers a large number of pixels in the soft shadow map. But this situation corresponds to an object with a large penumbra zone, and if we have a large penumbra zone, we can use a lower resolution for the shadow maps. So we can compensate the cost for the algorithm by running it with bitmaps of lower resolution.

Using our algorithm with a large resolution shadow map in a situation of large penumbra results in relatively high computing costs, but a low resolution shadow map would give the same visual results, for a smaller computation time.

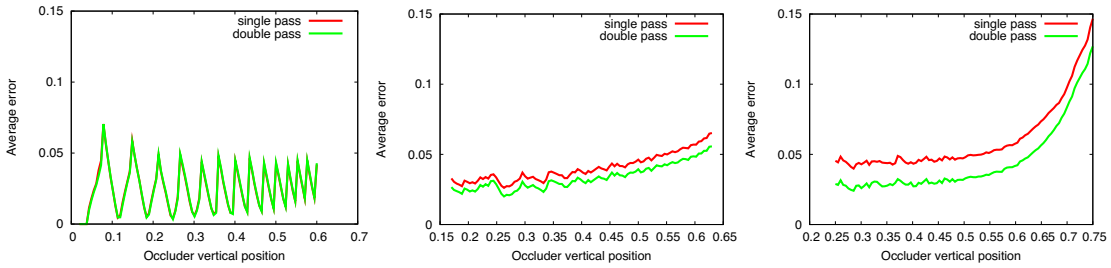
6.2. Experimental complexity

All measurements in this section were conducted on a 2.4 GHz Pentium4 PC with a GeForce 6800 Ultra graphics card. All framerates and rendering times correspond to *observed* framerates, that is the framerate for a user manipulating our system. We are therefore measuring the time it takes to display the scene *and* to compute soft shadows, not just the time it takes to compute soft shadows.

6.2.1. Number of polygons

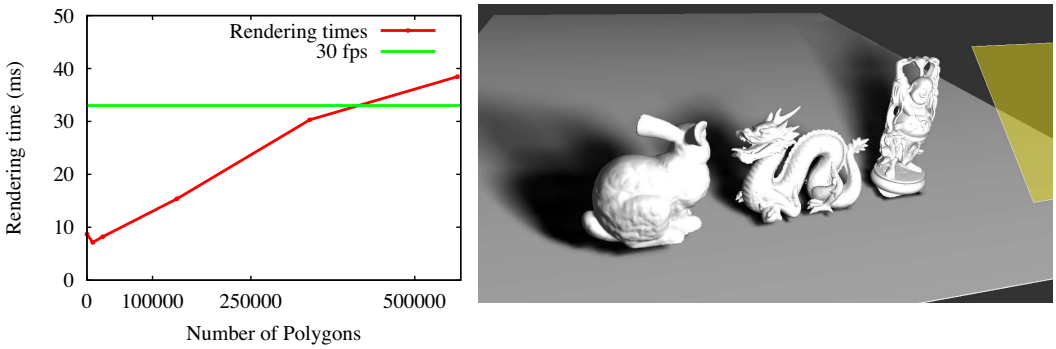
We studied the influence of the polygon count. Fig. 6.2 shows the observed rendering time (in ms) as a function of the polygon count, with a constant occluder map size of 128×128 pixels. The first thing we note is the speed of our algorithm: even on a large scene of $340,000$ polygons, we achieve real-time framerates (more than 30 frames per second). Second, we observe that the rendering time varies linearly with respect to the number of polygons. That was to be expected, as we must render the scene twice (once for the occluder map and once for the actual display), and the time it takes for the graphics card to display a scene varies linearly with respect to the number of polygons. For smaller scenes (less than 10,000 polygons, rendering time below 10 ms), some factors other than the polygon count play a more important role.

Our algorithm exhibits good scaling, and can handle significantly large scenes without incurring a high performance



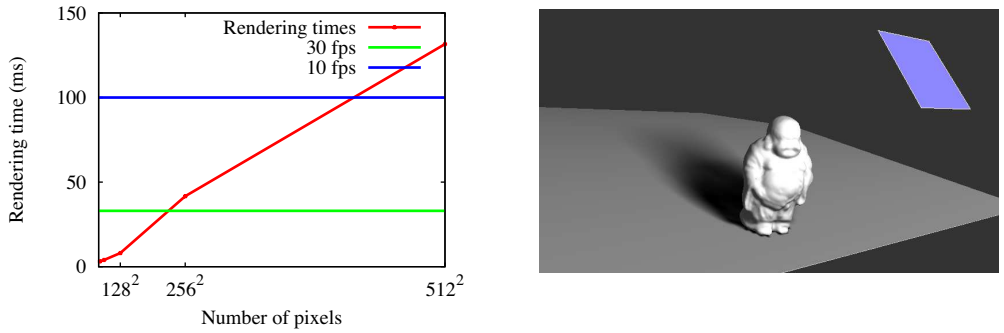
(a) Square plane (b) Buddha (c) Bunny

Figure 20: *Variation of the error with respect to the vertical position of the occluder*



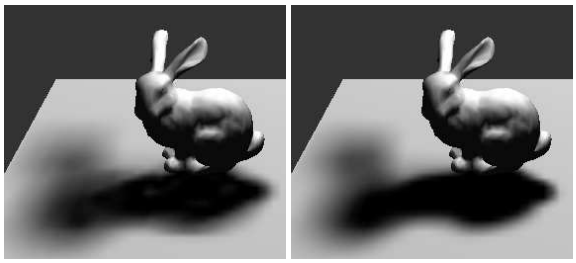
(a) Rendering times (in ms) (b) Our largest test scene (565,203 polygons)

Figure 21: *Influence of polygon count*



(a) Rendering times (in ms) (b) Test scene (24,000 polygons)

Figure 22: *Influence of the size of the occluder map*



(a) Bitmap of 64^2 (184 fps) (b) Ground truth

Figure 23: *Large light sources with small bitmaps*

cost. The maximum size of the scene depends on the requirements of the user.

6.2.2. Size of occluder map

Fig. 6.2.1 shows the observed rendering times (in ms) of our algorithm, on a scene with 24,000 polygons (Fig. 22(b)), when the size of the occluder map changes. We plotted the rendering time as a function of the number of pixels in the occluder map (that is, the *square* of the size of the occluder map) to illustrate the observed linear variation of rendering time with respect to the total number of pixels.

An occluder map of 512^2 pixels gives a rendering time of 150 ms — or 7 fps, too slow for interactive rendering. An occluder map of 128^2 or 256^2 pixels gives a rendering time of 10 to 50 ms, or 20 to 100 fps, fast enough for real-time rendering. For a large penumbra region, an occluder map of 128^2 pixels qualitatively gives a reasonable approximation, as in Fig. 22(b). For a small penumbra region, our algorithm behaves like the classical shadow mapping algorithm and artifacts can appear with a small occluder map of 128^2 pixels; in that case, it is better to use 256^2 pixels.

The fact that the rendering time of our algorithm is proportional to the number of pixels in the occluder map confirms that the bottleneck of our algorithm is its transfer to the CPU. Due to the cost of this transfer, we found that for some scenes it was actually faster to use textures whose dimensions are not a power of 2: if the difference in pixel count is sufficient, the gain in transfer time compensates the losses in rendering time.

6.2.3. Light source size

Another important parameter is the size of the light source, compared to the size of the scene itself. A large light source results in a large penumbra region for each micro-patch, resulting in more pixels of the soft shadow map covered, and a larger computational cost. Fig. 24(a) shows the observed framerate as a function of the size of the light source. We did the tests with several bitmap resolutions (256^2, 128^2, 64^2). Fig. 24(b) shows the error as a function of the size of the light source, for the same bitmap resolutions.

As you can see from Fig. 24(a), the rendering time increases with the size of the light source. What is interesting is the error introduced by our algorithm (see Fig. 24(b)). The error logically increases with the size of the light source, and for small light sources, larger bitmaps result in more accurate images. But for large light sources, a smaller bitmap will give a soft shadow of similar quality. A visual comparison of the soft shadows with a small bitmap and ground truth shows the small bitmap gives a very acceptable soft shadow (see Fig. 23).

This effect was observed by previous researchers: as the light source becomes larger, the features in the soft shadow become blurrier, hence they can be modeled accurately with a smaller bitmap.

6.3. Comparison with Soft-Shadow Volumes

Finally, we performed a comparison with a state-of-the art algorithm for computing soft shadows, the Soft-Shadow Volumes by Assarsson and Akenine-Möller [AAM03].

Fig. 25 shows the same scene, with soft shadows, computed by both algorithms. We ran the tests with a varying number of jeeps, to test how both algorithms scale with respect to the number of polygons. Fig. 25(c) shows the rendering times as a function of the number of polygons for both algorithms. These figures were computed using a window of 512×512 pixels for both algorithms, and with the two-pass version of our algorithm, with an occluder map resolution of 210×210.

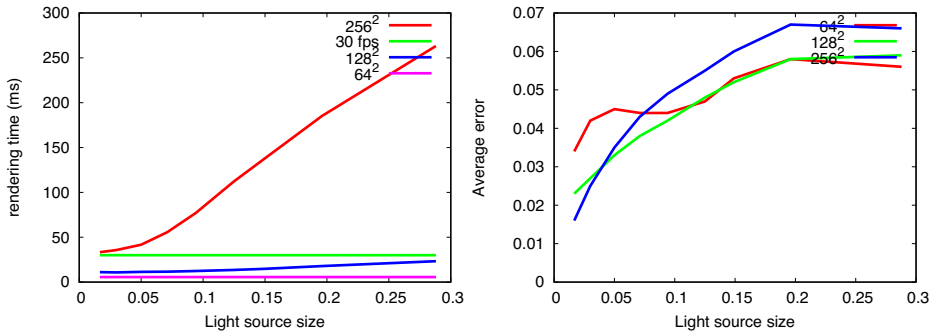
Our algorithm scales better with respect to the number of polygons. On the other hand, soft shadow volumes provide a better looking shadow (see Fig. 25(b)), closer to the actual truth.

It is important to remember that the rendering time for the Soft- Shadow Volumes algorithm varies with the number of screen pixels covered by the penumbra region. If the viewpoint is close to a large penumbra region, the rendering time becomes much larger. The figures we used for this comparison correspond to an observer walking around the scene (as in Fig. 25(b)).

7. Conclusion and Future Directions

In this paper, we have presented a new algorithm for computing soft shadows in real-time on dynamic scenes. Our algorithm is based on the shadow mapping algorithm, and is entirely image-based. As such, it benefits from the advantages of image-based algorithms, especially speed.

The largest advantage of our algorithm is its high framerate, hence there remains plenty of computational power available for performing other tasks, such as interacting with the user or performing non-graphics processing such as physics computations within game engines. Possibly the



(a) Rendering time (b) Average error

Figure 24: *Changing the size of the light source (floating bunny scene)*

(a) Soft Shadow Maps (b) Soft Shadow Volumes (c) Rendering times

Figure 25: *Comparison with Soft-Shadow Volumes*

largest limitation of our algorithm is the fact that it does not compute self-occlusion and it requires a separation between occluders and receivers. We know that this limitation is very important, and we plan to remove it in future work, possibly by using layered depth images.

An important aspect of our algorithm is that we can use low-resolution shadow maps in places with a large penumbra, even though we still need higher resolution shadow maps for places with small penumbra, for example close to the contact between the occluder and the receiver. An obvious improvement to our algorithm would be the ability to use hierarchical shadow maps, switching resolutions depending on the shadow being computed. This work could also be combined with perspective-corrected shadow maps [SD02, WSP04, MT04, CG04], in order to have higher resolution in places with sharp shadows close to the viewpoint.

In its current form, our algorithm still requires a transfer of the occluder map from the GPU to the main memory, and a loop, on the CPU, over all the pixels in the occluder map. We would like to design a GPU only implementation of our algorithm, using the future render-to-vertex-buffer capabilities.

8. Acknowledgments

Most of this work was conducted while Charles Hansen was on sabbatical leave from the University of Utah, and was a visiting professor at ARTIS/GRAVIR IMAG, partly funded by INPG and INRIA.

The authors gratefully thank Ulf Assarsson for doing the tests for the comparison with the soft-shadow volumes algorithm.

The Stanford Bunny, Happy Buddha and dragon 3D models appearing in this paper and in the accompanying video were digitized and kindly provided by the Stanford University Computer Graphics Laboratory.
The smaller Buddha 3D model appearing in this paper was digitized and kindly provided by Inspeck.
The Jeep 3D model appearing in this paper was designed and kindly provided by Psionic.
The horse 3D model appearing in the accompanying video was digitized by Cyberware, Inc., and was kindly provided by the Georgia Tech. "Large Geometric Models Archive".
The skeleton foot 3D model appearing in the accompanying video was digitized and kindly provided by Viewpoint Datalabs Intl.

References

[AAM03] ASSARSSON U., AKENINE-MÖLLER T.: A geometry-based soft shadow volume algorithm using

graphics hardware. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2003) 22*, 3 (2003), 511–520.

[AHT04] ARVO J., HIRVIKORPI M., TYYSTJÄRVI J.: Approximate soft shadows using image-space flood-fill algorithm. *Computer Graphics Forum (Proc. of Eurographics 2004) 23*, 3 (2004), 271–280.

[AMH02] AKENINE-MÖLLER T., HAINES E.: *Real-Time Rendering*, 2nd ed. A. K. Peters, 2002.

[BS02] BRABEC S., SEIDEL H.-P.: Single sample soft shadows using depth maps. In *Graphics Interface* (2002).

[CD03] CHAN E., DURAND F.: Rendering fake soft shadows with smoothies. In *Rendering Techniques 2003 (Proc. of the Eurographics Symposium on Rendering)* (2003), pp. 208–218.

[CD04] CHAN E., DURAND F.: An efficient hybrid shadow rendering algorithm. In *Rendering Techniques 2004 (Proc. of the Eurographics Symposium on Rendering)* (2004), pp. 185–195.

[CG04] CHONG H., GORTLER S. J.: A lixel for every pixel. In *Rendering Techniques 2004 (Proc. of the Eurographics Symposium on Rendering)* (2004), pp. 167–172.

[Cro77] CROW F. C.: Shadow algorithms for computer graphics. *Computer Graphics (Proc. of SIGGRAPH '77) 11*, 2 (1977), 242–248.

[DF94] DRETTAKIS G., FIUME E.: A fast shadow algorithm for area light sources using backprojection. In *SIGGRAPH '94* (1994), pp. 223–230.

[GBP06] GUENNEBAUD G., BARTHE L., PAULIN M.: Real-time soft shadow mapping by backprojection. In *Rendering Techniques 2006 (Proc. of the Eurographics Symposium on Rendering)* (2006).

[HLHS03] HASENFRATZ J.-M., LAPIERRE M., HOLZSCHUCH N., SILLION F.: A survey of real-time soft shadows algorithms. *Computer Graphics Forum 22*, 4 (2003), 753–774.

[KD03] KIRSCH F., DOELLNER J.: Real-time soft shadows using a single light sample. *Journal of WSCG (Winter School on Computer Graphics) 11*, 1 (2003).

[KMK97] KERSTEN D., MAMASSIAN P., KNILL D. C.: Moving cast shadows and the perception of relative depth. *Perception 26*, 2 (1997), 171–192.

[LWGM04] LLOYD B., WENDT J., GOVINDARAJU N., MANOCHA D.: Cc shadow volumes. In *Rendering Techniques 2004 (Proc. of the Eurographics Symposium on Rendering)* (2004), pp. 197–205.

[McC00] MCCOOL M. D.: Shadow volume reconstruction from depth maps. *ACM Transactions on Graphics 19*, 1 (2000), 1–26.

[MT04] MARTIN T., TAN T.-S.: Anti-aliasing and continuity with trapezoidal shadow maps. In *Rendering Techniques 2004 (Proc. of the Eurographics Symposium on Rendering)* (2004), pp. 153–160.

[SD02] STAMMINGER M., DRETTAKIS G.: Perspective shadow maps. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2002) 21*, 3 (2002), 557–562.

[SS98] SOLER C., SILLION F. X.: Fast calculation of soft shadow textures using convolution. In *SIGGRAPH '98* (1998), pp. 321–332.

[Wan92] WANGER L.: The effect of shadow quality on the perception of spatial relationships in computer generated imagery. In *Symposium on Interactive 3D Graphics* (1992), pp. 39–42.

[WFG92] WANGER L., FERWERDA J. A., GREENBERG D. P.: Perceiving spatial relationships in computer-generated images. *IEEE Computer Graphics and Applications 12*, 3 (1992), 44–58.

[WH03] WYMAN C., HANSEN C.: Penumbra maps: Approximate soft shadows in real-time. In *Rendering Techniques 2003 (Proc. of the Eurographics Symposium on Rendering)* (2003), pp. 202–207.

[Wil78] WILLIAMS L.: Casting curved shadows on curved surfaces. *Computer Graphics (Proc. of SIGGRAPH '78) 12*, 3 (1978), 270–274.

[WPF90] WOO A., POULIN P., FOURNIER A.: A survey of shadow algorithms. *IEEE Computer Graphics & Applications 10*, 6 (1990), 13–32.

[WSP04] WIMMER M., SCHERZER D., PURGATHOFER W.: Light space perspective shadow maps. In *Rendering Techniques 2004 (Proc. of the Eurographics Symposium on Rendering)* (2004), pp. 143–152.

Appendix A: Floating-point blending accuracy

In this section, we review the issues behind the hardware blending accuracy problems we have encountered and propose a temporary fix for these issues.

All the accuracy issues are linked to the fact that hardware blending is, at the time of writing, only available for 16-bits floating point numbers. NVidia graphics hardware stores these floating-point numbers using s10e5 format: one bit of sign, 10 bits of mantissa, 5 bits of exponent, with a bias of 15 for the exponent. The important point for addition is that the mantissa is stored on 10 bits. As a result, adding a large number X and a small number ε will give an inaccurate result if $\varepsilon < 2^{-10}X$:

$$X + \varepsilon = X \quad \text{if} \quad \varepsilon < 2^{-10}X \ \ (\text{inFP16})$$

For example, $2048 + 1 = 2048$ (in FP16 format) and $0.5 + \frac{1}{2049} = 0.5$ (also in FP16 format).

In some cases, the addition of the contribution from all micro-patches will be 1 (meaning complete occlusion of the light source). As a consequence, we can expect numerical

accuracy issues if some micro-patches hide less than 2^{-10} of the light source. Because $32^2 = 2^{10}$, it means that the width of the reprojection of one micro-patch should be larger than $\frac{1}{32}$ of the width of the light source.

This translates easily into conditions for the position of the occluder:

$$\frac{1}{z_O} < \frac{1}{z_R} + \frac{64 \tan \alpha}{NL} \tag{4}$$

where L is the width of the light source, N is the resolution of the bitmap, α is the half-angle of the camera used to generate the shadow map, z_O is the distance between the light source and the occluder and z_R is the distance between the light source and the receiver.

