



Du Rendu de Haute Qualité à l'interactivité

George Drettakis

► To cite this version:

George Drettakis. Du Rendu de Haute Qualité à l'interactivité. Interface homme-machine [cs.HC]. Université Joseph-Fourier - Grenoble I, 1999. tel-00004826

HAL Id: tel-00004826

<https://theses.hal.science/tel-00004826>

Submitted on 18 Feb 2004

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

MÉMOIRE D'HABILITATION

Présenté par :

George Drettakis

pour obtenir le grade de

**L'Habilitation à Diriger des Recherches
de**

**L'UNIVERSITÉ JOSEPH FOURIER
(Grenoble I)**

Spécialité : Informatique

Du Rendu de Haute Qualité à l'interactivité

Date de soutenance : 13 septembre 1999

Composition du jury :

Pr. Jacques VOIRON

Pr. Eugene FIUME

Pr. Peter SHIRLEY

Pr. Jean-Daniel BOISSONNAT

Pr. Jean-Claude PAUL

Pr. Claude PUECH

Habilitation préparée au sein du projet iMAGIS
Projet commun entre le CNRS, l'INRIA, l'INPG et l'UJF.

Remerciements

Je tiens avant tout à remercier Claude Puech pour son aide et son soutien pendant ces années à iMAGIS. J'ai eu l'occasion de travailler dans six laboratoires de recherche dans six pays différents, je ne peux que constater que Claude parvient à offrir à son équipe un environnement de recherche exceptionnel sur le plan international. Il sait protéger ses chercheurs de l'administration et les laisser travailler dans les meilleures conditions possibles. Ses conseils et nos nombreuses discussions m'ont permis d'apprendre petit à petit le fonctionnement très complexe de la recherche française. Enfin, notre co-direction de thèses a également été une expérience très enrichissante. Pour tout cela je lui suis très reconnaissant.

Je remercie également tous les autres membres de mon jury pour leur participation, leurs remarques et les échanges intéressantes : les rapporteurs Peter Shirley, Eugene Fiume et Jean-Daniel Boissonnat, l'examinateur Jean-Claude Paul et le président Jacques Voiron.

Mon travail à iMAGIS est marqué par mes nombreuses collaborations avec François Sillion, comme le témoignent nos nombreuses publications. Grâce à lui, j'ai pu aborder plusieurs nouveaux domaines de recherche en images de synthèse. Ses grandes connaissances du domaine et sa grande expérience sont des sources inépuisables d'inspiration. Travailler ensemble fut un plaisir, et j'espère que cela continuera longtemps.

Le plus grand plaisir de mon travail à iMAGIS a indéniablement été l'encadrement d'étudiants. Je remercie mes thésards Frédo Durand, Céline Loscos, Eric Paquette et Xavier Granier, grâce auxquels j'ai commencé à apprendre combien il est difficile d'encadrer. Ils m'ont tous appris beaucoup de choses ; j'espère avoir pu les aider pour la suite de leurs carrières.

Le travail avec Bruce Walter pendant son post-doc a été un vrai plaisir et une expérience très enrichissante. Ses connaissances et son talent m'ont beaucoup marqué.

Pierre Poulin est un ami de long date ; travailler avec lui s'est avéré un vrai plaisir. Ses visites fréquentes en France sont une source d'inspiration professionnelle et de détente personnelle.

La vaste majorité du travail présenté ici est le fruit de collaborations. Je tiens à remercier tous mes co-auteurs : Frédéric Cazals, Nicolas Holszchuch, Cyril Soler, Benoît Bodelet, Cyrille Damez, Marie-Claude Frasson et Jean-Marc Hasenfratz d'iMAGIS, Luc Robert et Sylvain Bougnoux de Sophia, Hartmut Schirmacher, Katja Daubert, Marc Stamminger et Annette Scheel (d'ex-Erlangen et actuellement Max-Planck), Fédé Perez-Cazorla de Girona et enfin Steve Parker de l'université d'Utah. Cela a été un grand plaisir de travailler avec vous tous.

Tous les iMAGISIens méritent d'être mentionnés, mais faute de place je me limite à ceux qui ont eu à beaucoup me supporter : François Faure, Mathieu Desbrun, Alexis Lamouret et Rachel Orti, à la fois pour le boulot et pour les pauses. Merci à Jean-Dominique Gascuel et Jean-Luc Douvillé, sans lesquels rien ne marcherait à iMAGIS, ainsi qu'à Patricia Mathieu et Rosine Ah-Tchou qui nous ont sauvé la vie tant de fois.

Une partie du travail s'est fait en postdoc ERCIM ; je remercie Peret Brunet et Xavier Pueyo en Espagne et Simon Gibson, Christian Breiteneder et Costas Arapis au GMD de leur accueil.

Je dois beaucoup à Eugene Fiume, mon ancien directeur de thèse qui m'a si bien enseigné le métier de « directeur » d'étudiant et qui m'a soutenu dans bien des moments difficiles. Merci aussi à Dionisis Tsichritzis, sans qui je n'aurais jamais fait d'informatique.

Enfin, je n'aurais jamais eu la force de faire tout ce qui se trouve dans ce document sans le soutien et l'amour de Laurence et la joie que nous donne notre petite Anna. Merci d'avoir supporté si patiemment tant de longues nuits, les week-ends d'absence au labo, et le travail pendant les vacances de Noël.

Table des matières

1	Introduction	3
1.1	Structure du Mémoire	3
2	Calcul de haute précision en utilisant des méthodes analytiques de visibilité	5
2.1	Echantillonnage structuré et Maillages de discontinuités	5
2.1.1	Éclairage Global	8
2.1.2	Éclairage Direct pour des Scènes Dynamiques	9
2.2	Structures Analytiques pour la Visibilité Globale	9
2.2.1	Complexe de visibilité	9
2.2.2	Squelette de visibilité	11
2.2.3	Éclairage global et Squelette de Visibilité	13
2.3	Discussion	14
2.4	Articles	15
2.4.1	Structured Penumbral Irradiance Computation (IEEE TVCG'96)	17
2.4.2	Simplifying the Representation of Radiance from Multiple Emitters (EGRW'94)	35
2.4.3	Accurate Visibility and Meshing Calculations for Hierarchical Radiosity (EGRW'96)	51
2.4.4	Interactive High-Quality Soft Shadows in Scenes with Moving Objects (EG'97)	67
2.4.5	The 3D Visibility Complex, a new approach to the problems of accurate visibility (EGRW'96)	83
2.4.6	The Visibility Skeleton : A Powerful and Efficient Multi-Purpose Global Visibility Tool (SIGGRAPH'97)	97
2.4.7	The 3D Visibility Complex : a unified data-structure for global visibility of scenes of polygons and smooth objects (CCCG'97)	111
2.4.8	Fast and Accurate Hierarchical Radiosity Using Global Visibility (ACM TOG'99)	119
3	Éclairage pour des scènes de grande complexité	161
3.1	Structures de Données Hiérarchiques pour le lancer de Rayons	161
3.2	Algorithmes Hiérarchiques d'éclairage	163
3.2.1	Étude et Améliorations de la Radiosité Hiérarchique	163
3.2.2	Visibilité Multi-résolution	165
3.2.3	Réduction de la mémoire utilisée par la Radiosité Hiérarchique	166
3.3	Traitement de la Complexité Photométrique	167
3.3.1	L'éclairage non-diffus	168
3.3.2	L'éclairage de scènes d'extérieur	169
3.4	Discussion	169
3.5	Articles	170
3.5.1	Filtering, Clustering and Hierarchy Construction (EG'95)	171
3.5.2	A Light Hierarchy for Fast Rendering of Scenes with Many Lights (EG'98)	185
3.5.3	An Efficient Progressive Refinement Strategy for Hierarchical Radiosity (EGRW'94)	199

3.5.4	Feature-Based Control of Visibility Error : A Multiresolution Clustering Algorithm for Global Illumination” (SIGGRAPH’95)	217
3.5.5	Controlling Memory Consumption of Hierarchical Radiosity with Clustering (GI’99)	227
3.5.6	A Practical Analysis of Clustering Strategies for Hierarchical Radiosity (EG’99) .	237
3.5.7	A Clustering Algorithm for Radiance Calculation in General Environments (EGRW’95)	253
3.5.8	Efficient Glossy Global Illumination with Interactive Viewing (GI’99)	267
3.5.9	Hierarchical Lighting Simulation for Outdoor Scenes (EGRW’97)	277
4	Algorithmes interactifs :	
	rendu à base d’images, rendu haute qualité et	
	environnements mixtes, réels-virtuels	291
4.1	Rendu Interactif à Base d’Images	291
4.2	Rendu Interactif de Haute Qualité	293
4.2.1	Radiosité Interactive pour des Scènes Dynamiques	293
4.2.2	Rendu Interactif par Tracer de Rayon	294
4.3	Réalité Augmentée	296
4.4	Discussion	300
4.4.1	Rendu Haute Qualité	300
4.4.2	Réalité Augmentée	301
4.5	Articles	301
4.5.1	Efficient Impostor Manipulation for Real-Time Visualization of Urban Scenery (EG’97)	303
4.5.2	Interactive Update of Global Illumination Using A Line Space Hierarchy (SIGGRAPH’97)	317
4.5.3	Interactive Rendering using the Render Cache (EGRW’99)	327
4.5.4	Interactive Common Illumination for Computer Augmented Reality (EGRW’97) .	343
4.5.5	Interactive Virtual Relighting and Remodeling of Real Scenes (EGRW’99)	359
5	Conclusion et Perspectives	375
5.1	Perspectives	376
	Bibliographie	377

Introduction

La synthèse d'images a connu au cours des dernières années un développement impressionnant. L'amélioration du matériel d'une part, et la conception des nouveaux algorithmes d'autre part, nous permettent aujourd'hui de synthétiser des images réalistes de très haute qualité.

Malgré cela, beaucoup de problèmes restent à résoudre. Dans ce mémoire nous allons nous intéresser principalement aux problèmes du *rendu* des images, c'est-à-dire la génération des images depuis un modèle géométrique bien défini. Les domaines qui nous intéressent sont la détermination de la visibilité et le rendu haute qualité, le calcul de l'éclairage sur les objets du modèle, le rendu interactif de ces images y compris pour la réalité augmentée, (les scènes mixtes, contenant à la fois des objets réels et virtuels). Ces trois domaines regroupent l'ensemble de nos travaux pendant cette période.

Dans le domaine de la visibilité analytique nous avons étendu le travail de la thèse [Dre94b] sur les maillages de discontinuité ainsi que sur l'échantillonnage structuré de l'éclairage. Ensuite nous avons introduit une nouvelle structure de visibilité globale, le *Complexe de Visibilité*. Le complexe permet de coder toutes les informations de visibilité d'une scène dans l'espace des segments libres maximaux, en regroupant les droites qui voient le même objet. Cette structure a été ensuite simplifiée, donnant lieu au *Squelette de Visibilité*, qui est une structure bien plus légère en mémoire, et plus facile à implémenter. Le squelette a été d'ailleurs utilisé pour une application de simulation d'éclairage.

Nos travaux sur le calcul de l'éclairage pour des scènes très complexes, se penchent sur deux aspects : le premier concerne la complexité géométrique des scènes et consiste à améliorer les algorithmes de radiosité hiérarchique ; le deuxième essaie de surmonter les problèmes liés à la simulation de l'éclairage dans des environnements non-diffus, en utilisant une représentation directionnelle.

Nos travaux plus récents se concentrent sur les problèmes liés au rendu interactif. Nous avons développé un algorithme à base d'images pour améliorer la visualisation des scènes très complexes. Pour un rendu interactif de haute qualité nous avons introduit deux approches, une basée sur la radiosité en utilisant des structures de données qui permettent une mise à jour rapide de l'éclairage, et une autre basée sur le lancer de rayons qui présente à l'utilisateur une image approximative mais avec un temps de réaction interactif. Pour la réalité augmentée enfin, nous avons développé deux approches. La première permet l'ajout et la manipulation d'objets virtuels dans une scène réelle. La deuxième permet également de changer les conditions d'éclairage, d'ajouter des lampes virtuelles et même d'enlever des objets réels, tout en gardant un temps de mise à jour interactif.

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 les détails des travaux. Nous présentons

ensuite les conclusions et quelques perspectives pour l'avenir.

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

Calcul de haute précision en utilisant des méthodes analytiques de visibilité

Les algorithmes d'éclairage, connus comme « algorithmes de radiosité », ont réussi d'une part à produire des images contenant des ombres douces causées par des sources étendues (non-ponctuelles), et d'autre part à simuler de façon satisfaisante des effets secondaires causés par la diffusion de l'éclairage.

Les algorithmes de radiosité, malgré leur succès, essaient surtout de calculer une image *rapidement*. Le calcul de la visibilité, c'est-à-dire la partie d'une source qui est visible depuis un récepteur, est central dans tout calcul d'éclairage. La précision de ce calcul est déterminante pour la simulation d'éclairage de haute qualité. De plus, ce calcul est généralement très coûteux : sa complexité est au moins linéaire par rapport au nombre n d'objets contenus dans la scène, pour *chaque* échange source-récepteur, et il peut en avoir n^2 .

Dans ce chapitre, nous présentons nos travaux qui portent sur des algorithmes d'éclairage qui utilisent des structures dites « analytiques », c'est-à-dire qui donnent une réponse exacte ou très précise à la question : « quelle est la partie visible d'une source depuis un récepteur ? ». Dans ce cadre nous avons été amenés à développer de nouvelles structures de visibilité globale.

Le résultat de l'utilisation de ces méthodes est le calcul d'images de haute précision et de grande qualité en ce qui concerne l'éclairage.

2.1 Echantillonnage structuré et Maillages de discontinuités

Une façon efficace de représenter la partie *visible* d'une source étendue (non ponctuelle) depuis un point sur un récepteur est l'utilisation de *maillages de discontinuité*. Un maillage de discontinuité par rapport à une source S divise l'environnement en cellules (ou *faces*) qui ont une vue homogène de la source S . Cette division résulte de la propagation des *surfaces de discontinuité* qui génèrent des changements de la structure de la partie visible de la source.

Les deux principaux types de surfaces de discontinuité sont présentés dans la Figure 2.1, calculées par notre système de maillage de discontinuité.

La vue homogène de la source depuis une face, que l'on appelle « projection-arrière », donne la capacité de calculer exactement la partie visible de la source d'un point arbitraire dans une scène, sans avoir besoin de recalculer la visibilité chaque fois.

L'algorithme que nous avons développé [DF94] a étendu le travail de Heckbert [Hec92], en ajoutant le traitement des surfaces de discontinuité quadriques causées par l'interaction de trois arêtes de l'environnement (voir Figure 2.1(b)), et l'utilisation d'une subdivision de l'espace pour propager d'une façon efficace les surfaces de discontinuité. Étant donné ce maillage, le calcul de la projection-arrière est possible d'une façon très efficace avec l'introduction d'un algorithme local, qui ne dépend pas directement de la com-

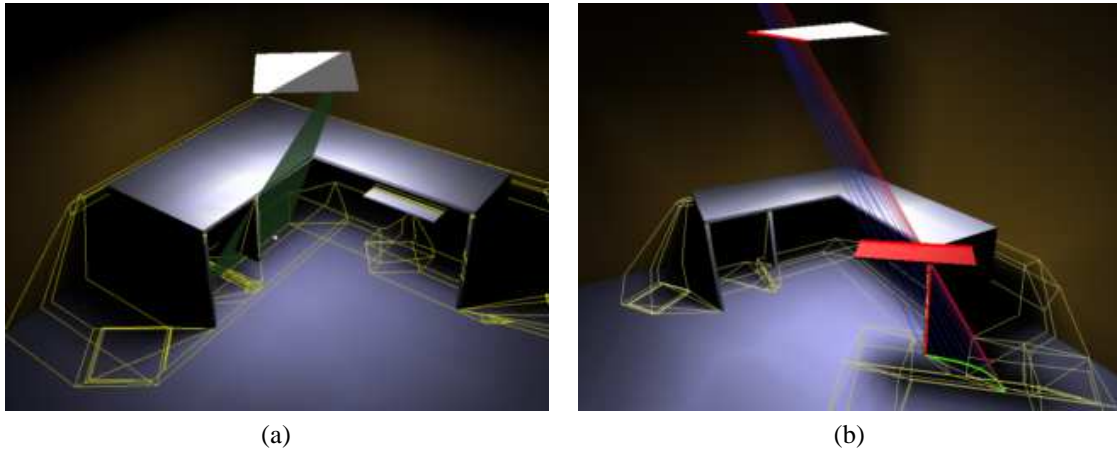


FIG. 2.1: (a) Une surface de discontinuité, créée par une arête et un sommet (EV pour edge-vertex en anglais). (b) Une surface de discontinuité EEE (triple arête).

plexité de la scène. L'utilisation des statistiques des environnements typiques a permis de déterminer que les problèmes de complexité théoriquement insurmontables, ne sont pas gênants en pratique pour beaucoup d'environnements d'intérieur (bureaux etc.).

Les maillages calculés de cette manière ont été ensuite utilisés pour le calcul des images exactes (analytiques), qui sont utiles comme images de référence, et aussi pour créer des visualisations interactives avec des images contenant des ombres de haute qualité. Pour calculer ces images rapidement nous avons utilisé des interpolants quadratiques pour représenter la lumière. Un exemple d'une telle image est présenté dans la Figure 2.2.



FIG. 2.2: Image calculée par l'algorithme [DF94] des maillages de discontinuité. Cette scène comporte 1000 polygones d'origine.

Les maillages calculés sont indispensables pour la création d'une image exacte, pour laquelle aucune approximation n'est permise. Par contre, il est souvent suffisant de faire une approximation quand la précision requise n'est pas très importante.

Dans la poursuite des travaux après la thèse, nous avons développé une méthode pour la simplification des maillages, basée sur des méthodes purement géométriques. Les expériences ont montré qu'il est possible d'avoir une représentation d'assez haute qualité de l'éclairage en simplifiant les maillages d'une

façon assez importante (40-60 %) par l'enlèvement d'arêtes peu utiles du maillage. Nous avons également montré qu'il est suffisant d'utiliser des interpolants linéaires ou mixtes linéaire/quadratique en combinant les techniques pour les méthodes sans ombres [DF93] avec l'approche des maillages [DF96].

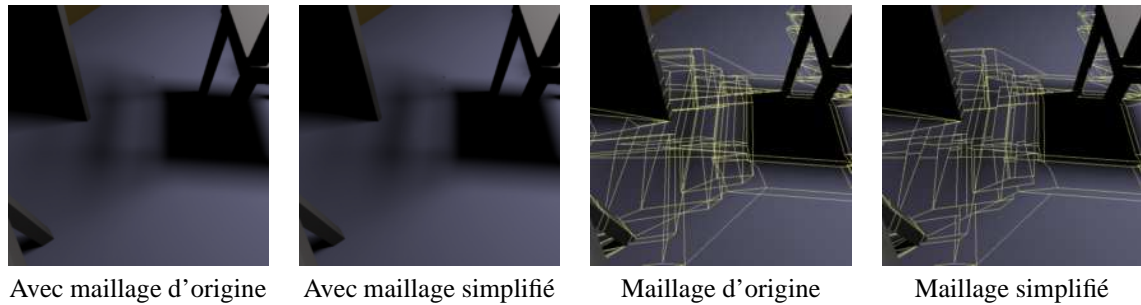


FIG. 2.3: Comparaison de l'image rendu avec le maillage d'origine et l'image avec le maillage simplifié de 56% en utilisant la méthode de [DF96].

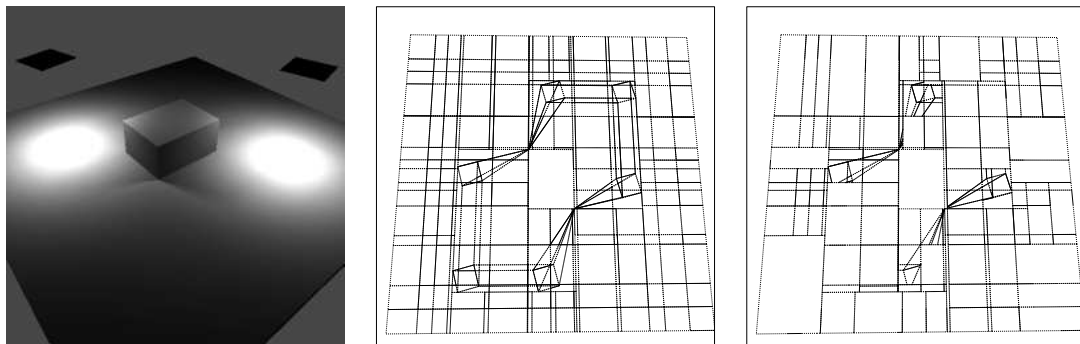


FIG. 2.4: (a) Une scène comportant un objet simple et deux sources lumineuses. (b) Le maillage au sol complet, et (c) le maillage simplifié en considérant l'effet des deux sources [Dre94a].

Un exemple de cette méthode est montré dans la Figure 2.3, où nous montrons l'effet de la réduction de nombre d'arêtes du maillage par 56%.

Dans les situations où plusieurs sources éclairent la même scène, il est également possible de simplifier les maillages. Dans ce cas, il arrive souvent que l'éclairage dû à une deuxième source rende invisibles les détails des ombres de la première source, et donc le calcul dépensé pour le premier maillage est gaspillé.

Pour traiter ce problème, nous avons développé une méthode [Dre94a] où l'on calcule d'abord un maillage très simplifié, avec un gain de temps de calcul significatif, et où l'on applique ensuite des critères d'estimation d'erreur pour déterminer si le calcul du maillage complet est nécessaire localement. Cette méthode donne des maillages simplifiés, en considérant l'effet simultané des sources multiples. Un exemple est présenté dans la Figure 2.4.

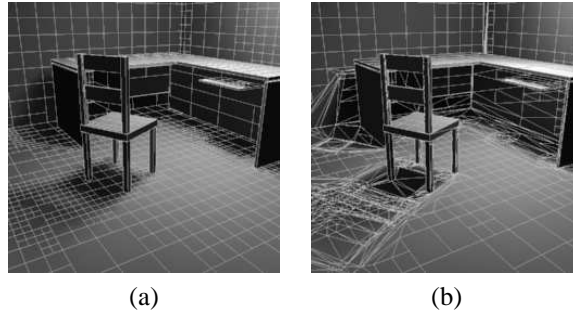


FIG. 2.5: Les maillages respectifs pour le cas de (a) « quadtree » et (b) de maillage de discontinuité.

2.1.1 Éclairage Global

La recherche décrite ci-dessus porte exclusivement sur l'éclairage direct. L'utilisation de la richesse d'information existant dans la structure de la projection-arrière est plus utile encore dans le cadre de la simulation de l'éclairage global.

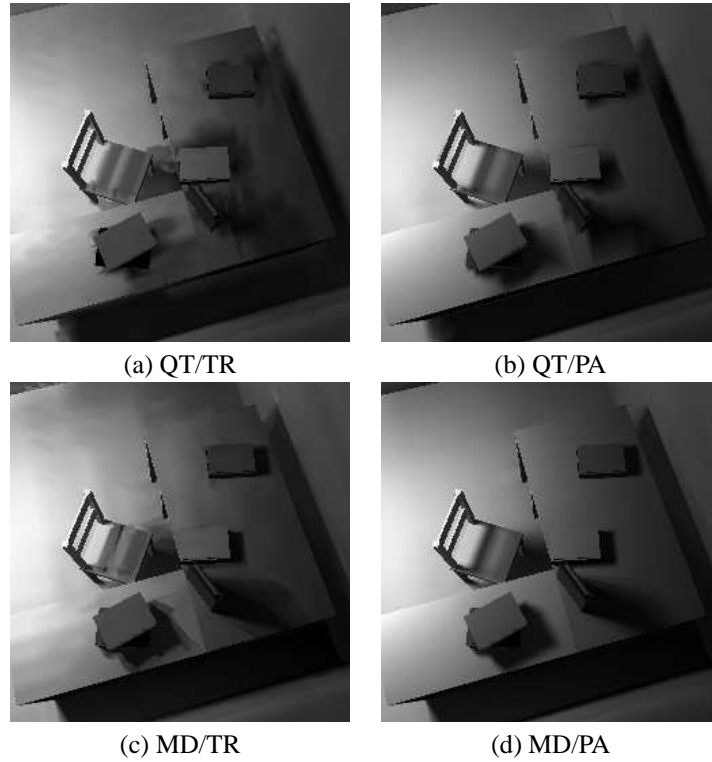


FIG. 2.6: Dans (a)-(d) nous présentons les résultats des différentes configurations. QT signifie maillage « quadtree » et MD maillage de discontinuité. TR signifie que la visibilité est calculée par lancer de rayons, et PA par la projection arrière.

En introduisant une nouvelle structure hiérarchique à base de maillages de discontinuité et de projections-arrières nous avons développé un outil d'investigation des sources d'erreur dans le calcul de la radiosité hiérarchique. Cette étude a permis de déterminer l'importance de la précision des calculs de la visibilité et des conditions pour lesquelles un maillage adapté est aussi important [DS96].

Considérez par exemple les images présentées dans la Figure 2.5. Dans (a) nous présentons un maillage type « quadtree » et (b) le nouveau maillage pour une scène simple. Pour une scène similaire, considérez les résultats présentés de Figure 2.6 (a)-(d). Nous voyons clairement que le meilleur résultat est acquis en utilisant à la fois les maillages de discontinuité (MD) *et* la projection arrière (PA) pour la visibilité. Les maillages de discontinuité sans visibilité exacte par contre, donnent des artefacts très visibles (cas (c) MD/TR).

2.1.2 Éclairage Direct pour des Scènes Dynamiques

Un problème particulièrement important des systèmes d'éclairage actuels est la nécessité de déplacer des objets, traitant ainsi des scènes « dynamiques ». Le déplacement interactif d'un objet dans une scène éclairée en utilisant les maillages de discontinuité n'est pas possible avec les algorithmes existants. Cela est dû au fait que le calcul du maillage est long, même pour des scènes de taille modérée. Dans le cadre du stage DEA de Céline Loscos, que j'ai encadré, nous avons développé un algorithme de mise à jour incrémentale du maillage et des projections arrières qui permet un affichage presque interactif pour des scènes de centaines de polygones, en utilisant la cohérence spatiale et la cohérence de la visibilité [LD97].

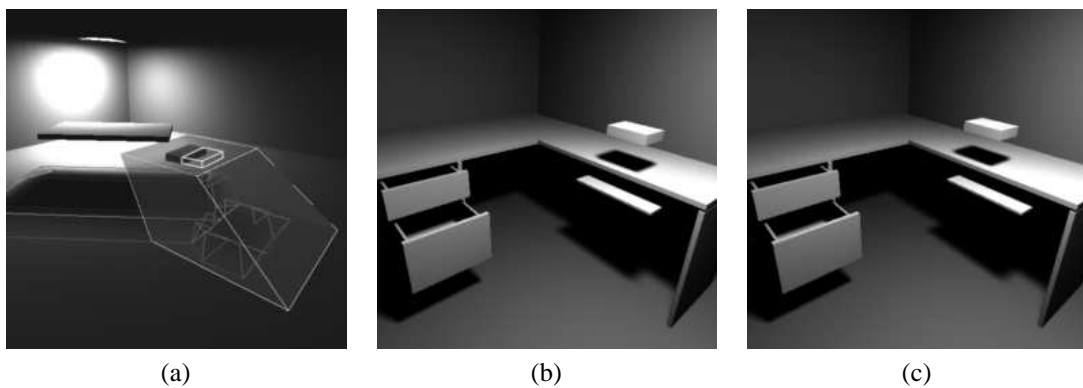


FIG. 2.7: (a) Le volume de déplacement. (b) Le petit objet se déplace au dessus du bureau. Chaque mise à jour nécessite moins de deux secondes.

Pour atteindre des mises à jour interactives, nous utilisons un « volume de déplacement » (voir Figure 2.7(a)). Le déplacement de petits objets (Figure 2.7(b) et (c)) peut être fait avec notre méthode en quelques secondes par image.

En conclusion, la recherche menée sur les maillages de discontinuité a démontré l'importance de la précision des calculs de visibilité dans le cas de l'éclairage direct mais aussi dans le cas global, ainsi que l'importance des maillages adaptés aux frontières des ombres pour le cas des sources étendues.

2.2 Structures Analytiques pour la Visibilité Globale

La généralisation directe du calcul des projections arrières est le développement de structures de données adaptées à la description compacte de la visibilité *globale* dans une scène. Une telle structure est l'objet de la thèse de Frédo Durand que je co-dirige avec Claude Puech.

2.2.1 Complexe de visibilité

Nous avons introduit la structure du « Complexe de Visibilité 3D » dans l'espace de droites. Le complexe est de dimension quatre (une paramétrisation de l'espace de droites) plus une « demi-dimension » pour ordonner l'occultation [DDP96]. L'intuition derrière cette structure est de partitionner l'espace de droites en groupes, chaque groupe contenant les droites qui « voient » le même objet. Pour ceci nous utilisons

l'espace des segments libres maximaux, c'est-à-dire les segments de droites qui ont leurs extrémités sur des objets.

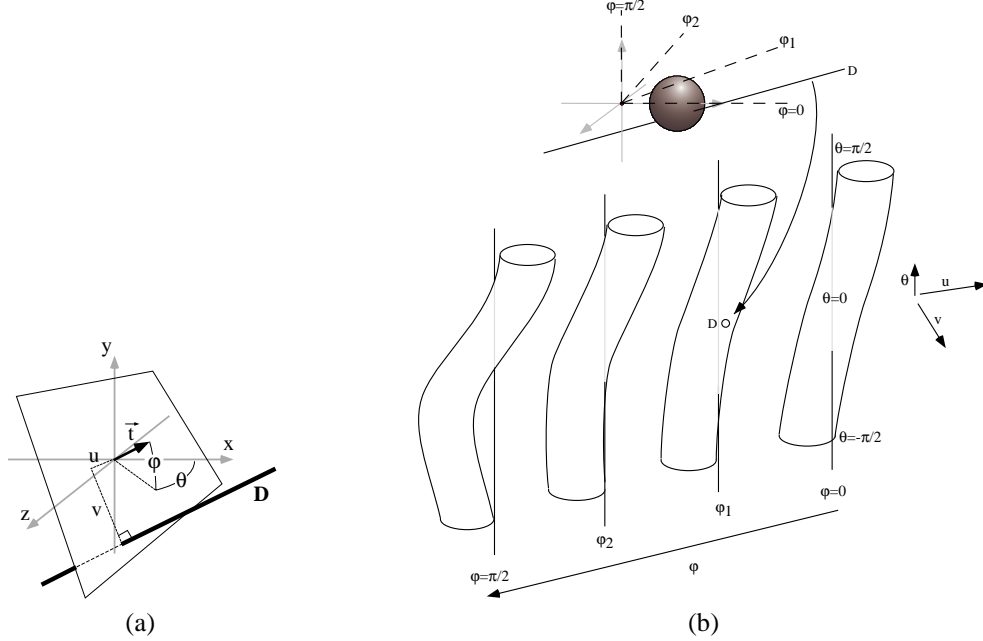


FIG. 2.8: (a) La dualité utilisée pour la représenter l'espace des droites (voir texte). (b) Le volume de tangence d'une sphère, présentée en haut de la figure. Nous considérons les différentes directions de vue en pointillés. En utilisant les tranches de φ , nous pouvons mieux visualiser la dualité. Sur chaque tranche, nous présentons l'axe θ , pour $u = 0, v = 0$. À l'extrême gauche, nous voyons le cas de la droite $\varphi = \frac{\pi}{2}$, qui n'intersecte pas la sphère ; cela est représenté par volume de tangence qui n'est pas coupé par l'axe θ . La droite D (la droite $\varphi = 0$ également), intersecte la sphère ; son point dual est à l'intérieur du volume de tangence (montré par la flèche).