Bitmap resolution: The most important thing is that increasing N makes this error *more* likely to appear. This explains why using a bitmap of 512×512 pixels we see a poor looking shadow, while the 128×128 bitmap gives the correct shadow (see Fig. 14).

Light source size: In equation 4, the size of the light source appears in a product with the resolution of the bitmap. If the light source is large, the bitmap must be low resolution in order to avoid FP16 blending errors. Fortunately, a large light source means a large penumbra for most occluders, so a low resolution bitmap might be enough for these penumbra effects.

Occluder position: As the occluder moves closer to the receiver, the likeliness of blending errors gets lower.

Camera half-angle: Similarly, increasing the camera half-angle improves the FP16 blending accuracy.

Basically, all these conditions amount to the same thing: using less pixels to describe the occluder in the shadow map. While this improves the FP16 blending accuracy, it obviously degrades the discretization of the occluder and also increases the overlapping between reprojections of neighboring pixels.

In our experiments (see Fig. 14) the blending accuracy problem appears very often when the resolution of the shadow map is larger than 512×512, sometimes with a shadow map resolution of 256×256 and very rarely with a shadow map resolution of 128×128.

The problem will disappear when hardware blending will become available on higher accuracy floating point numbers. FP32 have a mantissa of 23 bits, allowing the use of micro-patches that block less than 2^{-23} of the light source, meaning that the width of the back-projection of the micro-patch should be at least larger that 2^{-11} than the width of the light source (64 times smaller than the current threshold). Compared with the current method, it would allow the use of shadow maps with a resolution above 4096×4096.

With FP16 blending only, the best solution is to use a hierarchical shadow map for soft-shadow computations, as was suggested by Guennebaud *et al.* [GBP06]: the low resolution shadow map would be used for large penumbra regions, and the high-resolution shadow map for areas with hard shadows, *e.g.* when the occluder and the receiver are in contact.

4.7.4 Fast Precomputed Ambient Occlusion for Proximity Shadows (JGT 2006)

Auteurs : Mattias M , Fredrik M , Ulf A et Nicolas H
Journal : *Journal of Graphics Tools* (accepté, à paraître)
Date : accepté en octobre 2006.

Vol. [VOL], No. [ISS]: 1–13

Fast Precomputed Ambient Occlusion for Proximity Shadows

Mattias Malmer and Fredrik Malmer
Syndicate

Ulf Assarsson
Chalmers University of Technology
Nicolas Holzschuch
ARTIS-GRAVIR/IMAG INRIA

Abstract. Ambient occlusion is used widely for improving the realism of real-time lighting simulations. We present a new method for precomputed ambient occlusion, where we store and retrieve unprocessed ambient occlusion values in a 3D grid. Our method is very easy to implement, has a reasonable memory cost, and the rendering time is independent from the complexity of the occluder or the receiving scene. This makes the algorithm highly suitable for games and other real-time applications.

1. Introduction

An "ambient term" is commonly used in illumination simulations to account for the light that remains after secondary reflections. This ambient term illuminates areas of the scene that would not otherwise receive any light. In first implementations, ambient light was an uniform light, illuminating all points on all objects, regardless of their shape or position, flattening their features, giving them an unnatural look.

To counter this effect, *ambient occlusion* was introduced by [Zhukov et al. 98]. By computing the *accessibility* to ambient lighting, and using it to modulate the effects, they achieve a much better look. Ambient occlusion is widely used in special ef-

1086-7651/06 $0.50 per page

Figure 1. Example of contact shadows. This scene runs at 800 fps.

fects for motion pictures [Landis 02] and for illumination simulations in commercial software [Christensen 02, Christensen 03].

Ambient occlusion also results in objects having *contact shadows*: for two close objects, ambient occlusion alone creates a shadow of one object onto the other (see Figure 1).

For offline rendering, ambient occlusion is usually precomputed at each vertex of the model, and stored either as vertex information or into a texture. For real-time rendering, recent work [Zhou et al. 05, Kontkanen and Laine 05] suggest storing ambient occlusion as a field around moving objects, and projecting it onto the scene as the object moves. These methods provide important visual cues for the spatial position of the moving objects, in real-time, at the expense of extra storage. They pre-process ambient occlusion, expressing it as a function of space whose parameters are stored in a 2D texture wrapped around the object. In contrast, our method stores these *un-processed*, in a 3D grid attached to the object. The benefits are numerous:

- faster run-time computations, and very low impact on the GPU, with a computational cost being as low as 5 fragment shader instructions per pixel,
- very easy to implement, just by rendering one cube per shadow casting object,
- shorter pre-computation time,
- inter-object occlusion has high quality even for receiving points inside the occluding object's convex hull,
- handles both self-occlusion and inter-object occlusion in the same rendering pass.
- easy to combine with indirect lighting stored in environment maps.

The obvious drawback should be the memory cost, since our method's memory costs are in $O(n^3)$, instead of $O(n^2)$. But since ambient occlusion is a low frequency

phenomenon, in only needs a low resolution sampling. In [Kontkanen and Laine 05], as in our own work, a texture size of $n = 32$ is sufficient. And since we are storing a single component per texel, instead of several function coefficients, the overall memory cost of our method is comparable to theirs. For a texture size of 32 pixels, [Kontkanen and Laine 05] report a memory cost of 100 Kb for each unique moving object. For the same resolution, the memory cost of our algorithm is of 32 Kb if we only store ambient occlusion, and of 128 Kb if we also store the average occluded direction.

2. Background

Ambient occlusion was first introduced by [Zhukov et al. 98]. In modern implementations [Landis 02, Christensen 02, Christensen 03, Pharr and Green 04, Bunnell 05, Kontkanen and Laine 05], it is defined as the percentage of ambient light blocked by geometry close to point $\boldsymbol{p}$:

$$ao(\boldsymbol{p}) = \frac{1}{\pi}\int_{\Omega}(1 - V(\omega))\lfloor \boldsymbol{n}\cdot\omega\rfloor \,\mathrm{d}\omega \tag{1}$$

Occlusion values are weighted by the cosine of the angle of the occluded direction with the normal $\boldsymbol{n}$: occluders that are closer to the direction $\boldsymbol{n}$ contribute more, and occluders closer to the horizon contribute less, corresponding to the importance of each direction in terms of received lighting. Ambient occlusion is computed as a percentage, with values between 0 and 1, hence the $\frac{1}{\pi}$ normalization factor.

Most recent algorithms [Bunnell 05, Kontkanen and Laine 05] also store the average occluded direction, using it to modulate the lighting, depending on the normal at the receiving point and the environment.

[Greger et al. 98] also used a regular grid to store illumination values, but their grid was attached to the scene, not to the object. [Sloan et al. 02] attached radiance transfer values to a moving object, using it to recompute the effects of the moving object on the environment.

3. Algorithm

3.1. Description of the algorithm

Our algorithm inserts itself in a classical framework where other shading information, such as direct lighting, shadows, etc. are computed in separate rendering passes. One rendering pass will be used to compute ambient lighting, combined with ambient occlusion. We assume we have a solid object moving through a 3D scene, and we want to compute ambient occlusion caused by this object.

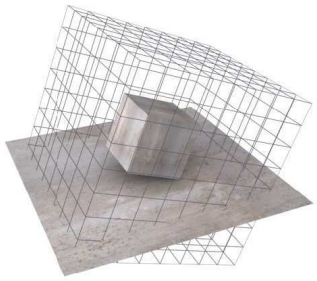


Figure 2. We construct a grid around the object. At the center of each grid element, we compute a spherical occlusion sample. At runtime, this information is used to apply shadows on receiving objects.

Our algorithm can either be used with classical shading, or with deferred shading. In the latter case, the world-space position and the normal of all rendered pixels is readily available. In the former, this information must be stored in a texture, using the information from previous rendering passes.

Precomputation: The percentage of occlusion from the object is precomputed at every point of a 3D grid surrounding the object (see Figure 2). This grid is stored as a 3D texture, linked to the object.

Runtime:

- render world space position and normals of all shadow receivers in the scene, including occluders.
- For each occluder:
  1. render the back faces of the occluder's grid (depth-testing is disabled).
  2. for every pixel accessed, execute a fragment program:
     (a) retrieve the world space position of the pixel.
     (b) convert this world space position to voxel position in the grid, passed as a 3D texture
     (c) retrieve ambient occlusion value in the grid, using linear interpolation.
  3. Ambient occlusion values a from each occluder are blended in the frame buffer using multiplicative blending with $1 - a$.

The entire computation is thus done in just one extra rendering pass. We used the back faces of the occluder's grid, because it is unlikely that they are clipped by the far clipping plane; using the front faces could result in artifacts if they are clipped by the front clipping plane.

3.2. Shading surfaces with ambient occlusion alone

The ambient occlusion values we have stored correspond to the occlusion caused by the occluder itself:

$$ao'(\boldsymbol{p}) = \frac{1}{4\pi}\int_{\Omega}(1 - V(\omega))\,\mathrm{d}\omega \tag{2}$$

that is, the percentage of the entire sphere of directions that is occluded. When we apply these occlusion values at a receiving surface, during rendering, the occlusion only happens over a half-space, since the receiver itself is occluding the other half-space. To account for this occlusion, we scale the occlusion value by a factor 2. This shading does not take into account the position of the occluder with respect to the normal of the receiver. It is an approximation, but we found it performs quite well in several cases (see Figure 1). It is also extremely cheap in both memory and computation time, as the value extracted from the 3D texture is used directly.

We use the following fragment program (using Cg notation):

```
1 float4 p_world = texRECT(PositionTex, p_screen)
2 float3 p_grid = mul(M_WorldToGrid, p_world)
3 out.color.w = 1 - tex3D(GridTexture, p_grid)
```

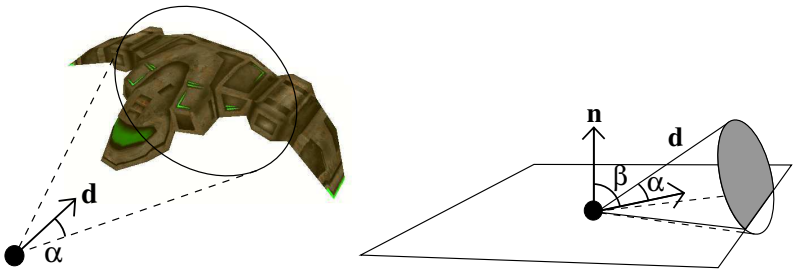
There are two important drawbacks with this simple approximation: first, the influence of the occluder is also visible where it should not, such as a character moving on the other side of a wall; second, handling self-occlusion requires a specific treatment, with a second pass and a separate grid of values.

3.3. Shading surfaces with ambient occlusion and average occluded direction

For more accurate ambient occlusion effects, we also store the average occluded direction. That is equivalent to storing the set of occluded directions as a cone (see Figure 3). The cone is defined by its axis ($\boldsymbol{d}$) and the percentage of occlusion a (linked to its aperture angle α). Axis and percentage of occlusion are precomputed for all moving objects and stored on the sample points of the grid, in an RGBA texture, with the cone axis $\boldsymbol{d}$ stored in the RGB-channels and occlusion value a stored in the A-channel.

3.3.1. Accounting for surface normal of receiver

In order to compute the percentage of ambient occlusion caused by the moving occluder, we clip the cone of occluded directions by the tangent surface to the receiver (see Figure 3(b)). The percentage of effectively occluded directions is a function of two parameters: the angle between the direction of the cone and the normal at the receiving surface (β), and the percentage of occlusion of the cone (a). We precompute this percentage and store it in a lookup table T_{clip}. The lookup table also stores



(a) The cone is defined by its direction $\boldsymbol{d}$ and its aperture α.

(b) The cone is clipped by the tangent plane to the receiver to give the ambient occlusion value.

Figure 3. Ambient occlusion is stored as a cone.

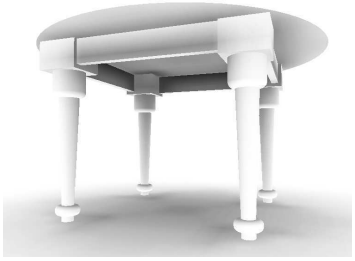


Figure 4. Ambient occlusion computed with our algorithm that accounts for the surface normal of the receiver and the direction of occlusion.

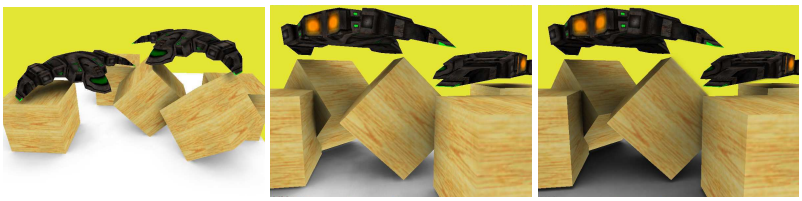
the effect of the diffuse BRDF (the cosine of the angle between the normal and the direction). For simplicity, we access the lookup table using $\cos\beta$.

We now use the following fragment program:

```
1 float4 p_world = texRECT(PositionTex, p_screen)
2 float3 p_grid  = mul(M_WorldToGrid, p_world)
3 float4 {d_grid, a} = tex3D(GridTexture, p_grid)
4 float3 d_world = mul(M_GridToWorld, d_grid)
5 float3 n = texRECT(NormalTex, p_screen)
6 float  cosβ = dot(d_world, n)
7 float  AO = texRECT(T_clip, float2(a, cosβ))
8 out.color.w = 1-AO
```

This code translates to 16 shader assembler instructions. Figure 4 and 5 were rendered using this method, with a grid resolution of 32^3.

Compared to storing only ambient occlusion values, using the average occluded direction has the advantage that results are more accurate and self-occlusion is naturally treated.



(a) (b) (c) Ground Truth

Figure 5. Ambient occlusion values, accounting for the normal of the occluder and the direction of occlusion (135 to 175 fps).

(a) Gouraud shading (b) Blending occlusion from multiple occluders (c) Ground truth

Figure 6. Checking the accuracy of our blending method: comparison of Ambient Occlusion values computed with ground truth.

3.3.2. Combining occlusion from several occluders

When we have several moving occluders in the scene, we compute occlusion values from each moving occluder, and merge these values together. The easiest method to do this is to use OpenGL blending operation: in a single rendering pass, we render the occlusion values for all the moving occluders. The occlusion value computed for the current occluder is blended to the color buffer, multiplicatively modulating it with $(1 - a)$.

[Kontkanen and Laine 05] show that modulating with $(1 - a_i)$, for all occluders i, is statistically the best guess. Our experiences also show that it gives very satisfying results for almost all scenes. This method has the added advantage of being very simple to implement: the combined occlusion value for one pixel is independent from the order in which the occluders are treated for this pixel, so we only need one rendering pass.

Each occluder is rendered sequentially, using our ambient occlusion fragment program, into an occlusion buffer. The cone axes are stored in the RGB channels and the occlusion value is stored in the alpha channel. Occlusion values are blended multiplicatively and cone axes are blended additively, weighted by their respective solid

angle:

$$\begin{aligned} \alpha_R &= (1-\alpha_A)(1-\alpha_B) \\ \boldsymbol{d}_R &= \alpha_A \boldsymbol{d}_A + \alpha_B \boldsymbol{d}_B \end{aligned}$$

This is achieved using **glBlendFuncSeparate** in OpenGL. See Figures 5 and 6 for a comparison of blending values from several occluders with the ground truth values, computed with distributed ray-tracing: The two pictures exhibit the same important features, although our method is noticeably lighter (see also Section 4.3).

We have designed a more advanced method for blending the occlusions between two cones, taking into account the respective positions of the cones and their aperture (see the supplemental materials), but our experiments show that the technique described here generally gives similar results, runs faster and is easier to implement.

3.3.3. Illumination from an environment map

The occlusion cones can also be used to approximate the incoming lighting from an environment map, as suggested by [Pharr and Green 04]. For each pixel, we first compute the lighting due to the environment map, using the surface normal for Lambertian surfaces, or using the reflected cone for glossy objects. Then, we subtract from this lighting the illumination corresponding to the cone of occluded directions.

We only need to change the last step of blending the color buffer and occlusion buffer. Each shadow receiving pixel is rendered using the following code:

PSEUDO CODE

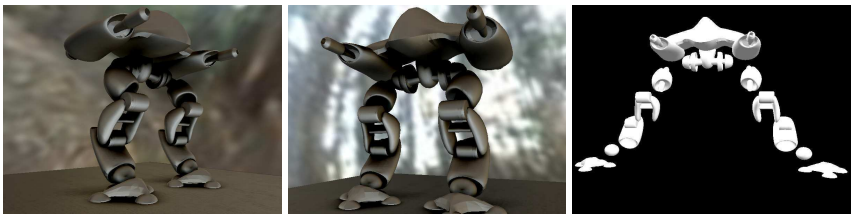
```
1  Read cone d, α from occlusion buffer
2  Read normal from normal buffer
3  Compute mipmap level from cone angle α
4  A = EnvMap(d, α). i.e., lookup occluded light within the cone
5  B = AmbientLighting(normal). i.e., lookup the incoming light due to the environment map.
6  return B-A.
```

In order to use large filter sizes, we used lat-long maps. It is also possible to use cube maps with a specific tool for mip-mapping across texture seams [Scheuermann and Isidoro 06].

3.4. Details of the algorithm

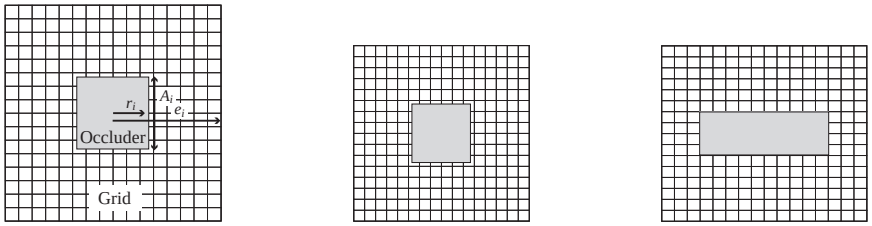
3.4.1. Spatial extent of the grid

An important parameter of our algorithm is the *spatial extent* of the grid. If the grid is too large, we run the risk of under-sampling the variations of ambient occlusion,



(a) (b) (c) Robot parts

Figure 7. Using ambient occlusion with environment lighting. These images are rendered in roughly 85 fps.



(a) Our notations (b) Cubic object (c) Elongated object. Notice the grid is *thinner* along the longer axis

Figure 8. Our notations for computing the optimal grid extent based on the bounding-box of the occluder (a), and optimal grid extents computed with $\epsilon = 0.1$ (b-c).

otherwise we have to increase the resolution, thus increasing the memory cost. If the grid is too small, we would miss some of the effects of ambient occlusion.