La paramétrisation de droites utilisée est la direction θ, φ de la droite, ainsi que l'intersection u, v sur le plan orthogonal à la droite (voir Figure 2.8(a)). La visualisation de ces structures est particulièrement difficile à cause de ses 4 dimensions et demie. Nous avons développé une représentation en « tranches » selon φ (voir Figure 2.8(b)), qui permet la visualisation de la dualité d'une façon intuitive. Dans cette visualisation, l'ensemble de droites qui sont tangentes à un objet, pour φ constant forment un volume de forme « cylindrique » comme l'illustre la Figure 2.8(b). Nous appelons ce volume le *volume de tangence*.

Pour le cas de plusieurs objets (voir Figure 2.9), les volumes de tangence s'intersectent dans l'espace dual, représentant ainsi les différentes situations possibles. Par exemple, les droites qui sont tangentes à deux objets correspondent à l'intersection des deux volumes de tangence.

L'espace de droites ne suffit pas à représenter les occultations. Pour cela, nous utilisons des *segments* de droites. Avec les volumes de tangence et la structure auxiliaire, nous pouvons définir la structure complète du complexe. Par exemple, (voir Figure 2.9), A est l'ensemble de segments qui « voient » le devant de la sphère R (gris clair) et B l'ensemble de segments qui voient l'arrière de l'autre sphère L . Un cas intéressant est le cas C , qui correspond aux segments qui sont entre les deux sphères ; dans l'espace dual il s'agit de l'intersection $A \cap B$. Les autres cas sont décrits dans la légende de la Figure 2.9.

Nous pouvons ainsi décrire les éléments du complexe de visibilité : une face correspond (comme le cas C du Tableau 2.1) à un élément de dimension 4. Une face de tangence correspond aux extrémités d'une telle face, et est de dimension 3. Les autres cas sont présentés au Tableau 2.1. Un sommet du complexe correspond à une droite de zéro degré de liberté, car c'est une droite tangente à quatre objets.

Nous avons décrit [DDP96] d'une façon abstraite un algorithme de construction pour le cas des scènes composées de polygones, et nous avons également présenté les modalités d'utilisation de ces structures, notamment pour des applications graphiques comme le calcul des facteurs de formes pour l'éclairage et le

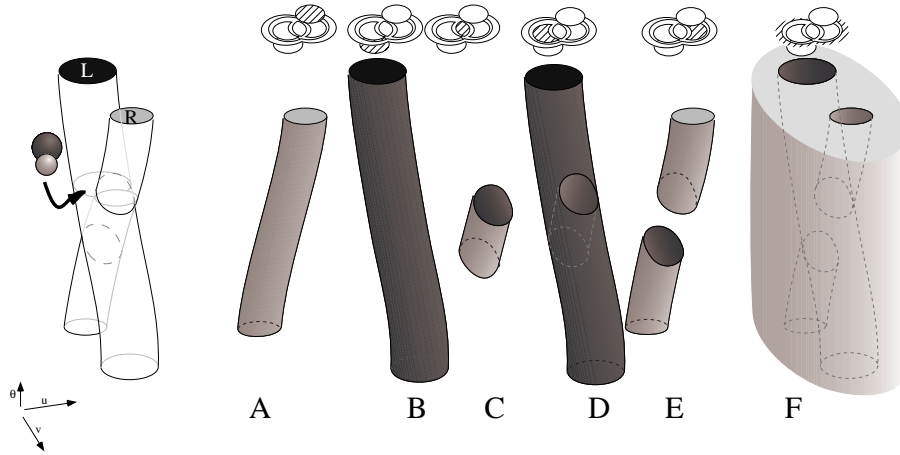


FIG. 2.9: Le cas de deux objets. En haut, nous présentons le complexe auxiliaire. Les ensembles A à C sont décrits dans le texte. D est l'ensemble de segments qui voient le devant de L . Comme la visibilité est occultée par R dans cette direction, D a la forme $B - A$. D'une façon similaire, E est l'ensemble de segments qui voient l'arrière de R . F est l'ensemble de segments qui ne voient aucune sphère ; dans l'espace dual ça correspond au complément de $A \cup B$.

Dim	Configuration	tranche en φ en espace dual	Nom
4			face
3			face de tangence
2			face de bi-tangence
1			arête de tritangence
0			sommet

TAB. 2.1: Éléments du Complexe de Visibilité

calcul de vue.

Une généralisation théorique du complexe a également été développée pour le cas des objets lisses et convexes, en présentant un algorithme de construction « sensible à la sortie » [DDP97a].

2.2.2 Squelette de visibilité

Le complexe de visibilité est une façon élégante de décrire les relations de visibilité globale dans une scène. En contrepartie, le coût en mémoire de cette structure est très élevé, à cause de ses 4 dimensions et demie. De plus, cette structure serait assez compliquée à implémenter, car elle nécessiterait la résolution d'équations de degré élevé, ainsi que des calculs géométriques instables.

Pour répondre à ces difficultés, nous avons développé une structure simplifiée, nommée « Squelette de Visibilité ». Pour le squelette, nous ne construisons que les composantes de dimensions 0 et 1. Cette structure a l'avantage d'éviter le coût élevé en mémoire du complexe complet, et en même temps a permis son implémentation et son utilisation pour répondre à une série de requêtes utiles de visibilité globale [DDP97b].

La nouvelle structure est assez intuitive. Comme décrit ci-dessus, les éléments de degré zéro sont les droites tangentes à quatre objets, ou quadritangentes. Pour le cas des scènes polygonales, cela correspond aux droites tangentes à quatre arêtes. Un exemple est illustré par la Figure 2.10(b). L'intersection de deux surfaces de discontinuité arête-sommet (comme décrit pour le cas des maillages de discontinuité - voir Figure 2.10(a)) induisent une droite qui est tangente à quatre arêtes (les deux du sommet et les deux arêtes

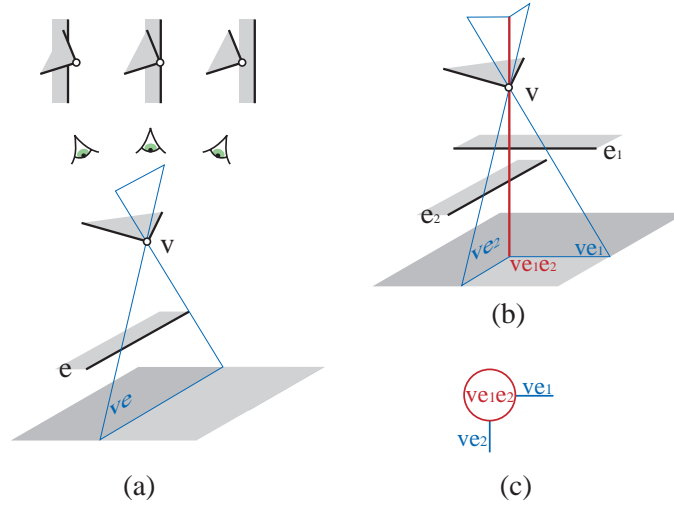


FIG. 2.10: (a) La surface de discontinuité eV décrit un changement de visibilité comme le montre la vue du dessus illustrée en haut. (b) Une quadritangente VEE définie par l'intersection de deux EV et (c) le graphe ou squelette induit par cette configuration [DDP97b].

e_1 et e_2).

Les surfaces de discontinuité e_1V et e_2V qui sont adjacentes à cette droite sont des arêtes de tri-tangence du complexe, car elles sont tangentes à e_1 ou e_2 et aux deux arêtes composant le sommet V . Nous représentons donc les éléments de degré 0 et 1 du complexe par une structure de graphe : les nœuds sont les quadritangentes, éléments de degré zéro, et les arcs qui lient les nœuds sont les surfaces de discontinuité adjacentes aux nœuds respectifs. Voir Figure 2.10(c), qui montre le nœud et les arcs du graphe correspondant à la configuration de la Figure 2.10(b).

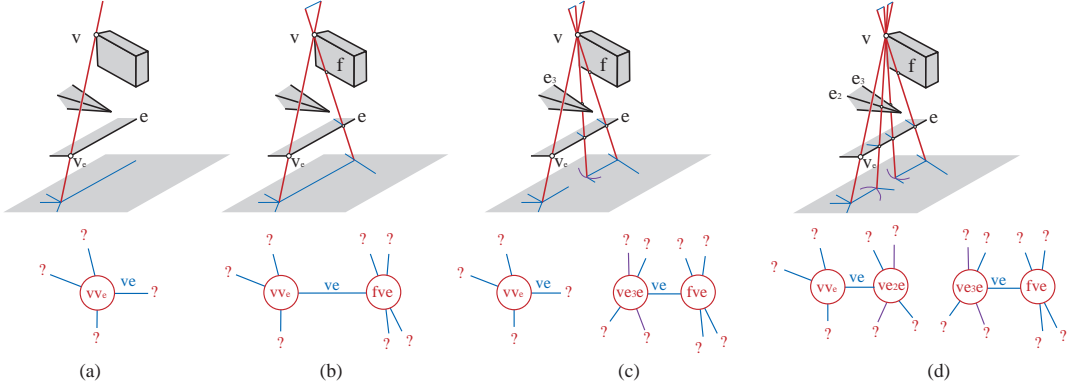


FIG. 2.11: Construction du squelette de visibilité. Les noeuds sont calculés par lancer de rayons, ce qui permet la détermination des adjacences et la construction du graphe (en bas de la figure).

Pour éviter le calcul instable des intersections de surfaces de discontinuité, nous calculons directement les nœuds du complexe par lancer de rayons. Par exemple, on commence par le calcul d'un événement de degré 0, vv (les plus simples) (Figure 2.11(a)), et après par un fve , où f est une face (Figure 2.11(b)). Ces deux nœuds du complexe partagent la surface de discontinuité ve , et ils sont donc liés dans le graphe par un arc. La construction continue pour les événements ve_3e et ve_3e . Malgré la forte diminution du coût en mémoire par rapport au complexe de visibilité, les besoins en mémoire du squelette restent très importants (des centaines de mega-octets pour des scènes de quelques centaines de polygones). Par rapport

aux maillages de discontinuité, la construction est beaucoup plus stable ; pour des scènes contenant des configurations fortement dégénérées, notre implémentation peut cependant avoir des problèmes.

Malgré ces problèmes, les avantages de cette structure sont multiples : par rapport aux maillages de discontinuité, elle est beaucoup plus robuste à la construction, car elle ne nécessite que des intersections objet/rayon, au lieu d'intersections objet/surface de discontinuité (souvent quadriques) ; la structure encapsule la visibilité globale, car elle peut répondre aux requêtes de visibilité de n'importe quelle paire d'objets de l'environnement ; enfin la structure est flexible à la construction (contrairement à beaucoup d'algorithmes géométriques) car elle pourrait être construite localement (par exemple au fur et à mesure des besoins des requêtes).

2.2.3 Éclairage global et Squelette de Visibilité

Dans le calcul de l'éclairage global, c'est-à-dire l'échange de lumière depuis les sources lumineuses, les objets réfléchissants et les réflexions multiples, il est très important d'avoir un calcul précis de la visibilité.

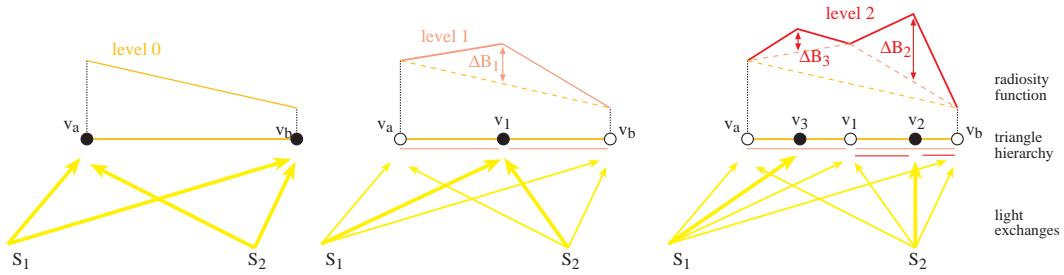


FIG. 2.12: Représentation cohérente et multirésolution par ondelettes. Au lieu de stocker de valeurs de radiosité (éclairage) nous stockons la différence.

Dans le contexte de l'algorithme de la radiosité [GTGB84] et plus précisément de la radiosité hiérarchique [HSA91], le calcul de la visibilité est souvent la partie la plus lourde du coût total de la simulation (voir également paragraphe 3.2.1). Le squelette de visibilité paraît une structure tout à fait adaptée à ce calcul, car elle nous fournit une description complète de toutes les relations de visibilité dans une scène.

Pour exploiter ces propriétés du squelette, nous avons étendu le squelette pour pouvoir calculer des informations de visibilité aux sommets introduits par une subdivision de la scène originale [DDP99]. D'une façon semblable aux méthodes de maillage de discontinuité, nous représentons l'éclairage par des maillages irréguliers. Nous représentons également le transfert de lumière par des liens face (polygone)-sommet ; ce calcul peut être fait analytiquement par les informations fournies par le complexe.

Contrairement aux méthodes précédentes, nous avons été conduits à utiliser des triangulations hiérarchiques, pour permettre l'élaboration d'une méthode *hiérarchique* de simulation de l'éclairage. La méthode d'ondelettes « paresseuses » [SDS96] a été adaptée à ces fins. La construction est illustrée par la Figure 2.12 ; en stockant la différence au lieu de la valeur de l'éclairage (radiosité), nous pouvons maintenir une représentation multi-résolution. En particulier, nous n'avons pas besoin de représenter par des valeurs multiples les échanges entre les sources différentes (par exemple S_1 et S_2 vers les sommets v_a , v_b etc. dans le cas de la Figure 2.12). Nous maintenons également des liens entre polygones, pour décider si une surface doit être subdivisée. Cette décision est prise en utilisant les informations fournies par le squelette sur la visibilité entre deux polygones, ainsi que par des considérations liées à la perception humaine [War94].

Le résultat de notre nouvel algorithme est le calcul d'images de haute qualité pour des scènes comportant des configurations de lumière difficiles à simuler par des algorithmes précédents, comme des scènes éclairées par plusieurs sources (Figure 2.13) ou des scènes éclairées principalement par la lumière indirecte, c'est-à-dire par le biais d'un ou plusieurs réflexions (Figure 2.14).

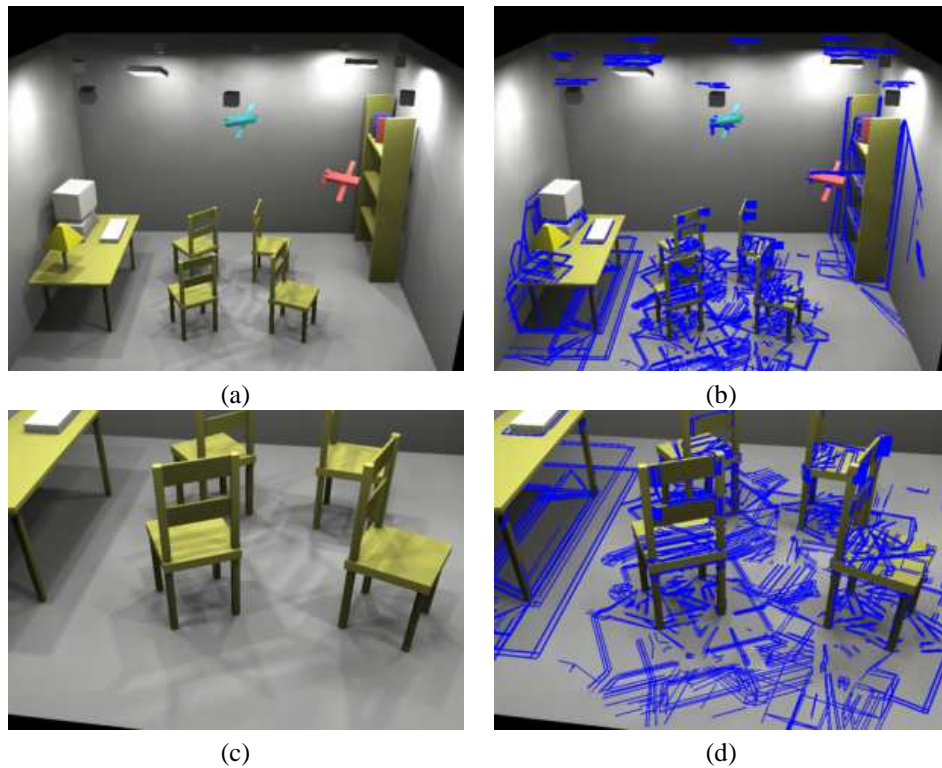


FIG. 2.13: Scène comportant plusieurs lumières (a) image finale, (b) les discontinuités insérées. (c) et (d) vue de près du sol [DDP99].

2.3 Discussion

Nos travaux de 1994 à 1999 sur les méthodes analytiques de visibilité pour le calcul d'éclairage ont donné lieu à plusieurs algorithmes et à de nouvelles structures représentant la visibilité, pour l'éclairage direct d'abord et ensuite pour l'éclairage global.

Ces méthodes ont donné des images de grande qualité, parfois même exactes, ce qui était très difficile, ou même impossible, auparavant. De plus, elles ont donné lieu à une meilleure compréhension des facteurs importants pour la qualité d'image (le maillage, le calcul de la visibilité), qui nécessitaient une solution analytique du problème de la visibilité pour l'éclairage.

Malgré ces progrès importants, plusieurs problèmes intéressants demeurent pour l'avenir. Nous mentionnons les deux qui nous semblent les plus importants :

- *La place mémoire.* Les structures du complexe et du squelette de visibilité sont trop coûteuses en place mémoire (des centaines de mega-octets pour des centaines de polygones). Pour des scènes d'une complexité du « monde réel » il est indispensable de développer de nouvelles méthodes, soit hiérarchiques, soit « paresseuses » afin de permettre l'utilisation de ces structures pour des problèmes réels.
- *La robustesse.* Tous les calculs décrits dans ce chapitre comportent des composantes géométriques, très sensibles aux configurations dégénérées. Il est possible qu'une approche hiérarchique, évoquée ci-dessus, puisse être utilisée pour permettre à ces algorithmes et ces structures de donner une réponse approximative, mais cohérente. La définition de la visibilité approximative, et de ce qui est une réponse cohérente, restent des questions intéressantes et difficiles.

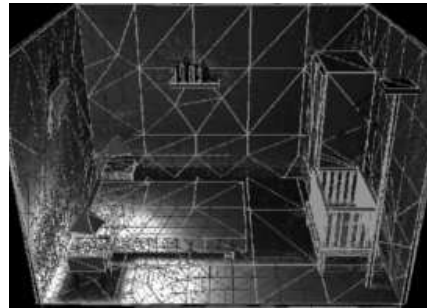
Par la suite, nous décrirons une tout autre approche aux problèmes de l'éclairage, qui consiste à ne pas essayer de trouver une solution exacte, mais à traiter rapidement des scènes de grande complexité, d'une façon approximative.



(a)



(b)



(c)

FIG. 2.14: Scène éclairée par la lumière indirecte (a) image finale, (b) les discontinuités insérés. (c) maillage

2.4 Articles

2.4.1 Structured Penumbral Irradiance Computation (IEEE TVCG'96)

Auteurs : George Drettakis et Eugene Fiume

Revue : IEEE Transactions on Visualization and Computer Graphics

Date : décembre 1996

Structured Penumbra Irradiance Computation

George Drettakis, Eugene Fiume

Abstract—A definitive understanding of irradiance behavior in penumbral regions has been hard to come by, mainly due to the computational expense of determining the visible parts of an area light source. Consequently, sampling strategies have been mostly ad hoc, and evaluation of the resulting approximations has been difficult. In this paper, the structure of penumbral irradiance is investigated empirically and numerically. This study has been made feasible by the use of the discontinuity mesh and the *backprojection*, an efficient data structure representing visibility in regions of partial occlusion. Regions of penumbræ in which irradiance varies non-monotonically are characterized empirically, and numerical tests are performed to determine the frequency of their occurrence. This study inspired the development of two algorithms for the construction of interpolating approximations to irradiance: one algorithm reduces the number of edges in the mesh defining the interpolant domain, and the other algorithm chooses among linear, quadratic, and mixed interpolants based on irradiance monotonicity. Results from numerical tests and images are presented that demonstrate good performance of the new algorithms for various realistic test configurations.

I. IRRADIANCE PROPERTIES IN SCENES WITH PARTIAL OCCLUSION

IN scenes illuminated with area light sources, regions of partial occlusion or *penumbra* readily occur. Understanding how illumination varies within these regions is both important and difficult. It is important since such an understanding allows us to pick both a suitable sampling strategy, and a good way to compactly represent illumination in the penumbra, using piecewise-polynomial functions for example. These representations can be used for fast high-quality rendering of scenes with area sources, and are important in global illumination calculations (e.g., [16], [21], [34], [12]). However, gaining an understanding of irradiance behavior in the penumbra is difficult because the problem reduces to determining how the visible part of the source changes as one moves from one partially occluded point to another. These changes depend on the interaction of the edges and vertices in the environment; analyzing this geometric interaction is a non-trivial problem. In addition, determining the visible part of the source at any point is expensive if done naively.

The *backprojection* and the *discontinuity mesh* are data structures that permit efficient calculation of the visible portions of a polygonal light source in a penumbra. In polyhedral environments, the irradiance contribution of each portion can be computed analytically using standard techniques. However, the overall irradiance at a point is the sum of all such contributions, and can exhibit visually-significant variations over a small region. By performing a thorough empirical study of penumbral irradiance behavior, it is possible to glean insights that can be exploited in an efficient approximation. Doing so has allowed us to isolate the causes of multiple extrema in the penumbra.

George Drettakis is at iMAGIS/GRAVIR-INRIA, BP 53, Grenoble Cedex 9, F-38041, FRANCE. iMAGIS is a joint research project of CNRS, INRIA, UJF, INPG. Part of this work was performed when George Drettakis was a Ph.D. student at the University of Toronto.

Eugene Fiume is at the Department of Computer Science, University of Toronto, Toronto, Ont. CANADA M5S 1A4.

E-mail: George.Drettakis@imag.fr
elf@dgp.toronto.edu

We have also gathered statistics on the frequency of the different configurations affecting penumbral irradiance.

Our study has led to the development of two algorithms that exploit the properties of irradiance in the penumbra. In the first, the number of edges in the discontinuity mesh is reduced without significant deterioration of image quality, while the second algorithm chooses appropriate interpolant degrees (linear or quadratic) again with only moderate quality degradation. Numerical and visual results for both are presented and discussed.

II. PREVIOUS WORK IN SAMPLING AND SHADOW COMPUTATIONS

Approximate and compact representations of illumination or irradiance (impinging light power/area), are useful for the efficient display of illumination for direct lighting and are also necessary for the purposes of global illumination algorithms, such as those developed in radiosity-based approaches (e.g., [6], [17]). In early global illumination algorithms, piecewise constant representations were used for radiosity or irradiance, but it quickly became clear that this representation was insufficient. As an alternative, higher order methods have been since proposed for the solution process (i.e., the light transport phase of global illumination algorithms) with the use of approximation schemes that are of higher degree [31], [16], [34].

A. Observed Properties of Irradiance

Campbell and Fussell [3] observed that irradiance in a penumbral region can exhibit multiple minima and maxima. Numerical optimization was used to determine these critical points. Tampieri [29] and Lischinski et al. [22] segmented the penumbral domain by the mesh generated solely from visual events caused by planar discontinuity surfaces including a source edge or vertex. They then postulated that within each face or cell of this mesh the irradiance varies little. A subsequent adaptive subdivision step was however used when large irradiance discrepancies were observed.

In [9] we proposed that the structure of illumination should be studied in more detail in the hope that a better understanding would lead to more efficient and accurate sampling strategies. Such a structure-driven approach for unoccluded (i.e., shadow-free) environments lit by area light sources was presented in [11]; we conjectured that the illumination from convex polygonal light sources is unimodal, and an effective structured sampling algorithm based on this conjecture was developed. The algorithm first finds the overall maximum of the irradiance function over the surface, if it exists, and then segments the function into convex and concave regions along two axes passing through the maximum. A mixed quadratic/linear interpolant is fit to the irradiance function satisfying tight and relevant error bounds.

B. Discontinuity Meshing and Backprojections

The visible regions of a polygonal area light source from a point are polygons whose vertices are either formed by the pro-

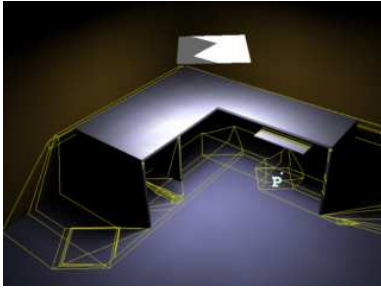


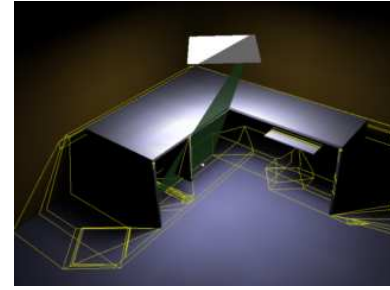
Fig. 1. A Complete Disc. Mesh and Backprojection Instance

jection of scene edges onto the source, or are vertices of the original light source. A *backprojection instance* at, or induced by, a point P , with respect to a source, is the set of polygons forming the visible parts of the source at that point (e.g., the gray region on the source in Fig. 1). The *backprojection* in a region is a data structure containing the set of ordered lists of emitter vertices and edge pairs such that at every point P in that region, the projection through P of these elements onto the plane of the source form the backprojection instance at P [10].

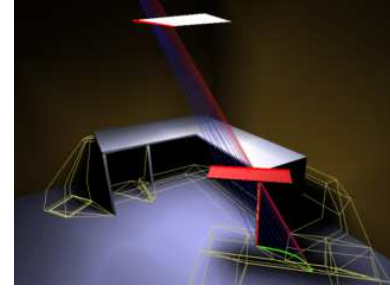
Given a polygonal light source σ and polygonal scene, the partition of the scene into regions having the same backprojection is the *complete discontinuity mesh* of σ (shown as a yellow mesh in Fig. 1). A region of the complete mesh with the *same* backprojection is a *face* of the mesh. At any point P within a mesh face, the backprojection instance can be efficiently determined by projecting the scene edges in the backprojection structure through the point P to find the coordinates of the relevant points. An example of a scene, its discontinuity mesh, and a backprojection instance (the shaded region on the source) can be seen Fig. 1. The backprojection instance corresponds to the white spot marked P under the drawer.

Early proposals to compute shadows involved numerous techniques dealing with point sources, as well as approximate solutions for linear or area sources (see [33] for a good survey). This research naturally lead to the computation of partial discontinuity meshes. In [24] the extremal boundaries were computed, that is the boundary between umbra and penumbra as well as the boundary between penumbra and light for simple geometries. To compute backprojection instances where required, the light source was intersected with the *entire* environment each time, to determine the visible part of the source. The expense of computing exact irradiance values in the penumbra was thus prohibitive. Campbell and Fussell [2] first computed shadow boundaries for complex environments using BSP trees from point light sources, and then extended the method to compute the extremal boundaries for area sources in [3]. Chin and Feiner [4] performed a similar computation, and also presented an extension to area sources in [5]. Additional lines of the mesh, interior to the penumbræ, were computed in [22] again using BSP trees, and in [19] using a two-dimensional visibility algorithm. In all these approaches, computing the exact visible portion of the source at a given point involves a visibility computation requiring the intersection of all the scene objects, since the complete mesh has not been computed; thus the backprojection is not unique within each mesh face.

The computation of the complete mesh is performed by cast-



(a)



(b)

Fig. 2. (a) EV and (b) EEE discontinuity surfaces.

ing *discontinuity surfaces* into the environment. These surfaces are of two types: *EV surfaces* which are planar “wedges” caused by the interaction of an edge and a vertex (Fig. 2(a)), and *EEE surfaces* which are ruled quadric surfaces caused by the interaction of three edges (Fig. 2(b)). Algorithms to compute the equivalent problem in computer vision, that of computing the aspect graph, have been proposed among others, by [15] and [14]. An algorithm which computes an equivalent structure for visibility was presented in [30]. An algorithm specifically for shadow computation with good theoretical complexity bounds has been proposed by Stewart and Ghali [27]. A extension and implementation of this approach was presented in [28].

The authors have developed and implemented a fast, practical algorithm for computing the complete discontinuity mesh with backprojections ([8] and [10]). All relevant visual events are properly treated and the algorithm displays fast running times in the number of objects in the scene, for scenes of moderate complexity. This approach has recently been used to develop a hierarchical global illumination algorithm permitting accurate visibility calculations using backprojections [12].

C. General Discontinuity Meshing and the Irradiance Jacobian

In the work presented in [10] certain special cases (such as EEE surfaces consisting exclusively of edges of the environment) had not been implemented. The work reported here is based on a complete implementation which includes all possible configurations of discontinuity surfaces. We have also developed techniques to treat various degenerate cases of discontinuity surfaces which arise in general environments, permitting the treatment of scenes with arbitrary positioning of the source and scene objects.

Another important addition to structured sampling for unoccluded environments and discontinuity meshing is the use of the

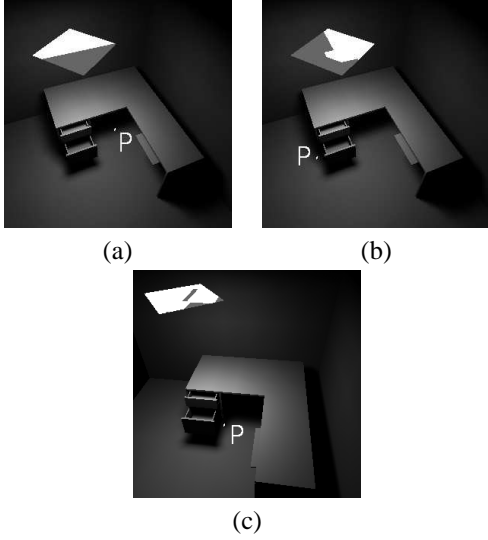


Fig. 3. (a) Convex, (b) concave and (c) disconnected backprojection instances.

irradiance Jacobian as presented by Arvo [1]. This formulation permits the analytic computation of the gradient of irradiance at a cost equivalent to the cost of irradiance $I(p)$ where p is a point of a surface. In particular we can compute and store $\nabla I(p)$. If we wish to determine the derivative dI/dt of irradiance $I(t)$ in a certain direction \bar{u} , we simply perform $\nabla I \cdot \bar{u}$ to obtain the corresponding value. This calculation renders the structured sampling approach of [11] much more efficient and accurate, since the need for numerical approximation of the derivatives is obviated. Arvo presented a formulation for partially occluded points, which allows efficient computation of analytic derivatives, as presented above. In the work presented here the backprojection data structure is used, which provides all the information necessary for the computation in [1].