To compute the optimal spatial extent of the grid, we use the bounding box of the occluder. This bounding box has three natural axes, with dimension $2r_i$ on each axis, and a projected area of A_i perpendicular to axis i (see Figure 8(a)).

Along the i axis, the ambient occlusion of the bounding box is approximately:

$$a_i \approx \frac{1}{4\pi} \frac{A_i}{(d - r_i)^2} \tag{3}$$

where d is the distance to the center of the bounding box.

If we decide to neglect occlusion values smaller than ϵ, we find that the spatial extent e_i of the grid along axis i should be:

$$e_i = r_i + \sqrt{\frac{A_i}{4\pi\epsilon}} \tag{4}$$

We take $\epsilon = 0.1$, giving an extent of $e_i \approx 3r_i$ for a cubic bounding box (see Figure 8(b)). For elongated objects, equation 4 gives an elongated shape to the grid,

(a) Using raw values, discontinuities can appear

(b) After re-scaling, ambient occlusion blends continuously

Figure 9. We need to re-scale occlusion values inside the grid to avoid visible artifacts.

following the shape of the object, but with the grid being thinner on the longer axes of the object (see Figure 8(c)).

We use a relatively large epsilon value (0.1), resulting in a small spatial extent. As a consequence, there can be visible discontinuities on the boundary of the grid (see Figure 9(a)). To remove these discontinuities, we re-scale the values inside the grid so that the largest value at the boundary is 0. If the largest value on the boundary of the grid is V_M, each cell of the grid is rescaled so that its new value V' is:

$$V' = \begin{cases} V & \text{if} \quad V > 0.3 \\ 0.3\frac{V - V_M}{0.3 - V_M} & \text{if} \quad V \leq 0.3 \end{cases}$$

The effect of this scaling can be seen on Figure 9(b). The overall aspect of ambient occlusion is kept, while the contact shadow ends continuously on the border of the grid.

3.4.2. Voxels inside the occluder

Sampling points that are inside the occluder will have occlusion values of 1, expressing that they are completely hidden. As we interpolate values on the grid, a point located on the boundary of the occluder will often have non-correct values. To counter this problem, we modify the values inside the occluder (which are never used) so that the interpolated values on the surface are as correct as possible.

A simple but quite effective automatic way to do this is: for all grid cells where occlusion value is 1, replace this value by an average of the surrounding grid cells that have an occlusion value smaller than 1. This algorithm was used on all the figures in this paper.

4. Results

All timings and figures in this paper were computed on a Pentium 4, running at 2.8 GHz, with a NVidia GeForce 7800GTX, using a grid resolution of 32^3.

4.1. Timing results

The strongest point of our method is its performance: adding ambient occlusion to any scene increases the rendering time by $\approx$ 0.9 ms for each occluder. In our experiments, this value stayed the same regardless of the complexity of the scene or of the occluder. We can render scenes with 40 different occluders at nearly 30 fps.

The cost of the method depends on the number of pixels covered by the occluder's grid, so the cost of our algorithm decreases nicely for occluders that are far from the viewpoint, providing an automatic level-of-detail.

The value of 0.9 ms corresponds to the typical situation, visible in all the pictures in this paper: the occluder has a reasonable size, neither too small nor too large, compared to the size of the viewport.

4.2. Memory costs

Precomputed values for ambient occlusion are stored in a 3D texture, with a memory cost of $O(n^3)$ bytes. With a grid size of 32, the value we have used in all our tests, the memory cost for ambient occlusion values is 32 Kb per channel. Thus, storing just the ambient occlusion value gives a memory cost of 32 Kb. Adding the average occluded direction requires three extra channels, bringing the complete memory cost to 128 Kb.

4.3. Comparison with Ground Truth

Figure 5(b)-5(c) and 6(b)-6(c) show a side-by-side comparison between our algorithm and ground truth. Our algorithm has computed all the relevant features of ambient occlusion, including proximity shadows. The main difference is that our algorithm tends to underestimate ambient occlusion.

There are several reasons for this difference: we have limited the spatial influence of each occluder, by using a small grid, and the blending process (see Section 3.3.2) can underestimate the combined occlusion value of several occluders.

While it would be possible to improve the accuracy of our algorithm (using a more accurate blending method and a larger grid), we point out that ambient occlusion methods are approximative by nature. What is important is to show all the

relevant features: proximity shadows and darkening of objects in contact, something our algorithm does.

Acknowledgments. ARTIS is an INRIA research project and a research team in the GRAVIR laboratory, a joint research unit of CNRS, INRIA, INPG and UJF.

This work was started while Ulf Assarsson was a post-doctoral student at the ARTIS research team, funded by INRIA.

The space ship model used in this paper was designed by Max Shelekhov.

References

[Bunnell 05] Michael Bunnell. "Dynamic Ambient Occlusion and Indirect Lighting." In *GPU Gems 2*, edited by Matt Pharr, pp. 223–233. Addison Wesley, 2005.

[Christensen 02] Per H. Christensen. "Note 35: Ambient occlusion, image-based illumination, and global illumination." In *PhotoRealistic RenderMan Application Notes*. Emeryville, CA, USA: Pixar, 2002.

[Christensen 03] Per H. Christensen. "Global Illumination and All That." In *Siggraph 2003 course 9: Renderman, Theory and Practice*, edited by Dana Batall, pp. 31 – 72. ACM Siggraph, 2003.

[Greger et al. 98] G. Greger, P. Shirley, P. M. Hubbard, and D. P. Greenberg. "The Irradiance Volume." *IEEE Computer Graphics and Applications* 18:2 (1998), 32–43.

[Kontkanen and Laine 05] Janne Kontkanen and Samuli Laine. "Ambient Occlusion Fields." In *Symposium on Interactive 3D Graphics and Games*, pp. 41–48, 2005.

[Landis 02] Hayden Landis. "Production Ready Global Illumination." In *Siggraph 2002 course 16: Renderman in Production*, edited by Larry Gritz, pp. 87 – 101. ACM Siggraph, 2002.

[Pharr and Green 04] Matt Pharr and Simon Green. "Ambient Occlusion." In *GPU Gems*, edited by Randima Fernando, pp. 279–292. Addison Wesley, 2004.

[Scheuermann and Isidoro 06] Thorsten Scheuermann and John Isidoro. "Cubemap Filtering with CubeMapGen." In *Game Developer Conference 2006*, 2006.

[Sloan et al. 02] Peter-Pike Sloan, Jan Kautz, and John Snyder. "Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments." *ACM Transactions on Graphics (Proc. of Siggraph 2002)* 21:3 (2002), 527–536.

[Zhou et al. 05] Kun Zhou, Yaohua Hu, Steve Lin, Baining Guo, and Heung-Yeung Shum. "Precomputed Shadow Fields for Dynamic Scenes." *ACM Transactions on Graphics (proceedings of Siggraph 2005)* 24:3.

[Zhukov et al. 98] S. Zhukov, A. Iones, and G. Kronin. "An Ambient Light Illumination Model." In *Rendering Techniques '98 (Proceedings of the 9th EG Workshop on Rendering)*, pp. 45 – 56, 1998.

Web Information:

Two videos, recorded in real-time and demonstrating the effects of pre-computed ambient occlusion on animated scenes are available at:

`http://www.ce.chalmers.se/~uffe/ani.mov`
`http://www.ce.chalmers.se/~uffe/cubedance.mov`

A technique for better accuracy in blending the occlusion from two cones is described in a supplemental material.

Mattias Malmer, Syndicate, Grevgatan 53, 114 58 Stockholm, Sweden.
(www.syndicate.se)

Fredrik Malmer, Syndicate, Grevgatan 53, 114 58 Stockholm, Sweden
(www.syndicate.se)

Ulf Assarsson, Department of Computer Science and Engineering, Chalmers University of Technology, S-412 96 Gothenburg, Sweden.
(uffe@ce.chalmers.se)

Nicolas Holzschuch, ARTIS/GRAVIR IMAG INRIA, INRIA Rhône-Alpes, 655 avenue de l'Europe, Innovallée, 38334 St-Ismier CEDEX, France.
(Nicolas.Holzschuch@inria.fr)

Received [DATE]; accepted [DATE].

4.7.5 Accurate specular reflections in real-time (EG 2006)

Auteurs : David R et Nicolas H
Conférence : Eurographics 2006, Vienne, Autriche. Cet article a également été publié dans *Computer Graphics Forum*, vol. 25, n^{o} 3.
Date : septembre 2006

EUROGRAPHICS 2006 / E. Gröller and L. Szirmay-Kalos
(Guest Editors)

Volume 25 (2006), Number 3

Accurate Specular Reflections in Real-Time

David Roger and Nicolas Holzschuch

ARTIS–GRAVIR† IMAG INRIA

Figure 1: Left: Specular reflections computed with our algorithm. Middle: ray-traced reference. Right: Environment map reflection.

Abstract

Specular reflections provide many important visual cues in our daily environment. They inform us of the shape of objects, of the material they are made of, of their relative positions, etc. Specular reflections on curved objects are usually approximated using environment maps. In this paper, we present a new algorithm for real-time computation of specular reflections on curved objects, based on an exact computation for the reflection of each scene vertex. Our method exhibits all the required parallax effects and can handle arbitrary proximity between the reflector and the reflected objects.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

1. Introduction

Reflections on specular objects are important in our perception of a synthetic 3D scene. They convey important information about the specular reflector itself, conveying its shape and its fabric. They can also give information about the relative spatial positions of objects or the distance between the reflector and the reflected object. Finally, they give information about objects that are not directly visible (see Figure 1).

Real-time computation of specular reflections is usually done using environment mapping. While these techniques perform quite well in a wide variety of cases, they have their shortcomings. They perform best if the reflected object is at a large distance from the reflector, but as the reflected object moves closer to the specular reflector, reflection errors become more visible. The worst case for environment mapping techniques is when the reflector is in contact with the object being reflected, as in Figure 1. Environment mapping technique also suffer from the parallax problem: from all the points on the specular reflector, we are seeing the same side of the reflected objects, even if the specular reflector is large enough to see the different sides of an object.

In this paper, we present a new method for computing specular reflections. Our method is *vertex based:* we com-

† GRAVIR is UMR 5527 GRAVIR, a joint research laboratory of CNRS, INRIA, INPG and UJF.

pute the accurate reflected position of each vertex in the scene, then interpolate between these positions. The advantage of our method is that it is computing the reflection of the object depending on the position on the reflector. We are therefore exhibiting all parallax effects, and we can handle proximity and even contact between the reflector and the reflected objects.

However, our method also has obvious limitations: as it is vertex-based and uses the graphics hardware for linear interpolation between the projections of the vertices, artifacts can appear if the model is not finely tessellated enough. These artifacts can be overcome using either adaptive tessellation or curvilinear interpolation. If the model is finely tesselated, these artifacts are not visible. Our algorithm provides solutions for situations where no convincing solutions existed before.

Our paper is organized as follows: in the next section, we review previous work on real-time computation of specular reflections. Then, in section 3, we present our algorithm for computing vertex-based specular reflections on curved surfaces. In section 4, we present experiments on various scenes and comparisons with existing methods. Finally, in section 5, we conclude and present future directions for research.

2. Previous Works

Ray-tracing has historically been used to compute reflections on specular objects. Despite several advances using either highly parallel computers [WSB01, WBWS01, WSS05] or GPUs [CHH02, PBMH02], ray-tracing is not, currently, available for real-time computations on a standard workstation.

Planar specular reflectors are easy to model, at the cost of a second rendering pass, with a camera placed in the mirror position of the viewpoint [McR96]. Curved reflectors are more complex; the easiest method uses environment mapping [BN76].

Environment mapping computes an image of the scene and maps it on the reflector as if it was located at an infinite distance. The reflection only depends on the direction of the incoming vector from the viewpoint, and can be easily computed in real-time on graphics hardware. Obviously, environment mapping suffers from parallax issues, since the reflection depends on a single image computed from a single point of view. There is also the question of accuracy: since all objects are assumed to be at an infinite distance, their reflection is not necessarily accurate, and the difference becomes larger as the object gets closer to the reflector.

There has been much research to improve the original environment mapping algorithm. To remove the parallax issues, Martin and Popescu [MP04] interpolate between several environment maps. Yu *et al.* [YYM05] used an *environment light-field*, containing all the information of a light field, but organized like an environment map. Both methods remove parallax issues, at the cost of a longer precomputation time. The specular reflector is also restricted, and can only be moved inside the area where the light field or the environment maps were computed. If it is moved outside of this area, the environment light field must be recomputed, a costly step.

Other research have dealt with distance-based reflection. The simplest method is to replace the infinite-radius sphere associated with the environment map by a finite-radius sphere [Bjo04]; the reflection changes with the position of the reflector in the environment, but parallax effects can not be modeled.

More accurate methods use the Z-buffer to compute a distance map along with the environment map. For each pixel of the environment map, they know both its color and the distance to the center of the reflected object. Patow [Pat95] and Kalos *et al.* [SKALP05] used this information to select the proper pixel inside the environment map. Their reflections change depending on the distance between the reflector and the reflected object. Kalos *et al.* [SKALP05] use the GPU for a fast computation of the reflected pixel, and achieve real-time rendering for moderately complex scenes. Still, image based methods are inherently limited to the information included in the original image.

For planar reflectors, the easiest way to compute the reflection is vertex-based, using an alternative camera to compute the image of the scene as reflected by the planar reflector. For curved reflectors, there is no simple rule to tell the position of the reflection of the objects. Even for a finite-radius sphere, the simplest specular reflector, the position of the reflection depends on a 4th-order polynomial.

Mitchell and Hanrahan [MH92] used the equation of the underlying surface to compute the characteristic points in the caustic created by a curved reflector. Ofek [Ofe98] and Ofek and Rappoport [OR98] computed the *explosion map* to find intersected triangles ID based on the reflected vector. Chen and Arvo [CA00b, CA00a] used ray-tracing to compute the reflection of some vertices, then applied perturbation to these reflections to compute the reflection of neighboring vertices.

Estalella *et al.* [EMD*05] computed the reflection of scene vertices on curved specular objects by an iterative method. At each iteration, the position of the reflection of the vertex is modified, using the angles between the normal, the vertex and the viewpoint, in the direction where these angles will follow Descartes' law. They did a fixed number of iterations, and have implemented the method only on the CPU. In a subsequent work, developed concurrently with ours, Estalella *et al.* [EMDT06] extended this work to the GPU, searching the position of the reflection of the vertex in image space.

Our method is comparable to that of Estalella *et al.* [EMD*05, EMDT06], but we use a different refinement

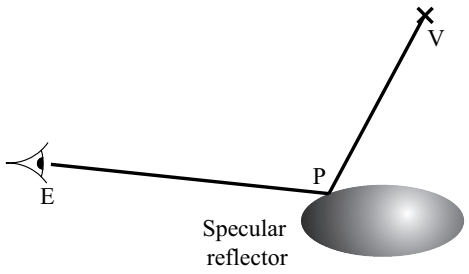


Figure 2: *Finding the reflection of a given vertex*

criterion, keeping geometric bounds on the reflected position for robustness. We use these geometric bounds for adaptive refinement, stopping the iteration as soon as we reach sub-pixel accuracy. In our experience, these two elements are of great importance: in all the scenes we used, we encountered robustness-related issues, especially for reflections at grazing angle. We also noticed that the number of iterations required to reach convergence varies greatly with the position of the reflection.

3. Algorithm

3.1. Principles

Our algorithm is vertex-based: we compute the reflected position of all the scene vertices, then let the graphics hardware interpolate between these vertices and solve visibility issues with a Z-buffer. Our algorithm therefore inserts itself as a replacement for the usual projection of the vertices. Knowing the position of the viewpoint, E, for each vertex V, we find the point P on the specular reflector that corresponds to the position of V (see Figure 2).

The difficult part in this algorithm is computing P as a function of V and E. Except in the most basic case of planar specular reflectors, there is no simple relationship between P, V and E. Even for a sphere, the explicit position of P depends on a polynomial of the fourth order; finding the roots of this polynomial is feasible, but takes actually longer than the iterative method we use.

According to Fermat's principle, light travels along paths of extremal length, so P must correspond to an extremum of the optical path length $\ell = EP + PV$. We are searching for extrema of ℓ, or equivalently, for zeros of its first order derivative, the gradient $\nabla\ell$.

This is an optimization problem, with a function of two parameters (the surface of the specular reflector is a 2D manifold). Usually, optimization problems are solved with *line search* methods, such as the gradient descent or the conjugate gradient methods. These method progress iteratively from an initial guess. At each step, they know the *direction* in which they should progress, but not necessarily the *distance* along this direction. Knowing this distance accurately requires knowledge about the second derivatives of the function.

Our application is inherently graphical: we are displaying the result of our computations on the screen, and changing parameters — the viewpoint, the reflected scene, the reflector — dynamically. One of the most important points for such graphical applications is temporal coherency: the reflection of one point must not change suddenly between frames. We therefore need *spatial* information about the accuracy of the computations: if we have not yet computed the position of one point with sub-pixel accuracy, we run the risk of seeing temporal discontinuities at the next frame. We also observed in our experiences that the number of iterations required for convergence varies greatly with the configuration of the vertex. Spatial information about convergence help in adapting the number of iterations to the current case.

Line search methods typically use residuals to check the numerical accuracy of the computations, but they do not provide information about the spatial accuracy. At each step, we know the distance traveled from the previous step, but this information is only *linear*. Since the reflector is a 2-dimension surface, it can happen that the algorithm has closed in on the result along one dimension, but is still far from it on the other dimension.

The *secant method* searches for roots of one function f by replacing it with a linear interpolation between samples, picking the root of the linear interpolation and iterating. While the secant method does not guarantee that the root remains bracketed, it provides a good information about the accuracy achieved so far, and converges faster than the simpler bisection method. Newton's method converges faster than the secant method, but requires computing the derivative of f.

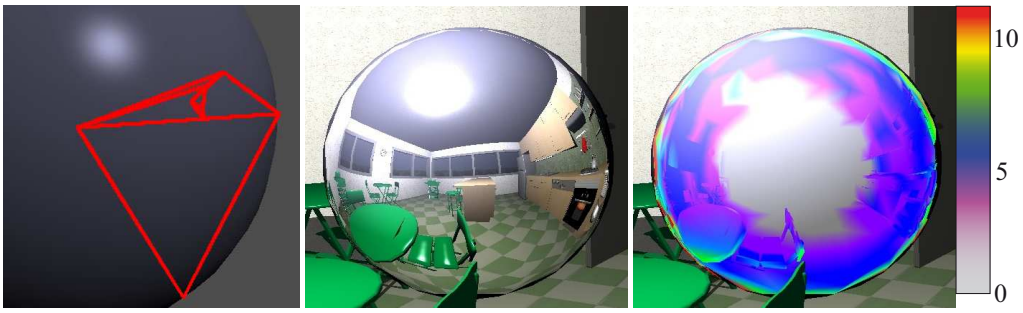
Since we are looking for zeros of $\nabla\ell$, we apply to it a variant of the secant method. At each step, we maintain a triangle of sample points where we compute $\nabla\ell$ and linearly interpolate between these gradients. At each step, the triangle of sample points gives us approximate geometric bounds on the projection of the vertex.

3.2. Algorithm for specular vertex reflection

Our algorithm for computing the reflection of a 3D scene in a specular reflector uses the following steps:

1. render the scene into the framebuffer, with direct lighting and shadowing;
2. for all vertices of the scene, find their reflection on the specular reflector;
3. interpolate between these vertices, computing lighting and doing hidden surface removal.

For each vertex, finding the position of its reflection is done iteratively, using a variant of the secant method on the gradient of the optical path length: at each step, we maintain a triangle of sample points, and we:



(a) Example of successive triangles generated by our algorithm (b) Example image rendered with our algorithm (c) Number of iterations required for convergence

Figure 3: *Convergence of our iterative system.*

- compute the gradient of the optical path length for each sample point (see section 3.3.2),
- linearly interpolate between these gradients,
- find the resulting gradient with the smallest norm (see section 3.3.3).
- discard the original sample point with the largest gradient, replace it by the new sample point and iterate (see Figure 3(a)).

At each step, the projected area of the triangle gives us an indication of the accuracy of our computations. We stop the computation if this area falls below a certain threshold.

Our method converges quickly in most cases, in 5 to 10 iterations in moderately complex cases but can require up to 20 iterations for certain difficult points, such as vertices whose reflection is close to the boundary of the reflector (see Figure 3(c)).

The method is robust enough to converge even if the initial set of sample points is poorly chosen. However, it converges faster if the sample points are close to the actual solution. Section 3.3.4 describes our strategy for picking the initial sample points.

Once we have computed the reflection of each vertex, we project it on the screen and let the graphics hardware does linear interpolation between the vertices. We exploit the fact that we know the spatial position of the point being reflected to compute direction-dependent lighting (see Section 3.3.5).

Hidden surface removal requires special handling, as we have several possible sources of occlusion: the scene and the reflector may be hiding each other, parts of the reflector may be hiding themselves, and parts of the reflected scene are hiding other parts of the reflected scene. Section 3.3.6 describes our solution to these combined occlusion issues.

The entire algorithm was implemented on the GPU, using programmable capabilities for vertex and fragment processing. Hardware implementation issues are described in section 3.3.7.

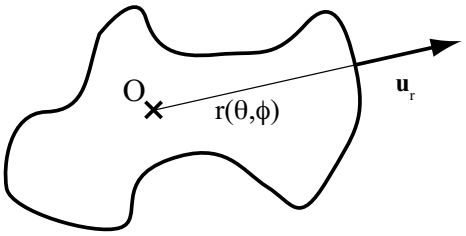


Figure 4: *To reduce dimensionality, we assume that the reflector is star-shaped.*

3.3. Details of the algorithm

3.3.1. Specular reflector parameterization

In order to provide interesting reflections, it is better if our reflector is actually smooth. We also assume that it is parameterizable. Finally, to reduce the dimensionality of the problem, we assume that the reflector is star-shaped: there is a point O that is directly connected to all the points on the surface of the reflector (see Figure 4).

This reduces the equation of the specular reflector to a scalar function, r. Using spherical coordinates, for example, all point $P(\theta, \phi)$ on the receiver can be expressed as:

$$P(\theta,\phi) = O + r(\theta,\phi)\boldsymbol{u}_r \quad \text{with} \quad \boldsymbol{u}_r = \begin{pmatrix} \sin\theta\cos\phi \\ \sin\theta\sin\phi \\ \cos\theta \end{pmatrix}$$

For our algorithm, we will also need the variations of the surface of the reflector: we also compute the derivatives of the function r.

In a preliminary step, r and its partial derivatives are computed and stored in a texture. Although our algorithm works with any kind of reflector, the star-shaped hypothesis allows us to retrieve all the required information about the specular reflector at any given point with a single texture read. This

will be useful for implementing our algorithm efficiently on the GPU.

Using spherical coordinates introduces singularities in the parameterization, at the poles. To avoid numerical issues in our computations, we do not use r or its partial derivatives directly, but we only use 3-dimensional vectors such as P or ∇r. All computations and interpolations are done in 3D space, never in parameter space.

3.3.2. Optical path derivatives

Assuming we have a sample point on the surface of the reflector, we can compute the length ℓ of the optical path length from the viewpoint E to the vertex V through P (see Figure 2):

$$\ell = EP + PV$$

The gradient of the optical path length depends on the derivative of point P on the reflector surface:

$$\begin{aligned} \nabla\ell &= \nabla(EP) + \nabla(PV) \\ \nabla\ell &= \mathrm{d}(P)\left(\frac{\overrightarrow{EP}}{EP} + \frac{\overrightarrow{PV}}{PV}\right) \end{aligned}$$

Here $\mathrm{d}(P)$ is the derivative of point P, a linear form operating on a vector. With our parameterization of P on a star-shaped reflector, $\mathrm{d}(P)$ is also reduced in dimension, and we can express $\nabla\ell$ as a function of ∇r:

$$\nabla\ell = (\nabla r \cdot \boldsymbol{e})\boldsymbol{u}_r + (\boldsymbol{u}_\theta \cdot e)\boldsymbol{u}_\theta + (\boldsymbol{u}_\phi \cdot e)\boldsymbol{u}_\phi \qquad (1)$$

with:

$$\boldsymbol{e} = \frac{\overrightarrow{EP}}{EP} + \frac{\overrightarrow{PV}}{PV}$$

$$\boldsymbol{u}_\theta = \begin{pmatrix} \cos\theta\cos\phi \\ \cos\theta\sin\phi \\ -\sin\theta \end{pmatrix} \qquad \boldsymbol{u}_\phi = \begin{pmatrix} -\sin\phi \\ \cos\phi \\ 0 \end{pmatrix}$$

∇r can be expressed as a function of the partial derivatives of r, but it is not actually necessary in our case. We are storing information about r and its derivatives in a texture, which will be accessed by the GPU. As a single texture read gives access to 4 channels, we store r and its gradient ∇r, saving computations.

3.3.3. Finding a better estimate for vertex reflection

At each step, we have a triangle of sample points (A, B, C). For all points D, expressed in barycentric coordinates with respect to (A, B, C):

$$D = \alpha A + \beta B + (1 - \alpha - \beta)C$$

we compute an approximation $\tilde{\nabla} d_D$ the gradient of ℓ using linear approximation:

$$\begin{aligned} \tilde{\nabla} d_D &= \alpha\nabla d_A + \beta\nabla d_B + (1 - \alpha - \beta)\nabla d_C \\ &= \alpha\boldsymbol{a} + \beta\boldsymbol{b} + \boldsymbol{c} \end{aligned}$$

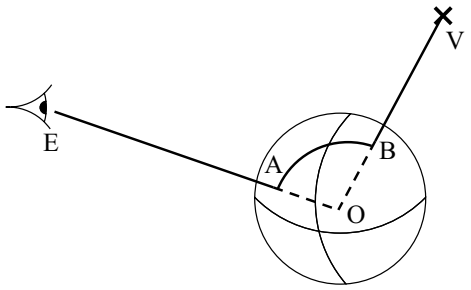


Figure 5: *On a sphere, the reflection lies on the arc* (A, B)

Ideally, we would like to select (α, β) such that $\tilde{\nabla} d_D = 0$. However, this is not always possible, unless the vectors $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{c}$ are linearly dependent. So we pick (α, β) so that $\|\tilde{\nabla} d_D\|$ is minimum: we derivate $\|\tilde{\nabla} d_D\|^2$ with respect to α and β, and find (α, β) such that both derivatives are null. This is equivalent to solving the linear system:

$$\begin{cases} \alpha\boldsymbol{a}^2 + \beta(\boldsymbol{a} \cdot \boldsymbol{b}) + (\boldsymbol{a} \cdot \boldsymbol{c}) &= 0 \\ \alpha(\boldsymbol{a} \cdot \boldsymbol{b}) + \beta\boldsymbol{b}^2 + (\boldsymbol{b} \cdot \boldsymbol{c}) &= 0 \end{cases}$$

whose determinant is:

$$\delta = \boldsymbol{a}^2\boldsymbol{b}^2 - (\boldsymbol{a} \cdot \boldsymbol{b})^2$$

The (α, β) parameters give us a new point D. We discard the point in (A, B, C) with the largest gradient and replace it with point D, then iterate.

In some circumstances, the determinant δ of the system can be null or very small, making the system ill-conditioned. When it happens, we backtrack in time, replacing one of the points $\{A, B, C\}$ by the most recently discarded point. Of course, we cannot replace the most recently added point, or the system would enter an infinite loop.

3.3.4. Initialization

Our method is efficient and converges even if arbitrary sample points are used as a starting triangle. However, the convergence is faster if the starting triangle is small and close to the result. It is not necessary for our initial guess to actually enclose the result, since our algorithm is able to extrapolate outside the triangle if necessary.

For a spherical reflector, the reflection of a vertex V is in the plane defined by V, the eye E and the center of the sphere O. Ofek [Ofe98] shows that the reflected vertex is bound on the arc of circle $[AB]$ where A (resp. B) is the projection of V (resp. E) on the reflector (see Figure 5).

For non-spherical reflectors, this property does not hold. We nevertheless use A and B as as two of our initial points. The third point C is chosen so that ABC is an equilateral triangle.

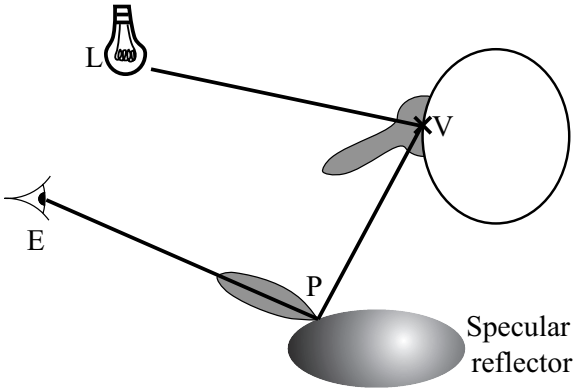


Figure 6: *Computing the illumination of the reflected scene: illumination at the reflected point is computed using its BRDF, with $\overrightarrow{VL}$ and $\overrightarrow{VP}$ as incoming and outgoing directions; it is then multiplied by the BRDF on the reflector, with $\overrightarrow{PV}$ and $\overrightarrow{PE}$ as incoming and outgoing directions.*

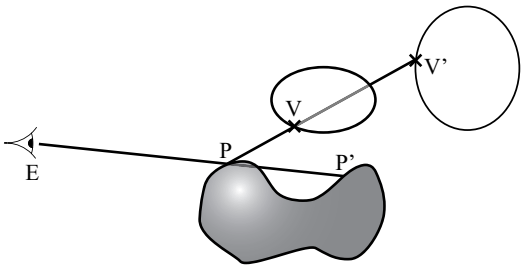


Figure 7: *For a ray originating from the eye, we have to resolve visibility issues both between P and P′, on the reflector, and between V and V′, on the reflected ray.*

3.3.5. Direction-dependent lighting on the reflected scene

When we display a fragment of the reflected scene, we know its spatial position V and the approximate spatial position of its reflection P. We use this information to compute directionally-dependent lighting:

- compute illumination at point V, using its BRDF, with the light source L as the incoming direction and the reflected point P as the outgoing direction (see Figure 6).
- multiply this by the BRDF of the specular reflector at point P, using the reflected point V as the incoming direction, and the viewpoint E as the outgoing direction.

This simple rule allows us to have directional lighting on the reflected scene. The lighting on the reflected scene is thus not necessarily the same as the lighting on the original scene.

3.3.6. Multiple Hidden-Surface Removal

Hidden surface removal requires special handling, as we have several possible sources of occlusion (see Figure 7): the scene and the reflector may be occluding each other, and we also have to conduct hidden-surface removal on the reflected scene. The ideal solution would be to use several depth buffers, or a multi-channel depth-buffer. As these are not available, we have designed a workaround.

For each vertex V, when we compute its projection P, we store in the depth buffer the distance between P and V. This way, the Z-buffer of the graphics card naturally removes fragments of the reflected scene that are hidden by other objects.

To solve the other occlusion issues, we use the following strategy:

- pre-render the frontmost back-facing polygons of the reflector into a depth texture; clear the Z-buffer and framebuffer.
- render the scene, with lighting and shadowing; clear the stencil-buffer.
- render the reflector, with hidden surface removal. For pixels that are touched by the reflector, set the stencil buffer to 1.
- clear the depth buffer and render the reflected scene using our algorithm. The fragments generated are discarded if the stencil buffer is not equal to 1 (using the classical stencil test) and if they are further away than the backfaces of the reflector (using the depth texture computed at the first step).
- (optional) enable blending and render the reflector, computing its illumination.

Our strategy correctly handles occlusions between the reflector and the scene (using the stencil test), as well as self occlusion of the reflector, using the depth texture. Note that we have to use frontmost back-facing polygons: using the frontmost front-facing polygons would falsely remove all the reflected scene for locally convex reflectors, since we are linearly interpolating between reflected points that are on the surface of the reflector.

3.3.7. GPU implementation

We have implemented our algorithm on the GPU for better efficiency. To compute the reflected position of one vertex, we need access to the equation and derivatives of the specular reflector. Since we stored these in a texture to handle arbitrary specular reflectors, this limits us to two possible implementation strategies:

- place our algorithm in vertex shader, using graphics hardware with vertex texture fetch (NVidia GeForce 6 and above).
- place our algorithm in a fragment shader and render the reflected positions of the vertices in a Vertex Buffer Object. In a subsequent pass, render this VBO. This requires hardware with render-to-vertex-buffer capability, which was not available to us at the time of writing.

We have used the first strategy, but found that it suffers from several limitations: there are less vertex processing units than fragment processing units on GPUs, so we are not taking full advantage of its parallel engine; a texture fetch in a vertex processor has a large latency; vertex processors can not currently read from cube maps or rectangular textures, forcing us to use square textures.

As pointed out by [EMD*05], it makes sense to use a cube map to store the information about the specular reflector, since reflector information is queried based on a direction vector $\boldsymbol{d}$, as cube maps are. In the current implementation, we have to convert the vector $\boldsymbol{d}$ into spherical coordinates (θ, ϕ), a costly step.

An implementation of our algorithm using the second strategy is likely to have much better rendering times, as well as a simpler code.

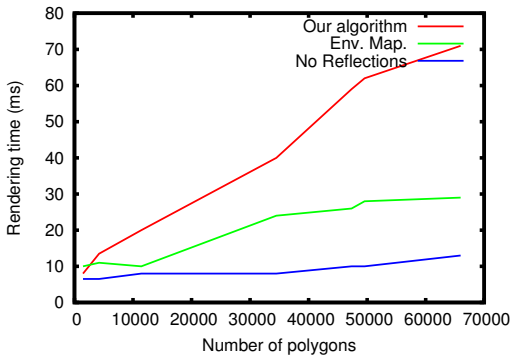


Figure 10: *Observed rendering time (in ms) for rendering scenes, with no specular reflections, with environment-mapping specular reflections and with our algorithm.*

4. Experiments and Comparisons

4.1. Comparison with other reflection methods

The strongest point of our algorithm is its ability to produce reflections with great accuracy. Figure 1 and Figure 8 show, for comparison, pictures generated with our algorithm, ray-traced pictures for reference, and pictures generated with environment mapping. Our method handles all the reflection issues, including contacts between the reflector and the reflected object. Differences between our method and the environment mapping method especially appear for objects that are close to the reflector, such as the hand in Figure 1 and the handle of the kettle in Figure 8. Notice how the reflection of the handle of the kettle appears to be flying in the reflection of the room in Figure 8(c).

For objects that are close to the reflector, our algorithm exhibits all the required parallax effects. One of the problems with environment mapping techniques is when objects are visible from some parts of the reflector but not from its center. In figure 9, our algorithm properly renders the back of the chair.

Another strong point of our algorithm is its robustness and temporal stability. As shown in the accompanying video, reflections computed by our algorithm exhibit great temporal stability, without temporal aliasing. This property is essential for practical applications, such as video games.

4.2. Rendering speed

As we have seen in Figure 3(c), the number of iterations required for convergence depends greatly on the position of the reflection. Reflections close to the center of the reflector converge quickly, in less than 5 iterations, while reflections of objects located close to the silhouette of the reflector take longer to reach convergence.

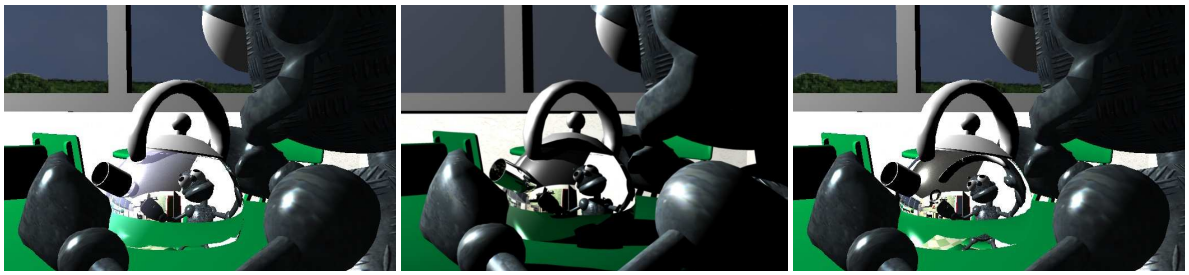
As a consequence, the rendering time depends on the respective position of the object and the reflector. We observe the worse timing results if the scene being reflected completely surrounds the reflector. In that case, many objects are reflected on the silhouette, dragging the rendering process. These scenes are also more interesting to render, which is why we used them nevertheless in all our timings results.

Figure 10 shows the rendering times for scenes of various sizes, surrounding the specular reflector. For comparison, we plotted the rendering time for the scene, without specular reflections, with specular reflections simulated by environment mapping and with specular reflections computed by our algorithm. The extra cost introduced by our algorithm is always larger than that of environment maps, but it remains within the same order of magnitude. We observe satisfying performances for scenes up to 40,000 polygons, and we also observe that rendering times depend linearly on the number of vertices (all timings in this section were measured on a 2-processor Pentium IV, running at 3 GHz, with a NVidia GeForce 7800 graphics card). We measure rendering times in ms by taking the reciprocal of the observed framerate, multiplied by 1000.