III. PROPERTIES OF IRRADIANCE FUNCTIONS IN PENUMBRAL REGIONS

Given the complete discontinuity mesh and the backprojection, the exact value of irradiance at any point in the penumbra can be efficiently determined. It thus becomes possible to perform a careful empirical study of irradiance in the penumbra, even for moderately complex scenes. We shall now present the results of an empirical study that isolates key configurations that induce significant variations in penumbral irradiance. We must be aware of these configurations when constructing approximations. Snapshots of such observations are shown in the figures which include the mesh and backprojection geometries, as well as analytically computed irradiance and first derivative values and numerically computed second and third derivatives. Before discussing the results of the experiments, we define some important scene properties.

Backprojection Type. The backprojection of a face is said to be *convex* or *concave* if every instance of the backprojection in that face is itself a convex or concave polygonal subset of the source, respectively. In addition, a backprojection of a face is *disconnected* if every instance of the backprojection in that face consists of more than one polygon, while it is *simple*, if every instance consists of only one polygon. An example of a convex

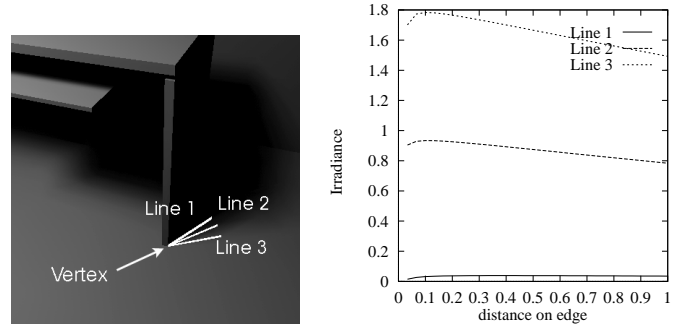


Fig. 4. A singular vertex

backprojection instance is shown in grey on the light source in Fig. 3(a) and a concave example is shown in Fig. 3(b), which are both simple. An example of a disconnected backprojection is shown in Fig. 3(c).

Singularities. As noted in [29] and [22], irradiance along a surface is singular at points at which two surfaces touch. Thus at a vertex in the mesh joining faces of umbra, penumbra and light, the irradiance is multi-valued and is defined as a limit depending on the direction from which the vertex is approached. A singular vertex is shown in Fig. 4, while the graph on the right shows the variation of the irradiance values on three lines joining at the singular vertex.

A. Empirical Characterization of Penumbral Irradiance Behavior

A set of empirical tests were performed within the penumbra for moderately complex scenes. These tests attempted to isolate configurations that cause local extrema. Three influential factors affecting the appearance of extrema were identified:

- *Backprojection type* within the mesh faces of interest; this can cause simple irradiance extrema within a mesh face but also opposite extrema along different directions in the face.
- *Interaction with unoccluded areas*; in particular when unoccluded irradiance increases and the visible part of the source decreases simultaneously or vice-versa.
- *Position of irradiance maximum* along mesh edges with a constant backprojection instance.

In what follows we present examples for each category with an illustration of the backprojection instance shown in grey on the source, and a corresponding graph illustrating the irradiance variation along a line on the floor (shown as a thick white line). We briefly discuss the effects of each configuration on irradiance behavior.

Disconnected or Concave Backprojections. If the backprojection in a face (or along an edge) is not simple, it is likely that there exists a line in the face or an edge for which the irradiance will display one or more maxima. Illumination in a region with a disconnected backprojection containing n polygons is equivalent to illumination from n separate unoccluded light sources, and may have up to n maxima. Non-monotonic behavior in this case may cause “troughs” in the irradiance function, as shown in Fig. 5(top), from the overlap of two “tail” regions [11] of the irradiance function due to a disconnected backprojection with instances containing two polygons. Similar non-monotonic be-

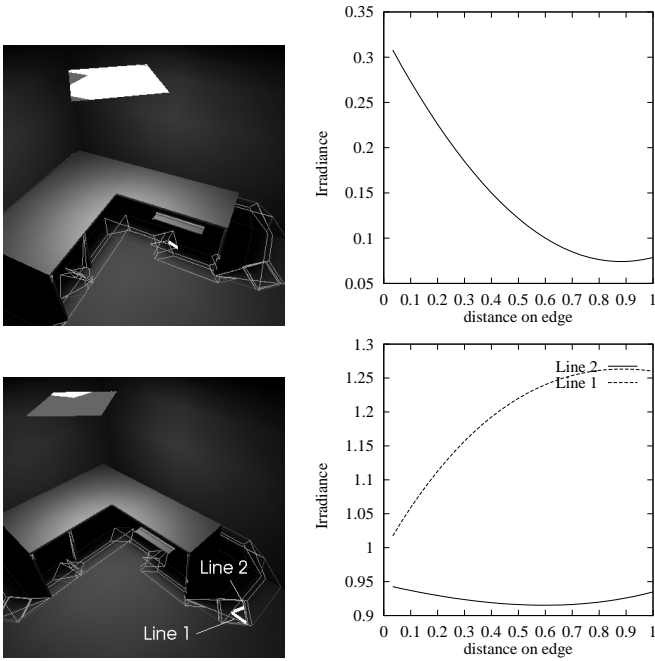


Fig. 5. (top) Irradiance minimum from disconnected backprojection (bottom) Opposite extrema within a face.

havior may occur in faces or along mesh edges that have concave backprojections.

Faces with concave or disconnected backprojections, particularly in the presence of light source edges that are very long, can cause the existence of opposite extrema (i.e. a minimum and a maximum) along different directions in a face. An example is shown in Fig. 5(bottom). In this case the maximum is caused because the corresponding edge in the mesh ends in a light region (see below), while the trough is caused because this face has a concave backprojection.

Interaction with Unoccluded Regions. Consider a line in the mesh that does not lie on the external penumbral boundary between light and shadow but has however one endpoint in light (i.e. an unoccluded region, also called a *light region*). An example is shown as a thick white line in the mesh of Fig. 6(top). The visible area of the source increases along this line, as it goes from penumbra to light, since a smaller portion of the source becomes occluded. As a consequence the irradiance on a line is generally increasing as the region of light is approached. The unoccluded illumination along this line may be increasing or decreasing. If the unoccluded illumination is decreasing (as in Fig. 6(top)), the irradiance in the penumbra, which tends to be an increasing function, will smoothen as it approaches the unoccluded regions. Again, this can result in non-monotonic behavior, although it is necessarily a local maximum.

Position of Maximum from a Constant Backprojection Instance. Consider a scene that is illuminated by a polygonal source, and in which the complete discontinuity mesh of that source has been computed. For some edges in the mesh, the backprojection remains constant along that edge. Thus the analysis used for unoccluded sources in [11] can be used directly. For example, the edge shown as a thick line in Fig. 6(bottom) contains the maximum of the irradiance function

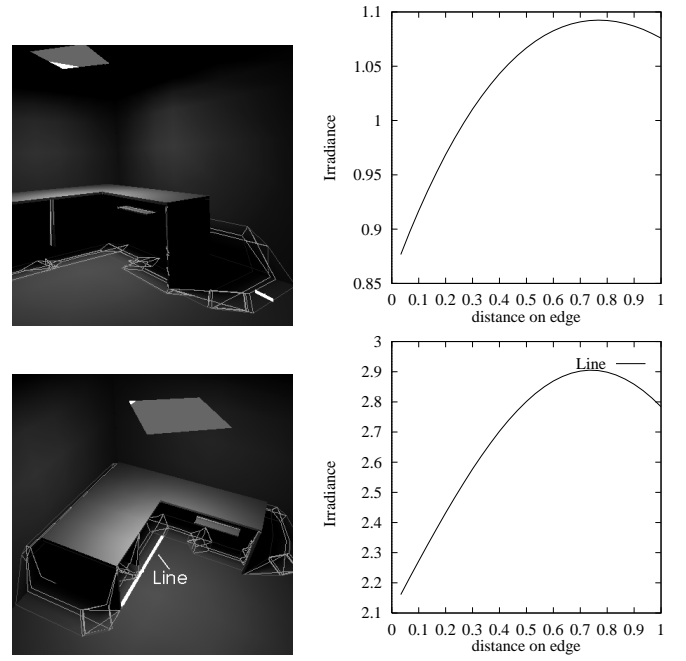


Fig. 6. (top) Interaction with unoccluded regions. (bottom) Constant backprojection instance.

from the polygonal source defined by the backprojection instance, shown shaded on the rectangular source. This maximum can be seen in the irradiance plot on the right.

B. Scene Statistics and Identification of Non-monotonic Irradiance

A test program has been written which analyses the behavior of irradiance in a set of scenes. The goal is to determine whether irradiance tends to be monotonic within a single face of the discontinuity mesh. This has obvious implications in the necessity for additional adaptive subdivision, but also in the development of error bounds in radiosity calculations [21].

The set of scenes considered consists of a simple desk model in a simple configuration (e.g., Fig. 6) containing 73 polygons and a more complex configuration including drawers (145 polygons) which cause complex visibility interactions (e.g, Fig. 9). For each of the two geometries, the light source was placed in nine different positions. Six of these are shown in Fig. 9. In addition two different light sources were tested, namely a small light source (Fig. 7(a)) and an elongated light source, which behaves in a manner similar to a linear source (Fig. 7(b)) These two source types were tested on both desk geometries (with and without drawers). Finally a larger scene was tested, containing two complex desks and two chairs containing 373 polygons (Fig. 8), which was used to confirm the more exhaustive examples with the multiple source positions. These scenes will also be used to compare the relative performance of the new structured sampling algorithms proposed in later sections (see Section VII-A).

It is important to note that the tests performed using these scenes are of course not definitive. Nonetheless, all types of visibility events occur in these scenes (EV, EEE, D1 [13] and other degenerate configurations) and in addition relatively com-

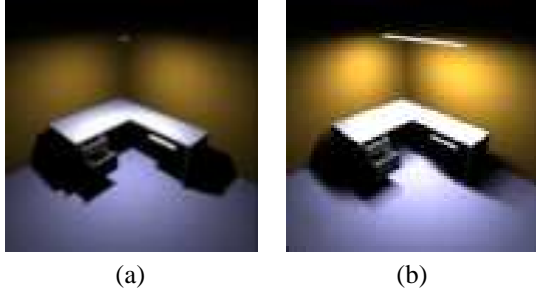


Fig. 7. (a) Image with small source (b) Image with elongated source



Fig. 8. Big Scene

plex shadow behaviors can be observed (e.g., in the regions below the open drawers or the shadows caused by the back of the chairs). As a consequence we believe that the trends identified in this experimental study are a strong indication of irradiance behavior in the penumbra for interior scenes. We restrict our approach to a single source, since multiple sources present numerous specific issues (as pointed out in [7]).

Given a line embedded in a receiver plane, defined by two endpoints p_t and p_h , we consider the irradiance as it varies along this line as a function $I(t)$ of a single parameter (in a manner similar to that presented previously). In this case we can eval-

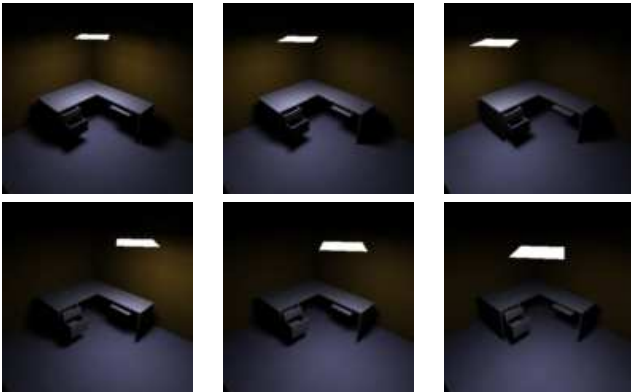


Fig. 9. 6 of the 9 light positions used

Scene	e_p	e_{nm}	%NM	f_l	f_d	f_{conc}
Average values over 9 runs						
Simple Desk	721.9	25.1	3.5	23.1	0.9	1.1
Desk Drawers	2386.1	157.6	6.6	33.7	42.2	81.7
Average over 2 runs						
Small Source	906.0	73.5	8.1	28.5	12.5	32.5
Long Source	1449.0	114.5	7.9	27.5	30.5	56.5
Single run						
Big Scene	8661.0	746.0	8.6	78.0	374.0	294.0

TABLE I

MESH EDGE MONOTONICITY OF PARTIALLY OCCLUDED FACES

uate $\frac{dI(p_t)}{dt}$ and $\frac{dI(p_h)}{dt}$. If these values are of opposite sign, we consider that the irradiance along the line is *not* monotonic. If the derivative values are of the same sign, we consider the irradiance along this line monotonic.

We have chosen to test the irradiance along all the edges of the mesh on penumbral (i.e., partially occluded) faces, and also along the diagonals of the penumbral faces connecting two vertices of a mesh face not belonging to the same mesh edge. We thus have two tables of statistics for edges (Table I) and for diagonals (Table II), which report *average* values from the 9 different light source positions for “Simple Desk” and “Desk and Drawers”, average of two runs (both geometries) for “Long Source” and “Small Source” and a single run for “Big Scene”. We considered testing faces for monotonicity (i.e. the faces for which either an edge or a diagonal are non-monotonic), but it was observed in test runs that the statistics for faces are very similar to those for the diagonals (Table II).

The monotonicity test is not infallible: edges which are non-monotonic may be ignored because of a change of sign in the derivative in the edge interior. We have however run tests comparing the approach presented above with an exhaustive test of 20 samples of the derivative on each edge, which show that for the scenes in question the simple monotonicity test is accurate for 85% of the tests, which we judged to be satisfactory. Developing other signatures for nonmonotonicity is an interesting open problem.

In Table I, the field e_p is the average number of edges neighboring at least one mesh face in penumbra, e_{nm} is the number of edges along which the irradiance is non-monotonic, while “%NM” indicates the percentage of edges with non-monotonic irradiance. The breakdown f_l , f_d , f_{conc} , shows the number of the edges with non-monotonic irradiance that neighbor a face respectively in light, with a disconnected backprojection or a concave backprojection. To classify an edge, both neighboring faces are tested, and the edge is designated as either light, disconnected or concave, in that order, if either neighbor is in the appropriate category. For Table II, similar statistics are reported, but for the diagonals *contained* in faces with corresponding properties, with d_p and d_{nm} the total number of penumbral diagonals and those with non-monotonic irradiance respectively, while f_d , f_{conc} , are the number of the diagonals with non-monotonic irradiance inside a face respectively with a disconnected backprojection or a concave backprojection.

From these tables it is clear that irradiance is monotonic along a large majority of edges (more than 92%) of the mesh, consistently, even for different source types and more complex geom-

Scene	d_p	d_{nm}	%NM	f_d	f_{conc}
Average values over 9 runs					
Simple Desk	665.3	189.6	28.5	74.4	115.1
Desk Drawers	2181.1	150.0	6.9	43.8	106.2
Average over 2 runs					
Small Source	704.5	61.0	8.7	13.0	48.0
Long Source	1366.5	219.0	16.0	65.0	154.0
Single run					
Big Scene	8116.0	776.0	9.6	438.0	338.0

TABLE II

MESH DIAGONAL MONOTONICITY IN PARTIALLY OCCLUDED FACES

etry. Similarly, irradiance is monotonic along a large majority of the diagonals. The disparity in Table II between the “Simple Desk” and the “Desk Drawers” scenes is due to the fact that, in the absence of drawers which involves less complex visibility interactions, and consequently larger mesh faces, more significant irradiance variation is observed.

Finally, we can see that a majority of edges or diagonals with non-monotonic irradiance neighbor, or are contained in, faces with disconnected or concave backprojections, as suggested by the empirical study presented above.

C. Discussion

The statistics presented above are meant as a first indication of the behavior of irradiance in penumbral regions. In future work geometric *a priori* determination of which regions of penumbra display non-monotonic behavior should be performed (similar in spirit to the discontinuity “ranking” approach presented recently in [18]).

These first measurements however indicate that irradiance is largely monotonic within regions of equivalent visibility. Such piecewise monotonic functions, especially those for which the values do not differ significantly, are good candidates for lower order interpolation, specifically linear or quadratic. The experiments presented below will indicate that this is sufficient in general, obviating the need for cubic interpolants as proposed in [26], for a large class of scenes.

In the following section we will present an algorithm for edge elimination, and an algorithm for degree selection. The fact that irradiance in the penumbra is largely monotonic motivates the need to simplify the mesh: if the function is well behaved a coarser subdivision is sufficient. The empirical observations of which factors are important in the penumbra also influenced the construction of the edge elimination algorithm. Similarly the locally well-behaved nature of penumbral irradiance suggests the use of linear interpolation wherever possible, leading to the algorithm for degree selection.

IV. COMBINING UNOCCLUDED STRUCTURED SAMPLING AND DISCONTINUITY MESHING

In [10] and [8] we presented a first attempt at combining the structured sampling approach developed for unoccluded environments and discontinuity meshing. This method consisted of the collection of mesh faces in penumbra into “penumbral groups” which were enclosed in a bounding box and then combined with the subdivision induced by finding the maximum and the inflection points as described in [11]. This approach

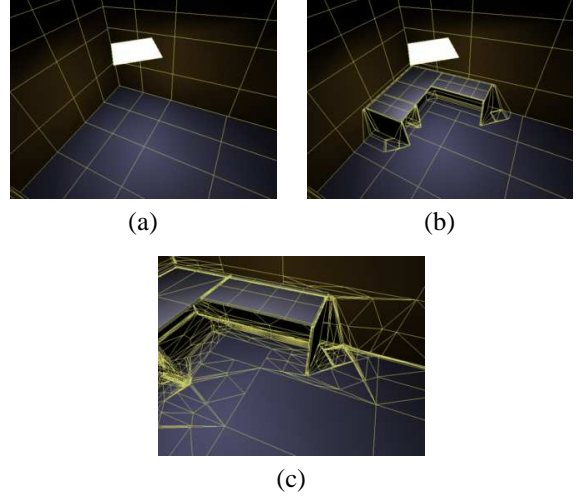


Fig. 10. (a) Unoccluded Structured Sampling (b) Combination with Discontinuity Mesh (c) Triangulation

resulted in reconstruction problems on the boundaries between these bounding boxes of penumbral zones and the unoccluded zones [8].

We have since developed a simpler approach, which appears to avoid these problems. We begin by applying the structured sampling algorithm on the receiver polygon as if it were unoccluded, using the algorithm of [11] (see also Section II-A). This results in a segmentation such as that shown in Fig. 10(a).

In the new approach used here we simply insert the lines of subdivision up to the border of the penumbra and light (Fig. 10(b)). In the quadrilateral regions of the mesh entirely in light we construct bi-quadratic tensor product interpolants. In the regions of penumbra and the regions between regular light regions and penumbra we perform a constrained Delaunay triangulation following [32]. We then construct triangular bi-quadratic interpolants on the resulting triangles (Fig. 10(c)).

It is however clear that this construction is too expensive. The size of the discontinuity mesh faces are often small. In addition, in many cases the variation of irradiance in the faces (as well as their size) is so small that linear or even constant approximations are largely sufficient. These facts, supported by the empirical and numerical study presented above, lead us to the development of two algorithms, allowing the reduction of the size of the mesh, and the use of lower degree polynomial approximations presented below.

A. Adaptive Subdivision

Overall, the subdivision effected by the complete discontinuity mesh appears sufficiently fine for the construction of interpolants. In some scenes however, large faces can occur, over which irradiance may vary significantly (e.g., the face containing the thick white edge in Fig. 6(bottom)). As noted by Tampieri [29], adaptive subdivision can be required in such faces.

We apply two simple criteria. The first requires that an edge defined by two points p_t and p_h for which $|I(p_t) - I(p_h)| > \epsilon$, is subdivided. The tolerance ϵ is user-defined. The second requires the subdivision of every edge for which $\dot{I}(p_t)\dot{I}(p_h) = 0$,

where $\dot{I}(t)$ is the first derivative. In this manner, edges with non-monotonic irradiance are subdivided. In practice, these criteria result in reasonable triangulations.

Adaptive subdivision is always performed before any simplification or degree selection (see below).

V. INTERPOLANT DOMAIN CONSTRUCTION FOR PENUMBRA REGIONS

The problem of constructing an interpolant for irradiance in a general scene can be split into the construction of interpolants for regions that are unoccluded, and into the construction of interpolants for regions in partial shadow. The former is treated using the extension of the methods of [11] for unoccluded environments as described above, while the determination of interpolants for penumbral regions is presented next. There are two aspects to the interpolant definition: the determination of the domains on which the interpolants are defined, and the choice of basis functions, including their degree. We will start with the determination of domains.

A. Constraints on Interpolant Domains

As mentioned above, the complete discontinuity mesh and the accompanying backprojection information is a natural segmentation of the irradiance function over a surface. Edges in this mesh represent discontinuities either of value, or of first or second derivative. Characterizations of these discontinuities can be found in [20], [19], [22], [29].

Value discontinuities occur only where objects touch, and therefore the boundary of such regions delineates areas in which the irradiance function has value zero, because they are completely hidden from the light. Discontinuities of second derivative (or first derivative when degenerate events occur) occur inside regions of penumbra. These discontinuities constitute the majority of edges in the discontinuity mesh. The geometry of the mesh is complicated: it includes highly irregular regions that can be small, concave and with small angles between edges (see Fig. 13(d)). The mesh constrains the construction of interpolant domains, since some of these edges of discontinuity need to be maintained to achieve high quality reconstruction of irradiance.

The relative importance of the discontinuities is difficult to assess without significant numerical computation. Some of the second derivative “jumps” can be small, while others can be quite large. In some cases the effect on the actual irradiance function is more evident (Fig. 11(bottom)) while in others the effect is negligible (Fig. 11(top)). In addition, irradiance in very small regions cannot display extremely large differences in value, because the shape of the polygons in the backprojection instances cannot change much, and neither can the (point-to-area) form-factors that determine the value of irradiance at any point in a face.

To accommodate the highly irregular nature of the faces in the mesh, triangles are selected as the domains over which to construct interpolants. Specifically, a proper triangulation of the penumbral domain is performed. A triangulation $\tau = \{T_0, T_1, \dots, T_n\}$ into n triangles T_i is *proper* if each pair of triangles intersect at a vertex, a complete side or not at all, and the union of all triangles equals the domain (Prenter [25]). In a manner similar to that for irregular regions in light, we use a constrained Delaunay triangulation of the original mesh faces

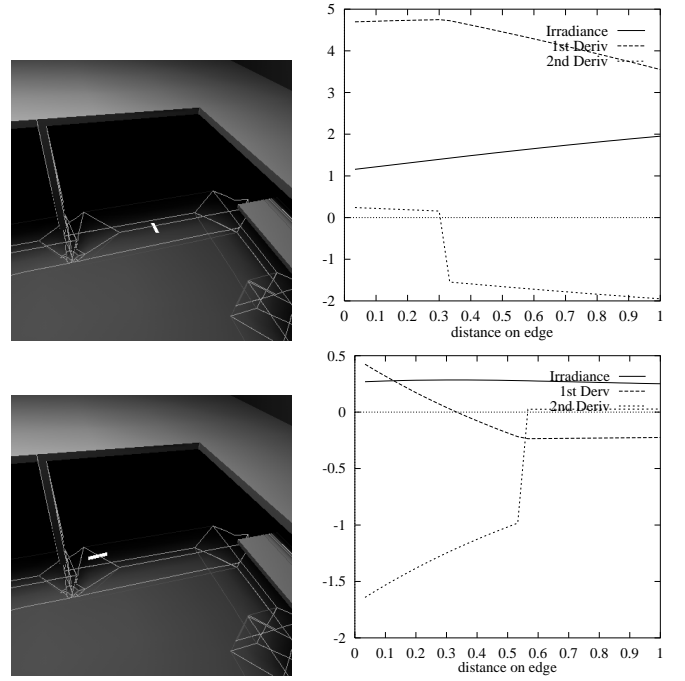


Fig. 11. Discontinuities (top) large “jump” (bottom) small “jump”

[32]. The basis functions used (presented in Section VI-B) are linear, quadratic or mixed linear/quadratic Lagrange interpolating polynomials.

B. A Mesh Reduction Algorithm

As mentioned above, faces of the discontinuity mesh can be arbitrarily small and may also have edges with high aspect ratios. In addition, the triangulation of concave regions can result in triangles that are very small or that have small angles. For the construction of piecewise smooth interpolant domains, it is desirable to reduce the number of such triangles as much as possible. Larger triangles, and triangles with roughly-equal interior angles are more stable numerically, and in addition provide the benefit of a better theoretical error estimate. Specifically, consider a six-point quadratic interpolant on a triangle, and the irradiance function f . The max norm in an interval $[a, b]$ is defined as follows [25]:

$$\|f(x)\| = \max_{a \leq x \leq b} |f(x)|. \quad (1)$$

Following [25], the error bound with respect to the max norm for an interpolant s_N is:

$$\|f - s_N\| \leq \frac{8 M_3}{\sin \theta} h^3, \quad (2)$$

where h is the longest edge of the triangle, and θ is the smallest angle of the triangle. The constant M_3 is equal to the maximum value of the first, second and third derivative of f within the triangle. In general:

$$M_n = \max\{\|D^1 f\|, \|D^2 f\|, \dots, \|D^n f\|\}. \quad (3)$$

where $D^i f$ is the i 'th derivative of f . In the case of unoccluded illumination, and for some of the faces in the penumbra, the

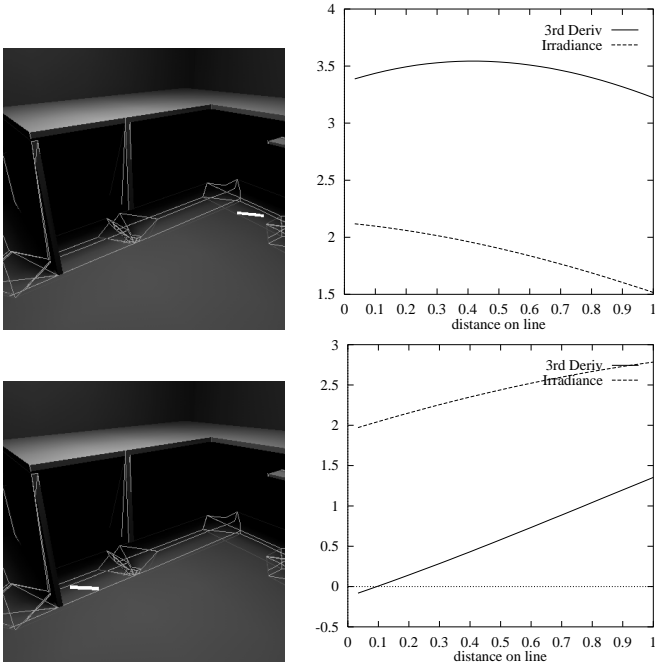


Fig. 12. Absolute values of third derivative in the mesh faces along the thick white lines.

magnitude of the third derivative is many times larger than that of the irradiance itself, rendering Eq. (2) somewhat meaningless (which does not necessarily imply that the approximation is poor). This is shown in Fig. 12(top) where irradiance along the thick white line in the image is plotted in the graph. However, there are other faces in the mesh in which the third derivative is small, and thus the size of the smallest angle θ can play an important role in the quality of the approximation achieved. In Fig. 12(bottom) the absolute value of the third derivative is consistently smaller than the irradiance value.

B.1 An Area-Based Edge Elimination Algorithm

For the reasons outlined above it is desirable to eliminate all overly small faces in the mesh. A smaller mesh size is desirable in general, among other reasons because it allows the irradiance function to be represented with a smaller number of triangular interpolants. This allows faster rendering and is also important when such a representation is used for light transfer simulation (see [12], [8]). In the results it will be shown that a large mesh reduction with negligible error is achievable.

Our guiding strategy is to remove all faces that have area less than $\text{area-tol}\%$ of the largest face, where area-tol is a user defined tolerance. When eliminating edges from the mesh, corresponding faces are deleted. It is therefore necessary to maintain the backprojection information, together with the geometry of the deleted faces in the resulting merged face. This information can be discarded after the construction of the interpolants if the exact solution is no longer desired.

Eliminating Concavities. It is often the case that the tip of a smaller face will bite into a larger face, creating a concavity. For the reasons outlined above, it is desirable to eliminate such faces. Each concave face is visited, and the vertices for which the two edges on the face form an angle greater than 180 de-

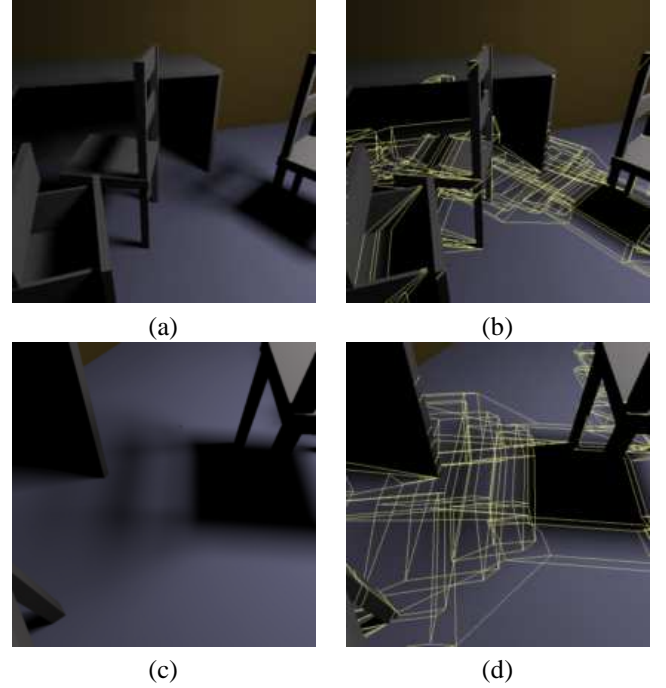


Fig. 13. (a) Image and (b) mesh before reduction (c) detail image (d) detail mesh before reduction

grees are identified. Edges are removed from the concave face if the corresponding small face has area below the tolerance. If all the faces around the vertex can be removed, the remaining unconnected edge is also removed from the mesh.

Eliminating Small Faces. After the concave faces have been treated, a number of small faces that have area less than the maximum may still remain. For each such face the following procedure is applied:

```
remove_small_face( Face f )
{
    sort all edges e of f
      by area of neighboring face
    for each edge e in sorted list do
      if can_remove( e ) then
        remove_edge( e );
        return;
      endif
    done
}
```

The procedure `can_remove(e)` determines whether the removal of the edge is possible. The following rules are applied:

1. No edge of the boundary between umbra and penumbra is removed.
2. No edge of the boundary between umbra and light is removed.
3. If the removal of an edge results in a convex face becoming concave, it will not be removed.

After this procedure is applied, all remaining unconnected edges are removed from the remaining face. After an edge is deleted, the tail and head vertex are searched to determine if the edge extended into the neighboring faces. If it did, the neighboring

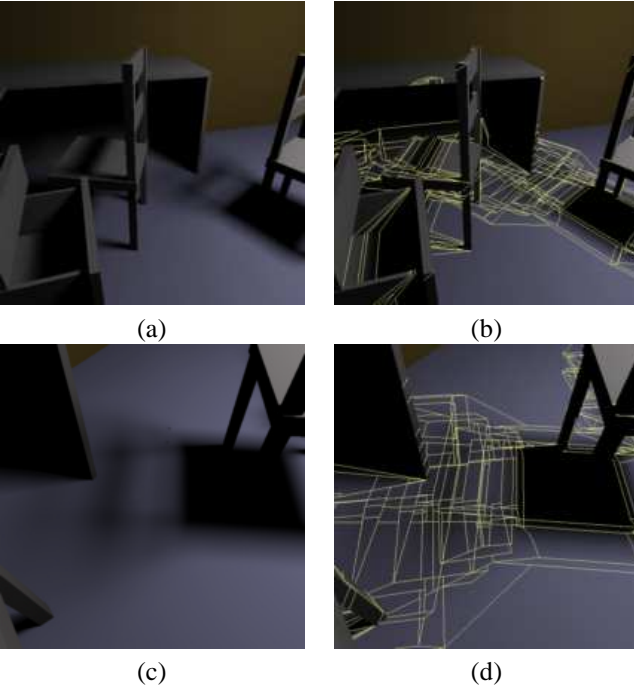


Fig. 14. (a) Image and (b) mesh after 56.0% reduction (c) detail image (d) detail of reduced mesh

edges are removed, if such a removal does not violate rules 1-3. The removal process (concavity and small face removal) is repeated until all faces that are smaller than the tolerance are removed, or until no more faces can be removed.

Polygon rendering hardware on an SGI Indigo 2 XZ was used to generate images in Fig. 13 to 15. In Fig. 13 part of “Big Scene” is shown using the combined structured and discontinuity mesh algorithm (Section IV) before mesh reduction, while in Fig. 14, the same scene and mesh are shown after reduction. Although the mesh has been reduced by 56.0% of the original faces before reduction, the quality of interpolation is still good. The reduction of the mesh is shown in detail in Fig. 14(d), compared to Fig. 13(d). The images with a reduced mesh, even when looking at details (Fig. 14(c) vs. Fig. 13(c)) are still of acceptable quality.

VI. CONSTRUCTING INTERPOLANTS FOR PENUMBRAL REGIONS

Once the triangular domains have been constructed, we have to choose suitable basis functions and then calculate the coefficients to construct the interpolant. The empirical and numerical analysis presented in Section III suggests that for many cases in the penumbra, linear interpolation of irradiance is sufficient. Thus the interpolants constructed by our algorithm are chosen to be of low degree: linear, quadratic or mixed.

Because the complete mesh has been computed, the umbral regions are well defined. On these domains, constant basis functions are used with a value of zero. For this step, the computation of the complete mesh is a necessity, since boundaries of the umbra are often (EEE-induced) curved edges, which were not computed by previous discontinuity meshing algorithms (e.g., [22], [19]) although they have been treated for a different application in [30].

To achieve the construction of mixed linear/quadratic interpolants, it is first necessary to characterize the edges of the mesh and the edges interior to the faces with a required degree of interpolation. Each edge or face is designated as linear or quadratic, and the appropriate basis function is assigned to each triangle of the face, maintaining C^0 continuity.

A. Selecting a Degree for the Triangular Domains

The choice of degree required on an edge of the discontinuity mesh (DM) or a triangle edge in the interior of a DM face is determined by whether or not irradiance along it is monotonic, and whether the difference between irradiance values at the endpoints is larger than a user-defined tolerance `linear-tol`. The algorithm first determines monotonicity on each edge using the same approach as that presented in Section III-B. An edge is then marked “constant” if the difference of irradiance at its endpoints is zero and “linear” if it is less than `linear-tol`. Otherwise it is marked “quadratic”.

As can be seen from the statistics presented in Section III-B, a large proportion of non-monotonic edges are adjacent to faces with disconnected and concave backprojections. Edges neighboring a light face (i.e. a face of the mesh in an unoccluded region), and mesh edges that include the maximum of irradiance can also be non-monotonic. For edges of triangles interior to faces, similar properties hold. Specifically, faces with concave or disconnected backprojections, as well as faces neighboring light regions often display non-monotonic behavior.

Once each edge of the triangular domains has been classified as “constant”, “linear” or “quadratic”, a triangular basis function is selected and the coefficients (irradiance values) are calculated at the triangle vertices and appropriate interior points. In Fig. 15 the same scene as that of Fig. 13 is rendered after the application of the degree selection algorithm. Notice that the visual quality of interpolation is still high despite the fact that 58.3% of the interpolants are linear. The red lines show the edges of the mesh that have been assigned linear interpolants, while the green lines show edges with quadratic interpolants. The choice of basis functions and their construction is described in the following section.

B. Basis Function Design

The basis functions chosen to interpolate the irradiance are triangular Lagrange polynomials. For triangles in which all three edges are linearly interpolated, the method of plates is used to construct a linear basis function over the triangle. For this interpolant the formal error bound is given by [25]:

$$\max |f(p) - s_N(p)| \leq 4M_2h^2 \quad (4)$$

where f is the irradiance function and s_N is the interpolant, and h and M_2 are as in Eqs. (2) and (3). For triangles on which all edges are of degree 2, a six-point interpolant is used [25]. The error bound for this interpolant was given in Eq. (1).

As a consequence of degree selection, there are triangles for which some edges are linear and some are quadratic (see Fig. 16(a) which follows the model of [25]). A special linear/quadratic basis function was designed to guarantee value continuity across such interpolants. For a given vertex, the six-point bi-quadratic interpolant is used, which interpolates the

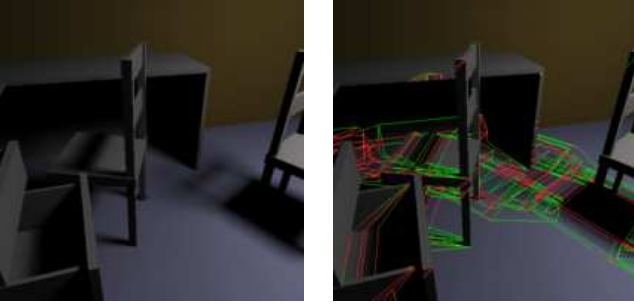


Fig. 15. Image and mesh degree selection (58.3% of edges are linear shown in red)

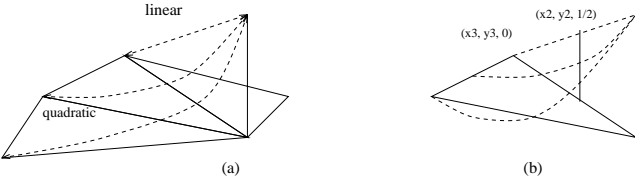


Fig. 16. Linear/quadratic basis functions.

value 1 at the vertex of interest and the line passing through the values $0, \frac{1}{2}, 1$ along the linear edge. This basis function is depicted in Fig. 16(b). These samples are of course reused on neighboring triangles.

For triangles with one linear edge, two linear/quadratic basis functions are used at the vertices of the linear edge, while the remaining vertex is assigned a quadratic basis function, and two quadratic basis functions along the quadratic edges. A total of 5 function (irradiance) values are required for the triangle with one linear edge, since we require one value at each vertex, plus two values at the midpoints of the edges characterized as quadratic.

For triangles with two linear edges, all vertices have linear/quadratic basis functions, and an additional quadratic basis function is defined at the point along the quadratic edge. A total of 4 function values are required for this interpolant.

C. An Interpolant for Triangles at a Singularity

A typical singular vertex will have many edges joining at that point, as described in Section III. The value at the vertex is defined as a limit of the function as it approaches the singular point, and therefore it is multi-valued. To represent this using interpolants, as suggested by Tampieri [29], we use degenerate tensor product interpolants. Nonetheless, the multiple values of irradiance at this vertex must be approximated. To achieve this, the total angle θ_{tot} of the triangles neighboring the vertex is first computed.

The value of the unoccluded illumination at the singular vertex L_{max} is calculated next. For triangle i , whose edges joining at the singularity form angle θ_i , (see Fig. 17) the values of irradiance for the bi-quadratic tensor products along the edge corresponding to the singular point are assumed to vary in the interval:

$$\left[L_{max} \sum_{i=0}^{j-1} \frac{\theta_i}{\theta_{tot}}, L_{max} \sum_{i=0}^j \frac{\theta_i}{\theta_{tot}} \right]. \quad (5)$$

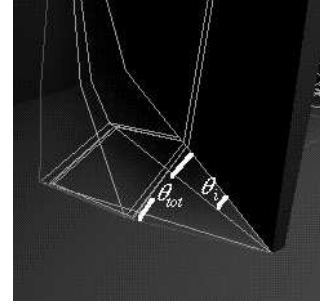


Fig. 17. Singular vertex interpolant construction

This is an approximation, since it would be necessary to compute the limit values of irradiance along each edge to determine the exact range of irradiance in each triangle. The midpoint vertex in the tensor product is assigned the average of these two values. In this fashion a continuous representation is constructed for the irradiance around a singularity.

A special case occurs when the light source touches a different object. In this case we simply displace the vertex slightly so that we can apply the analytical point-to-area form-factor (e.g., [1]) to determine L_{max} .

D. Error Bounds for the Triangular Interpolants

The theoretical error bounds for the linear and quadratic triangular interpolants have been given in Eqs. (2) and (4). These bounds are conservative, since they depend on the maximal magnitude of the derivatives within the domain of interest. In some faces of the discontinuity mesh the derivatives are sufficiently small to permit these bounds to be meaningful. Identifying these cases however is expensive, and not a practical way of assessing quality.

For faces with monotonic irradiance, a simple error bound is given from the maximum difference between irradiance values at the three points of the polygon. The error bound B of the interpolant s_N , on triangles over which irradiance is monotonic, is given as:

$$B = \max |f(p_i) - f(p_j)|, \quad i, j = 1, 2, 3. \quad (6)$$

It is important to note that since we do not guarantee correct monotonicity characterization, in terms of our algorithm B is a heuristic rather than a strict bound.

Error bounds have not been strictly established for faces with non-monotonic irradiance for which adaptive subdivision has not been performed. However, for such faces with a single extremum, the quality of the interpolant can be estimated closely by the maximum of B in Eq. (6) and the maximum difference of the irradiance values of the interior points along each edge of the triangle used in the quadratic interpolant construction.

VII. NUMERICAL TESTS AND QUALITY EVALUATION

To evaluate the quality of the algorithms for mesh reduction and degree selection we ran the algorithms on the “Simple Desk” and “Desk Drawers” scenes for the 9 light positions as described in Section III-B, as well as the two geometries for “Small” and “Long” light source configurations and finally “Big Scene” with two desks and two chairs. All images used for the

numerical evaluation (Tables III to VII) were computed using ray-casting, with the value of the visible point at each pixel being determined by evaluating the interpolant of calculating the exact backprojection (for the reference images). To measure error we use two error metrics: an area-weighted object-space square root error and an image-based square root error.

For the object-space error we compute a set of sampling points $\{p_j\}$ on each face f_i of the mesh, both in penumbra and in light. Define $I(p)$ to be the irradiance on a surface at point p and $\hat{I}(p)$ the approximation constructed by piecewise linear/quadratic interpolation. We have F faces in the scene and each face f_i has area A_i . The object space error ϵ_{os} is thus given as:

$$\epsilon_{os} = \sqrt{\frac{1}{\sum A_i} \sum_i \frac{A_i}{n} \sum_{j=0}^n (\hat{I}(p_j) - I(p_j))^2} \quad (7)$$

The error metric ϵ_{os} has the same units as irradiance (power per area). We also compute a reference image with analytic radiance values $E(i, j)$ and an approximate image $\hat{E}(i, j)$ using the interpolants. The image-based error ϵ_{is} for a $ni \times nj$ image, is given as:

$$\epsilon_{is} = \sqrt{\frac{\sum_i^{ni} \sum_j^{nj} (E(i, j) - \hat{E}(i, j))^2}{ni \, nj}} \quad (8)$$

The error metric ϵ_{is} is in pixel value differences and thus varies between 0 and 255. We compute ϵ_{is} for three difference view-points, the first is the view shown in Fig. 9, the second is similar to the image in Fig. 13 and a third which is a view on the other side of the desk. For “Big Scene” the view of Fig. 8 is also used.

A. Results for Mesh Reduction and Degree Selection Algorithms

Results are presented next in Tables III to VII. Table III gives the results for the simple combination of structured sampling with discontinuity meshing, Tables IV, V show the statistics for the mesh reduction algorithm, while Tables VI, VII shows the results for the degree selection approach.

In Tables IV, V, n_f is the original number of faces in the mesh, and n'_f is the reduced number, ϵ_{os} and ϵ_{is} are as defined above and % *Red.* is the percentage reduction of the number of faces. In Tables VI, VII, n_{eq} is the number of edges with quadratic interpolants, n_{el} is the number of edges with linear interpolants, and % *Lin.* the percentage of edges with linear interpolants.

The mesh reduction achieved is satisfactory (between 29% to 63%), while the error is globally low. Similarly, the number of edges characterized as linear is high (from 27% to 69.1%), allowing the use of cheaper, lower degree polynomial interpolants. Object-space error ϵ_{os} is low, growing slightly more than in other cases for mesh reduction in the case of the long light source (Table IV), for which the tolerance value results in higher mesh reduction. Image space error ϵ_{is} is also low, since it is less than a unit RGB pixel value for almost cases of mesh reduction or degree selection for the simple scene, and in the order less than 4 RGB pixel values for mesh reduction of the big scene (Table V).

We next present a first visual comparison of the images presented in Sections VI and V by showing the difference images

Scene	ϵ_{os}	ϵ_{is}
Average over 9 runs		
Simple Desk	0.019	0.461
Desk Drawers	0.025	0.664
Average over 2 runs		
Small Source	0.015	0.716
Long Source	0.024	1.623
Single run		
Big Scene	0.004	0.851

TABLE III
ERROR FOR COMBINED ALGORITHM

Scene	n_f	n'_f	%Red.	ϵ_{os}	ϵ_{is}
Average over 9 runs					
Simple Desk	344.9	244.1	29.2	0.016	0.052
Desk Drawers	1025.4	673.0	34.4	0.019	0.076
Average over 2 runs					
Small Source	421.5	260.5	38.2	0.018	0.444
Long Source	647.0	385.0	40.5	0.106	0.808
Single run					
Big Scene	3840.0	2529.0	34.1	0.016	1.260

TABLE IV
MESH REDUCTION: AREA TOLERANCE 0.01 (1%)

Scene	n_f	n'_f	%Red.	ϵ_{os}	ϵ_{is}
Average over 9 runs					
Simple Desk	344.9	159.9	53.6	0.021	0.067
Desk Drawers	1025.4	375.0	63.4	0.021	0.102
Average over 2 runs					
Small Source	421.5	187.0	55.6	0.028	0.854
Long Source	647.0	237.0	63.4	0.099	1.129
Single run					
Big Scene	3840.0	1689.0	56.0	0.022	3.946

TABLE V
MESH REDUCTION: AREA TOLERANCE 0.09 (9%)

Scene	n_{eq}	n_{el}	% Lin.	ϵ_{os}	ϵ_{is}
Average over 9 runs					
Simple Desk	1049.7	396.0	27.4	0.002	0.051
Desk Drawers	1909.2	1526.4	44.4	0.004	0.075
Average over 2 runs					
Small Source	863.0	993.5	53.5	0.012	1.348
Long Source	1426.5	866.0	37.8	0.012	0.813
Single run					
Big Scene	4736.0	6630.0	58.3	0.004	0.865

TABLE VI
DEGREE SELECTION TOLERANCE 0.001

Scene	n_{eq}	n_{el}	% Lin.	ϵ_{os}	ϵ_{is}
Average over 9 runs					
Simple Desk	1780.2	1111.1	38.4	0.019	0.063
Desk Drawers	2982.0	3889.3	56.6	0.024	0.097
Average over 2 runs					
Small Source	1702.5	2010.5	54.1	0.030	1.348
Long Source	2327.5	2257.5	49.2	0.121	1.124
Single run					
Big Scene	2443.0	8923.0	89.3	0.021	0.955

TABLE VII
DEGREE SELECTION TOLERANCE 0.008

Scene	t_{tot}	t_m	t_{bp}	t_{ss}	$t_r^{0.01}$	$t_r^{0.09}$
Average over 9 runs						
Simple Desk	21.40	15.43	2.98	0.79	0.16	0.16
Desk Drawers	116.56	73.83	33.68	1.20	1.04	0.74
Average over 2 runs						
Small Source	23.55	18.61	1.88	0.97	0.23	0.17
Long Source	56.56	40.46	10.75	0.80	0.49	0.36
Single run						
Big Scene	649.22	440.60	175.05	2.14	13.33	9.24

TABLE VIII
TIMING RESULTS



Fig. 18. Difference images (x20) for: (a) simple interpolant (b) reduced mesh.

multiplied by a factor of 20. As can be seen in Figs. 18 and 19 the differences are limited and small in magnitude (a dark pixel signifies no difference, and a totally white pixel a difference of 255 pixel levels in all three channels).

Finally, timing results are presented in Table VIII. All timings are in CPU seconds on an SGI Indy R5000 processor running at 150Mhz. The value t_{tot} is the total time spent to create the mesh, triangulate and construct the interpolants, t_m is the mesh construction time, t_{bp} is the backprojection calculation time, t_{ss} is the cost of structured sampling (as in [11] but using irradiance gradients [1]), $t_r^{0.01}$ is the cost of the edge removal algorithm for an area tolerance of 0.01 (1%) and $t_r^{0.09}$ is the removal time for area tolerance of 0.09 (9%). As can be seen from these statistics, the cost of the edge removal algorithm is negligible compared to the total cost of the algorithm.

B. Discussion of Numerical and Visual Results

The mesh reduction algorithm has presented good results for the scenes tested. For satisfactory mesh reduction (30-60%), the increase in error is in general minimal, indicating that the



Fig. 19. Difference image (x20) after degree selection.

complete discontinuity mesh is much larger than required for satisfactory reconstruction. However, there are cases in which the reconstruction quality can degrade, particularly when small faces are left in the mesh which force the creation of small or elongated triangles. Further geometric manipulation of the mesh can be used to eliminate these artifacts. One issue that is more difficult to address is that of animation. No provision is currently made for consistency in mesh reduction, and thus over animated sequences flickering can occur as the mesh changes from frame to frame. This should be addressed in the context of a more general incremental meshing algorithm for animation.

For degree selection, the results are also encouraging. With a good percentage of linear interpolants (30-70%) the increase in error is small, both in object and image space. Nonetheless, the method can be improved by incorporating some criterion based on the possible visible impact of the degree reduction, to avoid occasionally objectionable artifacts. This will require the use of perceptual error metrics.

VIII. SUMMARY AND DISCUSSION

An empirical and numerical study of irradiance in penumbral regions has been presented. Such a study was previously impractical due to the expense of computing irradiance values in the penumbrae. The use of the discontinuity mesh and the back-projection now makes such a study possible. It was found that in the majority of cases, irradiance in the penumbra is monotonic and thus amenable to reconstruction by linear or quadratic interpolants. Configurations that cause the appearance of extrema or irregular behavior were characterized. This study offers a better understanding of irradiance behavior in regions of partial occlusion, and guided the construction of an interpolating, interpolating, piecewise polynomial representation.

For the construction of the interpolant domains, the complete discontinuity mesh is used as a starting point. The faces of the mesh are triangulated, and the irradiance information is stored compactly with the mesh. The observations made from the empirical study suggested that many of the edges in the mesh are not actually required for satisfactory reconstruction of illumination in the penumbra. A mesh reduction algorithm is thus introduced, based on the removal of faces with small area and faces that cause concave regions in the mesh.

The observations of the empirical study also suggest that in many cases linear interpolation is sufficient for illumination reconstruction in the penumbra. An algorithm was presented which characterizes the edges in the mesh and the triangulation as requiring linear or quadratic degree polynomials to achieve high quality reconstruction. A set of basis functions was designed that allows the use of mixed degree polynomials for reconstruction.

Numerical tests were performed for a suite of moderately complex environments in which complicated shadow structures appear. The results show that both the mesh reduction and the degree selection algorithm can be applied without significant degradation of quality in the reconstruction.

This paper is a first attempt at comprehending irradiance behavior in penumbral regions. Much more work remains to be done. More detailed studies are needed of the geometric conditions leading to irradiance extrema in the penumbra. This will hopefully lead to a priori algorithms which will allow the

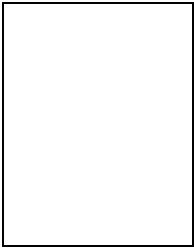
casting only of those discontinuity surfaces which contribute to significant illumination changes. In this manner the mesh will be simplified overall, and the meshing will be computationally cheaper. Such work will also result in much more reliable and effective error bounds, which are indispensable for global illumination algorithms [21]. The hierarchical global illumination algorithm incorporating discontinuity meshing and backprojections presented in [12] can use the mesh reduction approach in a straightforward manner. A first attempt at simplification in the presence of multiple sources was presented in [7], and evidently more work needs to be performed in combining the different simplification strategies, in particular for the application of discontinuity meshing to global illumination [23], [12].

IX. ACKNOWLEDGEMENTS

The authors would like to thank the various members of the University of Toronto Dynamic Graphics Project for their contributions to the discontinuity meshing system, notably Rob Lansdale, James Stewart and Pierre Poulin. The referees are thanked for the careful review of the manuscript and the many suggestions and corrections which greatly improved the quality of this paper. The research in this paper was funded in part by the University of Toronto, ITRC (Ontario), NSERC (Canada) and ERCIM (European Commission).

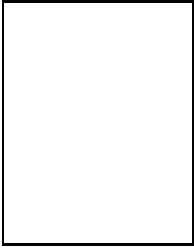
REFERENCES

- [1] James Arvo. The irradiance jacobian for partially occluded polyhedral sources. In *Computer Graphics Proceedings, Annual Conference Series: SIGGRAPH '94* (Orlando, FL). ACM SIGGRAPH, New York, July 1994.
- [2] A. T. Campbell, III and Donald S. Fussell. Adaptive mesh generation for global diffuse illumination. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 155–164, August 1990.
- [3] A. T. Campbell, III and Donald S. Fussell. Analytic illumination with polygonal light sources. Technical Report R-91-15, Dept. of Computer Sciences, Univ. of Texas at Austin, April 1991.
- [4] Norman Chin and Steven Feiner. Near real-time shadow generation using BSP trees. In Jeffrey Lane, editor, *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23, pages 99–106, July 1989.
- [5] Norman Chin and Steven Feiner. Fast object-precision shadow generation for areal light sources using BSP trees. In David Zeltzer, editor, *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, volume 25, pages 21–30, March 1992.
- [6] Michael F. Cohen and Donald P. Greenberg. The hemi-cube: A radiosity solution for complex environments. *Computer Graphics*, 19(3):31–40, July 1985. Proceedings SIGGRAPH '85 in San Francisco (USA).
- [7] George Drettakis. Simplifying the representation of radiance from multiple emitters. In *Proceedings of 5th EG Workshop on Rendering*, Darmstadt, Germany, June 1994.
- [8] George Drettakis. *Structured Sampling and Reconstruction of Illumination for Image Synthesis*. PhD thesis, Department of Computer Science, University of Toronto, January 1994. PhD Thesis, CSRI Tech. Report No. 293:ftp:ftp.csri.toronto.edu:csri-technical-reports/293.
- [9] George Drettakis and Eugene Fiume. Concrete computation of global illumination using structured sampling. *Third Eurographics Workshop on Rendering*, May 1992.
- [10] George Drettakis and Eugene Fiume. A fast shadow algorithm for area light sources using back projection. In *Computer Graphics Proceedings, Annual Conference Series: SIGGRAPH '94* (Orlando, FL), pages 223–230. ACM SIGGRAPH, New York, July 1994.
- [11] George Drettakis and Eugene L. Fiume. Accurate and consistent reconstruction of illumination functions using structured sampling. *Computer Graphics Forum (Proc. of Eurographics '93)*, 13(3):273–284, September 1993.
- [12] George Drettakis and Francois Sillion. Accurate visibility and meshing calculations for hierarchical radiosity. In *Proceedings of Seventh Eurographics Workshop on Rendering in Porto, Portugal*. Eurographics, June 1996.
- [13] Sherif Ghali and A. James Stewart. A complete treatment of d1 discontinuities in a discontinuity mesh. *Proceedings of Graphics Interface '96*, pages 122–131, May 1996.
- [14] Ziv Gigus, John Canny, and Raimund Seidel. Efficiently computing and representing aspect graphs of polyhedral objects. *IEEE Trans. on Pat. Matching & Mach. Intelligence*, 13(6), June 1991.
- [15] Ziv Gigus and Jitendra Malik. Computing the aspect graph for the line drawings of polyhedral objects. *IEEE Trans. on Pat. Matching & Mach. Intelligence*, 12(2), February 1990.
- [16] Steven Gortler, Michael F. Cohen, and Philipp Slusallek. Radiosity and relaxation methods. *submitted for publication*, 1993.
- [17] Pat Hanrahan, David Saltzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. *Computer Graphics*, 25(4):197–206, August 1991. SIGGRAPH '91 Las Vegas.
- [18] Stephen Hardt and Seth Teller. High-fidelity radiosity rendering at interactive rates. In *Proceedings of Seventh Eurographics Workshop on Rendering in Porto, Portugal*. Eurographics, June 1996.
- [19] Paul Heckbert. Discontinuity meshing for radiosity. *Third Eurographics Workshop on Rendering*, pages 203–226, May 1992.
- [20] Paul S. Heckbert and James M. Winget. Finite element methods for global illumination. Technical Report UCB/CSD 91/643, Computer Science Division (EECS), University of California, July 1991.
- [21] Dani Lischinski, Brian Smits, and Donald P. Greenberg. Bounds and error estimates for radiosity. In *Computer Graphics Proceedings, Annual Conference Series: SIGGRAPH '94* (Orlando, FL), pages 67–74. ACM, July 1994.
- [22] Dani Lischinski, Filippo Tampieri, and Donald P. Greenberg. Discontinuity meshing for accurate radiosity. *IEEE Computer Graphics and Applications*, 12(6):25–39, November 1992.
- [23] Dani Lischinski, Filippo Tampieri, and Donald P. Greenberg. Combining hierarchical radiosity and discontinuity meshing. In *Computer Graphics Proceedings, Annual Conference Series: SIGGRAPH '93* (Anaheim, CA, USA), pages 199–208. ACM, August 1993.
- [24] Tomoyuki Nishita and Eihachiro Nakamae. Continuous tone representation of three-dimensional objects taking account of shadows and inter-reflection. In B. A. Barsky, editor, *Computer Graphics (SIGGRAPH '85 Proceedings)*, volume 19, pages 23–30, July 1985.
- [25] P. M. Prenter. *Splines and Variational Methods*. John Wiley & Sons Inc, New York, 1989.
- [26] David Salesin, Dani Lischinski, and Tony DeRose. Reconstructing illumination functions with selected discontinuities. *Third Eurographics Workshop on Rendering*, pages 99–112, May 1992.
- [27] A. James Stewart and Sherif Ghali. An output sensitive algorithm for the computation of shadow boundaries. In *Canadian Conference on Computational Geometry*, pages 291–296, August 1993.
- [28] A. James Stewart and Sherif Ghali. Fast computation of shadow boundaries using spatial coherence and backprojections. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94* (Orlando, Florida, July 24–29, 1994), Computer Graphics Proceedings, Annual Conference Series, pages 231–238. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.
- [29] Filippo Tampieri. *Discontinuity Meshing for Radiosity Image Synthesis*. PhD thesis, Department of Computer Science, Cornell University, Ithaca, New York, 1993. PhD Thesis.
- [30] Seth J. Teller. Computing the antipenumbra of an area light. *Computer Graphics*, 26(4):139–148, July 1992. Proceedings of SIGGRAPH '92 in Chicago (USA).
- [31] Roy Troutman and Nelson L. Max. Radiosity algorithms using higher order finite element methods. In *Computer Graphics Proceedings, Annual Conference Series: SIGGRAPH '93* (Anaheim, CA, USA), pages 209–212. ACM SIGGRAPH, New York, August 1993.
- [32] Marc Vigo. An incremental algorithm to construct restricted delaunay triangulations. Technical Report LSI-95-43-R, Departament de Llenguatges i Sistemes Informatics, Universitat Politcnica de Catalunya, Barcelona, Spain, 1995.
- [33] Andrew Woo, Pierre Poulin, and Alain Fournier. A survey of shadow algorithms. *IEEE Computer Graphics and Applications*, 10(6):13–32, November 1990.
- [34] Harold R. Zatz. Galerkin radiosity: A higher-order solution method for global illumination. In *Computer Graphics Proceedings, Annual Conference Series: SIGGRAPH '93* (Anaheim, CA, USA), pages 213–220. ACM SIGGRAPH, New York, August 1993.



George Drettakis received his Ptychion (B.Sc. equivalent) in Computer Science at the University of Crete, Greece (1988). He received an M.Sc. and Ph.D. in Computer Science at the University of Toronto, Canada, in 1990 and 1994 respectively. He received an ERCIM postdoctoral fellowship which was performed at INRIA, (France), UPC, (Spain) and GMD (Germany) in 1994-1995. He holds the position of researcher at INRIA Rhône-Alpes, Grenoble, where he works in the iMAGIS/GRAVIR-INRIA project since October 1995. His research interests include global il-

lumination, visibility and shadow calculations, sampling, error estimation and the treatment of complex environments. He is a member of ACM and Eurographics.



Eugene Fiume is in the Department of Computer Science at the University of Toronto, where he has been on faculty since 1987. Over that period he has also been an NSERC University Research Fellow, and co-directs the Dynamic Graphics Project. Prior to his faculty appointment, he was Maitre Assistant at the University of Geneva, Switzerland. He holds Ph.D. and M.Sc. degrees from the University of Toronto, and a B.Math. from the University of Waterloo. He was a Visiting Professor at the University of Grenoble, France in 1996. He is currently Associate Di-

rector of CSRI the University of Toronto, and is a member of the Scientific Advisory Board of GMD, Germany. He is also a consulting Senior Scientist at Alias|Wavefront Inc. His research interests include most aspects of realistic computer graphics, including computer animation, modelling natural phenomena, illumination, image processing, sampling and filtering, and graphics software systems. He is a member of the ACM and Eurographics, and is an associate member of the IEEE.

2.4.2 Simplifying the Representation of Radiance from Multiple Emitters (EGRW'94)

Auteur : George Drettakis

Actes : 5th Eurographics Workshop on Rendering

Date : juin 1994

Simplifying the Representation of Radiance from Multiple Emitters

George Drettakis

iMAGIS / IMAG *

In recent work radiance function properties and discontinuity meshing have been used to construct high quality interpolants representing radiance. Such approaches do not consider the combined effect of multiple sources and thus perform unnecessary discontinuity meshing calculations and often construct interpolants with too fine subdivision. In this research we present an extended structured sampling algorithm that treats scenes with shadows and multiple sources. We then introduce an algorithm which simplifies the mesh based on the interaction of multiple sources. For unoccluded regions an a posteriori simplification technique is used. For regions in shadow, we first compute the maximal umbral/penumbral and penumbral/light boundaries. This construction facilitates the determination of whether full discontinuity meshing is required or whether it can be avoided due to the illumination from another source. An estimate of the error caused by potential simplification is used for this decision. Thus full discontinuity mesh calculation is only incurred in regions where it is necessary resulting in a more compact representation of radiance.