4.3. Robustness and early exit

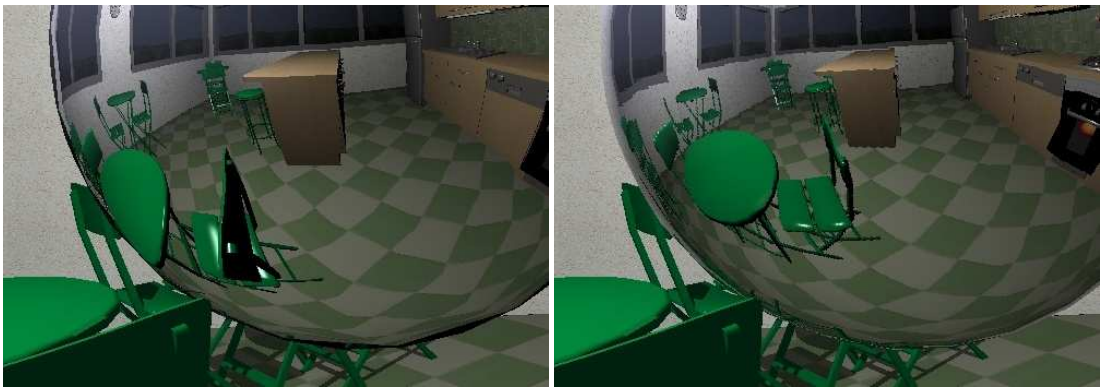
As our method is based on a triangle of sample points and uses the gradient of the optical path at each sample point, it has several advantages:

- we make large steps if we are far from the solution, and smaller steps as we approach the solution (see Figure 3(a)). This ensures faster convergence, even with poor initial conditions.
- the method is remarkably robust, and converges even for difficult cases, such as vertices reflected at grazing angles; in that case, it takes longer to reach a converged solution, but it reaches one, sometimes after more than 10 iterations (see Figure 3(c)).



(a) Our method (b) Ray traced reference (c) Environment mapping

Figure 8: *Comparison of our results (left) with ray-tracing (center, for reference) and environment-mapping (right). The difference are especially visible for objects that are close to the kettle, such as its handle and the right hand of the character.*



(a) Our algorithm (b) Environment mapping

Figure 9: *Our algorithm is able to display objects that are not visible from the center of the reflector. Notice here how the back of the chair is properly rendered.*

We note that for simple cases, our method reaches convergence very quickly (less than 5 iterations), while for difficult cases it requires more computations. As we are doing our computations on the vertex processing units, the fact that different vertices require different computation times is not a big issue. In our tests, we found that using the early exit greatly improved the speed of the computations compared to using a fixed number of iterations.

Spatial consistency could become a larger issue if we moved the computations to the fragment processing unit, but we observe (see Figure 3(c)) that all the vertices from one object have similar complexities; all these vertices should take roughly the same computation time, ensuring that early exit also works well in this situation.

4.4. Concave reflectors

Concave reflectors are a special case. As noted by [Ofe98, OR98], concave reflectors divide space into three zones. Objects that are in the first zone, close to the reflector, are reflected only once and upside-up. Objects that are in the

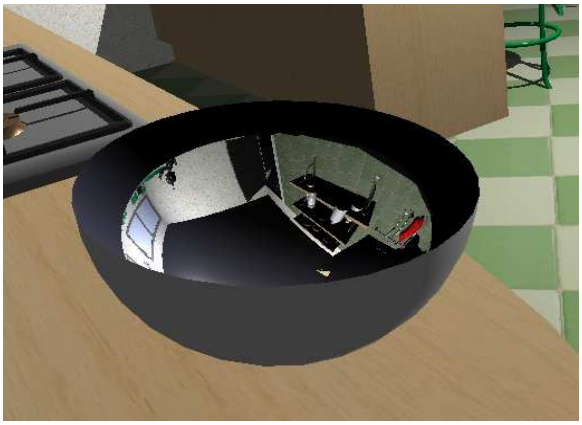


Figure 11: *Example of a reflection with a concave reflector. As our algorithm only captures the first reflection of the scene in the bowl, the top of the bowl looks empty.*

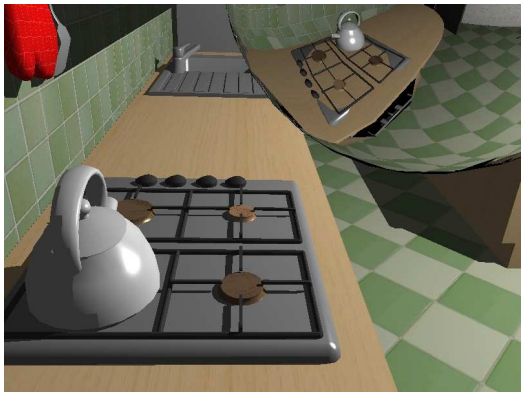


Figure 13: *Example of Z-fighting when a small object is layered on top of a larger object.*

third zone, far from the reflector, are reflected only once, and upside-down (as in Figure 11). Objects that are in the second zone, between the other two, can have several reflections, sometimes an infinite number, and their reflection is numerically unstable.

As with [Ofe98, OR98], our algorithm properly handles objects that are either completely in the first or the third zone, but not objects that cross or are in the second zone.

In our experiments, another problem appeared: concave objects are highly likely to cause secondary reflections (reflections with several bounces inside the specular reflector). As our algorithm only captures the first reflection of the scene by the specular reflector, the place where these secondary reflections should be looks empty.

4.5. Tesselation issues

One of the biggest drawback of our algorithm is that we are only computing the exact reflection position at the vertices, and we let the graphics hardware interpolate between the reflected positions. Currently, the graphics hardware is only able to interpolate linearly. This has several consequences. The first one is that the interpolated objects are located *behind* the front face of the reflector if the reflector is locally convex. Thus, the front face of the reflector would hide the reflection. We had to ensure that the front face of the reflector was not present in the Z-buffer to avoid this problem. The second one is that for objects that are not finely tesselated, we see interpolation artifacts. These artifacts can either be discontinuities between neighboring faces with different levels of tessellation, or a reflection that looks straight, as in Figure 12(a). The third consequence appears for thin objects layered on top of another, larger object (see Figure 13). Because we are linearly interpolating Z-values as well as position, the back object may pop in front of the other object, partially occluding it.

The solution to these issues would be to use curvilinear interpolation, or adaptive tessellation. In the meantime, we apply our algorithm to well-tessellated scenes (see Figure 12(b)). Note that curvilinear interpolation of depth values would be easier with current graphics hardware than curvilinear interpolation in pixel space.

5. Conclusion and Future Works

We have presented an algorithm for computing reflections on curved specular surfaces, using vertex-based computations. Our algorithm produces realistic specular reflections in real-time, showing all the required parallax effects. Our algorithm is iterative, with an adaptive number of iterations, and has a geometry-based criterion for deciding convergence.

In its current form, our algorithm uses linear interpolation between the projections of the vertices, resulting in artifacts for scenes that are not finely tessellated. Solutions to this problem are either adaptive tessellation or curvilinear interpolation techniques.

The strongest point of our algorithm is that it can handle arbitrary geometry on the reflector and the reflected object, including contact between the two surfaces. It is for this situation — close proximity between the reflected object and the reflector — that current environment-map methods do not provide convincing results. We think that our algorithm would be best used as a complement to existing methods, handling the reflection of close objects, while environment-map based methods would be used for the reflection of further objects and the background.

As our algorithm provides a method to compute the reflected ray passing by two endpoints, it can be used for other computations, such as caustics and refraction computations.

References

[Bjo04] B K.: Finite-radius sphere environment mapping. In *GPU Gems*. Addison-Wesley, 2004.

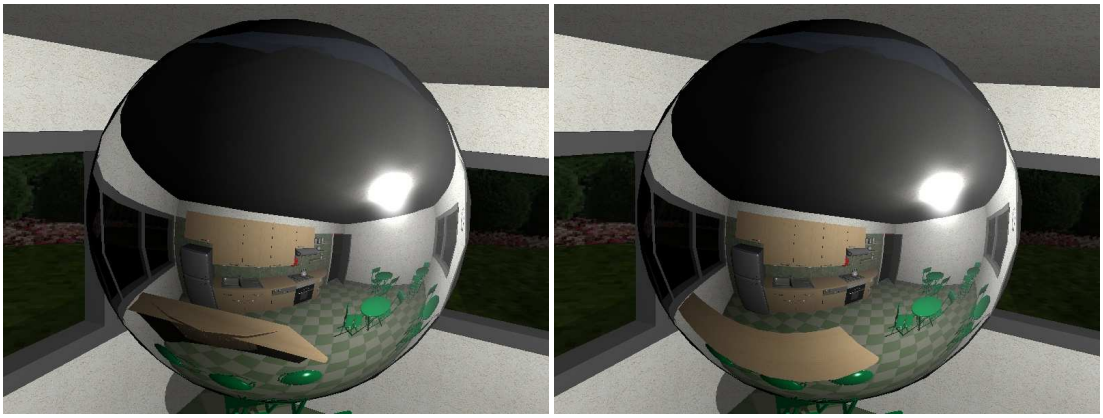
[BN76] B J. F., N M. E.: Texture and reflection in computer generated images. *Communications of the ACM 19*, 10 (1976), 542–547.

[CA00a] C M., A J.: Perturbation methods for interactive specular reflections. *IEEE Transactions on Visualization and Computer Graphics 6*, 3 (2000), 253–264.

[CA00b] C M., A J.: Theory and application of specular path perturbation. *ACM Transactions on Graphics 19*, 4 (2000), 246–278.

[CHH02] C N. A., H J. D., H J. C.: The ray engine. In *Graphics Hardware 2002* (2002).

[EMD*05] E P., M I., D G., T D., D O., C F.: Accurate interactive specular reflections on curved objects. In *Proceedings of VMV 2005* (Nov. 2005).



(a) The bar is not tessellated, and its reflection is not curved — as it should be.

(b) The problem disappears if we tessellate the bar.

Figure 12: *The scene has to be well tesselated, or artifacts appear because we cannot render curved triangles.*

[EMDT06] E P., M I., D G., T D.: A gpu-driven algorithm for accurate interactive reflections on curved objects. In *Rendering Techniques 2006 (Proc. EG Symposium on Rendering)* (June 2006).

[McR96] M R T.: Programming with OpenGL: Advanced rendering. Siggraph'96 Course, 1996.

[MH92] M D., H P.: Illumination from curved reflectors. *Computer Graphics (Proc. of SIGGRAPH '92) 26*, 2 (1992), 283–291.

[MP04] M A., P V.: *Reflection Morphing*. Tech. Rep. CSD TR#04-015, Purdue University, 2004.

[Ofe98] O E.: *Modeling and Rendering 3-D Objects*. PhD thesis, Institute of Computer Science, The Hebrew University, 1998.

[OR98] O E., R A.: Interactive reflections on curved objects. In *Proc. of SIGGRAPH '98* (1998), pp. 333–342.

[Pat95] P G. A.: Accurate reflections through a Z-buffered environment map. In *Proceedings of Sociedad Chilena de Ciencias de la Computacin* (1995).

[PBMH02] P T. J., B I., M W. R., H - P.: Ray tracing on programmable graphics hardware. *ACM Transactions on Graphics (Proc. of Siggraph 2002) 21*, 3 (July 2002), 703–712.

[SKALP05] S -K L., A B., L I., P - M.: Approximate ray-tracing on the GPU with distance impostors. *Computer Graphics Forum(Proceedings of Eurographics '05) 24*, 3 (2005).

[WBWS01] W I., B C., W M., S P.: Interactive rendering with coherent ray tracing. *Computer Graphics Forum (Proc. of EUROGRAPHICS 2001) 20*, 3 (2001).

[WSB01] W I., S P., B C.: Interactive distributed ray tracing of highly complex models. In *Rendering Techniques 2001 (Proc. 12th EUROGRAPHICS Workshop on Rendering)* (2001), pp. 277–288.

[WSS05] W S., S J., S P.: Rpu: a programmable ray processing unit for realtime ray tracing. *ACM Transactions on Graphics (Proc. of Siggraph 2005) 24*, 3 (2005), 434–444.

[YYM05] Y J., Y J., M M L.: Real-time reflection mapping with parallax. In *Proc. I3D 2005* (2005), pp. 133–138.

4.7.6 Wavelet radiance transport for interactive indirect lighting (EGSR 2006)

Auteurs : Janne K , Emmanuel T , Nicolas H et François X. S
Conférence : Eurographics Symposium on Rendering 2006.
Date : juin 2006

Eurographics Symposium on Rendering (2006)
Tomas Akenine-Möller and Wolfgang Heidrich (Editors)

Wavelet Radiance Transport for Interactive Indirect Lighting

Janne Kontkanen[1], Emmanuel Turquin[2], Nicolas Holzschuch[2] and François X. Sillion[2]

[1]Helsinki University of Technology [2]ARTIS† GRAVIR/IMAG INRIA

Maze scene | Direct lighting | Indirect lighting | Complete illumination

Figure 1: *This scene is rendererd 15 FPS by our system with full global illumination. The light sources (spotlights) and the viewpoint can be modified interactively. The precomputation time was 23 minutes.*

Abstract

Global illumination is a complex all-frequency phenomenon including subtle effects caused by indirect lighting. Computing global illumination interactively for dynamic lighting conditions has many potential applications, notably in architecture, motion pictures and computer games. It remains a challenging issue, despite the considerable amount of research work devoted to finding efficient methods. This paper presents a novel method for fast computation of indirect lighting; combined with a separate calculation of direct lighting, we provide interactive global illumination for scenes with diffuse and glossy materials, and arbitrarily distributed point light sources. To achieve this goal, we introduce three new tools: a 4D wavelet basis for concise radiance expression, an efficient hierarchical pre-computation of the Global Transport Operator representing the entire propagation of radiance in the scene in a single operation, and a run-time projection of direct lighting on to our wavelet basis. The resulting technique allows unprecedented freedom in the interactive manipulation of lighting for static scenes.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

1. Introduction

Illumination simulation methods have many interesting applications, for example in architectural design, lighting design, computer games or motion pictures. These applications make use of global illumination algorithms, known for their high computational demands, and would greatly benefit from improved interactivity. We note that these applications are often dealing with indoor scenes, illuminated by local light sources whose position and orientation are subject to change. In this specific setup, interactive global illumination remains a particularly challenging issue.

However, by taking advantage of the linearity of the rendering equation it is possible to precompute light transport offline, and to use this data during run-time to obtain convincing global illumination effects: *Precomputed Radiance Transport* methods (PRT) [SKS02, Leh04] precompute the relationship between the emitted light and the radiance outgoing from the surfaces of the scene. In order to keep the complexity manageable, these methods usually express the emission in a low dimensional basis. The most common way to do this is to consider only infinitely distant lighting, and

† ARTIS is a research project in the GRAVIR/IMAG laboratory, a joint unit of CNRS, INPG, INRIA and UJF.

thus reduce the dimensionality of the emission to 2. Yet, local light sources (a.k.a. *near-field illumination*) have 5 degrees of freedom, that can be narrowed down to 4 without loss of generality if we consider that light travels through a vacuum; this high dimensionality tends to make classical PRT methods extremely costly.

In this paper, we present a technique for interactive computation of global illumination in static scenes with diffuse and glossy materials, and arbitrarily placed dynamic point/spotlights. Our algorithm uses a precomputed *Global Transport Operator* that expresses the relationship between incident and outgoing surface radiance. During run-time we project the direct light form the light sources to the surfaces, and apply this precomputed operator to get full global illumination. Rather than following the common *compute, then compress* scheme, we try to generate the operator directly in a compact representation.

Our contributions are: a new 4D wavelet basis for compact representation of radiance, a method to efficiently compute the Global Transport Operator, greatly speeding up the precomputation time, and a method to efficiently project direct lighting from point light sources on our hierarchical basis at runtime. These three contributions, combined together, result in interactive manipulation of light sources, with immediately visible results in the global lighting.

The most noticeable limitation of our approach is directly linked to a well-known problem of finite-element methods for global illumination: our basis functions have to be expressed on the surfaces of the scene. Incidentally, our example scenes are exclusively composed of large quads. Another important limitation is that BRDFs must be relatively low-frequency to be efficiently representable in our wavelet basis.

2. Previous work

Global illumination has been the subject of research in Computer Graphics for decades. Dutré *et al.* [DBB03] give a complete survey of the state-of-the art of global illumination techniques. There have been plentiful research efforts to speed up global illumination computations and achieve real-time or interactive framerates.

Ray-tracing has been ported to the GPU [PBMH02, PDC*03] or to specific architectures [WSB01, SWS05]. The same has been done with the radiosity algorithm [Kel97, CHL04], while others use the GPU for fast computation of hierarchical form-factors [LMM05].

Nijasure *et al.* [NPG05] compute a representation of the incident radiance at several sample points sparsely covering the volume enclosed by the scene. Incident radiance is stored using spherical harmonics. Spherical harmonics coefficients are interpolated between the sample points and applied to the surfaces of the scene. The system can be iterated to compute multiple bounces of light, at the expense of rendering time. This approach can be seen as a dynamic generalization of Greger's *irradiance volumes* [GSHG98].

Dachsbacher and Stamminger [DS06] introduce an extended shadow map to create first-bounce indirect light sources. They splat the contribution of these sampled sources onto the final image using deferred shading. They only compute the first indirect light bounce, without taking visibility into account. Nevertheless, they observe that the results look plausible in most situations.

Despite using GPUs or even custom hardware, the above methods currently barely run interactively, unless they restrict themselves to small scenes or degrade the accuracy of the simulation.

In a separate research direction, PRT techniques [Leh04] precompute light exchanges and store the relationship between incoming lighting and outgoing global illumination on the surface of an object. The result of these precomputations, the light transport operator, is compressed and used at runtime for interactive display of global illumination. Most PRT techniques start by precomputing the light transport operator with great accuracy, *then* compress it, typically using *clustered principal component analysis* [SHHS03].

The cost of the uncompressed light transport is directly related to the degrees of freedom (DOF) n in the operator, growing with $O(k^n)$. As discussed in section 1, the general expression for emission space, assuming no participating media, has 4 DOF. Given that the outgoing surface radiance also has 4, the general form of the operator end up with 8 DOF.

To keep memory and precomputation costs tractable, most PRT techniques somehow restrict these degrees of freedom. It is generally achieved by assuming infinitely distant lighting, as done Sloan *et al.* [SKS02] and many others. Another option is to fix the locations of the light sources in space [DKNY95]. Yet another one is to fix the viewpoint [NRH03]: in this work, Ng *et al.* demonstrate that all-frequency lighting from an infinitely distant environment can be rendered efficiently by using a light transport operator expressed in Haar wavelet basis and non-linearly compressed. The fixed viewpoint restriction applies when the scene contains glossy materials. In a subsequent publication [NRH04] the authors remove this restriction by introducing triple wavelet product integrals. As a result they are able to generate high quality pictures that solve the 6-dimensional transport problem, but not with real-time or interactive rates.