1 Sampling Illumination from Multiple Sources

To accurately render scenes illuminated by area light sources, it is necessary to represent the illumination on surfaces by a simpler, approximating function, even when considering only direct illumination. Piecewise polynomial interpolants are often chosen for this purpose. Such representations are an essential requirement for global illumination computation, in particular for the finite-element style approaches (e.g. [Zatz93, GSCH93]), which extend the radiosity-based method [CoGr85].

In the interpolant construction algorithms presented to date, much effort has been devoted to correctly treating shadow boundaries and identifying the behaviour of radiance. These methods have thus achieved high quality representation of illumination using simple functions. However, despite the significant advances in the field, little has been done to actually compensate for the cumulative effects of illumination from multiple emitters, be they light sources or secondary reflectors.

The importance of identifying these interactions is easy to see: when a single source is present, it may cast a detailed shadow which may require significant computation to

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

represent correctly. However, if a second source illuminates the same region in an unobstructed fashion, the shadow will be “washed out” leaving little need for the detailed representation. This phenomenon is illustrated in Fig. 7(a) and (b) (see colour section), in which one and two sources illuminate the environment respectively.

In this paper we propose a solution to this problem, by extending the techniques developed for discontinuity meshing and structured sampling [DrFi93, DrFi94, Dret94]. Throughout we consider only environments of diffusely reflecting surfaces lit by diffusely emitting sources. In the following section we present relevant previous work; we then present the extended structured sampling approach. We then briefly describe the two-pass discontinuity meshing algorithm which incurs the cost of full discontinuity meshing only in the regions required. In the sections that follow, we describe the simplification criteria for two sources, first for the intersection of unoccluded regions and then for the intersection of penumbral/unoccluded regions. For both cases we present first results of a prototype implementation. We then present the extension to multiple sources and summarise the results of the paper.

2 Previous Work

In previous work, the approximations used to represent radiance or radiosity have generally been guided by the requirements of the global illumination calculations. A simpler approach to constructing radiance representations is to examine illumination from a single emitter. The first such approach, in which the nature or structure of radiance for unoccluded regions is examined, was presented by Campbell and Fussell [Camp91]. They observed that radiance for these environments displays a single maximum. This idea was extended by Drettakis and Fiume [DrFi93], who constructed quadratic or linear interpolants tensor-product interpolants which can be shown to satisfy tight error bounds.

It has recently been shown that the computation of shadow boundaries, which are subsequently used to guide interpolant construction, is fundamental for high quality approximation of illumination. The first such work was performed in [Camp91] in which the boundary between penumbral and unoccluded regions was computed. The resulting mesh was then used to build an approximation of radiance of constant-radiance triangular elements. Similar work was performed by Chin and Feiner [ChFe92]. Lischinski et al. [LiTG92] were the first to consider discontinuity surfaces interior to the penumbra, that signify a change in the topological view of the light source (e.g. the appearance or disappearance of a vertex or an edge in the visible portion of the source). They subsequently built a triangulation of the receiver surfaces based on the subdivision of this mesh, and constructed quadratic interpolants over these triangles. A different algorithm was presented by Heckbert [Heck92], in which a similar mesh is computed. Lischinski et al. [LiTG92] also merged the meshes from multiple sources, but no simplification was attempted. In this paper we extend the approach developed in [DrFi94, Dret94]. In this approach the *complete discontinuity mesh* is calculated: the environment is segmented into regions (mesh *faces*), in which the topological structure of the visible region of the source does not change. An abstract representation of the visible part of source, called a *backprojection*, is stored with each face. An example of such a mesh is shown

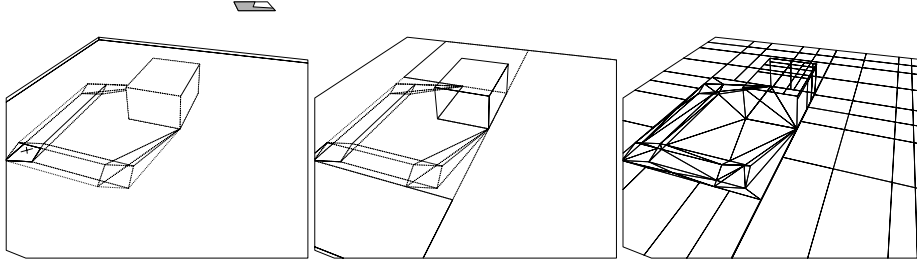


Fig. 1. (a) Mesh and Backprojection (b) Segmentation into Light and Penumbra and (c) Triangular/Tensor Product Interpolants

in Fig. 1(a), where the backprojection of the point marked with a cross is shaded on the source. In [Dret94], the complete mesh is used to construct linear and quadratic interpolants representing radiance in the penumbra. In addition, the structured sampling approach of [DrFi93] was extended ([Dret94]) to handle environments with shadows in the following way. First all regions of shadow are identified and enclosed in a bounding box. Such a regular region enclosing a region of penumbral and umbral faces is called a *penumbral group*. The remaining parts of the receivers (which are unoccluded) are segmented into parallelograms (Fig. 1(b)) on which the structured sampling algorithm is used to create tensor-product interpolants as in [DrFi93] (Fig. 1(c)). Notice in Fig. 1(c) how in the regions of penumbra triangular interpolants are used, while in the unoccluded regions sparse tensor product representations suffice.

3 Extending Structured Sampling for Multiple Emitters

For the purpose of computing reference images in scenes with multiple sources, the discontinuity mesh from each source can be computed independently, and stored with the surface. When rendering using ray-casting, each mesh is queried, the backprojection retrieved for each mesh corresponding to each source and the exact radiance value computed.

To obtain the merged mesh due to several sources, the meshes corresponding to each source are combined. This is performed simply by adding the faces of one mesh into the other. If two light regions with tensor-products are combined, the merged region will contain tensor-product interpolants, while in every other case (penumbra or umbra combined with penumbra, penumbra or umbra combined with unoccluded) the resulting mesh faces will be triangulated and a combined interpolant built.

By using the structured algorithm in [DrFi93] the radiance function in unoccluded regions for each source is split into regions in which the radiance is well behaved. The algorithm then creates quadratic interpolants and guarantees that the interpolants satisfies tight error bounds. Thus, the combined illumination function over the intersection of two light regions will continue to satisfy these error bounds. Similarly, for the other regions the combination of triangular or tensor-product interpolants is also guaranteed to give high quality results, since the regions have been segmented based on the complete discontinuity mesh.

Because of the guaranteed error bounds for the interpolants representing unoccluded illumination, we can safely use these approximations in our calculations for simplification (see below), instead of the more expensive direct illumination calculation.

4 Two-Pass “On Demand” Discontinuity Meshing

The main cost of the complete discontinuity meshing algorithm is due to the relatively large number of discontinuity surfaces that must be traced in the environment. In addition, it is necessary to search for the existence of discontinuity surfaces (either edge-vertex wedges (EV) or triple-edge quadric surfaces (EEE)), formed by edges and vertices not on the source. To reduce the cost of this computation, we must reduce the number of surfaces traced into the environment.

To do this we separate the mesh computation into two phases: first, the computation of the boundary between light and penumbra, and an estimate of the region between umbra and penumbra, and second the full computation of all discontinuity surfaces interior to the penumbra *only* when required. We call the boundary between penumbra and light the maximal boundary, and the boundary between umbra and penumbra the minimal boundary. The combined maximal and minimal boundary is called the *extremal* boundary.

4.1 Extremal Boundary Approximation

The computation of the maximal boundary can be performed exactly, since it is formed exclusively by EV surfaces [Camp91]. Thus these events can be identified in constant time for each object, and subsequently propagated into the environment. The minimal boundary can include EEE events [Tell92], which can be treated by the method described in [DrFi94].

As an example consider the scene shown in Fig. 2. On the left we see the full discontinuity mesh, and on the right the extremal boundary.

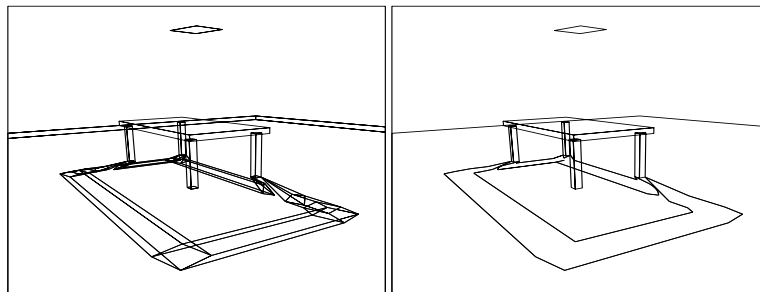


Fig. 2. Complete Mesh vs. Extremal Boundary

The number of discontinuity surfaces traced through the environment is thus reduced significantly. In addition, since no internal detail of the mesh is computed, all non-emitter events are ignored, and the search time for such events is eliminated.

The computation time for the extremal boundary is significantly reduced compared to the computation of the full mesh. In Table 1 we compare the cost of complete discontinuity meshing to the cost of the extremal boundary for the scenes shown in Fig. 1 and Fig. 2, as well as two other more complex scenes. As we can see, the cost of the complete mesh computation is three to four times higher than the just the extremal boundary. It is thus evident that large gains can be achieved if the complete mesh need be computed only when required.

Table 1. Computation time for Complete Mesh and Extremal Boundary

Scene	Polygons	Complete Mesh	Extremal Boundary	Ratio Complete/ Boundary
<i>Box Scene</i>	14	0.74 sec	0.16 sec	4.6
<i>Table Scene</i>	36	1.01 sec	0.31 sec	3.2
<i>Desk Scene</i>	182	17.20 sec	4.42 sec	3.8
<i>Desk & Chair Scene</i>	288	35.20 sec	9.20 sec	3.8

4.2 Local Complete Mesh Construction

As discussed earlier, one of the goals of the approach presented here is to compute portions of the discontinuity mesh only when necessary. The discontinuity meshing algorithm presented in ([Dret94, DrFi94]) is particularly well suited to such an extension.

Given a convex region defined on a receiver for which the complete mesh is desired, a convex volume defined by the source and that region is defined. Using the same spatial subdivision structure as in [DrFi94], the objects contained in this volume can be found efficiently.

To create the full mesh *locally* in the desired region of the receiver, the discontinuity meshing algorithm of [DrFi94] is applied using only the objects within the volume. In this manner, a much smaller number of discontinuity surfaces are traced (only those corresponding to edges and vertices of the selected objects), and the number and expense of searches for non-emitter events is also limited.

5 Simplification Criteria

In this section we discuss the simplification criteria used when two meshes are combined. In Section 6 we show how this applies to an arbitrary number of sources. Consider a source S_1 and a source S_2 . We assume that we have computed the extremal boundaries for the discontinuity mesh for each source, that the environment has been segmented into parallelogram regions of light and penumbra/umbra. We also assume that the structured sampling algorithm has been applied, subdividing the regions of light. In each such unoccluded region a biquadratic tensor product interpolant has been built, which represents the radiance function accurately within strict error bounds. We call each such mesh the simplified mesh for source S . In Fig. 3 we show the simplified mesh for each source for the scene of Fig. 7 (see colour section).

Given the two meshes, M_1 and M_2 respectively, we proceed to “add” M_2 into M_1 . Merging is performed this way purely for reasons of algorithmic simplicity. There are three cases that must be treated:

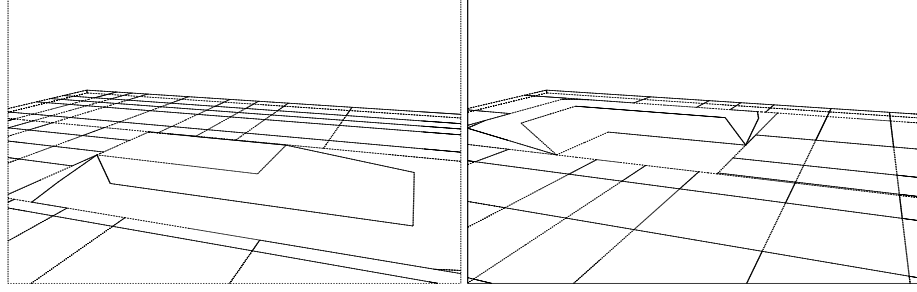


Fig. 3. Simplified Meshes for Box Scene

1. Merging light faces of M_1 with light faces of M_2 .
2. Merging light faces of one mesh with penumbral faces of the other.
3. Merging penumbral faces of one mesh with penumbral faces of the other.

For the first case, since we have the structured representation in the form of tensor product interpolants for both meshes, we use an a posteriori error estimation to determine whether simplification can be performed. For the second case, we determine the regions of the penumbral group for which complete meshing is necessary. For the third case we currently perform no simplification.

5.1 Light-Light Simplification

The simplest case is the insertion of an unoccluded (light) face F_2 of M_2 , into the mesh M_1 . The mesh M_1 is searched to find all faces contained inside the boundary of the face F_2 being added. Call these faces $\{f_1, f_2, \dots, f_n\}$. Within each such light face f_i of mesh M_1 , a (structured) biquadratic interpolant $s_i^1(x, y)$ has been defined. Correspondingly, the structured interpolant in F_2 is $s_2^2(x, y)$.

To determine whether simplification is possible, we proceed to construct two biquadratic interpolants: first a high quality representation of the combined radiance with the region of F_2 , denoted $s_\cap(x, y)$ and second a simplified representation, $\tilde{s}(x, y)$. The error incurred by the simpler interpolant (compared to the high-quality interpolant) is used to determine whether simplification can be achieved.

The high-quality interpolant $s_\cap(x, y)$ is defined as follows, in a piecewise fashion over each f_i (this is the interpolant created when combining the meshes as in Section 3):

$$s_\cap = s_2^2(x, y) + s_i^1(x, y), (x, y) \in f_i \quad (1)$$

Since the interpolants $s_i^j(x, y)$, $j = 1, 2$ already constructed are good approximations of the actual radiance function, $s_\cap(x, y)$ is considered to be an accurate approximation of the combined function over the entire domain $F_2 = \cup f_i$.

The second interpolant $\tilde{s}(x, y)$ is defined over F_2 as a simple 9-point biquadratic tensor product, for which the midpoints are used as internal defining nodes. The nodal values are found by querying $s_\cap(x, y)$.

To determine whether the combined illumination from two sources can be represented accurately by the simplified interpolant $\tilde{s}(x, y)$, we use standard approximation theoretic error estimate [Pren89]. As a first approach we compute the L_2 -norm of the difference of the simplified and the accurate interpolants.

For the L_2 -norm the following quantity is computed:

$$L_2 = \sqrt{\int \int_{F_2} (\tilde{s}(x, y) - s_{\cap}(x, y))^2 dx dy} \quad (2)$$

This integral is computed in a piecewise fashion over each tensor product domain f_i . Since both $\tilde{s}(x, y)$ and $s_{\cap}(x, y)$ are quadratic functions, the integral of Eq.(2) can be computed analytically. In practice, the analytic expression is large and numerically unstable, so a two-dimensional Gauss-Legendre quadrature rule is used. In many cases, the quadrature can give exact results.

If the L_2 -norm is less than a user-specified tolerance, the edges of the faces f_i are removed, and radiance in the domain of F_2 is represented by the simplified interpolant $\tilde{s}(x, y)$.

In Fig. 4, we show the result of the simplification criteria applied to a scene of two sources with no shadows. In Fig. 4(a) the original mesh is shown. From Fig. 4(b) it can be seen that T-vertices have been introduced into the mesh. To ensure C^0 continuity, T-vertices are treated as “slave-nodes”. First all interpolants of simplified faces are constructed. For each T-vertex, the corresponding value of the neighbouring simplified interpolant replaces the previously assigned nodal value. In Fig. 8(a) (in the colour

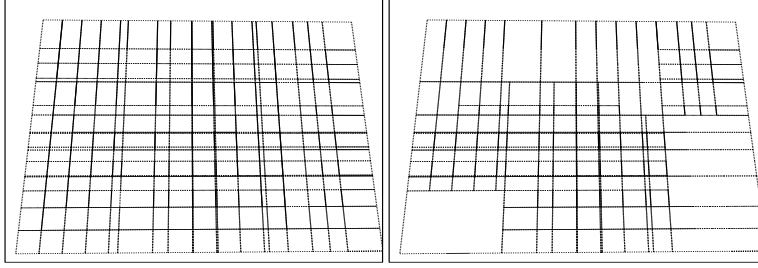


Fig. 4. (a) Original Unoccluded Mesh, and (b) Simplified Mesh

section) we show the image rendered using the original full mesh interpolant. In Fig. 8(b) the result of the construction of the continuous interpolants for the simplified mesh is shown. As can be seen, the resulting images show little difference. However, a more graded variation between simplified and unsimplified regions would be beneficial, using a form of restricted meshing.

5.2 Light-Penumbra Simplification

Consider a penumbral group of a mesh M_P caused by source S_P and a set of light faces of the mesh M_L caused by source S_L , which are contained or cut the penumbral

group. We wish to add the light faces into the mesh M_P , and to determine the regions of the penumbral group for which the complete discontinuity mesh must be computed. In contrast to the light-light case, we do not have an accurate representation of the radiance in the penumbra.

To determine whether detailed mesh computation is required, we first construct a medium quality approximation $\hat{s}_\cap(x, y)$ to the radiance in the penumbra, using the extremal boundary, within each light face of M_L . This piecewise approximation takes into account the extremal boundaries of the various sources, and its use is equivalent to the accurate interpolant $s_\cap(x, y)$ for the light-light case. We then construct the simplified interpolant by defining a single biquadratic tensor product $\hat{s}(x, y)$. The simplification criteria used are the same as in the light-light case.

The construction of $\hat{s}_\cap(x, y)$ proceeds as follows. We first construct an independent mesh defined by the bounding box of the penumbral group. We then add in the extremal boundary of the group of mesh M_P . We show this construction for the box scene and the penumbral group of one source in Fig. 5(a) (refer to Fig. 3(a) and Fig. 7 (colour section) to understand the geometry). In this way, a coarse segmentation of the penumbral group into regions of light, penumbra and umbra has been achieved.

For each vertex inserted into the independent mesh the appropriate illumination value due to source S_P is assigned. For the vertices on the maximal boundary or in the unoccluded regions this is the direct unoccluded illumination from S_P and for the points on the minimal boundary the value is 0. We then insert all the light faces of M_L

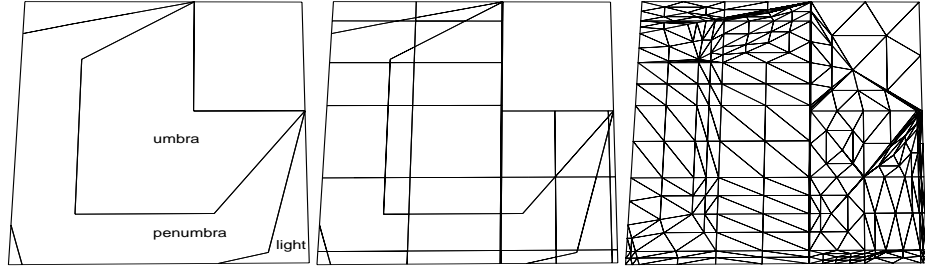


Fig. 5. Mesh for Error Testing: (a) Maximal/Minimal Boundary of penumbral group of M_P , (b) Light faces of M_L added, (c) Triangulation (domain of $\hat{s}_\cap(x, y)$)

that intersect or are contained in the penumbral group boundary (Fig. 5(b)). For the resulting vertices the value of unoccluded illumination is retrieved from the appropriate interpolants of M_L , but it is then necessary to add the appropriate (penumbral) value due to the source S_P . For regions of umbra and light (due to source S_P) this can be found simply. For vertices in regions within the penumbra however it is necessary to retrieve an estimate of the radiance value. This can be achieved by estimating the derivative value of radiance (see below).

The resulting combined mesh is then triangulated (Fig. 5(c)), and the piecewise elements of the interpolant $\hat{s}_\cap(x, y)$ are built. Interior nodal values are computed either directly (if in a region of light or umbra) from the appropriate interpolants in M_L and

M_P , or are averages of the neighbouring nodes if the node is within the penumbra.

For each region corresponding to a light face F_L , the interpolant $\hat{s}(x, y)$ is constructed. This interpolant is a simple 9-point bi-quadratic Lagrange interpolant. The values for nodes corresponding to vertices in the combined mesh have already been assigned and those that remain are found by querying the interpolant $\hat{s}_\cap(x, y)$. We then compute the L_2 -norm error in the same manner as for the unoccluded case for the triangles of $\hat{s}(x, y)$ which lie in umbra or penumbra. The integral is computed over each triangle included in the domain of the light face F_L . If the L_2 error is less than the pre-defined tolerance, the edges of F_L are inserted into M_P , the extremal boundary edges contained in F_L are removed from the mesh M_P and radiance within this region is represented by the simplified interpolant.

If the error is greater than the user specified tolerance, the region of the original light face is marked as requiring further meshing. After processing all light faces, the complete mesh is locally computed only for the regions required.

5.3 First Implementation and Discussion

To verify the algorithm, we have implemented the light-penumbra simplification by first computing the complete mesh, and then simplifying the mesh where appropriate. The full construction of the extremal boundary and the simplification algorithm have been implemented as described above, with the exception of the local backprojection estimate. Instead, for the light-face vertices within penumbra, the exact penumbral radiance is retrieved from the (complete) mesh of source S_P .

As mentioned above, for the penumbral regions only the portions of the simplified mesh in penumbra or umbra are taken into consideration for the L_2 -norm computation. As noted in [Dret94], edges leading to a singular vertex display a particularly rapid variation. To correctly account for this, in faces for which singular edges exist the light faces are also considered in the L_2 -norm calculation.

The results of the implementation are shown in Fig. 6. We first show the unsimplified combined mesh (a), and then the simplified mesh for tolerance values 0.005 and 0.001 respectively (b) and (c). The corresponding shaded images are shown in Fig.

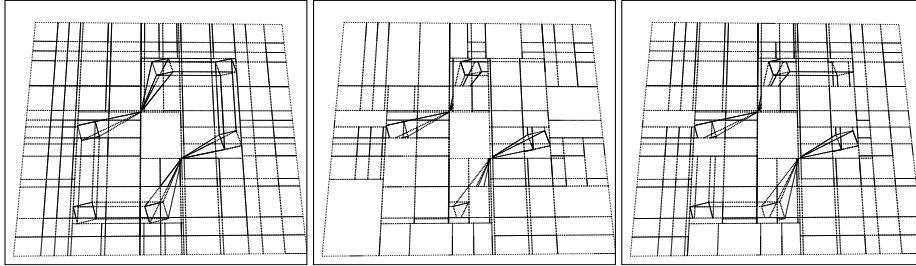


Fig. 6. (a) Unsimplified Combined Mesh and Simplified Mesh for (b) Tolerance 0.005 and (c) 0.001

9(a),(b) in the colour section. The results of a more complicated test are shown in Fig.

?? to ??. The complete mesh and resulting image are shown in Fig. ??, and the reduced meshes and images in Fig. ?? and ?? for tolerances equal to 0.1 and 0.005 respectively.

Overall the method shows promising first results. Little difference can be seen in the simplified images compared to complete mesh image for the simple scene (Fig. 7 (b)), and the simplification appears to occur in desirable regions of the mesh as the tolerance grows. Similarly the simplified images for the table scene (Fig. ??, ??) appear to maintain relatively high quality, since simplification occurs in the regions in which the detail of the penumbra is not very important.

In the tests performed it can be seen that the use of the L_2 -norm can sometimes cause undesirable simplification (e.g., the shadow boundary of the front leg in Fig. ??(a)). A possible solution is to maintain the extremal boundary instead of substituting with a tensor product.

5.4 Penumbra Radiance Estimates

Given the maximal and minimal boundary we propose here an estimate of the radiance at any point in the penumbra using local backprojection information. By construction, the minimal or maximal edges of the discontinuity mesh include information about the local change of the backprojection. Thus a good estimate of the radiance at a point P , known to be in penumbra, can be found by approximation.

To perform this approximation we first find the edge on the minimal boundary for which the two endpoints are closest to P . We then calculate the backprojection into the penumbra locally in a direction defined by the midpoint of the minimal edge and the point P . Given the backprojection, we estimate the radiance derivative, then build a Hermite cubic from the values and the derivative estimates, and determine the radiance value at P using the cubic. Experimental verification will determine the quality of this approach.

6 Treating Multiple Sources

The simplification algorithm begins by computing the extremal boundary for each of the n sources in the scene. The light regions are computed, and the structured algorithm run for each surface. The result is a list of simplified meshes for each surface: $\{M_1, \dots, M_n\}$. The algorithm proceeds by merging the first two meshes. The combined mesh M_c is then merged with mesh M_3 etc.

For a pair $\{M_c, M_j\}$ we first insert the light faces of M_j into the mesh. If a light face of M_j contains exclusively light faces of M_c , or there is a parallelogram subregion of M_j with this property, the light-light simplification is applied. The penumbral regions of both meshes are then visited, and the simplification algorithm is run for each penumbral group. A list of regions marked as “potentially requiring meshing” is stored, together with a pointer to the appropriate source. In addition, the interpolant $\hat{s}_\cap(x, y)$ is stored and used in subsequent tests for error bound checking. If a subsequent source eliminates the need for the meshing, the corresponding regions are deleted from the list. At the end of this process, there will be a list of regions for which the complete mesh is applied.

7 Conclusions

In this paper we have presented an algorithm which allows more compact representation of radiance due to multiple emitters based on careful error analysis, and allows the cost of discontinuity meshing to be deferred until it is required.

To achieve this goal, the L_2 -norm is used to compare an accurate representation of radiance over a domain with a simpler one. When the simpler interpolant satisfies a given error tolerance, it is used. For regions with unobstructed views of all sources, this is performed as an a posteriori step. For regions in penumbra for one source and light for another, a low-quality discontinuity mesh is first computed, and an approximation to radiance built, which is then compared to a simpler interpolant. Results of a first implementation show promising reduction of the mesh, and good quality images when using the simplified interpolant.

For the future, it is extremely interesting to apply these ideas to complex environments with many sources, to determine the savings, both in the representation of unoccluded regions, but more importantly in the computation time for discontinuity meshing. The subsequent step is the usage of these algorithms in a global illumination context, since for secondary reflection the need for complete meshing is highly unlikely.

8 Acknowledgments

The author is a post-doctoral fellow hosted by INRIA, under an ERCIM fellowship. Many of the ideas presented originate in the authors' Ph.D. thesis [Dret94] at the University of Toronto, under the supervision of Prof. Eugene Fiume. The software system used for the implementation includes many software components written by researchers at the Dynamic Graphics Project at Toronto.

References

- [Camp91] Campbell, A. T. III, Fussell, D. S.: An Analytic Approach to Illumination with Area Light Sources, Technical Report TR-91-25, Computer Science Dept., University of Texas at Austin, August 1991.
- [ChFe92] Chin N., Feiner S.: Fast Object Precision Shadow Generation for Area Light Source using BSP Trees, ACM Computer Graphics (SIGGRAPH Symp. on Interactive 3D Graphics), March 1992.
- [CoGr85] Michael F., Greenberg D. P. : The Hemi-cube, A Radiosity Solution For Complex Environments, ACM Computer Graphics (SIGGRAPH '85 Proceedings), Vol. 19, No. 3, July 1985.
- [DrFi93] Drettakis G., Fiume E. L. : Accurate and Consistent Reconstruction of Illumination Functions Using Structured Sampling, Proceedings of the Eurographics Conference, Barcelona, Spain, September 1993.
- [DrFi94] Drettakis, G., Fiume E. L. : A Fast and Accurate Shadow Algorithm for Area Light Sources Using Backprojection ACM SIGGRAPH Computer Graphics, Annual Conference Series, July 1994.
- [Dret94] Drettakis G.: Structured Sampling and Reconstruction of Illumination for Image Synthesis Ph.D. Thesis, Dept. of Computer Science, University of Toronto, also available as CSRI Tech. Report-293 (ftp site ftp.csri.toronto.edu:csri-technical-reports/293), January 1994.

- [GSCH93] Gortler S. J., Schroeder P., Cohen M. F., Hanrahan P.: Wavelet Radiosity, ACM SIGGRAPH Computer Graphics, Annual Conference Series, August 1993.
- [HaSA91] Hanrahan P., Salzman D., Aupperle L.: A Rapid Hierarchical Radiosity Algorithm, ACM Computer Graphics (SIGGRAPH '91 Proceedings).
- [Heck92] Heckbert P.: Discontinuity Meshing for Radiosity, 3rd Eurographics Workshop on Rendering, Bristol, UK May 1992.
- [HeWi91] Heckbert P., Winget, J. M.: Finite Element Methods for Global Illumination, EECS/University of California Berkeley, Report No. UCB/CSD 91/643, July 1991.
- [LiTG92] Lischinski D., Tampieri F., Greenberg D. P.: Discontinuity Meshing for Accurate Radiosity, IEEE Computer Graphics and Applications, November 1992.
- [NiNa83] Nishita, T., Nakamae, E. : Half-tone Representation of 3-D Objects Illuminated by Area Sources or Polyhedron Sources, CompSAC, Proc. IEEE 7th Intl. Comp. Soft and Applications Conference, 237–242, November 1983.
- [Pren89] Prenter, P.M.: Splines and Variational Methods, John Wiley and Sons, New York, 1989.
- [SaLD92] Salesin D., Lischinski D., DeRose T. : Reconstructing Illumination Functions with Selected Discontinuities, 3rd Eurographics Workshop on Rendering, Bristol, UK May 1992.
- [Tell92] Teller S.: Computing the Antipenumbra of an Area Light Source ACM Computer Graphics (SIGGRAPH 92 Proceedings), July 1992.
- [Zatz93] Zatz H. R.: Galerkin Radiosity: A Higher Order Solution Method for Global Illumination, ACM Computer Graphics (SIGGRAPH '93 Proceedings), August 1993.