Haar wavelets have succesfully been used by others [LSSS04, WTL06] to efficiently express all-frequency transport from detailed environment maps to glossy surfaces. These method utilize separable decomposition, consisting of a purely light-dependent term and a purely view-dependent term.

The approaches closest to ours are those of Kristensen *et al.* [KAMJ05] and Hasan *et al.* [HPB06]. Both consider static scenes under near-field illumination, and they separate the computation of direct lighting, done on the fly using standard GPU-based methods, and indirect lighting precomputed on some specific basis functions. Kristensen *et al.* [KAMJ05] use a 3D unstructured point cloud basis, precomputing radiance transport from this basis to the surfaces of the scene. At run-time, uniform point light sources are projected onto the point-cloud basis, then they apply the precomputed transport operator to obtain indirect illumination on the surfaces.

In a work concurrent to ours, Hasan *et al.* [HPB06] precompute direct-to-indirect transport corresponding to our GTO and express it in wavelet basis. The receiving basis consist in the visible pixels, and the sending basis is build by distributing point samples into the scene, which are then hierarchically clustered. a preprocess. Each of these methods presents different limitations: the former is restricted to omni-directional point lights, and the later renders high-quality pictures but only for diffuse-to-diffuse indirect transfer (although the last reflection can be arbitrarily glossy) and fixed viewpoint. As a comparison, our method doesn't suffer from these restrictions, but is limited to simple geometry and works best with diffuse materials.

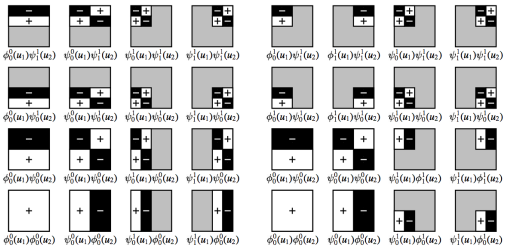
A common problem to many of the existing PRT methods is their fairly inefficient approach to precomputation. A lot of information computed during this step is discarded during a subsequent compression stage. One of our main concerns is precisely to avoid unnecessary computations and rather try to directly generate a concise operator (Hasan *et al.* share this objective). This way, we greatly reduce the memory cost and computation time for the precomputation step. We show clear improvement in precomputation times compared to Kristensen *et al.*, but as stated earlier, our finite element approach also brings restrictions not present in their work.

Our work draws inspiration from hierarchical finite element methods for global illumination [HSA91, GSCH93]. Wavelet and hierarchical algorithms adapt the solution to the geometry and lighting conditions: a coarse resolution is used where the illumination is smooth, and a finer resolution when there are sharp variations.

In our precomputation, we adopt some of the solutions used in Wavelet Radiance [CSSD94, CSSD96] which solves global illumination using a hierarchical 4D wavelet basis. Wavelet Radiance uses *non-standard* decomposition to represent surface radiance, but a *standard* decomposition for the transport operator. The latter enables direct computation of the transfer coefficients without needing a push-pull step.

3. Overview of the algorithm

Our method takes as input the geometric definition of a static and easily parametrized scene (*e.g.* composed by quads), and provides interactive visualization of global illumination effects in this scene under dynamic local lighting. We compute a 4D wavelet representation of indirect radiance on the surfaces of the scene. Our technique can be split into two high-level components: an offline component for precomputing the light transport operator, and a run-time component for rendering indirect lighting using this operator.



(a) Standard refinement (b) Non-standard refinement

Figure 2: *Forming multi-dimensional wavelet basis.*

The run-time component uses the precomputed transport operator to interactively render global illumination:

1. Project direct lighting onto our wavelet basis.
2. Apply the precomputed global transport operator, resulting in indirect lighting, expressed in our wavelet basis.
3. Convert this result to outgoing radiance, and blend with direct lighting.

We treat direct lighting separately because it contains sharp details that are best rendered using specific techniques. Our run-time component is explained in section 5.

The offline component consists of these two steps:

1. Compute a *Direct Transport Operator* (DTO) for the scene. The DTO expresses the propagation of light inside the scene, and corresponds to a single bounce of light.
2. Compute a *Global Transport Operator* (GTO) using the DTO. The GTO expresses the full radiance propagation inside the scene, *i.e.* an infinite number of light bounces.

For efficient computation and compact representation, we express the operators in wavelet basis. Our bases for expressing the surface radiance and the operators are described in section 4.2. The computation of the DTO is explained in detail in section 4.3. In section 4.4 we discuss how the DTO can be used to efficiently compute the GTO using Neumann series and non-linear compression.

4. Offline component

4.1. Wavelet Basis for Surface Radiance

Surface radiance is a 4-dimensional function: two dimensions for the surface location and two dimensions for the direction. Because of its 4D nature, storing a tabulated version of surface radiance is prohibitively costly. This problem is

even more pronounced for the 8-dimensional transport operator that expresses the radiance transport between surfaces.

An efficient expression and computation of the transport operator highly benefits from a hierarchical representation; hence we chose wavelets as our basis functions. We elected Haar wavelets for their computational simplicity, but our algorithm can be applied to any type of *tree* wavelet basis [GSCH93].

The building blocks of Haar basis are the following smooth function ϕ and wavelet ψ:

$$\phi(x) = \begin{cases} 1 & \text{for} \quad 0 \le x < 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\psi(x) = \begin{cases} 1 & \text{for} \quad 0 \le x < 1/2 \\ -1 & \text{for} \quad 1/2 \le x < 1 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

All the wavelets and smooth functions of Haar basis are formed by scales and translates of the above elementary functions as follows:

$$\phi^i_j = \phi(2^i x - j), \psi^i_j = \psi(2^i x - j) \tag{2}$$

Where i gives the scale, and j gives the translation. For comprehensive introduction to Haar wavelets and wavelets in general we refer to [SDS96].

Multi-dimensional wavelet bases are usually formed by combining one-dimension wavelet bases. There are two systems for creating multi-dimensional wavelet bases: the standard refinement (see Figure 2a), where the dimensions are refined separately, and the non-standard refinement (see Figure 2b), where refinement is performed alternatively along all dimensions.

The non-standard refinement method merges together the different dimensions, treating them equally. As a consequence, it is more widely used in fields such as Image Analysis and Image Synthesis, where the two spatial dimensions serve an equal purpose.

For Radiance computations, the spatial and angular dimensions are *not* equivalent. A surface can exhibit large variations on the spatial domain and be more continuous over the angular domain, and linking the resolutions of the spatial and angular dimensions is not always efficient. For this reason, we decouple the spatial and angular domains, using standard refinement between these dimensions. For the 2D sub-domains for angular and spatial dimensions, we still use non-standard refinement. Our wavelet basis for 4D radiance therefore uses a combination of standard and non-standard refinement.

For the angular domain, the hemisphere of directions is mapped to the unit square using a cosine-weighted concentric map [SC97]; we then apply wavelet analysis over the unit square. Using this mapping allows pre-integrated cosine on the hemisphere of directions, with a low angular distortion and constant area mapping.

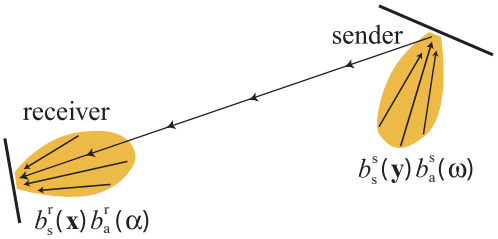


Figure 3: *Light transport from sending basis function $b^s_s(\mathbf{y})b^s_a(\omega)$ to receiving basis function $b^r_s(\mathbf{x})b^r_a(\alpha)$.*

4.2. Wavelet Basis for Transport Operator

The projected transport operator consists of coefficients that describe the influence of each basis function to all the other ones. The non-standard operator decomposition is a more common choice in hierarchical radiosity, as in theory it gives a more compact representation than standard decomposition. In spite of this we chose to follow [CSSD94] and used standard operator decomposition. We see two advantages in using the standard decomposition: it decouples the resolution for sender and receiver, and there is no need for a push-pull step. The former is an obvious advantage, for example when the sender and receiver differ greatly in size or in complexity.

The latter requires an explanation: conventional global illumination methods, using the non-standard representation, require a push-pull step between light bounces [HSA91]. For these methods, the cost of the push-pull step is not prohibitive. However, we are using the DTO to compute the Global Transport Operator, using Neumann series (see section 4.4). During this computation, we perform several multiplications between operators. During these operator multiplications, the fact that we do not require a push-pull step greatly accelerates the computation.

4.3. Direct Transport Operator

We compute the Direct Transport Operator to express a single bounce of light. As we are going to conduct operator multiplications, we require the output space of the DTO to be equal to its input space. This leaves a choice: either we express the DTO in terms of incident radiance or in terms of outgoing radiance. We chose to use the incident form of the Direct Transport Operator.

The incident form of the Direct Transport Operator is defined as follows:

$$(\mathcal{T}L)(\boldsymbol{x}, \boldsymbol{x} \leftarrow \boldsymbol{y}) = \int f_r(\omega, \boldsymbol{y}, \boldsymbol{y} \rightarrow \boldsymbol{x}) V(\boldsymbol{x}, \boldsymbol{y}) \lfloor \omega \cdot \boldsymbol{n}_{\boldsymbol{y}} \rfloor L(\boldsymbol{y}, \omega) \, \mathrm{d}\omega \tag{3}$$

The transport operator maps the incident radiance arriving to location $\boldsymbol{y}$ from direction ω to incident radiance at another location $\boldsymbol{x}$ from direction $\boldsymbol{x} \leftarrow \boldsymbol{y}$. Given a certain distribution of incident radiance, applying this operator once gives the

distribution of light that has been reflected once from the surfaces of the scene. Here f_r refers to BRDF and V to the visibility term. Along with the $\lfloor \omega \cdot \boldsymbol{n_y} \rfloor$ term, they form the kernel $k(\boldsymbol{x}, \boldsymbol{y}, \omega)$ of the light transport operator.

The projected form of the transport operator is obtained by integrating the 6D kernel against each 8D wavelet, in a similar fashion to [CSSD94]:

$$\int K(\boldsymbol{x}, \boldsymbol{y}, \omega) b_s^s(\boldsymbol{y}) b_a^s(\omega) b_s^r(\boldsymbol{x}) b_a^r(\boldsymbol{x} \leftarrow \boldsymbol{y}) \, \mathrm{d}\omega \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{y} \qquad (4)$$

Where $K(\boldsymbol{x}, \boldsymbol{y}, \omega) = k(\boldsymbol{x}, \boldsymbol{y}, \omega) \frac{\lfloor \boldsymbol{x} \leftarrow \boldsymbol{y} \cdot \boldsymbol{n_y} \rfloor}{r_{xy}^2}$ and b_s^r, b_a^r, b_s^s and b_a^s refer to the elementary non-standard basis functions of receiving spatial, receiving angular, sending spatial and sending angular dimensions, respectively (see section 4.1). $\boldsymbol{x}$ and $\boldsymbol{y}$ are integrated over surfaces, while ω is integrated over the hemisphere oriented according to corresponding surface normal. For a visual illustration, see Figure 3.

In the context of light transport, a wavelet coefficient obtained from Equation 4 has traditionally been called a *link*. We will use this term to refer to a group of coefficients for 8D basis functions sharing the same support on all 2D subspaces. In practice, this means that each link corresponds to 255 wavelets and a single smooth function coefficient. This can be seen by considering that each 2D sub-space has 4 elementary non-standard basis functions that share the same support, and $4^4 = 256$. As an example of elementary 2D functions, see the four functions in the lower left corner of Figure 2b.

We compute the Direct Transport Operator by progressively refining the existing links. We start by creating interactions between coarsest level basis functions in the scene, and then refine these. At each step, we consider 256 basis function coefficients. Note, however, that not all the 256 coefficients are stored. We only store the necessary parts of link: a link between two diffuse surfaces does not need wavelets in angular domain. In practice, each link contains between 1 and 256 wavelet coeffients depending on its type.

A refinement oracle (see section 4.3.2) tells us whether a link needs to be refined. For each link it has a choice to refine in any of the 2D sub-domains (spatial receiver, angular receiver, spatial sender, angular sender). The refinement oracle may independently choose each option, possibly refining both the sender and the receiver in space and angle, or simply refining the receiver in space, or any combination.

When the link is refined, we create the child wavelets, and recursively consider each newly created link. Consider a refinement of one of the spatial basis functions: when a spatial basis function is refined, four new child links are created (spatial patch is divided into four child patches). However, when two of the 2D sub-domains are refined, there will be $4 \times 4 = 16$ child links to consider, and finally if all the dimensions are refined $4^4 = 256$ child links are created.

When performing progressive refinement in standard basis, the refinement may arrive at a certain link from several parent links. This means that when we chose the standard method for combining dimensions in our 8D basis, we partially lost the tree-property of our basis.

However, we use the same solution for the problem as was used in the conventional wavelet radiance [CSSD94]. If, in the refinement process, we arrive at a 8D wavelet coefficient that has already been visited, we terminate the traversal. The difference with conventional wavelet radiance is that we have four independent subspaces instead of two.

An important point in our algorithm is that, as with Wavelet Radiance [CSSD94], we are computing wavelet transport coefficients directly between wavelet coefficients, not between smooth functions. This eliminates the need for push-pull step.

4.3.1. Numerical Integration

The actual coefficients corresponding to each link are computed by generating quasi-randomly distributed samples in the support area of the link. Thus, we are computing Equation 4 by quasi-Monte Carlo integration.

The coefficients of the coarsest links are difficult to compute accurately without a significantly large amount of samples. On the other hand the finer scale wavelets do not require as many samples since within a smaller support the kernel does not deviate as much. Because of this we adopt the adaptive integration procedure used in Wavelet Radiance [CSSD94]: we first refine the link structure to the finest level and then perform a wavelet transform to compute the coarser links in terms of the finer ones. As a result, only the finest scale wavelet coefficients are computed directly. In our implementation, this procedure is done during a single recursive visit.

4.3.2. Refinement Oracle

The refinement oracle considers each link, *i.e.* a cluster of coefficients of wavelets sharing the same support at the time. It works by testing quasi-random samples of the kernel, and using explicit knowledge of the BRDF. If the oracle finds that the operator is smooth, then the refinement stops and the kernel samples are used to compute the wavelet coefficients.

At each refinement step, the refinement oracle has to select whether to refine the sender or the receiver, or both, and whether to refine them spatially or angularly, or both. Thus, the oracle can refine between 0 to 4 dimensions, resulting in 16 possible combinations. The ability to make an independent refinement decision in each sub-space is a consequence of using standard refinement as described in sections 4.1 and 4.2.

The decision to refine the interaction in the angular domain is based solely on the BRDFs of the sender and receiver, unless the sender and the receiver are mutually invisible, in which case the interaction is not refined. The basis

functions are mutually invisible if no unobstructed ray can be generated between the supports of the sender and the receiver. Note that this can happen even if the spatial basis functions are not mutually occluded, but the angular support of the sending basis function is oriented in such a way that it does not point towards the receiving basis function.

Diffuse surfaces are never subdivided angularly. For an arbitrarily glossy BRDF, the maximum level of angular subdivision depends on the resolution of its wavelet representation. Note that at this point of the algorithm, the wavelet representation is only used to control the refinement resolution, whereas the kernel samples are evaluated using the original, possibly analytic representation of the BRDF.

Spatial refinement is based on the kernel deviation estimated on the samples. To take advantage of the standard operator decomposition, *i.e.* the ability to refine the sender and receiver independently, we apply the following heuristics:

- Refine sender if $(\max(K) - \min(K))A_s > \epsilon_s$
- Refine receiver if $(\max(K) - \min(K))A_r > \epsilon_r$

With K as defined in Equation 4; A_s, A_r refer to the surface areas covered by the supports of the basis functions; and ϵ_s, ϵ_r to the user selected thresholds. We use separate thresholds for sending and receiving refinement, since it is useful to generate asymmetrically refined matrices, where the sending basis functions are coarser than the receiving (see section 4.4).

4.4. Global Transport Operator

Having computed the direct transport operator $\mathcal{T}$, which expresses a single bounce of light transfer between the surfaces in the scene, we use it to compute the global transport operator, using the Neumann series:

$$\mathcal{G} = \mathcal{I} + \mathcal{T} + \mathcal{T}^2 + \mathcal{T}^3 + \mathcal{T}^4 + \ldots$$

The global transport operator expresses the relationship between the converged incident lighting and the incoming incident lighting. $\mathcal{G}$ is computed iteratively, from $\mathcal{T}$. At each step, we compute $\mathcal{T}^{n+1} = \mathcal{T}^n\mathcal{T}$.

The computation of above series is rather expensive using a high resolution representation of $\mathcal{T}$, since in the end all the basis functions interact with each other (unless the scene consists of separate local environments with no mutual visibility to each other). For this reason, we aggressively compress the matrices during the computation: after each computation of $\mathcal{T}^n$, we apply non-linear compression to the result, removing all coefficients below a certain threshold.

Because of the compression, the number of coefficients in $\mathcal{T}^n$ decreases when n increases (see Figure 6). We stop the computation when all the coefficients in $\mathcal{T}^n$ are smaller than our threshold, which in our experiments required up to ten iterations. To speed-up the computations, we pre-multiply by the sparsest matrix, computing $\mathcal{T}^n$ times $\mathcal{T}$.

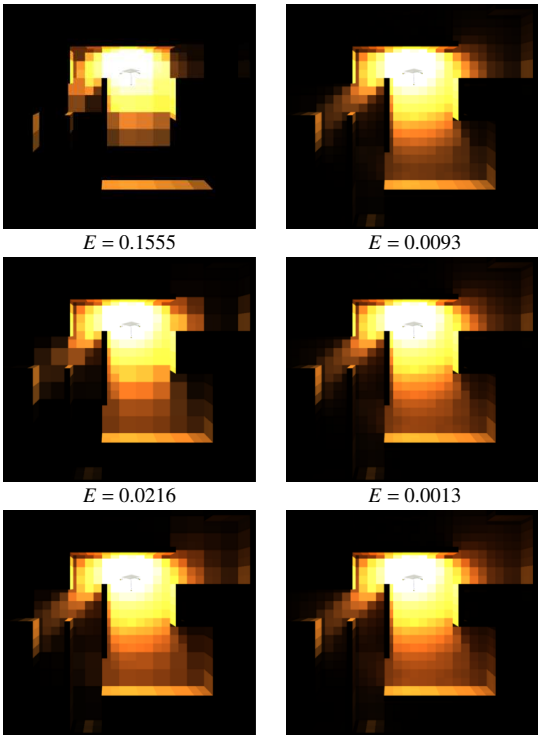


Figure 4: *Multiplying the GTO by the original fine-resolution DTO (right) improves the visual quality compared to the original GTO (left). Notice that even with a larger numerical error, the gathered GTO gives a more pleasing result (compare upper right corner with lower left corner). Error E refers to the sum of squared differences of wavelet coefficients when compared to uncompressed GTO.*

After a coarse GTO has been computed, we still perform one more step to improve the results: we multiply the series by the original fine-resolution DTO from the left. This operation can be thought as a kind of final gathering that improves the visual aspect of the result by using higher resolution representation for the last bounce before the light meets the eye (see Figure 4), in the same spirit as Hasan *et al.* [HPB06].