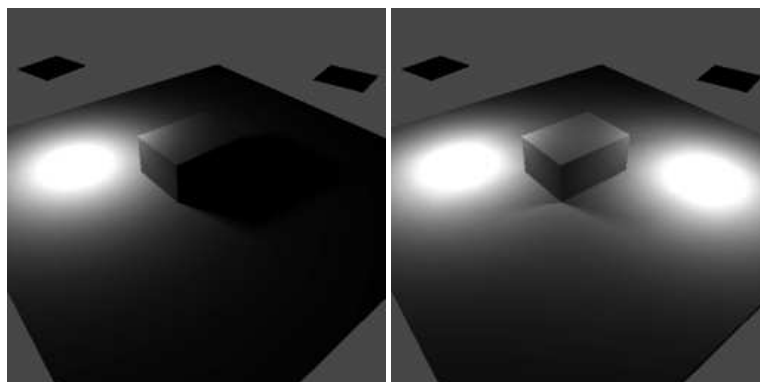


Fig. 7. (a) One Source and (b) Two Source Images

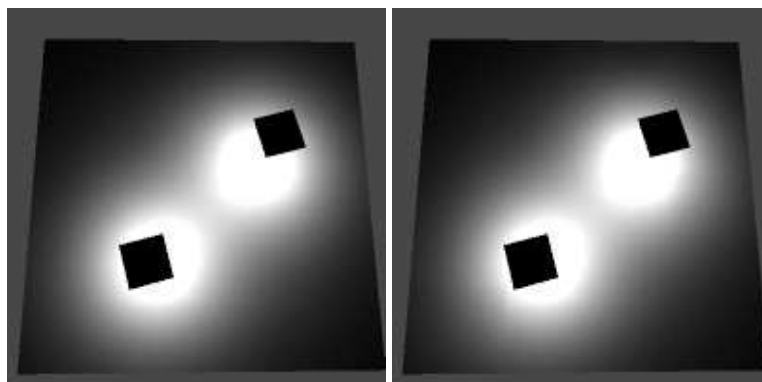


Fig. 8. Images for (a) Original and (b) Simplified Unoccluded Meshes

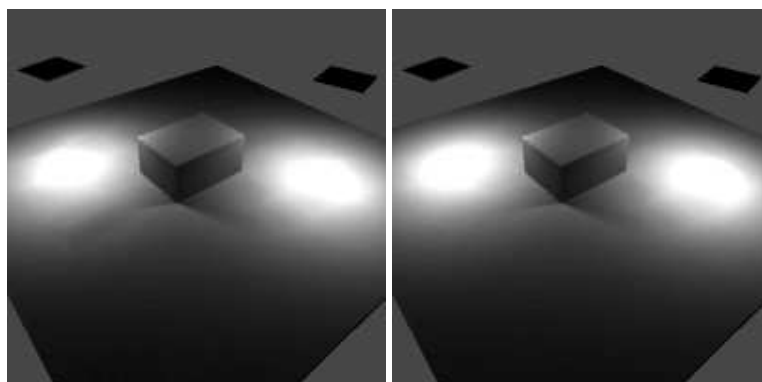


Fig. 9. Images of Simplified Meshes (a) Tolerance = 0.005, (b) Tolerance = 0.001

2.4.3 Accurate Visibility and Meshing Calculations for Hierarchical Radiosity (EGRW'96)

Auteur : George Drettakis et François Sillion

Actes : 6th Eurographics Workshop on Rendering

Date : juin 1996

Accurate Visibility and Meshing Calculations for Hierarchical Radiosity

George Drettakis, François Sillion

iMAGIS * Laboratoire GRAVIR/IMAG-INRIA

Abstract: Precise quality control for hierarchical lighting simulations is still a hard problem, due in part to the difficulty of analysing the source of error and to the close interactions between different components of the algorithm. In this paper we attempt to address this issue by examining two of the most central components of these algorithms: *visibility* computation and the *mesh*. We first present an investigation tool in the form of a new hierarchical algorithm: this algorithmic extension encapsulates exact visibility information with respect to the light source in the form of the *backprojection* data structure, and allows the use of *discontinuity meshes* in the solution hierarchy. This tool permits us to study separately the effects of visibility and meshing error on image quality, computational expense as well as solution convergence. Initial experimental results are presented by comparing standard quadtree-based hierarchical radiosity with point-sampling visibility to the approaches incorporating backprojections, discontinuity meshes or both.

1 Introduction

Hierarchical simulation techniques have received a lot of attention in research environments, but their practical use remains impaired by the difficulty of controlling the speed/accuracy tradeoff on which they are based. Error control and solution accuracy issues have been studied to a certain extent for global illumination algorithms [12, 1]. These studies provided a useful categorization of possible error sources, and offered a general framework for error-driven hierarchical refinement. Nonetheless, little has been done in terms of investigating the different causes of error in Hierarchical Radiosity (HR) in particular, and very little is currently known about the quantitative effects on error of different algorithmic choices used during the lighting simulation.

In this paper we attempt to address this problem by providing recommendations, based on theoretical discussion and initial experimental results. We will concentrate our efforts on meshing and visibility computation strategies. We begin by presenting a non-exhaustive list of important algorithmic components in HR and we mention the algorithms that have been proposed to improve these aspects of the simulation. For most of these factors, the precise impact on image or solution quality, as well as possible interactions between them, has not been thoroughly studied.

Important components of the HR simulation algorithm Broadly speaking, two main categories of factors affecting simulation can be identified (following [1]): *discretisation*, concerning mainly issues of mesh construction and data structures, and *computation* which involves the aspects of the algorithm related to form-factor and visibility computation as well as refinement strategy and convergence.

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

Discretisation

Meshing strategy HR relies on the ability to evaluate interactions (energy transfers) at different levels of a hierarchy in the description of the scene. Previous algorithms typically used simple recursive subdivision structures such as the quadtree to represent hierarchical meshes. Another approach consists of computing a *discontinuity mesh* (DM) for much improved representation of direct illumination.

Mixed meshes Simple hierarchical structures such as quadtrees are easy to implement and compact to store, because they rely on implicit information. However they are not suited to some geometrical or topological situations such as the representation of shadow boundaries. Mixed meshes with both triangles and quadrilaterals can therefore be used, and must provide access to connectivity information.

Computation

Visibility calculation Many radiosity implementations to date use point sampling to evaluate visibility factors. Discontinuity meshes, when equipped with the associated visibility information (backprojection) can provide exact visibility computation for direct illumination.

Refinement Strategy Refinement criteria (sometimes called “oracles”) are the core of the HR formulation. Many different criteria have been devised, using varying amounts of information. Possible variables for the refinement decisions are form factor estimates, visibility status, estimate of form factor variance, estimate of radiosity transfer, etc.

Point or Area-based form-factor computation As shown by the work of Wallace *et al.* [20] for progressive refinement radiosity, higher-quality solutions can be obtained when computing radiosity directly at mesh vertices. Area-to-area form factors require an extrapolation/interpolation step which effectively smoothes out some of the defects in the solution but also “blurs” the computed solution in ways which are difficult to quantify.

Convergence The benefits of HR really become apparent when a global solution is sought, i.e. with all interreflection effects. HR convergence is an issue that has received limited attention and it is not known whether some of the choices mentioned above have a significant impact on convergence.

2 Previous Work

2.1 Discontinuity meshing and backprojections

Discontinuity meshing ([9, 12, 5]) has been used to a certain extent for global illumination and HR calculations, but the meshes used have always been *partial* since they do not capture EEE and other important discontinuity surfaces. A result of this simplification is that *backprojections* (defined in e.g. [3, 16, 18]) cannot be computed by these algorithms. Backprojections are data structures permitting efficient determination of the visible part of the source anywhere in the penumbra, thus effectively eliminating all visibility calculation error with respect to the light sources. Backprojections can *only* be computed together with the complete discontinuity mesh (i.e. including EEE and degenerate events), and thus have never been used so far in the context of HR.

2.2 Combination of HR and DM

The most relevant approach to our work is that of Lischinski et al. [12]. In their work a global solution is computed using a BSP tree. When a node is split, an appropriate discontinuity line is chosen. A local pass is subsequently used for display, during which analytic visibility (mainly for primary sources) is computed using an expensive polygon clipping operation (as in [10, 17]). The main difference with the method we present here is the fact that exact visibility was not taken into account during the solution process and thus the different factors (visibility, meshing) affecting error could not be isolated or analysed. Gatenby and Hewitt [5] also developed a hierarchical solution for progressive refinement, but little was presented in terms of solution quality evaluation.

2.3 Error estimation and control

A detailed theoretical presentation of error analysis was developed by Arvo et al [1]. According to their classification, we will be dealing with *discretisation error* (meshing) and *computational error* (visibility calculations). For both types of error, little is known in practice or in quantitative terms. Lischinski et al. [11] have also developed an approach based on error bounds for HR. The approach we present here could be integrated into a system of this type, providing tighter upper and lower bounds in the most difficult cases, those of partial visibility.

3 Efficient combination of HR and backprojections

The first step to allow experimental comparisons of the effect of meshing and visibility calculations on the solution, is the introduction of a new algorithm incorporating backprojections (i.e. exact visibility calculations) and the complete discontinuity mesh in HR. Other than backprojections, this new algorithm constructs a full hierarchy on the input surfaces by clustering the elements of the discontinuity mesh. This allows the use of quadtrees in unoccluded regions as opposed to the use of BSP trees everywhere as in previous partial DM solutions ([12]), while using irregular triangles in shadow regions.

3.1 Accurate visibility computation using backprojections

Given a three-dimensional scene we construct the full discontinuity mesh (with EEE and degenerate events) which has the following property: each cell or face of the mesh contains a data structure called a backprojection which fully describes the partial visibility of the source at each point within the mesh face [3, 16]. An example of such a mesh is shown in Fig. 1(a).

The availability of the backprojections allows us to determine at a low cost the visible part of the source and thus the analytical value of the differential form-factor $dF_{x,S}$, from any point x in the penumbra to the source S . Because no visibility calculation is needed, an accurate estimate of the form factor value is available cheaply during refinement. As we will see below, the exact value is also used to determine irradiance values at vertices within the mesh.

3.2 Mixed triangle-quadrilateral meshes

The discontinuity meshing approach presented in [3, 4] constructs a mixed, non-hierarchical mesh containing triangles in penumbra and in irregular regions of light, and quadrilaterals in large regions of light. The goal here is to create a hierarchy suitable for HR solutions, starting with the DM.

To this end, quadtrees should be used where possible (in particular in unoccluded regions) due to their simplicity and ease of use, while irregular triangular meshes should be used around shadow boundaries.

The mixed mesh structure poses certain problems of connectivity, since inhomogeneous mesh elements co-exist. In particular, neighbour-finding is handled by adding simple adjacency information at the vertices of the original mesh. When a quadrilateral or a triangle are subsequently subdivided, regular quadtrees or regular triangle hierarchies are created. An example of an initial mesh (before subdivision) for our test scene is shown in Figure 1(b) (see next section for the construction algorithm); the same mesh subdivided after iterating is shown in Figure 4(b). When searching for a neighbour within a regular mesh (triangular or quadrilateral) implicit neighbourhood relationships are maintained, and when crossing a shared edge, the neighbourhood information stored at the vertex is used to find the appropriate quadtree or triangle mesh. We climb up the hierarchy until the parent maintaining the appropriate information is found.

3.3 Constructing a true hierarchy from the discontinuity mesh

For each receiver containing a penumbral or umbral zone, after the discontinuity meshing and triangulation steps, we have a set of triangles corresponding to this partially lit or occluded region. To construct a hierarchy we attach these triangles at appropriate levels of a standard quadtree, such that in unoccluded regions illumination is represented with the regular quadtree structure.

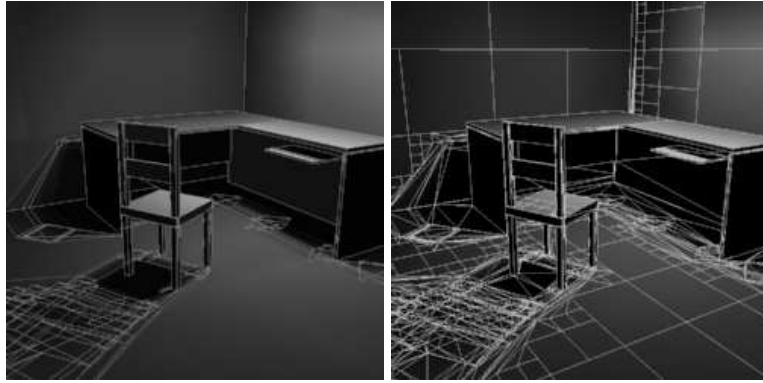


Fig. 1. (a) Discontinuity mesh and (b) HR/DM mesh for a test scene

Quadtree Subdivision We start by recursively subdividing the receiver using a standard quadtree. If a child of the quadtree contains no partially lit or occluded region, initial subdivision (i.e. the subdivision performed before BF-refinement) terminates. The unoccluded quadtree leaf elements are also inserted into a temporary face-edge-vertex data structure mesh. If on the other hand a child contains part of the penumbra or umbra, subdivision continues until a predefined maximal depth is reached.

When no more subdivision is possible, the penumbral/lit boundary edges are inserted into mesh. When this process is complete, mesh contains a set of faces corresponding to the completely unoccluded quadtree leaves, and a (usually highly irregular) face between the leaves and the penumbra/lit boundary. This face is triangulated, and the resulting triangles are then added to the list of penumbral/umbral triangles.

The final step requires “clustering” of these triangles (both original penumbra triangles from DM as well as the new triangles in lit regions) from mesh so that they can be correctly attached to an appropriate level of the quadtree.

“Clustering” for Penumbral and boundary regions To perform the clustering step, a 3D clustering bottom-up construction ([15]) is adapted to 2D. A multi-level grid is constructed, such that the smallest grid cell has the size of a maximal depth quadtree leaf. Each level of the grid is visited, and triangles entirely contained in a cell at a given level of the grid are attached to the corresponding quadtree inner node (if it exists). The triangles contained in a given cell at a given level are grouped to form an internal node. This node is then inserted at the appropriate level higher in the multi-level grid, if it is contained in a cell at that level. The contents of this grid cell will in turn be attached to the appropriate level of the quadtree.

In this manner we have a mixed hierarchy which starts at the root as a normal quadtree, and has children which may be regular quadtree subdivisions, or agglomerations of triangles or individual triangles (in the case of elongated triangles which can occur in the context of discontinuity meshing). An example is shown in Figure 1(b).

It must be noted that due to the bottom-up construction, we have the ability to insert the *entire* DM into the hierarchy (as is done here). In many cases simplification should probably be performed, but the generality of the method permits maximal flexibility.

4 Impact of algorithmic choices on solution and image quality

In this section we revisit the different algorithmic components of HR mentioned in the introduction, and discuss their relevance. This discussion serves both as a first attempt to investigate the influence these factors have on the solution as well as on each other, and as motivation for the experimental approach developed in the following section.

4.1 Meshing strategy

The use of quadtrees has many advantages: the structure is simple to handle and manipulate, it allows implicit neighbour finding operations, and provides well shaped elements which is important in the context of any numerical approximation ([13]). Interpolation and extrapolation operations are also readily performed in these structures since elements respect regular ratios. Nonetheless, the very regularity of the quadtree structure hides its inadequacy in representing high-frequency irregular information such as shadow boundaries.

Discontinuity meshing provides an appropriate solution to the problems of visual representation of shadow boundaries. The problems with such meshes are however numerous. Other than the issues related to numerical accuracy in construction [19], these meshes tend to contain far too many elements ([4]), and they result in badly formed triangles which pose problems for interpolation/extrapolation operations as well as being formally unadaptable to finite element approaches ([13]). It is difficult to determine a priori when the use of such meshes is advisable in the context of HR.

4.2 Visibility calculation

In traditional HR approaches, visibility is computed by ray-casting between two patches p and q . A form-factor disc approximation is then multiplied by the fraction of rays blocked, which is used as a visibility estimate. This approximation influences the form-factor estimation as well as refinement, and has repercussions which are difficult to isolate.

Given the mesh and backprojections, two important changes can be incorporated into the treatment of the direct illumination links: characterisation of links as partial, occluded or unoccluded (in the spirit of [19]) can be performed accurately immediately after the discontinuity meshing step and the calculation of irradiance values at the vertices during the solution is exact. The estimate of area-to-area form-factors can also be significantly improved since each sample of the kernel function is calculated with the exact visible portion of the source.

4.3 Direct Illumination at Vertices for HR

In standard HR (using a piecewise constant approximation), irradiance is “pushed” down the hierarchy to the leaf nodes ([6, 14]). This irradiance is then converted to radiosity which is subsequently extrapolated to the vertices of the leaves and “pulled” up the hierarchy. If the backprojections are available, we can compute exact irradiance values due to light sources at all vertices very cheaply, since no visibility computation is required (points are either in a penumbral (or umbral) mesh face or in light). It is thus only natural to skip the “push” step for the light source, and simply evaluate the exact irradiance at the vertices of the mesh. These values are then averaged, resulting in a radiosity value assigned to the leaf, and then “pulled” in the normal manner up the hierarchy. We note that this direct shading should only be performed at the vertices originally in the discontinuity mesh, or for vertices on hierarchy leaves with a link to the source (in the current implementation it is performed at all vertices).

This approach has a double advantage of producing visually accurate results (see Figure 5(iv) in Colour Section) while simultaneously providing a highly accurate, hierarchical representation of direct illumination, which will, hopefully, result in an overall higher quality global illumination simulation.

4.4 Refinement criteria

In traditional HR approaches, “BF” refinement has been used, which essentially requires a link between two surface elements to be refined if their mutual form-factor multiplied by the power on the link is larger than a threshold [8]. The philosophy of this approach is to reduce the respective size of the elements on the two sides of a link, thus reducing visibility error since all interactions tend to be either occluded or visible ([19]), and reducing “integration error” of the area form-factor integral since the kernel varies. Another approach involves a “smoothness” criterion for the kernel used in Wavelet-based HR ([7]).

When exact visibility is available, neither of these criteria is entirely satisfactory. Since we can cheaply determine the visible part of the source, we could apply a (point-to-area) form-factor variation criterion (this is similar in spirit to the “smoothness” criterion). Subdivision will be thus better adapted to the variation of irradiance on the receiver, since the source is never subdivided.

4.5 Convergence

The issue of convergence is rarely addressed in HR research. Following the original definition and structure of the algorithm [8], we define convergence as a sequence of iterations involving a Refine step followed by an (optional) loop of Gather-Push/Pull operations until the result is no longer modified. The iteration terminates when no new links are created (“convergence”).

One of the issues we wish to investigate is the effect of meshing strategy and visibility on the rate of convergence. Intuitively it seems that a good representation of direct illumination and accurate visibility computations should improve the convergence rate.

5 Comparison of some strategies and initial experimental results

The new algorithm presented in Section 3 provides an environment that allows a first investigation of the relative effect of meshing and visibility error on image and solution quality.

5.1 Experimental configurations

Four different algorithmic configurations are considered: (i) HR, using regular quadtree subdivision and ray-cast visibility calculation (QT/RT), (ii) HR regular quadtree subdivision and using backprojections (QT/BP), (iii) HR with full discontinuity meshing and ray-cast visibility (DM/RT) and finally (iv) HR with full discontinuity meshing and backprojections (DM/BP).

Comparing (i) and (ii) quantifies the effect of visibility error in form-factor computation, and the resulting effect on the solution. Comparing (i) and (iii) demonstrates the importance of the use of the discontinuity mesh as a basis for the subdivision in HR. Finally, the combined effect of the mesh and accurate visibility becomes evident by comparing configuration (iv) to the others.

Two test environments “Desk+Chair” (Fig. 1) and “Books” (Figure 5 in the colour section) with specific points of view have been chosen. The scenes are lit by large light sources giving rise to large regions of penumbra (which favours the use of backprojections). In addition, the “Books” scene contains many regions of small fine shadow, for which discontinuity meshing is advantageous. There are 133 polygons (268 distinct edges) in “Desk+Chair” and 241 polygons (484 distinct edges) in “Books”.

A reference solution is computed for both scenes, using a standard quadtree subdivided very finely, and run to convergence with a very small tolerance value. We compute an L_1 error on the pixel RGB values.

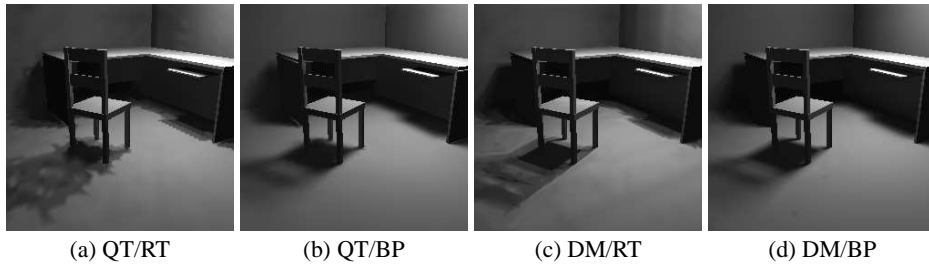


Fig. 2. “Desk+Chair” images

5.2 Quality evaluation and test suites

For each run we present the total computation time t_t to convergence, the final number of leaves at convergence l_c and the L_1 image error e after convergence for the given tolerance. We also report (where applicable) DM construction time t_{dm} ((ii)-(iv)) triangle clustering time t_c ((iii) and (iv)) and the number l_{dm} of leaf elements after the mixed hierarchy construction but before subdivision ((iii) and (iv)). All reported timings are on an SGI R4400 Indigo 2 at 150 Mhz.

For the two scenes, we have attempted to maintain approximately the same number of elements, to provide a “fair” comparison. This requires the judicious choice of parameters $BF-\epsilon$, minimum area size and the visibility factor as defined in [8]. The resulting images for “Books” are shown in colour in Figure 5, while small versions of “Desk+Chair” are shown in Figure 2. Meshes for “Desks+Chair” are shown for illustration in Fig. 4 for cases QT/BP and DM/BP. The numerical results are summarised in Table 1.

Solution	t_t (s)	t_{dm} (s)	t_c (s)	l_{dm}	l_c	e		t_t (s)	t_{dm} (s)	t_c (s)	l_{dm}	l_c	e
(i) QT/RT	413.6	-	-	-	3911	6.7		838.1	-	-	-	8759	9.2
(ii) QT/BP	531.3	71.4	-	-	3650	2.8		1076.6	394.5	-	-	8168	4.2
(iii) DM/RT	604.9	71.4	98.5	3365	4502	6.0		1332.8	394.5	95.0	5929	8320	8.2
(iv) DM/BP	310.1	71.4	98.5	3244	3982	3.0		782.7	394.5	95.0	5775	6969	3.7

Table 1. Test results: (left) “Desk+Chair” and (right) “Books”

5.3 Results of Experimental Study

The test results presented above are by no means definitive or complete. The nature of experimental work is such that it is difficult to come to concrete conclusions from a set of given tests. Nonetheless, we believe that the results presented provide interesting insight into the problems related to visibility and meshing in the context of HR.

For both scenes we see that DM/BP provides the most computationally efficient solution, despite the overhead of mesh creation and hierarchy construction. The image quality is always better (Fig. 2, and Colour Fig. 5), and numerical accuracy is either better or on a par with all available alternatives. In the case of fine shadow features, the discontinuity mesh is particularly advantageous.

Visibility Visibility accuracy is of predominant importance. Solution (ii) QT/BP is numerically the most accurate for “Desk+Chair”, but DM/BP is visually superior (see Fig. 2, 5). DM/BP is however more accurate numerically for “Books”. The use of backprojections enhances numerical and visual quality more than the use of DM alone. The visual quality of QT/BP can be very high, as is the case for “Desk+Chair”.

Convergence In Fig. 3 we compare the L_1 image error at each iteration for the two scenes. The accuracy of the visibility computation appears to directly influence convergence. Solution (ii) has the best behaviour for “Desk+Chair”, but the DM is more important for “Books”, for which the DM/BP solution has the lowest error. In the case where QT/BP is numerically better, the difference is insignificant.

Meshing Discontinuity meshing without exact visibility results in visual artifacts, and is computationally expensive. This is particularly evident for “Desk+Chair” (Fig. 2). Previous algorithms avoided this problem by using “final gather” type approaches [12]

Nonetheless, the irregular meshes produced add a high overhead in the global solution, simply by the sheer number of leaf elements at the outset (Table 1) (a similar observation was made by Lischinski et al. [12]). An obvious remedy is to investigate the use of simplification techniques for meshing (e.g., [2, 4]), while maintaining the original backprojection information for visibility computations.

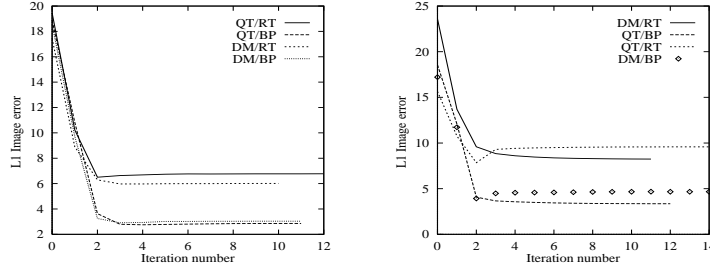


Fig. 3. Convergence for different approaches (a) “Desk+Chair” (b) “Books”

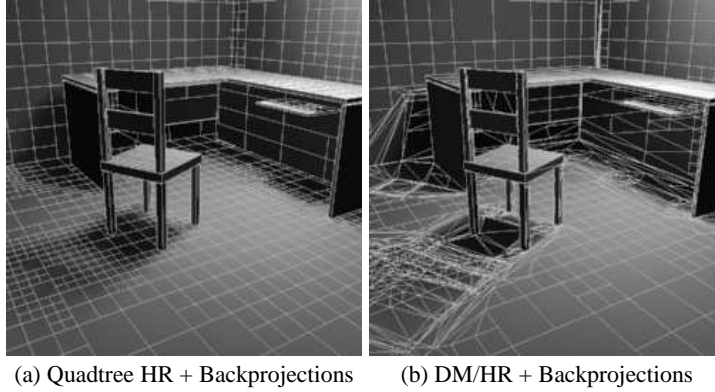


Fig. 4. Test suite mesh images

6 Conclusions

We have presented a first approach to investigating sources of error due to visibility computation and meshing strategies. A list of important factors affecting HR computations was presented and discussed. To facilitate experimental investigation we introduced a new hierarchical radiosity algorithm which incorporates backprojections (and thus exact visibility with respect to the source) and discontinuity meshes.

This approach has permitted the comparison of standard quadtree-based HR using traditional point-sampling for visibility with the case where visibility is calculated with backprojections, HR discontinuity mesh with point sampling and finally HR discontinuity meshing with backprojections.

A number of interesting observations were made from an experimental study relating the different effects of the use of analytic visibility (backprojections) and discontinuity meshes for HR light transport. Overall, it was observed that visibility accuracy is much more important than the use of meshing. Nonetheless, DM with BPs adds to overall visual quality. Many more experimental tests are required to confirm the observations made here as well as to investigate other aspects of the solution process.

Numerical difficulties and robustness problems are inherent in all discontinuity meshing approaches. A comprehensive solution to this problem is being pursued for scenes of moderate complexity. A algorithm to simplify discontinuity meshes for the mixed hierarchy is also currently being investigated.

Acknowledgements The first author thanks Xavier Pueyo for initial discussions on the subject.

References

1. James Arvo, Kenneth Torrance, and Brian Smits. A framework for the analysis of error in global illumination algorithms. In *Computer Graphics Proceedings, Annual Conference Series: SIGGRAPH '94* (Orlando, FL), pages 75–84. ACM, July 1994.
2. George Drettakis. Simplifying the representation of radiance from multiple emitters. In *Proceedings of 5th EG Workshop on Rendering*, Darmstadt, Germany, June 1994.
3. George Drettakis and Eugene Fiume. A fast shadow algorithm for area light sources using back projection. In *Computer Graphics Proceedings, Annual Conference Series: SIGGRAPH '94* (Orlando, FL), pages 223–230. ACM SIGGRAPH, New York, July 1994.
4. George Drettakis and Eugene Fiume. Structured penumbral irradiance computation. 1996. Submitted for publication.
5. Neil Gatenby and W. T. Hewitt. Optimizing discontinuity meshing radiosity. In *Fifth Eurographics Workshop on Rendering*, pages 249–258, Darmstadt, Germany, June 1994.
6. Reid Gershbein, Peter Schröder, and Pat Hanrahan. Textures and radiosity: Controlling emission and reflection with texture maps. In *Computer Graphics Proceedings, Annual Conference Series: SIGGRAPH '94* (Orlando, FL), pages 51–58. ACM, July 1994.
7. Steven J. Gortler, Peter Schröder, Michael F. Cohen, and Pat Hanrahan. Wavelet radiosity. In *Computer Graphics Proceedings, Annual Conference Series: SIGGRAPH '93* (Anaheim, CA, USA), pages 221–230. ACM SIGGRAPH, New York, August 1993.
8. Pat Hanrahan, David Saltzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. *Computer Graphics*, 25(4):197–206, August 1991. SIGGRAPH '91 Las Vegas.
9. Paul Heckbert. Discontinuity meshing for radiosity. *Third Eurographics Workshop on Rendering*, pages 203–226, May 1992.
10. A. T. Campbell III and Donald S. Fussell. An analytic approach to illumination with area light sources. Technical Report TR-91-25, CS Dpt. U of Texas, Austin, TX, August 1991.
11. Dani Lischinski, Brian Smits, and Donald P. Greenberg. Bounds and error estimates for radiosity. In *Computer Graphics Proceedings, Annual Conference Series: SIGGRAPH '94* (Orlando, FL), pages 67–74. ACM, July 1994.
12. Dani Lischinski, Filippo Tampieri, and Donald P. Greenberg. Combining hierarchical radiosity and discontinuity meshing. In *Computer Graphics Proceedings, Annual Conference Series: SIGGRAPH '93* (Anaheim, CA, USA), pages 199–208. ACM, August 1993.
13. P. M. Prenter. *Splines and Variational Methods*. John Wiley & Sons Inc, New York, 1989.
14. François Sillion. A unified hierarchical algorithm for global illumination with scattering volumes and object clusters. *IEEE Trans. on Vis. and Comp. Graphics*, 1(3), September 1995.

15. François Sillion and George Drettakis. Feature-based control of visibility error: A multiresolution cluster algorithm for global illumination. In *Computer Graphics Proceedings, Annual Conference Series: SIGGRAPH '95* (Los Angeles, CA), pages 145–152. ACM, August 1995.
16. A. James Stewart and Sherif Ghali. Fast computation of shadow boundaries using spatial coherence and backprojections. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 231–238. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.
17. Filippo Tampieri. *Discontinuity Meshing for Radiosity Image Synthesis*. PhD thesis, Department of Computer Science, Cornell University, Ithaca, New York, 1993. PhD Thesis.
18. Seth J. Teller. Computing the antipenumbra of an area light. *Computer Graphics*, 26(4):139–148, July 1992. Proceedings of SIGGRAPH '92 in Chicago (USA).
19. Seth J. Teller and Patrick M. Hanrahan. Global visibility algorithms for illumination computations. In *Computer Graphics Proceedings, Annual Conference Series: SIGGRAPH '93* (Anaheim, CA, USA), pages 239–246. ACM, August 1993.
20. John R. Wallace, Kells A. Elmquist, and Eric A. Haines. A ray tracing algorithm for progressive radiosity. *Computer Graphics*, 23(3):315–324, July 1989. Proceedings SIGGRAPH '89 in Boston.

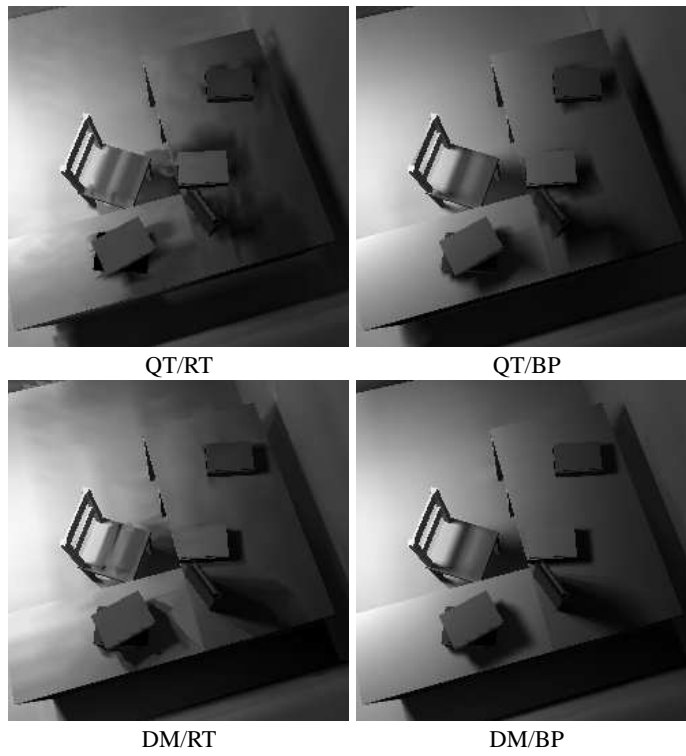


Fig. 5. Results of the four selected combinations for the “Books” image.

2.4.4 Interactive High-Quality Soft Shadows in Scenes with Moving Objects (EG'97)

Auteurs : Céline Loscos et George Drettakis

Actes : Congrès Eurographics '97

Date : septembre 1997

Interactive High-Quality Soft Shadows in Scenes with Moving Objects

Céline Loscos and George Drettakis[†]

iMAGIS[‡]/GRAVIR-INRIA,
BP 53, F-38041 Grenoble Cedex 9, FRANCE.

Abstract

Interactive rendering of soft shadows (or penumbra) in scenes with moving objects is a challenging problem. High quality walkthrough rendering of static scenes with penumbra can be achieved using pre-calculated discontinuity meshes, which provide a triangulation well adapted to penumbral boundaries, and backprojections which provide exact illumination computation at vertices very efficiently. However, recomputation of the complete mesh and backprojection structures at each frame is prohibitively expensive in environments with changing geometry. This recomputation would in any case be wasteful: only a limited part of these structures actually needs to be recalculated. We present a novel algorithm which uses spatial coherence of movement as well as the rich visibility information existing in the discontinuity mesh to avoid unnecessary recomputation after object motion. In particular we isolate all modifications required for the update of the discontinuity mesh by using an augmented spatial subdivision structure and we restrict intersections of discontinuity surfaces with the scene. In addition, we develop an algorithm which identifies visibility changes by exploiting information contained in the planar discontinuity mesh of each scene polygon, obviating the need for many expensive searches in 3D space. A full implementation of the algorithm is presented, which allows interactive updates of high-quality soft shadows for scenes of moderate complexity. The algorithm can also be directly applied to global illumination.

Keywords: Illumination, soft shadows, incremental update, discontinuity meshing, backprojection, dynamic scenes.

1. Introduction

High quality rendering for scenes lit by *area* light sources is an important component of any lighting system. Such display is typically performed using ray-casting to successfully render the soft shadows or *penumbra*¹. An alternative approach is the use of discontinuity meshing with backprojections. The *discontinuity mesh* provides an initial decomposition of the scene which is used to create a subdivision into simple polygons, whose edges are well adapted to the penumbra contours and the discontinuities of illumination in the interior of partially shaded regions^{2,3}. The computation of the full discontinuity mesh (capturing all illumination discontinuities due to the light source) permits the calculation

of *backprojections*^{2,4}. The backprojection structure encodes exact visibility of any point in the scene with respect to the light source, thus providing *exact* illumination (irradiance) values at the vertices of the subdivision and at any point in the penumbra. Very high quality rendering of soft shadows can be achieved in this manner, using a polygonal decomposition on a graphics hardware pipeline. We are therefore able to interactively visualise scenes with accurate soft shadows on graphics workstations as long as the objects in the scene do not move. If the geometry changes, existing algorithms require the complete recomputation of the discontinuity mesh and the backprojections, which is prohibitively expensive, and definitely precludes user interaction.

In this paper we present an algorithm which allows interactive rendering of high quality shadows for scenes where objects move, which we call *dynamic* scenes. Our new algorithm is based on discontinuity meshing and backpro-

[†] E-mail: {Celine.Loscos | George.Drettakis}@imag.fr

[‡] iMAGIS is a joint research project of CNRS/INRIA/UJF/INPG.

jections, thus providing accurate soft shadows for interactive display. To achieve interactive update rates for dynamic scenes, the algorithm exploits spatial coherence of the required modifications to the data structures related to shadows and the local nature of changes in the discontinuity mesh. This locality is encoded in the rich structure of the discontinuity mesh, which permits us to identify the visibility events by simply examining the planar discontinuity mesh on the polygons.

This novel algorithm is useful in several contexts. Since primary illumination is dominant in many situations, high quality direct lighting with soft shadows can be used as a standalone interactive visualisation program offering a much higher level of realism compared to traditional point-source interactive lighting systems. In addition the algorithm can be used as a first interactive design phase before a global illumination solution, for object placement and general modeling in a scene. Although we treat only direct illumination, this approach can be applied in the context of global illumination. Our method thus opens an interesting avenue of research for combined discontinuity meshing/hierarchical radiosity approaches such as those previous presented ^{5, 6}, in the context of dynamic scenes.

The strategy adopted to achieve interactive display of soft shadows with moving objects is based on two main components: (i) intelligent data structures which localise and thus accelerate access to changing visibility information and (ii) an efficient update algorithm which takes into account both spatial coherence and visibility information contained in the mesh. After presenting related previous work in Section 2, we present the data structures used in Section 3 and the incremental shadow update algorithm is described in Section 4. We next present the results of the implementation in Section 5. In Section 6 we discuss future work and conclude.

2. Previous work

2.1. Illumination in Dynamic Scenes

Most previous work in illumination for dynamic environments has concentrated on global solutions. Some research has been performed in ray-tracing (e.g., ⁷), which is specifically related to the view-dependent nature of ray-tracing, and is thus unsuitable for rendering approaches based on interactive visualisation using current graphics hardware.

The output of radiosity algorithms was used very early on with graphics hardware, permitting realistic interactive walkthroughs albeit with the restriction to static environments. The first attempt to remove this restriction was the approach of Baum ⁸, in which motion was predetermined and the region of space affected was preprocessed to accelerate the calculation of form-factors for each frame.

More involved approaches, based on the progressive refinement radiosity algorithm were presented by George et

al. ⁹, and also Chen et al. ¹⁰. In these approaches moving shadows were treated by re-shooting energy to remove them from their previous positions, shooting negative energy to reinstate them elsewhere. Modifications in the environment had to be ordered by a queue due to the nature of progressive refinement. Special attention was paid to efficiently treating shadows due to direct illumination. A more involved data structure for maintaining shadow form-factor lists has been presented ¹¹ for progressive refinement radiosity.

A first approach for hierarchical radiosity has been presented ¹². A similar approach was presented by Shaw ¹³. In this work, a “motion volume” was used to identify the links affected by the displacement of an object.

2.2. Discontinuity Meshing for High-Quality Illumination

For polygonal scenes lit by area sources, discontinuity meshing ^{14, 15, 2, 4}, was introduced to improve the quality of rendering for scenes containing soft shadows. To create the discontinuity mesh with respect to a source, *discontinuity surfaces* are cast into the environment. These surfaces are the interaction of an edge and a vertex (*EV* surface) or three edges (*EEE* surface ^{16, 17}). The reader unfamiliar with discontinuity surfaces is strongly encouraged to refer to the appropriate references ^{16, 2, 4}. Algorithms which treat all such events ^{16, 2, 4}, can then incrementally compute the *backprojection* data structure, which encodes all visibility information with respect to the source.

A first approach for dynamic environments rendering using discontinuity meshing with BSP trees was developed for point light sources by Chrysanthou and Slater ¹⁸.

Worral et al. have presented a new approach for area sources ¹⁹. In their method, illumination is computed on a triangulated discontinuity mesh in the context of a progressive-refinement radiosity method. The discontinuity mesh vertices are updated by taking into account certain visibility changes. Triangular mesh coherence is maintained and radiance values are updated for each triangle of the mesh by shooting the irradiance difference compared to the previous mesh. An interesting criterion is introduced, determining whether a change in visibility occurs in the mesh. This approach is limited to *EV* discontinuity surfaces, with vertex *V* on the source. Moreover, the focus of Worral et al.’s work is the update of the triangulation, whose cost is minimal compared to the casting of discontinuity surfaces, especially in complex environments. It is important to note that the approach presented in ¹⁹, computes an incomplete mesh, since *EEE* and other important discontinuity surfaces are ignored. As a consequence, backprojections cannot be computed. Visibility must thus either be approximated (e.g., by ray-casting), or be calculated by clipping the entire scene against the source, which is extremely expensive. Such visibility computation typically dominates the computation time ⁵.

In contrast, our new approach is totally different, since the complete discontinuity mesh and backprojections are incrementally updated at each frame. The exact visible part of the source can thus be determined very cheaply at any point in the penumbra, without a visibility calculation, since this information is encoded with the backprojections². Thus at every frame, we have exact (analytical) irradiance values for all the vertices in the mesh. Before presenting the complete algorithm, we describe important data structures used in the algorithm.

3. Data Structures for Efficient Update

In this section we present the data structures used to accelerate the update of shadows in dynamic scenes. In what follows we define as *static* the edges and vertices which belong to static objects, i.e. objects which do not move. The object that moves will be referred to as the *dynamic* object. We define as *dynamic* edges and vertices which belong to the dynamic object. As a consequence, we call *dynamic discontinuity surfaces*, the *EV* or *EEE* surfaces which are defined by at least one edge or vertex of the dynamic object, and *static discontinuity surfaces* those which are defined entirely by static edges or vertices. Finally, a *mesh* edge or *mesh* vertex, is a two-dimensional edge or vertex which is part of the planar discontinuity mesh calculated on each scene polygon. We use a *winged-edge* data structure used to store the mesh and access it efficiently². The deletion of mesh edges can thus be performed locally and rapidly, as well as the incremental update of backprojections.

The three data structures used to localise and thus accelerate access to information which modifies visibility and thus shadow calculations at each frame, are the following: (a) discontinuity surface storage in the spatial subdivision structure, (b) the motion volume and (c) intersection lists for modified discontinuity surfaces.

3.1. Storage of the Discontinuity Surfaces in the Spatial Subdivision

The scene is decomposed into a regular grid²⁰, used for efficient casting of discontinuity surfaces². Each voxel contains the list of polygons that cut it. In addition to this we add the list of discontinuity surfaces which intersect the voxel. This list is created on-the-fly, during the propagation of discontinuity surfaces. An example of this list is shown in Fig. 1.

The lists of discontinuity surfaces associated with each voxel allow the rapid identification of all visibility events affecting an area of space, by simply traversing the corresponding voxels. As a consequence, we can perform efficient incremental updates in the region of a moving object.

The storage overhead of the lists is small (between 65 and 300 Kb) for the test scenes presented in the results (see Section 5), which use a moderately-sized grid (15x15x15).

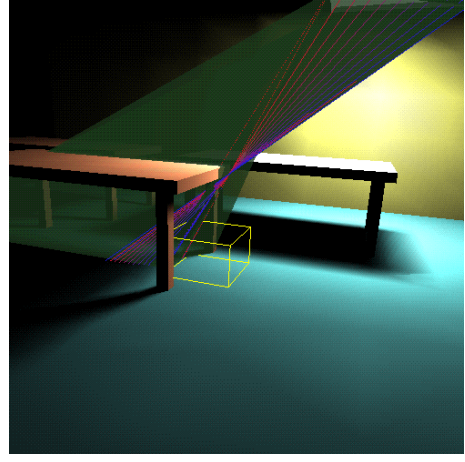


Figure 1: Discontinuity surfaces in a voxel (see also colour section).

3.2. Construction of a Motion Volume

The region of space for which visibility is affected by the motion of an object is entirely limited by the *maximal* (i.e. delimiting the frontier between light and penumbra) edge-vertex (*EV*) discontinuity surfaces defined by the light source and the polyhedron of the moving object for the initial and the final position. In addition there is no change in visibility in the region of space between the dynamic object and the light. As an illustration see Fig. 2, where the maximal discontinuity surfaces are shown as the dark grey surfaces, containing all interior discontinuity surfaces, such as that shaded in light grey.

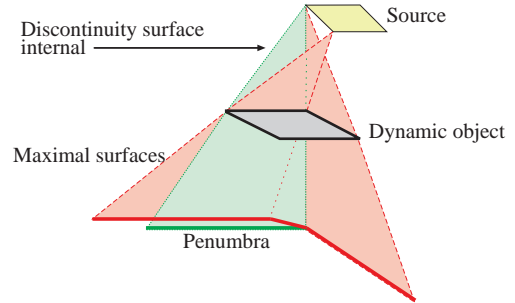


Figure 2: The maximal surfaces are shown in dark grey and the interior surfaces in light grey. Notice that the maximal surfaces encompass all the others.

Given this property, we can define a simplified approximation to the exact volume in space affected by the motion which we call a *motion volume*. This volume is delimited by a plane parallel to the source above the uppermost side (i.e. closest to the source) of the dynamic object, a plane

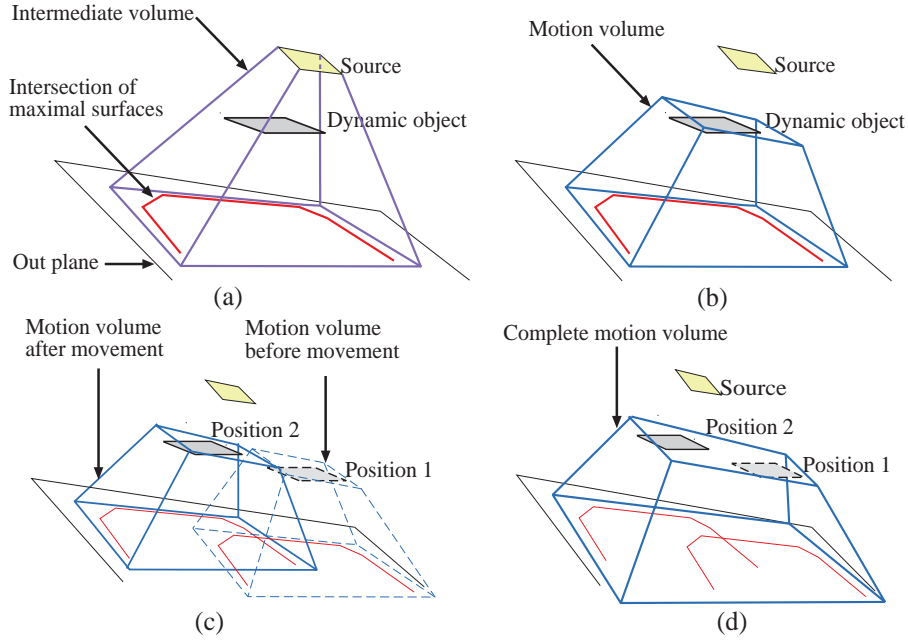


Figure 3: Motion Volume construction: (a) The maximal surfaces of the dynamic object are intersected with the outplane: a plane parallel to the source, and tangent to the bounding volume of the scene. The 2D bounding box of the contour of the maximal surfaces (dark grey segment on the outplane) is found. A four-plane volume is then constructed with the 2D bounding box of the source. (b) The volume is cut by a plane parallel to the source above the object, (c) Volumes for position before and after the move (d) Complete motion volume.

parallel to the source plane which is completely outside the scene, and four planes surrounding the maximal surfaces (see Fig. 3(a)-(b)).

In our current implementation three volumes are created. One for the first position of the object, one for the final position, and one that is the bounding volume of the two previous volumes (see Fig. 3(c)-(d)). Since we consider a small discrete motion at each frame, we currently use the bounding volume as the motion volume for updates. For larger displacements, the use of the two independent volumes would be more appropriate since their intersections would be small or inexistant. Otherwise the bounding volume would include too much unchanged space.

Note that this construction does not limit the dynamic object motion in any way. At any frame, the previous and current positions are available, and thus the user may interact freely with the dynamic object. Given the construction of the bounding volume, this motion can be of any type (translation, rotation), a scale operation, or a discrete curved trajectory.

3.3. Storage of Intersection Information with the Discontinuity Surfaces

The casting time for the discontinuity surfaces is mainly concentrated in the testing and intersection parts of the casting operation. Due to the richness of information already in the mesh, we can avoid a large part of this cost by storing some additional information with the discontinuity surfaces.

Consider the case of movement shown in Fig. 4(a) and (b), corresponding respectively to the position of the dynamic object before and after the move. We know that the only possible change in visibility for surfaces such as those shown in Fig. 4 can be caused by the dynamic object. As a consequence we do not need to search or intersect the discontinuity surface with any other object in the environment at each frame. If the dynamic object were moving away from the discontinuity surface after Fig. 4(a) it is evident that we would not need to recompute the intersection of the discontinuity surface with the environment, nor recompute the visibility on the surface.

The intersections of the polygons with the discontinuity surface are stored *before* visibility processing. An example is shown in Fig. 5(a). Notice that after visibility processing, which occurs as a 2D operation in the plane of the discon-

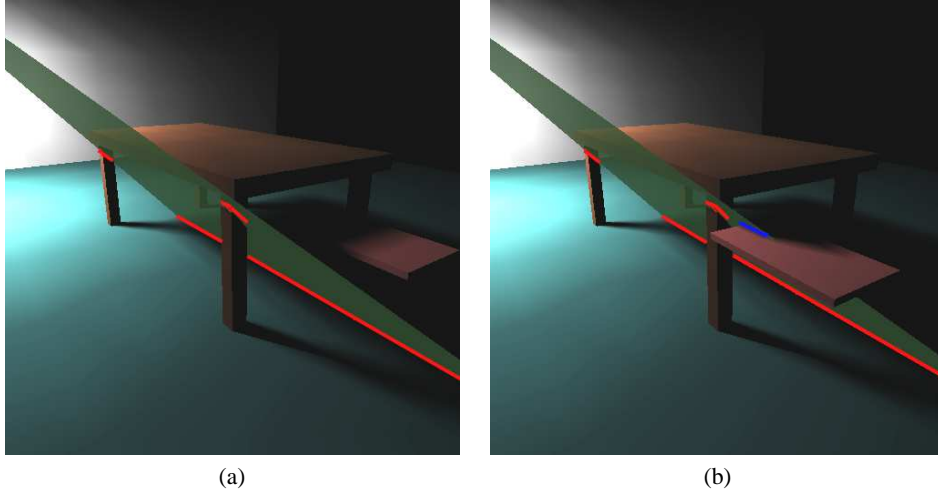


Figure 4: Dynamic object motion (a) the dynamic object (floating parallelepiped) does not cut the static EV discontinuity surface, (b) the object moves forward and cuts the discontinuity surface.

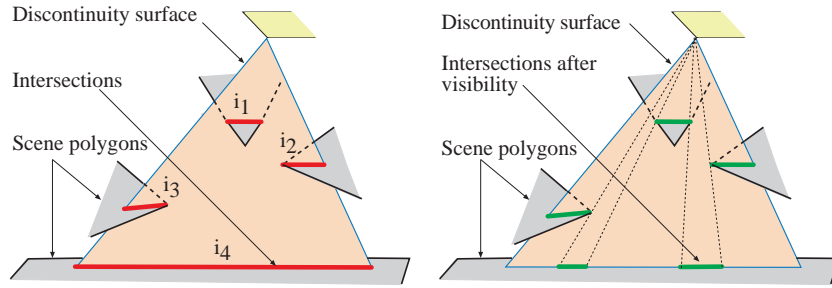


Figure 5: Intersection information storage (a) the intersections i_1, i_2, i_3, i_4 (in dark grey) are stored with the discontinuity surface before visibility computation. (b) the actual intersections (in light grey) after the visibility computation performed in the plane of the discontinuity surface.

tinuity surface (or 2-D parametric space for EEE), the intersections are changed, resulting in the final mesh edges inserted in the discontinuity mesh (e.g., two mesh edges for the floor - see Fig. 5(b)).

This list is stored with the discontinuity surface. For example the list i_1, i_2, i_3, i_4 in Fig. 5(a) is stored with the EV surface shown. When treating a static discontinuity surface at a given frame, we only perform a new intersection with the dynamic object. We thus avoid the cost of searching for and performing intersections with all the other (static) objects in the scene.

3.4. Input Scenes

As shall be seen later, we will be identifying visibility changes based on information in the mesh (see Section 4.3). In order to find all visibility changes, input scenes need to be closed environments. This ensures that all discontinuity surfaces have intersections with at least one scene polygon at any time. This guarantees that all the information required can be found in the mesh.

Considering only such scenes is not a strong restriction, since open environments can easily be changed by enclosing the scene in a box.

In addition, we suppose that the area source cannot move since the entire mesh would have to be updated. Techniques such as the Visibility Skeleton²¹ are probably more appropriate for this type of update, and will undoubtedly lead to efficient discontinuity mesh and backprojection algorithms for moving sources (see also Section 6).

4. Update Algorithm

Given the storage of discontinuity surfaces in the spatial subdivision structure, the creation of the motion volume and the storage of intersections with the discontinuity surfaces, we can now present the machinery required to perform efficient updates of the discontinuity mesh and backprojections.

The shadow update algorithm needs to perform the following steps: (a) identify the volume of space modified, and collect related discontinuity surfaces which need to be updated (function *findChangedSpaceAndDS*); (b) identify and process the region modified on each input polygon; (c) identify the visibility changes for each modified discontinuity surface (function *findAndProcessVisibilityChanges*); (d) cleanup the parts of the mesh which are invalid within each region; (e) update the mesh, and finally (f) update the shadows and the illumination.

In this manner we will have performed the necessary updates in the parts of the discontinuity mesh affected, and thus the soft shadows will correspond to the new position of the object. Both spatial coherence using the motion volume, and the information in the mesh are used to identify potential changes in visibility. Note that after these updates, the discontinuity mesh and backprojections are entirely recomputed, and the values of irradiance in the penumbra correct. We examine each step of the algorithm in detail.

4.1. Identification of Affected Discontinuity Surfaces and Mesh Region

We first identify (using the grid) all discontinuity surfaces and polygons contained in the motion volume. The discontinuity surfaces concerned are inserted into a list DS_d for the dynamic surfaces, and DS_s for the static surfaces. The intersection R_{2d} of the volume with each polygon is then computed, as shown in Fig. 6(a). The intersections R_{2d} of the volume with the polygons concerned are outlined in Fig. 6(b) in white. This two-dimensional polygon R_{2d} is in effect the modified region for each input polygon.

4.2. Processing of Mesh Edges in Modified Regions

Due to the winged-edge data structure used to store the mesh, we can efficiently identify the mesh edges which are modified. In particular, we find the mesh face containing a corner of R_{2d} and search all neighbouring faces recursively until no mesh edges crossing or contained in R_{2d} can be found.

```

processModifiedEdges() {
  foreach input polygon  $p$ 
    Poly2d  $R_{2d}$  = modified region of  $p$ 
    foreach mesh edge  $e$  in  $R_{2d}$ 
      if  $e$  is dynamic
        add  $e$  to dynEdgeDelList
      else if shouldDeleteStatic(  $e$  )
        add  $e$  to statEdgeDelList
}

```

Figure 7: Modified Mesh Edge Processing

For each mesh edge we identify those which need to be deleted. All dynamic mesh edges will be removed, as well as the static mesh edges for which a change in visibility occurs, with respect to the dynamic object. More precisely *shouldDeleteStatic*(e) is true only if the discontinuity surface associated to the edge e intersects the dynamic object at its initial or its final position. The corresponding static discontinuity surfaces are marked as changed. This process is summarised in Fig. 7, and detailed in what follows.

After processing the edges in the modified regions, we have two lists *dynEdgeDelList* and *statEdgeDelList* which are the mesh edges to be removed when the information they contain is no longer needed.

4.3. Finding and Processing the Visibility Changes in the Modified Regions

Recall that the routine *findChangedSpaceAndDS*() returns two lists which give us all the discontinuity surfaces passing through the motion volume: DS_s for the static discontinuity surfaces and DS_d containing the dynamic discontinuity surfaces.

For each surface, we identify the related visibility changes and perform the appropriate updates required to reflect the dynamic object motion. The process is summarised in Fig. 8.

4.3.1. Static Discontinuity Surfaces

For each static discontinuity surface which is on the list DS_s and has been marked changed, we compute new intersections with the polygons of the dynamic object, if such intersections exist. Note that a static discontinuity surface may intersect the dynamic object in its upper part, between an object vertex and the source edge, resulting in no mesh edges because of the object occlusion. Therefore a discontinuity surface may interact with the dynamic object without being detected by the previous mesh traversal. The use of the DS_s list is thus very important because it avoids the cost of an object-space search. With this list, we are able to consider such surfaces.

We then modify the intersection list of the discontinuity surface by either adding, deleting or modifying the information encapsulating the intersections of the surface with dynamic object.

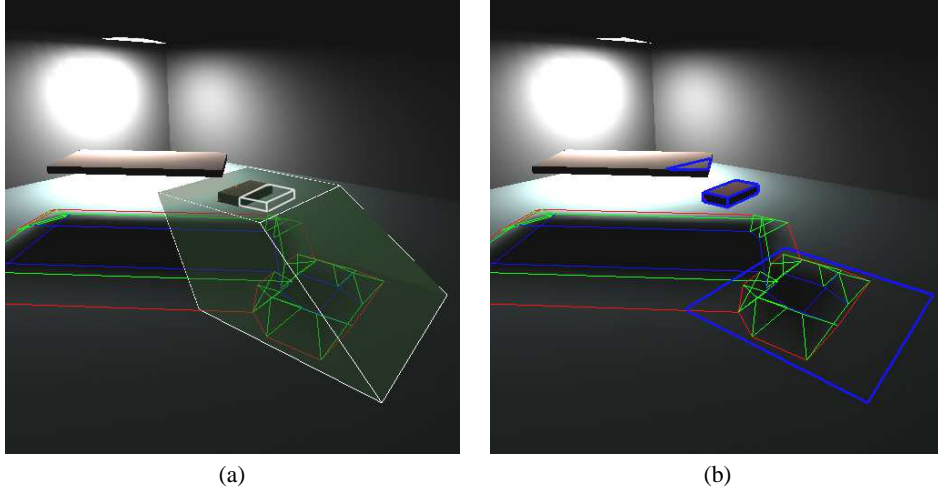


Figure 6: (a) Intersection of motion volume with polygons (see also colour section), (b) modified regions R_{2d} , (in white).

```

findAndProcessVisibilityChanges() {
  processStaticSurfaces( $DS_s$ )
  processDynamicSurfaces( $DS_d$ )
}
processDynamicSurfaces(list  $DS_d$ ) {
  foreach surface  $ds$  in  $DS_d$ 
    if  $ds$  is  $EV$ 
      processEV( $ds$ )
  ...
}

```

Figure 8: Finding and Processing Visibility Changes

4.3.2. Dynamic Discontinuity Surfaces

For dynamic discontinuity surfaces, we can easily see that their intersections with the scene polygons always change. In addition, the motion of the dynamic object can result in a change in the visibility configuration of each surface with respect to the static objects (new intersections, disappearance of intersections etc.). Much relevant information is contained in the mesh, and most notably is related to the static mesh edges. We thus avoid the cost of the search of intersections for each dynamic EV surface, involving an expensive traversal of many objects in the scene.

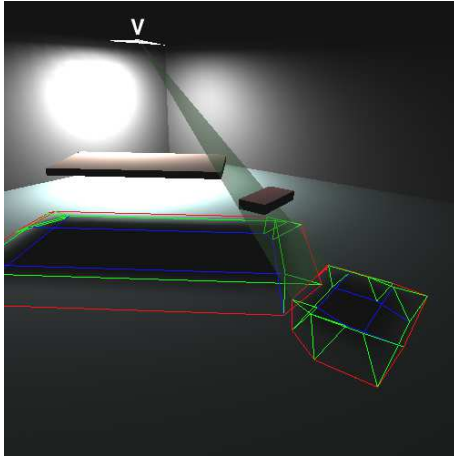
The treatment of EV surfaces was inspired by Worrall et al.¹⁹ who analyze the intersection of two edges of a mesh and decide whether a change in visibility occurs. In their work, a change occurs if the two corresponding discontinuity surfaces of the mesh edges share the same source vertex. We extend this idea to all types of EV edges and present a novel solution for the case of dynamic EEE surfaces.

EV Surfaces: Consider the example given in Fig. 9: the dynamic object (the small object on the right), has a dynamic EV surface related to the source vertex V . Initially it does not cut the static (larger) object. When moving inwards, the

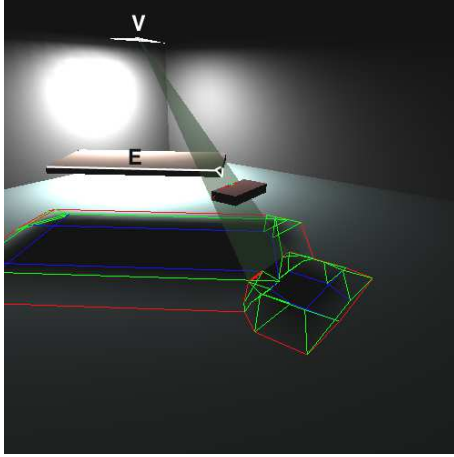
dynamic EV surface will intersect the corner of the static object. This can be detected of course in three-dimensions, but this would imply a costly search in space. Instead, we can directly identify this change in the discontinuity mesh. Consider the mesh edge e_d corresponding to the EV surface, shown in grey in Fig. 10(a). Due to the motion, the edge e_d will traverse the mesh vertex v (Fig. 10). The mesh vertex v is due the crossing of the mesh edges (in white), caused by two static EV discontinuity surfaces, due to the *same* source vertex V (Fig. 9(b)). Because of this traversal of a mesh edge generated by the same source vertex, we know that there is a visibility change concerning the dynamic EV surface, and that it is due to the static object in question.

To determine all such traversals, we need to perform a search in the mesh related to each dynamic discontinuity surface. For each dynamic discontinuity surface, we have stored the list of intersections with the polygons of the scene, for the *previous* position of the dynamic object. We will thus traverse this intersection list, and for each polygon which was intersected, we will find the region defined by the intersection points of the surface with the polygon, before and *after* the move. These correspond to the endpoints of e_d before (Fig. 9(a)) and after (Fig. 9(b)) the move. Within this region, we identify all static mesh edges. We again use the adjacency information of the winged-edge data structure to access these mesh edges rapidly. This is the reason why we do not remove any mesh edges before this step in the algorithm.

We then test to see if the conditions for a change in visibility are satisfied: that is whether the vertex V_s or the edge E_s of the corresponding static EV are the same as the edge E or vertex V of the dynamic discontinuity surface $evDs$. This process is summarised in Fig. 11 for the case of a EV_{src} surface (with vertex V on the source and edge E on the dynamic object). The $E_{src}V$ (edge on source, vertex on dynamic ob-



(a)



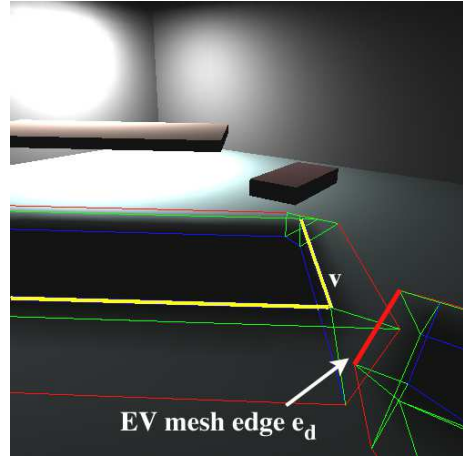
(b)

Figure 9: The small object is dynamic, moving towards the larger object. Dynamic EV discontinuity surface: (a) before the move there is no intersection with the static object, (b) after the move the dynamic surface intersects the static object.