5. Run-time component

The run-time component of our method works as follows:

1. Project direct lighting on the wavelet basis defined on the surfaces of the scene (section 5.1).
2. Use the precomputed GTO to transform the projected direct light to the converged incident radiance (section 5.2).
3. Transform incident radiance into outgoing radiance by applying the BRDF of the surfaces (section 5.3).
4. Render the indirect light using the wavelet basis and combine it with direct light computed separately (section 5.4).

5.1. Direct Light Projection

In order to use the GTO to generate indirect lighting, we need to project the light from the dynamic light sources to the 4D radiance basis defined on the surfaces of the scene.

For each light source and for each surface of the scene, our method proceeds as follows:

1. Estimate the level of precision required.
2. Compute all the smooth coefficients at this level of precision, by integrating direct lighting on the support of each coefficient.
3. Perform a wavelet transform on these smooth coefficients to compute the wavelet coefficients, then discard the smooth coefficients. This generates a wavelet representation of the direct light on all the surfaces of the scene.

This projection of direct lighting onto our wavelet basis is fundamental for interactive rendering, so it is important to perform these computations efficiently. Unfortunately, step 2 involves computing direct lighting for all the smooth coefficients, a costly step for arbitrary light sources.

To estimate the level of precision required (step 1), we look at the solid angle subtended by the geometry of the object, multiplied by the intensity of the light source in the direction of the object.

The computation of the smooth coefficients (step 2) involves computing direct lighting in the scene, including visibility between the light source and the support of the smooth coefficient. In our implementation, we tried both area light sources and point light sources, but we found that only point light sources were currently compatible with interactive framerates.

For point light sources, we compute the visibility using occlusion queries, *i.e.* we render the smooth functions from the view of light source using *GL_ARB_occlusion_query* extension of OpenGL, and estimate the solid angle each basis function subtends based on the number of visible pixels. Our current implementation only supports direct light projection to directionally smooth basis functions. This means that direct light falling on a specular surface gets reflected as if the surface was diffuse.

To benefit from our sparse wavelet representation for surface radiance, the elements need to be dynamically (de-)allocated. To avoid an excessive amount of dynamic memory management we use the following method: before projecting the direct light at each frame we set the existing allocated coefficients to zero. Then we project the light as described, and after the projection we de-allocate the entries that are still null. This minimizes the amount of dynamic allocations and de-allocations required during run-time.

5.2. Application of the GTO

Once we have a wavelet projected representation of direct incident lighting, we multiply it by the GTO to give the converged incident radiance:

$$\boldsymbol{X} = \mathcal{G}\boldsymbol{E}$$

where $\boldsymbol{E}$ represents the projected direct light, $\mathcal{G}$ is the GTO, and $\boldsymbol{X}$ is the resulting converged incident radiance. All the wavelet representations above are in sparse format, so that only non-zero coefficients are stored.

For efficient multiplication, it is important to take advantage of the sparseness of $\boldsymbol{E}$: typically the direct light can be expressed with a small number of wavelet coefficients, since it is often either spatially localized or falling from a far away light source, in which case only coarse basis functions are present in $\boldsymbol{E}$. We perform the multiplication by considering only the non-zero element of E and accumulating the results to $\boldsymbol{X}$.

We use the same technique to minimize the amount of dynamic memory allocations in $\boldsymbol{X}$ as we used for computing $\boldsymbol{E}$ (section 5.1).

5.3. Multiplication by the BRDF

$\boldsymbol{X}$ represents the incident indirect radiance, and yet we need the outgoing radiance for display. Thus, we need a final multiplication by the BRDF. In our implementation, we associate a wavelet representation of the BRDF with each surface of the scene and this step simply translates into a multiplication in wavelet space. Note that we use the same wavelet representation that the oracle uses to determine the angular refinement (section 4.3.2).

5.4. Rendering from the Wavelet Basis

To generate the final view for the user, we first render the scene representing the indirect light, and then additively blend in the direct light using standard techniques.

The indirect light is synthesized from the 4D wavelet basis to textures using the CPU. Then the whole scene is drawn using standard texture mapping and optionally bi-linear filtering (results without this filtering can be seen in Figure 4).

To get rid of the discontinuities that would appear between neighboring coarse level quads, we use border texels (supported by standard graphics hardware) to ensure a smooth reconstructed result across the edges of quads.

Each quad is associated with its own texture, and thus it is possible to use a specific texture resolution for each quad. In our current implementation we select the texture resolution according to the maximum of the spatial and angular resolutions present in the wavelet basis.

The texture synthesis is performed by traversing all non-zero wavelet coefficients for a given quad. For performance,

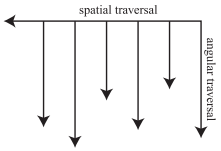


Figure 5: *For texture synthesis, we traverse the wavelet hierarchy in the order shown here. We terminate the angular traversal as soon as we detect that the angular sub-tree points away from the viewer.*

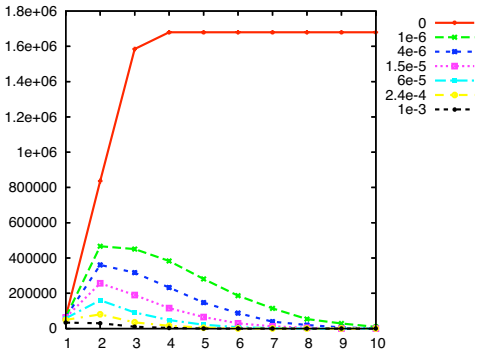


Figure 6: *Number of non-zero entries in DTO^n, as a function of number of bounces n, depending on the threshold used for wavelet compression, for the maze scene.*

we exploit our knowledge of the view direction and avoid traversing the subtrees of wavelets that do not point towards the eye: we traverse the wavelet hierarchy first in spatial order, then in angular order (see Figure 5), and terminate the angular traversal if we detect that the whole subtree points away from the viewer.

6. Experiments and comparison

We computed the GTO at different resolutions for three different scenes: the Cornell box, a maze and a simple scene for testing glossy illumination (see Figure 10). The results are summarized in Table 1. As can be seen, all the results run either in real-time or at least at interactive framerates.

6.1. Offline component

The most important result is the speed of the precomputation step. For comparison, the maze scene we used is an exact replica of the scene used by Kristensen *et al.* [KAMJ05]. The time it takes for our method to compute the GTO on this scene varies between 24 and 74 minutes depending on the required quality. Kristensen *et al.* report a precomputation time of 6.5 hours on a cluster of 32 PCs. Since their method is easily parallelizable, we may assume that the performance is almost linear with the number of machines, translating into a total computation time of approximately 8 days using a single PC. The comparison is based on a visual judgement. For more exact evaluation, a numerical error metric would need

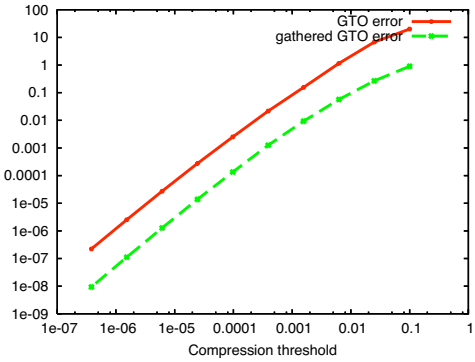


Figure 7: *GTO error in the maze scene as a function of the threshold on wavelet coefficients.*

to be used. Nevertheless, we believe it is fair to say that our method performs the precomputation faster.

This acceleration comes from our algorithm's ability to avoid the computation of unnecessary data. All the information computed during the pre-processing step is used for the runtime computation of indirect lighting. On the other hand, our technique suffers from well-known issues in finite element methods: we need a parameterization of our scene, which restricted us to easily parametrizable surfaces (quads in our current implementation).

Our precomputation time is dominated by the hierarchical refinement to compute the DTO, while the Neumann series evaluation to compute the GTO is relatively fast. The threshold used for non-linear wavelet compression in the GTO computations has an immediate impact on the memory cost of our algorithm (see Figure 6): not using compression in the maze scene results in approximately 2 million links stored. As each link stores 9 to 16 wavelet coefficients in floating point and in three channels, the average cost of a link is 150 bytes. Thus these 2 million links correspond to approximately 300 Mb which is not practical for real-time use.

Even a very small threshold ($\epsilon = 10^{-6}$) on our wavelet coefficients brings the number of links down to 400,000, corresponding to memory cost of 60 Mb. A more aggressive compression ($\epsilon = 10^{-4}$) further divides these numbers by 6, bringing the memory cost to 10 Mb.

We checked the relationship between the level of wavelet compression used on the GTO and the error we make on the operator. We tested both the standard GTO and the "gathered GTO", where the GTO is premultiplied by the original, fine-resolution DTO. Using the non-compressed operator as a reference, we computed the error as a function of the threshold used for compression (see Figure 7). The error on both operators decreases regularly with the threshold, with the error on the gathered GTO being consistently smaller than the error on the standard GTO.

We also checked the relationship between the memory

Table 1: *Summary of the performance of our algorithm. All matrices were computed with a single 3 GHz Pentium 4.*

	C	C	H -	M	M	H -	G	G	H -
t_{DTO} (precomp.)	2min	25min		23min	1h 12min		1min	9min	
t_{GTO} (precomp.)	<1s	2s		40s	1min		<1s	1min	
FPS (run-time)	60	25		15	7		8	3	
Links DTO	3477	30366		65501	288628		4260	36778	
Links GTO	418	648		24151	24599		1712	34176	
Links gathered GTO	14169	53100		164813	589361		44383	195037	
Memory cons. in MB	1.7	6.4		19.7	70		5.3	23.4	

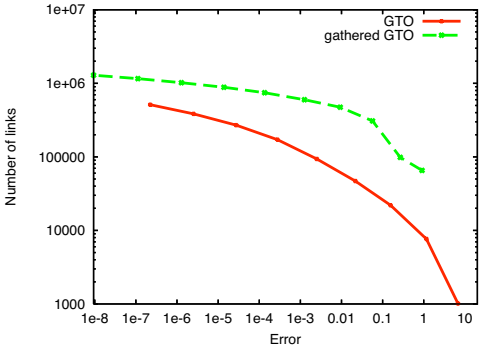


Figure 8: *Memory cost of the GTO as a function of the error on the operator (maze scene).*

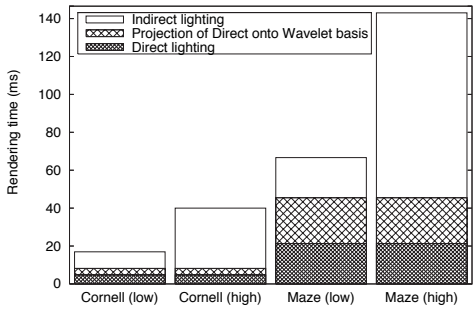


Figure 9: *Rendering times for the different steps of our runtime component. For each scene, we tried a high resolution (moderately compressed) and a low resolution (aggressively compressed) GTO.*

cost of the GTO and the error it represents (see Figure 8). For both versions of the GTO, the error decreases as the memory cost increases. We observe that, surprisingly, the GTO outperforms the gathered GTO: for a given error, it always provides a more compact representation of the operator. Even so, this compact representation does not always translate into visual quality (see Figure 4): comparing the two representations of the GTO with similar error levels, we found that the gathered GTO gives better visual results. The non-linear compression we used for computing the GTO removes links based on the energy they represent. However, the visual quality of the image is not directly linked to energy levels, but also to other spatial information.

6.2. Runtime component

We analyzed the performance of our runtime component. Timings for each step can be seen on Figure 9. We tried two different scenes, each of them with a different level of compression of the GTO. All these rendering times correspond to observed framerates.

Computations related to direct lighting only dependent on scene complexity, as the projection of direct lighting does not depend on the accuracy on the GTO. Computing the projection of direct lighting is about as expensive as the computation of visible direct lighting on the GPU.

Not surprisingly, the computations related to indirect lighting (multiplying the direct lighting by the GTO, the multiplication by the BRDF and conversion to textures) dominate, especially when using a high quality GTO (moderate compression). We observe that the rendering time for indirect lighting is related to the number of coefficients in the GTO.

7. Conclusions and Future work

In this paper, we have presented a novel algorithm for fast computation of indirect lighting. Combined with a separate computation of direct lighting, our algorithm allows interactive global illumination.

Our algorithm makes use of three different contributions: first a new wavelet basis for efficient storage of radiance, using standard refinement to separate the angular and spatial dimensions, secondly a hierarchical precomputation method for PRT, and third a fast projection of direct lighting on our basis. Our method works in a top-down approach, and therefore aims to only compute the information that is necessary for PRT computations.

Our main limitation is inherited from the finite element methods: the elements (*i.e.* the basis functions) need to be mapped to the surfaces of the scene. In the simplest case, the scene needs to initially consist of large quads as in the examples of this paper. Nonetheless, as a future work we wish to study the possibility of to relieve these restrictions in the spirit of bi-scale radiance transfer [SLSS03]. This would require an easily parameterizable coarse version of the scene

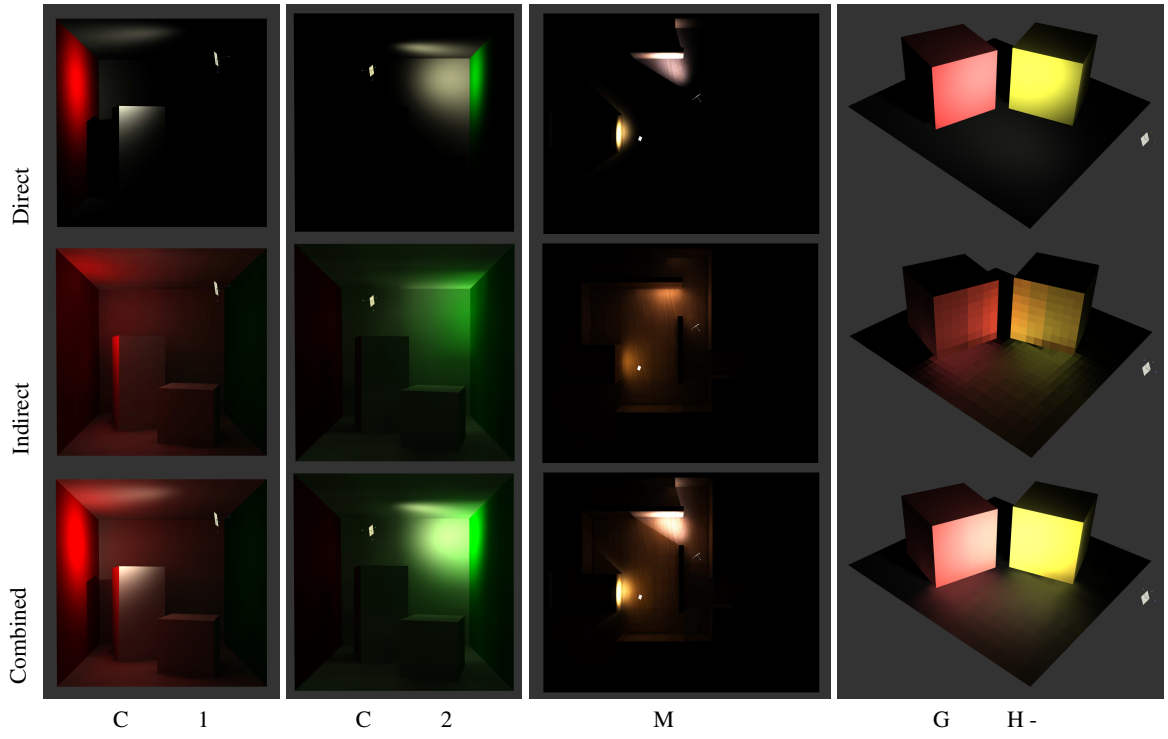


Figure 10: *Example results on our test scenes. In the glossy case (right column), we show the indirect illumination non-filtered, and the composited image using bi-linear filtering.*

and a method to transfer lighting from this coarse surface to a finer one.

Another direction for future work is establishing more explicit relationship between the different compressions done in our algorithm. In the current method, we have separate thresholds for hierarchical refinement and the non-linear compression used during the Neumann series computation. It would be advantageous to only have a single threshold related to the quality of GTO we want to obtain.

The oracle's ability to independently refine each subspace is both a strength and a challenge. The refinement heuristics we presented are not optimal, since they do not take into account the dependence of directional and spatial dimensions, as explained by Durand *et al.* [DHS*05]. For instance, it might not make sense to link a sender with a narrow angular support to a spatially large receiver. We believe that this is a very promising direction for future research and that clear performance improvements are possible.

Yet another idea concerns applying knowledge of runtime importance, projected from the camera to the surfaces of the scene. This projection could be used to speed-up the GTO multiplication as we would know beforehand which basis functions really contribute to the image. So we could use only the parts of the GTO that actually have an visible effect on the result.

Finally, to leverage the full potential of our 4D representation, we plan to explore run-time projection of arbitrarily distributed area light-sources instead of point lights. This would have to be coupled with an accurate display of direct lighting, which could benefit from the information we collect during the projection.

Acknowledgements

We would like to thank the Computer Graphics Group of Helsinki University of Technology, the ARTIS team, and our anonymous reviewers for their valuable feedback. This work has been supported by Bitboys, Hybrid Graphics, Remedy Entertainment, Anima Vitae, and the National Technology Agency of Finland.

References

[CHL04] C G., H M. J., L A.: Radiosity on graphics hardware. In *Graphics Interface 2004* (2004).

[CSSD94] C P. H., S E. J., S D. H., D R T. D.: Wavelet Radiance. In *Photorealistic Rendering Techniques (Proc. of EG Workshop on Rendering)* (June 1994), pp. 287–302.

[CSSD96] C P. H., S E. J., S

D. H., D R T. D.: Global Illumination of Glossy Environments Using Wavelets and Importance. *ACM Transactions on Graphics 15*, 1 (Jan. 1996), 37–71.

[DBB03] D ´P., B K., B P.: *Advanced Global Illumination*. AK Peters, 2003.

[DHS*05] D F., H N., S C., C E., S F. X.: A frequency analysis of light transport. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2005) 24*, 3 (Aug. 2005).

[DKNY95] D Y., K K., N H., Y - H.: A quick rendering method using basis functions for interactive lighting design. *Computer Graphics Forum (Proc. of EUROGRAPHICS 1995) 14*, 3 (Sept. 1995).

[DS06] D C., S M.: Splatting indirect illumination. In *Interactive 3D Graphics 2006* (2006).

[GSCH93] G S. J., S P., C M. F., H - P.: Wavelet Radiosity. In *SIGGRAPH '93* (1993), pp. 221–230.

[GSHG98] G G., S P., H P. M., G - D. P.: The irradiance volume. *IEEE Computer Graphics and Applications 18*, 2 (March/April 1998), 32–43.

[HPB06] H M., P F., B K.: Direct-to-indirect transfer for cinematic relighting. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2006) 25*, 3 (Aug. 2006).

[HSA91] H P., S D., A L.: A Rapid Hierarchical Radiosity Algorithm. *Computer Graphics (Proc. of SIGGRAPH '91) 25*, 4 (July 1991).

[KAMJ05] K A. W., A -M¨ T., J H. W.: Precomputed local radiance transfer for real-time lighting design. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2005) 24*, 3 (Aug. 2005).

[Kel97] K A.: Instant radiosity. In *SIGGRAPH '97* (1997), pp. 49–56.

[Leh04] L J.: *Foundations of Precomputed Radiance Transfer*. Master's thesis, Helsinki University of Technology, Sept. 2004.

[LMM05] L E. B., M K.-L., M N.: Calculating hierarchical radiosity form factors using programmable graphics hardware. *Journal of Graphics Tools 10*, 4 (2005).

[LSSS04] L X., S P.-P. J., S H.-Y., S J.: All-frequency precomputed radiance transfer for glossy objects. In *Rendering Techniques (Proc. of EG Symposium on Rendering)* (2004), pp. 337–344.

[NPG05] N M., P S. N., G V.: Real-time global illumination on GPUs. *Journal of Graphics Tools 10*, 2 (2005).

[NRH03] N R., R R., H P.: All-frequency shadows using non-linear wavelet lighting approximation. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2003) 22*, 3 (July 2003), 376–381.

[NRH04] N R., R R., H P.: Triple product wavelet integrals for all-frequency relighting. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2004) 23*, 3 (Aug. 2004), 477–487.

[PBMH02] P T. J., B I., M W. R., H - P.: Ray tracing on programmable graphics hardware. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2002) 21*, 3 (July 2002), 703–712.

[PDC*03] P T. J., D C., C M., J H. W., H P.: Photon mapping on programmable graphics hardware. In *Graphics Hardware 2003* (2003), pp. 41–50.

[SC97] S P., C K.: A low distortion map between disk and square. *Journal of Graphic Tools 2*, 3 (1997), 45–52.

[SDS96] S E. J., D R T. D., S D. H.: *Wavelets for Computer Graphics*. Morgan Kaufmann, 1996.

[SHHS03] S P.-P., H J., H J., S J.: Clustered principal components for precomputed radiance transfer. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2003) 22*, 3 (2003).

[SKS02] S P.-P., K J., S J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2002) 21*, 3 (July 2002).

[SLSS03] S P.-P., L X., S H.-Y., S J.: Bi-scale radiance transfer. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2003) 22*, 3 (2003).

[SWS05] S W J. S., S P.: RPU: A programmable ray processing unit for realtime ray tracing. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2005) 24*, 3 (Aug. 2005).

[WSB01] W I., S P., B C.: Interactive distributed ray tracing of highly complex models. In *Rendering Techniques 2001 (Proc. EG Workshop on Rendering)* (2001).

[WTL06] W R., T J., L D.: All-frequency relighting of glossy objects. *ACM Transactions on Graphics (to appear)* (2006).

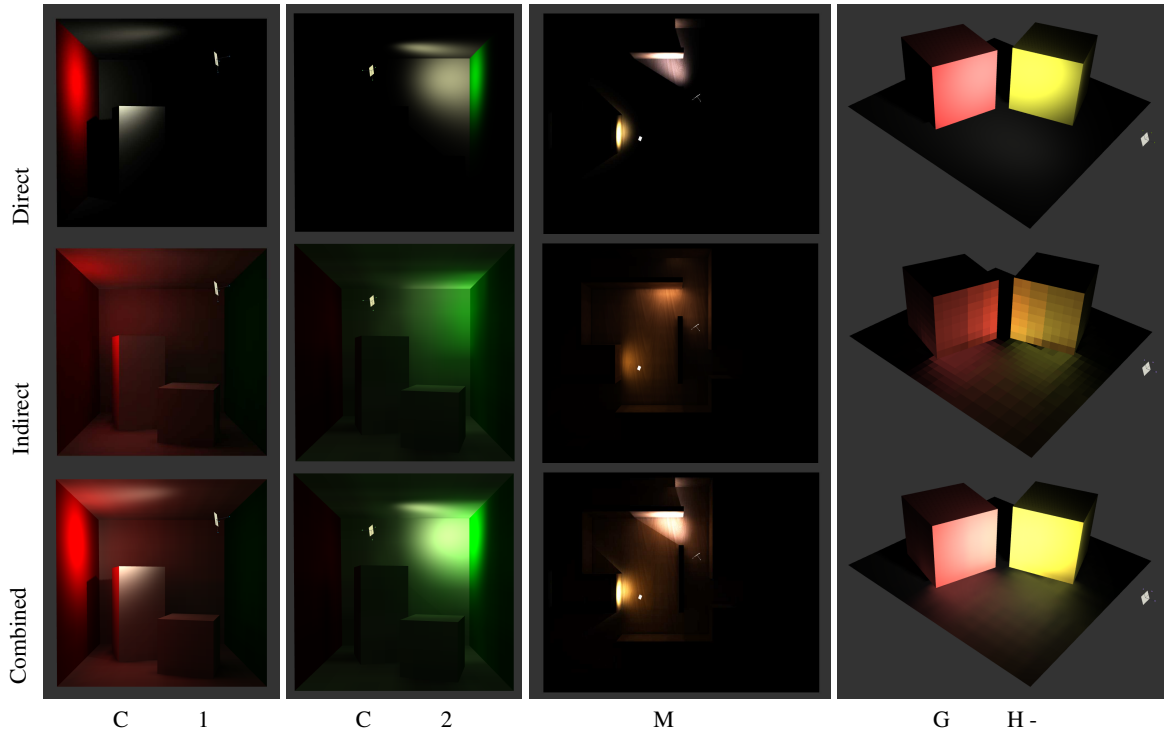


Figure 10: *Example results on our test scenes. In the glossy case (right column), we show the indirect illumination non-filtered, and the composited image using bi-linear filtering.*

5.
Conclusion et perspectives

Nous avons développé trois thèmes principaux dans ce mémoire : la simulation de l'éclairage par des méthodes multi-échelles à éléments finis, la détermination des caractéristiques de la fonction d'éclairage, et la simulation en temps-réel ou interactif de certains effets lumineux.

Nous allons tenter de résumer notre contribution dans chaque thème :

- En ce qui concerne la *simulation de l'éclairage par éléments finis*, nous avons montré l'efficacité de la représentation hiérarchique, y compris avec des fonctions d'ondelettes d'ordre élevé. Nous avons également montré combien les méthodes par éléments finis sont dépendantes du maillage original, et nous avons développé une méthode pour s'affranchir des limitations de ce maillage. Enfin, nous avons montré comment combiner les ondelettes d'ordre élevé avec un maillage de discontinuité.
- Pour l'*analyse des propriétés de la fonction d'éclairage*, nous avons développé une méthode pour prédire le contenu fréquentiel local de l'éclairage, pour chaque interaction, en fonction des obstacles rencontrés. Nous avons également développé une méthode pour le calcul des dérivées de la fonction d'éclairage.
- Enfin, dans le domaine du *rendu temps-réel*, nous avons développé des méthodes pour la simulation en temps réel de certains effets lumineux : ombres douces, réflexions spéculaires, éclairage indirect.

Ces travaux ont rencontré plusieurs limitations ou difficultés, et posent un certain nombre de problèmes intéressants à résoudre.

Ainsi, les méthodes par éléments finis, même hiérarchiques, sont fortement liées au maillage employé pour représenter la scène. Cette limitation est inhérente à la représentation par éléments finis, même si plusieurs méthodes ont été développées qui permettent de s'en affranchir partiellement (*clustering*, *face-clustering*, *instantiation*, *virtual mesh*...). D'autre part, une partie trop importante du temps de calcul est consacrée à des effets qui sont importants pour l'aspect visuel de la scène, comme les frontières d'ombre, mais moins importants pour le calcul de l'éclairage indirect.

L'analyse fréquentielle de la fonction d'éclairage ouvre la voie à de nombreuses études futures. Nous avons développé un outil pour la prédiction du comportement fréquentiel de l'éclairage en chaque point. Il reste beaucoup de recherches à faire dans ce domaine, à la fois pour le calcul effectif du contenu fréquentiel dans la scène et pour l'emploi de ces fréquences dans les méthodes de simulation de l'éclairage. Il est important que le gain en temps de calcul grâce à l'utilisation du contenu fréquentiel soit supérieur au temps pris pour calculer ce contenu fréquentiel.

Enfin, l'utilisation de cartes graphiques pour la simulation d'effets lumineux est un domaine très prometteur. En simulant certains effets lumineux sur la carte graphique, il est possible d'augmenter le réalisme des simulations d'éclairage tout en diminuant le temps de calcul. En même

temps, ces cartes programmables ont des limitations : elles fonctionnent sur un modèle SIMD, sans communications possibles entre les différents processeurs, avec un nombre limité d'instructions... Les algorithmes qui pourront le mieux exploiter la puissance de ces cartes seront ceux qui s'adaptent à ces limitations. En général, on pourra plutôt les utiliser pour simuler des phénomènes locaux ou semi-locaux.

5.1 Perspectives

Notre but, pour nos travaux futurs, est d'obtenir une simulation photoréaliste de l'éclairage global dans une scène quelconque, en temps-réel. Pour atteindre ce but, nous allons poursuivre plusieurs directions de recherche :

- D'une part, il est nécessaire de pouvoir simuler l'ensemble des phénomènes liés à l'éclairage direct en temps-réel. Les calculs d'éclairage direct avec des fonctions de réflectance quelconques et des sources étendues, les calculs d'ombre douce, les réflexions sur des surfaces semi-spéculaires sont quelques uns des phénomènes locaux ou semi-locaux que nous voulons pouvoir simuler de façon photoréaliste en temps-réel.
- D'autre part, un certain nombre de phénomènes liés à l'éclairage indirect ne sont observables qu'à proximité immédiate des objets, comme l'occlusion ambiante causée par un objet, ou les réflexions causées par des BRDF semi-diffuses. Pour ces phénomènes, il faudrait attacher aux objets mobiles une zone d'influence, à l'intérieur de laquelle on calculerait l'effet. Cette zone d'influence pourrait porter un certain nombre de coefficients pré-calculés pour la simulation.
- Dans la simulation de l'éclairage, on dispose généralement de plusieurs algorithmes pour simuler un effet donné, ou d'un ensemble de paramètres pour un algorithme donné. On a donc des choix à faire, et pour guider ces choix, nous proposons d'utiliser notre analyse fréquentielle de l'éclairage. On pourrait alors choisir l'algorithme le mieux adapté, ou encore limiter l'échantillonnage pour les phénomènes à basse fréquence.
- Cette analyse fréquentielle de l'éclairage a également des applications pour les simulations *offline* de l'éclairage. Notre approche devrait permettre de guider des calculs de simulation de l'éclairage par lancer de photons, ou encore d'adapter l'échantillonnage spatial dans les méthodes de *Precomputed Radiance Transfer*.

Liste des publications

Journaux internationaux

[1] Mattias Malmer, Fredrik Malmer, Ulf Assarsson et Nicolas Holzschuch. Fast Precomputed Ambient Occlusion for Proximity Shadows. *Journal of Graphics Tools*, 2007. (à paraître).

[2] Lionel Atty, Nicolas Holzschuch, Marc Lapierre, Jean-Marc Hasenfratz, Chuck Hansen et François Sillion. Soft shadow maps: Efficient sampling of light source visibility. *Computer Graphics Forum*, vol. 25, n° 4, décembre 2006.

[3] David Roger and Nicolas Holzschuch. Accurate specular reflections in real-time. *Computer Graphics Forum (Proceedings of Eurographics 2006)*, vol. 25, n° 3, septembre 2006.

[4] Aurélien Martinet, Cyril Soler, Nicolas Holzschuch et François Sillion. Accurate detection of symmetries in 3D shapes. *ACM Transactions on Graphics*, vol. 25, n° 2, avril 2006.

[5] Frédo Durand, Nicolas Holzschuch, Cyril Soler, Eric Chan et François Sillion. A frequency analysis of light transport. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2005)*, vol. 24, n° 3, août 2005.

[6] Cyrille Damez, Nicolas Holzschuch et François Sillion. Space-time hierarchical radiosity with clustering and higher-order wavelets. *Computer Graphics Forum*, vol. 23, n° 2, avril 2004.

[7] Jean-Marc Hasenfratz, Marc Lapierre, Nicolas Holzschuch et François Sillion. A survey of real-time soft shadows algorithms. *Computer Graphics Forum*, vol. 22, n° 4, décembre 2003.

[8] François Cuny, Laurent Alonso et Nicolas Holzschuch, A novel approach makes higher order wavelets really efficient for radiosity, *Computer Graphics Forum (Proceedings of Eurographics 2000)*, vol. 19, n° 3, septembre 2000.

[9] Laurent Alonso et Nicolas Holzschuch, Using graphics hardware to speed-up your visibility queries, *Journal of Graphics Tools*, vol. 5, n° 2, avril 2000.

[10] François Cuny, Laurent Alonso, Christophe Winkler et Nicolas Holzschuch, Radiosité à base d'ondelettes sur des mailles quelconques, *Revue internationale de CFAO et d'informatique graphique*, vol. 14, n° 1, octobre 1999.

[11] Nicolas Holzschuch et François Sillion, An exhaustive error-bounding algorithm for hierarchical radiosity, *Computer Graphics Forum*, vol. 17, n° 4, décembre 1998.

Conférences ou workshops internationaux avec comité de lecture et publication des actes

[12] Janne Kontkanen, Emmanuel Turquin, Nicolas Holzschuch et François Sillion. Wavelet radiance transport for interactive indirect lighting. In *Rendering Techniques 2006 (Eurographics Symposium on Rendering)*, juin 2006.

[13] Nicolas Holzschuch and Laurent Alonso. Combining higher-order wavelets and discontinuity meshing: a compact representation for radiosity. In *Rendering Techniques 2004 (Eurographics Symposium on Rendering)*, juin 2004.

[14] Nicolas Holzschuch, François Cuny et Laurent Alonso, Wavelet Radiosity on Arbitrary Planar Surfaces. In *Rendering Techniques 2000 (Eurographics Workshop on Rendering)*, juin 2000.

[15] Nicolas Holzschuch et François Sillion, Accurate Computation of the Radiosity Gradient for Constant and Linear Emitters. In *Rendering Techniques '95 (Eurographics Workshop on Rendering)*, juin 1995.

[16] Nicolas Holzschuch, François Sillion et George Drettakis, An Efficient Progressive Refinement Strategy for Hierarchical Radiosity, dans *Photorealistic Rendering Techniques (Eurographics Workshop on Rendering)*, juin 1994.

Conférences sans comité de lecture ou sans publication des actes

[17] David Roger and Nicolas Holzschuch. Réflexions spéculaires en temps réel sur des surfaces lisses. In *AFIG '05 (Actes des 18èmes journées de l'AFIG)*, novembre 2005.

[18] Aurélien Martinet, Cyril Soler, Nicolas Holzschuch et François Sillion. Accurately detecting symmetries of 3D shapes. In *Symposium on Geometry Processing 2005, Poster Session*, juin 2005.

[19] Aurélien Martinet, Cyril Soler, Nicolas Holzschuch et François Sillion. Organisation automatique de scenes 3D. In *AFIG '04 (Actes des 17èmes journées de l'AFIG)*, novembre 2004.

[20] Jean-Marc Hasenfratz, Marc Lapierre, Nicolas Holzschuch et François Sillion. A survey of real-time soft shadows algorithms. In *Eurographics State-of-The-Art Reports*, septembre 2003.

[21] Cyrille Damez, Nicolas Holzschuch et François Sillion, Space-Time Hierarchical Radiosity with Clustering and Higher-Order Wavelets, dans *Eurographics 2001 Short Presentations*, Manchester, Royaume-Uni, septembre 2001.

[22] François Cuny, Laurent Alonso, Christophe Winkler et Nicolas Holzschuch, Gestion efficace des polygones complexes pour la radiosité. In *AFIG 1998 (Actes des 6^{e}journées de l'AFIG)*, novembre 1998.

[23] Nicolas Holzschuch, Current trends in research in visualisation, *Intramural Conference on Digital Imaging*, University of Cape Town, Cape Town, 1996.

[24] Nicolas Holzschuch, Un critère de raffinement efficace pour la radiosité et la radiosité hiérarchique, poster aux Journées Jacques Cartier, Saint-Etienne, 1995.

[25] Nicolas Holzschuch, Les ondelettes en synthèse d'image, *Colloquium IMAG sur les ondelettes*, IMAG, Grenoble, 1994.

[26] Nicolas Holzschuch, Vers une radiosité évoluant en temps réel, *Journées Groplan*, Nantes, novembre 1992.

Rapports internes et articles soumis

[27] Mattias Malmer, Fredrik Malmer, Ulf Assarsson et Nicolas Holzschuch. Fast Precomputed Ambient Occlusion for Proximity Shadows. Raport de recherche RR-5779, INRIA, décembre 2005.

[28] Lionel Atty, Nicolas Holzschuch, Marc Lapierre, Jean-Marc Hasenfratz, Chuck Hansen et François Sillion. Soft shadow maps: Efficient sampling of light source visibility. Raport de recherche RR-5750, INRIA, novembre 2005.

[29] Aurélien Martinet, Cyril Soler, Nicolas Holzschuch et François Sillion. Accurately detecting symmetries of 3D shapes. Raport de recherche RR-5692, INRIA, septembre 2005.

[30] Nicolas Holzschuch, *Le Contrôle de l'erreur dans la méthode de radiosité hiérarchique*, Thèse, Université Joseph Fourier – Grenoble I, mars 1996.

[31] Nicolas Holzschuch, *Synthèse d'images et radiosité hiérarchique*, mémoire de Magistère, Magistère de Mathématiques Fondamentales et Appliquées et d'Informatique, École Normale Supérieure, Université Paris XI, novembre 1994.

[32] Nicolas Holzschuch, *Vers une radiosité évoluant en temps réel*, memoire de DEA, DEA Informatique, Mathématique et Applications, Université Paris XI, septembre 1992.