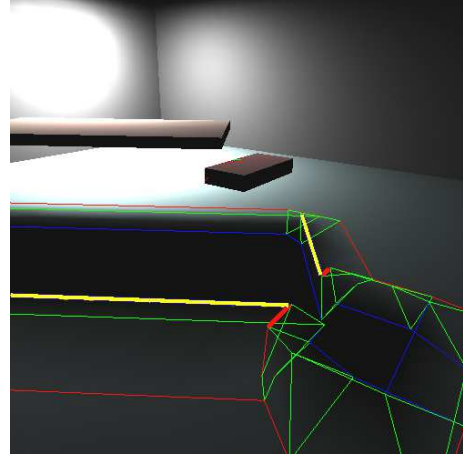
ject) case is treated similarly, by considering the equality of the generating edge E with E_s as well as the two vertices defining the edge.

If a visibility change is identified, the dynamic discontinuity surface is intersected with the corresponding static object, and its intersection list is updated. The same process could be applied to non-emitter EV surfaces.

EEE Surfaces. For EEE surfaces an algorithm which finds all visibility modifications from the mesh is much more involved, due to the complications implied by their curved nature. Nonetheless, we are capable of determining when a EEE surface will be created, maintained or destroyed, in particular for the case in which the discontinuity surface has



(a)



(b)

Figure 10: Dynamic discontinuity surfaces treatment: (a) the EV discontinuity surface of Fig. 9 results in edge e_d in the mesh. In (b) we see its new position. The modified search region for EV is defined by the two positions of e_d . Since the vertex v is crossed, a visibility change has occurred.

one edge on the source (this is the most common type of EEE surface). This allows us to avoid a costly search for EEE surfaces related to the source, which is otherwise required at each frame.

To understand this, consider the two EV surfaces in Fig. 12(a), created by a source polygon edge, a polygon edge on the dynamic object and a static polygon edge. When the dynamic object moves, the surfaces will intersect (Fig. 12(b)), and thus two EEE surfaces will be created. One such surface is shown in Fig. 12(c) with edges E of the source (adjacent to V), $E1$ and $E2$. The second EEE surface is built with $E1$, $E2$ and the second edge E' adjacent to V . These changes can be determined easily. When testing EV_{src} changes we check to see if the dynamic sur-

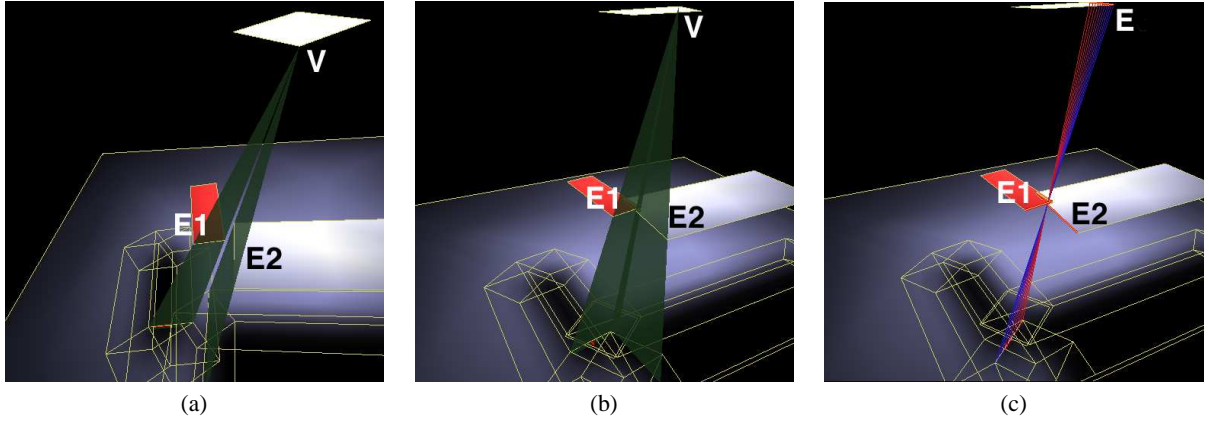


Figure 12: (a) Two EV surfaces which do not intersect (b) Intersection of the EV surfaces (c) One of the two EEE surfaces created. Figures replicated in the colour section.

```

processEV( DiscSurface evDs ) {
    updateDStoNewPosition( evDs )
    foreach intersection oldDsi of evDs
        newDsi = evDs →
            computeNewIntersection(dsi → polygon())
        Poly2d p2dds =
            findRegionAffected(oldDsi, newDsi )
        switch (evDs → type() )
            case : EVsrc
                findVisibilityChange(p2dds, evDs )
            ...
    }
    findVisibilityChangeEVsrc
    (Poly2d p2d, DiscSurface evDs ) {
        Edge E = evDs → edge(), Es
        Vertex V = evDs → vertex(), Vs
        foreach mesh edge em in p2d
            DiscSurface dss = em → getDiscSurface()
            Es = dss → edge()
            Vs = dss → vertex()
            if Vs == V {
                P = polygon containing edge Es
                updateIntersection(evDs, P)
                checkForEEE()
            }
    }
}

```

Figure 11: Finding Visibility Changes for Dynamic Discontinuity Surfaces

face crosses a static surface generated by the same vertex. The creation of the two EEE surfaces is performed by the routine *checkForEEE* (see Fig. 11).

4.4. Edge Cleanup, Mesh and Illumination Update

After processing all mesh edges in the modified region and identifying potential visibility changes we no longer need the mesh edges which will be modified. We thus traverse

the lists *dynEdgeDelList* and *statEdgeDelList* and remove the edges from the mesh within the modified region. The winged edge data-structure allows us to perform all removal-insertion operations efficiently and locally within the mesh. Adjacency information is accessed directly from the edge pointers stored in the aforementioned lists. After removal, we are ready to perform the visibility updates required for the static and dynamic surfaces which require them.

We first visit every modified surface of *DS_s* and *DS_d*, and perform a two-dimensional visibility operation on the discontinuity surface. This operation is a fast sweep algorithm which processes the intersection information stored with the discontinuity surface. Recall that this information always corresponds to the geometric state *before* visibility processing. This operation costs much less than a complete re-cast of a discontinuity surface which would involve a search in 3D and the re-intersection with the scene objects.

Once the visibility is performed, we insert the segments into the discontinuity mesh. These segments are thus correctly updated for occlusion.

We now have a discontinuity mesh which is completely up to date with respect to the new position of the dynamic object. We simply update the backprojection information in the faces which were modified. These faces were marked during the deletion and insertion of mesh edges. We incrementally traverse the faces changed and update the backprojections concerned. The same incremental algorithm as that presented in ^{17.2.4} is used. Since the number of modified faces is small, we can efficiently compute all the exact visibility information in the penumbra very efficiently.

The final step required is the update of the mesh vertex illumination values, which again is restricted to the mesh faces modified. This operation is again very efficient, since the backprojections compactly encode *complete* and *exact* visibility information. We thus rapidly compute exact irra-

dian values on the vertices in the penumbra which have changed.

It is important to note that the result of the update algorithm is not an approximation: at every frame, the solution is exact, and results in a mesh computed as if we were performing the entire re-computation of the discontinuity mesh and the backprojections.

5. Implementation and Results of the Update Algorithm

We have implemented the update algorithm and tested it on an Indigo2 R4400 running at 150MHz. The three test scenes are shown in Fig. 13. The scenes contain respectively 145, 307 and 475 polygons. The dynamic object is a box floating above the desk and its movement is given as four consecutive positions. We perform two different tests. The motion for Test 1 is shown in the sequence of Fig. 17, the motion for Test 2 in Fig. 18.

In Table 1 t_{TS} corresponds to the time which is required if the entire discontinuity mesh is to be recalculated at each frame. The time t_{TD} is the total time spent by our algorithm to update the mesh and the backprojections, as well as the illumination. All times are in seconds. The column s shows the speedup (ratio between t_{TS} and t_{TD}). The additional memory overhead for the storage of the intersection lists is on average 19 Kb for Scene 1, 46 Kb for Scene 2 and 60 Kb for Scene 3, in what concerns Test 1, and 17 Kb for Test 2.

As we can see, the update times are interactive for Test 1, between 1.2 seconds per frame for the simplest scene and (containing 145 polygons), to 2.5 seconds/frame for the most complex scene containing almost 500 polygons. In addition, notice that the additional memory required is small (less than 50Kb) for Scene 2. For Scene 3 (475 polygons) speedup can reach 90 times, compared to the recomputation of the complete mesh at each frame.

The localisation of the modified space has the benefit that the cost of the update algorithm does not depend heavily on the complexity of the rest of the scene. Notice that update times seem to grow sub-linearly with respect to scene complexity (number of polygons).

In Test 2 (performed only on Scene 1), a different movement of the dynamic object is performed, with greater interaction with the other objects (see Fig. 18). The visibility complexity is thus augmented by the number of static surfaces treated and the complication of the mesh. Notice that the update takes between 1.3 and 2.7 seconds. The additional cost is thus not overwhelming.

Our implementation is definitely unoptimised, and we thus believe that significantly improved update rates could be achieved by fine-tuning.

6. Summary, Discussion and Future work

The algorithm presented here provides accurate soft shadow updates for dynamic scenes at interactive rates. We first presented data structures that provide rapid access to relevant information, by exploiting spatial coherence. These structures permit local treatment of the visibility update. The inherent structure of the discontinuity mesh was then used to find and update local changes of visibility for both static and dynamic discontinuity surfaces. *EV* and *EEE* surfaces are treated in this manner. Finally, backprojections and lighting are efficiently updated exclusively in the parts of the mesh which have changed. Thus at each frame, analytic irradiance values are computed, resulting in high quality soft shadows.

Large scenes (more than several thousand polygons) cannot be directly treated with the implementation presented here. This is mainly due to problems of numerical precision. Numerical robustness problems occur during the computation of the intersections between objects and discontinuity surfaces and also during the calculation of the arrangement of the line segments forming the winged-edge data structure. Both problems can be addressed by adopting a symbolic computation approach based on *extremal stabbing lines* as described in the context of the Visibility Skeleton (VS) ²¹. All discontinuity surface/object intersection calculations can be replaced by the extremal stabbing lines, and the adjacency information available in the VS can be used to overcome the problems of the topological construction.

Other improvements of the our method should also be investigated. In particular, the use of a uniform grid, although simple to program, is definitely inefficient for more complex scenes. The use of a recursive grid or an octree type structure should provide interesting results.

Furthermore, depending on the movement of the object, we could also use individual volumes for the two positions and apply the algorithm presented for each. If the object moves only a little, we can use the bounding volume, whereas if the object moves a lot, resulting in negligible overlap between the initial and final volumes, it is probably better to use the two volumes.

More importantly, this paper opens a direction of research which will lead to an algorithm which limits the updates only to those strictly necessary. The incremental method presented for dynamic surfaces indicates a potential for such an approach. What is needed for this type of algorithm is an exhaustive classification of all events which occur in the discontinuity mesh in time, and the corresponding actions which must be taken. Optimality may thus be achieved by developing a “sweep” algorithm in time. Approaches similar to the Visibility Complex ²² or the more recent Visibility Skeleton ²¹ will prove useful in this direction.

Finally, the algorithm developed here should prove very useful in the context of mixed hierarchical radiosity/discontinuity meshing approaches ⁶. In particular, a

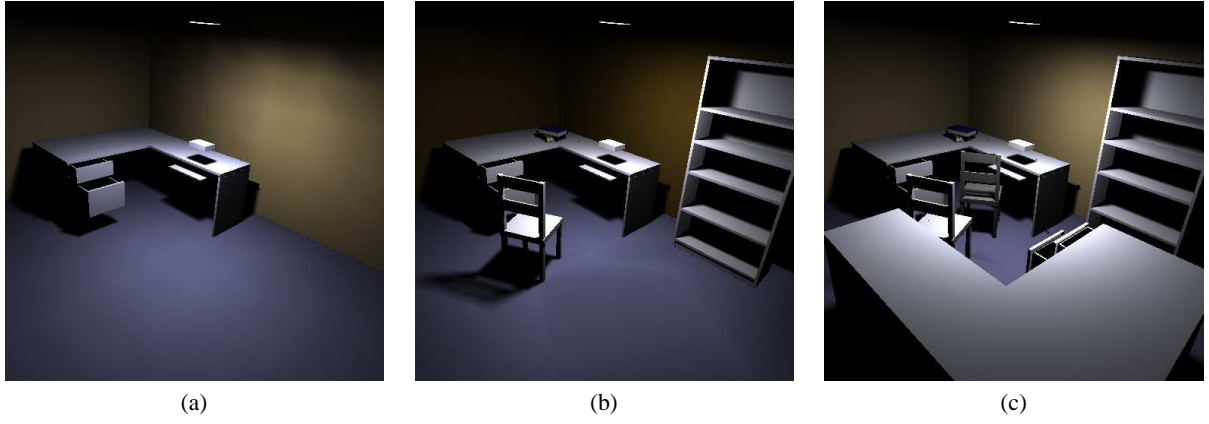


Figure 13: (a) Scene 1 (b) Scene 2 (c) Scene 3. Figures replicated in the colour section.

Pos.	t_{TS}	t_{TD}	s	t_{TS}	t_{TD}	s	t_{TS}	t_{TD}	s
Pos 1	57.67	1.36	42.4	122.33	1.79	68.3	222.51	2.33	89.4
Pos 2	57.86	1.25	46.3	123.91	1.81	68.5	225.75	2.35	79.7
Pos 3	58.13	1.24	46.9	125.16	1.91	65.5	227.60	2.48	84.3
Pos 4	58.55	1.27	46.1	125.97	1.99	63.3	230.22	2.46	85.3

Results for Test1: Scene 1 (145 polygons)				Scene 2 (307 polygons)			Scene 3 (475 polygons)		
Pos.	t_{TS}	t_{TD}	s						
Pos 1	58.74	1.29	45.5						
Pos 2	60.53	1.91	31.7						
Pos 3	61.24	2.23	27.5						
Pos 4	61.69	2.66	23.2						

Results for Test 2 (Scene 1)			
------------------------------	--	--	--

Table 1: Results for the Update Algorithm.

method similar to that described in ¹³ could be combined with our approach to achieve interactive updates for global illumination.

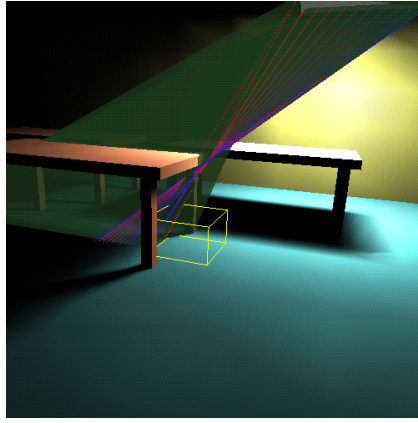
7. Acknowledgments

Thanks to Seth Teller for suggesting the treatment of static surfaces and Frdo Durand for a fruitful discussion on the treatment of dynamic surfaces.

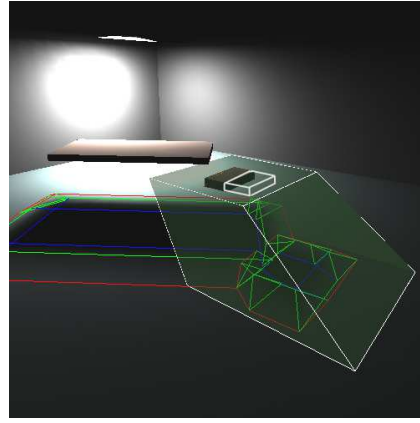
References

1. R. L. Cook, T. Porter, and L. Carpenter, “Distributed ray tracing”, *Computer Graphics*, **18**(4), pp. 137–147 (1984). Proceedings SIGGRAPH ’84 in Minneapolis (USA).
2. G. Drettakis and E. Fiume, “A fast shadow algorithm for area light sources using back projection”, in *SIGGRAPH 94 Conference Proceedings (Orlando, FL)* (A. Glassner, ed.), Annual Conference Series, pp. 223–230, ACM SIGGRAPH, (July 1994).
3. G. Drettakis and E. Fiume, “Structured penumbral irradiance computation”, *IEEE Transactions on Visualization and Computer Graphics*, **2**(4), pp. 299–313 (1996).
4. A. J. Stewart and S. Ghali, “Fast computation of shadow boundaries using spatial coherence and back-projections”, in *SIGGRAPH 94 Conference Proceedings (Orlando, FL)* (A. Glassner, ed.), Annual Conference Series, pp. 231–238, ACM SIGGRAPH, (July 1994).
5. D. Lischinski, F. Tampieri, and D. P. Greenberg, “Combining hierarchical radiosity and discontinuity meshing”, in *SIGGRAPH 93 Conference Proceedings (Anaheim, CA)* (J. T. Kajiya, ed.), Annual Conference Series, pp. 199–208, ACM SIGGRAPH, (Aug. 1993).
6. G. Drettakis and F. Sillion, “Accurate visibility and meshing calculations for hierarchical radiosity”, in

- Rendering Techniques '96* (X. Pueyo and P. Shroeder, eds.), pp. 269–279, Springer Verlag, (June 1996). Proceedings of 7th Eurographics Workshop on Rendering in Porto, Portugal.
7. N. Briere and P. Poulin, “Hierarchical view dependent structures for interactive scene manipulation”, in *SIGGRAPH 96 Conference Proceedings (New Orleans, LO)* (H. Rushmeier, ed.), Annual Conference Series, pp. 83–91, ACM SIGGRAPH, (Aug. 1996).
 8. D. R. Baum, J. R. Wallace, M. F. Cohen, and D. P. Greenberg, “The back-buffer algorithm : an extension of the radiosity method to dynamic environments”, *The Visual Computer*, **2**, pp. 298–306 (1986).
 9. D. W. George, F. Sillion, and D. P. Greenberg, “Radiosity redistribution for dynamic environments”, *IEEE Computer Graphics and Applications*, **10**(4), (1990).
 10. S. E. Chen, “Incremental radiosity: An extension of progressive radiosity to an interactive image synthesis system”, *Computer Graphics*, **24**(4), pp. 135–144 (1990). Proceedings SIGGRAPH '90 in Dallas (USA).
 11. S. Müller and F. Schoffel, “Fast radiosity repropagation for interactive virtual environments using a shadow-form-factor-list”, in *Photorealistic Rendering Techniques* (G. Sakas, P. Shirley, and S. Müller, eds.), (Darmstadt, Germany), Springer Verlag, (June 1994). Proceedings of Fifth Eurographics Workshop on Rendering.
 12. D. Forsyth, C. Yang, and K. Teo, “Efficient radiosity in dynamic environments”, in *Photorealistic Rendering Techniques* (G. Sakas, P. Shirley, and S. Müller, eds.), (Darmstadt, Germany), Springer Verlag, (June 1994). Proceedings of Fifth Eurographics Workshop on Rendering.
 13. E. Shaw, “Hierarchical radiosity for dynamic environments”, Master’s thesis, Cornell University, Ithaca, New York, (August 1994).
 14. P. S. Heckbert, “Adaptive radiosity textures for bidirectional ray tracing”, *Computer Graphics*, **24**(4), pp. 145–154 (1990). Proceedings SIGGRAPH '90 in Dallas (USA).
 15. D. Lischinski, F. Tampieri, and D. P. Greenberg, “Discontinuity meshing for accurate radiosity”, *IEEE Computer Graphics and Applications*, **12**(6), pp. 25–39 (1992).
 16. S. J. Teller, “Computing the antipenumbra of an area light source”, *Computer Graphics*, **26**(4), pp. 139–148 (1992). Proceedings of SIGGRAPH '92 in Chicago (USA).
 17. Z. Gigus and J. Malik, “Computing the aspect graph for the line drawings of polyhedral objects”, *IEEE Trans. on Pat. Matching & Mach. Intelligence*, **12**(2), (1990).
 18. Y. Chrysanthou and M. Slater, “Shadow volume BSP trees for computation of shadows in dynamic scenes”, in *1995 Symposium on Interactive 3D Graphics* (P. Hanrahan and J. Winget, eds.), pp. 45–50, ACM SIGGRAPH, (Apr. 1995).
 19. A. Worrall, C. Willis, and D. Paddon, “Dynamic discontinuities for radiosity”, in *Edugraphics + Compu-graphics Proceedings* (H. P. Santo, ed.), (Alvor, Portugal '95), pp. 367–375, GRASP- Graphic Science Promotions & Publications, P.O. Box 4076, Massama, 2745 Queluz, Portugal, (Dec. 12 1995).
 20. J. Amanatides and A. Woo, “A fast voxel traversal algorithm for ray tracing”, in *Eurographics '87*, pp. 3–10, North-Holland, (Aug. 1987).
 21. F. Durand, G. Drettakis, and C. Puech, “The Visibility Skeleton: A Powerful and Efficient Multi-Purpose Global Visibility Tool”, in *SIGGRAPH 97 Conference Proceedings (Los Angeles, CA)* (T. Whitted, ed.), Annual Conference Series, ACM SIGGRAPH, (Aug. 1997).
 22. F. Durand, G. Drettakis, and C. Puech, “The 3d visibility complex, a new approach to the problems of accurate visibility”, in *Rendering Techniques '96* (X. Pueyo and P. Shroeder, eds.), pp. 245–257, Springer Verlag, (June 1996). Proceedings of 7th Eurographics Workshop on Rendering in Porto, Portugal.

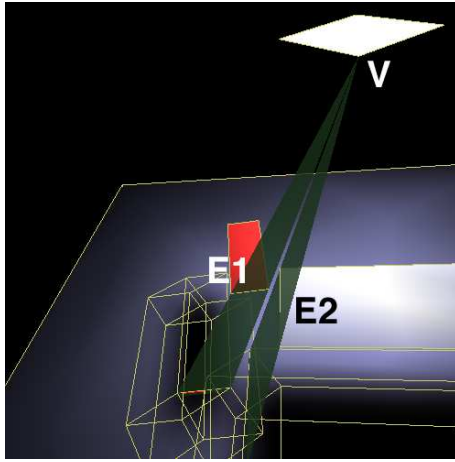


(a)

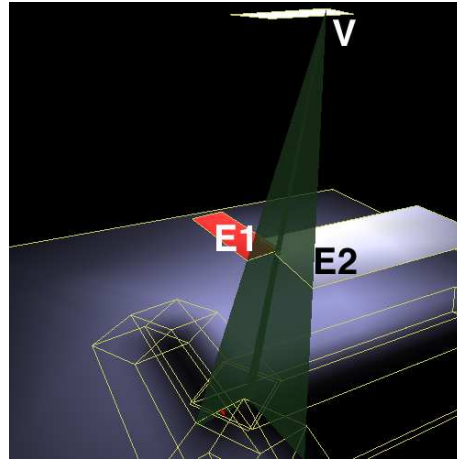


(b)

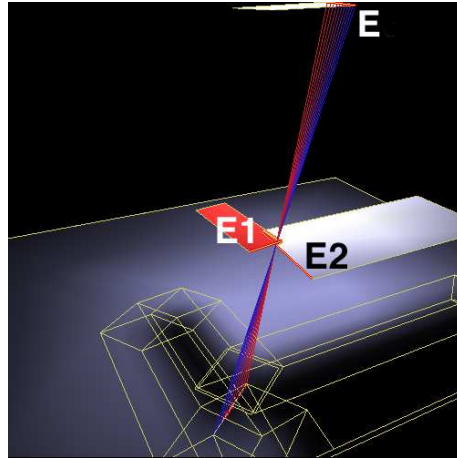
Figure 14: (a) Discontinuity surfaces in a voxel, (b) Intersection of motion volume with scene polygons



(a)



(b)



(c)

Figure 15: (a) Two EV surfaces which do not intersect (b) Intersection of the EV surfaces (c) One of the two EEE surfaces created



(a)

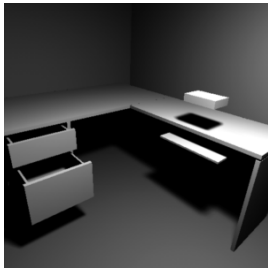


(b)

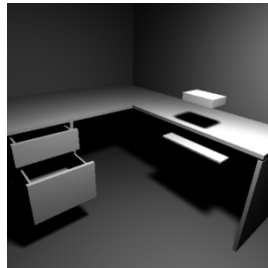


(c)

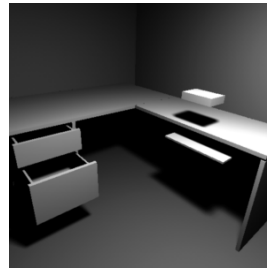
Figure 16: (a) *Scene 1* (b) *Scene 2* (c) *Scene 3*



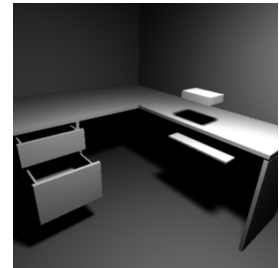
(a)



(b)

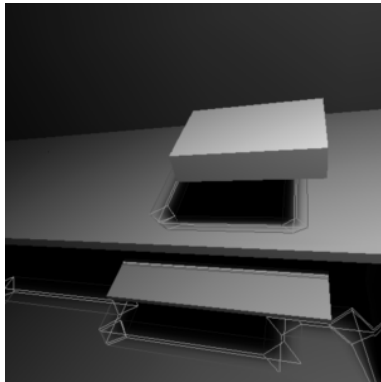


(c)

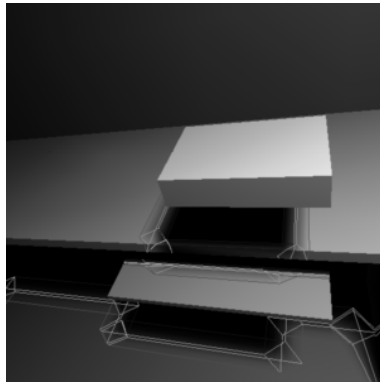


(d)

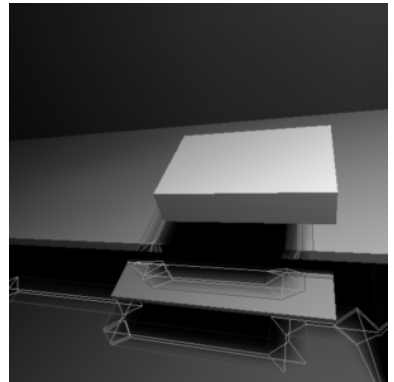
Figure 17: *Test 1:* (a) *position 1* (b) *position 2* (c) *position 3* (d) *position 4*



(a)



(b)



(c)

Figure 18: *Test 2:* (a) *position 1* (b) *position 3* (c) *position 4*

2.4.5 The 3D Visibility Complex, a new approach to the problems of accurate visibility (EGRW'96)

Auteurs : Frédo Durand, George Drettakis et Claude Puech

Actes : 6th Eurographics Workshop on Rendering

Date : juin 1996

The 3D visibility complex : a new approach to the problems of accurate visibility.

Frédo Durand, George Drettakis and Claude Puech

iMAGIS *

Abstract: Visibility computations are central in any computer graphics application. The most common way to reduce this expense is the use of approximate approaches using spatial subdivision. More recently analytic approaches efficiently encoding visibility have appeared for 2D (the visibility complex) and for certain limited cases in 3D (aspect graph, discontinuity meshes). In this paper we propose a new way of describing and studying the visibility of 3D space by a dual space of the 3D lines, such that all the visibility events are described. A new data-structure is defined, called the *3D visibility complex*, which encapsulates all visibility events. This structure is global and complete since it encodes all visibility relations in 3D, and is spatially coherent allowing efficient visibility queries such as view extraction, aspect graph, discontinuity mesh, or form factor computation. A construction algorithm and suitable data structures are sketched.

Keywords: visibility, visibility complex, spatial coherence, discontinuity meshing, form factor

1 Introduction

Visibility calculations are central to any computer graphics application. To date, no approach has been presented to encode all visibility information in a 3D scene.

In this paper we will present a new approach, which we call the *3D visibility complex*, which encodes all visibility information contained in a three dimensional scene. This research is in a preliminary phase, since an implementation has not yet been undertaken, but we believe that the importance and potential use of such a structure justify its presentation even at the stage of conception.

Related works The first attempts to cope with the cost of visibility computations involved space partitioning structures but they provided only local visibility information. Arvo and Kirk [1] subdivide the 5D ray-space for ray-tracing. Teller [13] uses the 5D Plücker duality to compute the antipenumbra cast by an area light source. He also developed algorithms for scenes naturally divided into cells [15] where the visibility is propagated through portals. In computer vision the *aspect graph* [7, 6] has been developed to group all the viewpoints for which an object has the same “aspect”. An aspect changes along visibility events which are the same as for the discontinuity meshing techniques [8]. These techniques have thus been extended with *backprojections* [3, 12] to provide the aspect of the source. Recently, efficient data structures have been developed for the 2D case [10, 5] and have inspired our research, although the new approach

* Laboratoire

GRAVIR

/

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

has been developed from scratch with the specifically three-dimensional problem in mind.

2 Description of the 3D Visibility Complex

In this discussion we will consider scenes of general convex objects, but the concepts will also be given for the polygonal scenes where appropriate. Visibility will be defined in terms of ray-objects intersections. If we consider the objects to be transparent, a ray is not blocked and all the objects a line intersects must be considered. If however we want to take occlusions into account, we will consider maximal free segments which are segments having no intersection with the inside of the objects and whose length is maximal (their two extremities lie on the boundary of two objects or are at infinity). In what follows we will often refer to them simply as *segments*. Segments can be interpreted as rays which can *see* the two objects on their extremities. A 3D line can be collinear to many segments, separated by the objects the line intersects. In this paper, we will introduce concepts first in terms of line visibility (where all the objects intersected by a line are considered) and then in terms of segment visibility (where the occlusions are taken into account).

We wish to group the segments (or the lines) which see the same objects. A partition of the set of segments into connected components according to their visibility is thus required. Since sets of segments are not intuitive objects, we will try to represent them in a dual space which will afford a better understanding of intricate visibility relationships. A suitable duality will thus be used for the purposes of illustration and presentation.

2.1 Duality

We have chosen to decompose the 4 dimensions of line space into two dimension of direction (the spherical coordinates (θ, φ) of the director vector of the lines) and a projection (u, v) onto the plane perpendicular to the line and going through the origin. The axes of the planes are chosen such as u is along $\mathbf{t} \wedge \mathbf{y}$ ². The intersections of a line with two parallel planes could also be used. Nonetheless, we believe that such an approach makes the interpretation of lines sharing one coordinate harder.

Visualizing 4D space is very hard. It can be seen as a moving 3D world with the 4th dimension being time. One approach is to use slices (in this paper we will fix $\varphi = ct$) which can be seen as frames in time. Such a slice will be called a φ -slice. Since each slice will be a 3D space (θ, u, v) , it will sometimes be useful to cut one more time and consider φ and θ constant. We will obtain a 2D slice where only u and v vary, composed of all the lines which are parallel and have the direction (θ, φ) . Such a slice will be called a $\theta\varphi$ -slice. These 2D $\theta\varphi$ -slices are easier to handle and visualize. They justify in part the choice of the duality because they can be interpreted as orthographic projections of the scene.

2.2 Tangency curves

Line Visibility Visibility changes when a line becomes tangent to an object. The set of lines tangent to one object is a 3-D set in the 4D dual space. This means, more intuitively, that a line has 3 degrees of freedom to stay tangent to one object. We will call the dual of the set of lines tangent to an object the *tangency volume* of this object.

² Discontinuities occur at $\varphi = \pm \frac{\pi}{2}$, but since we use this duality for the purpose of presentation and visualization we can ignore them without loss of generality.

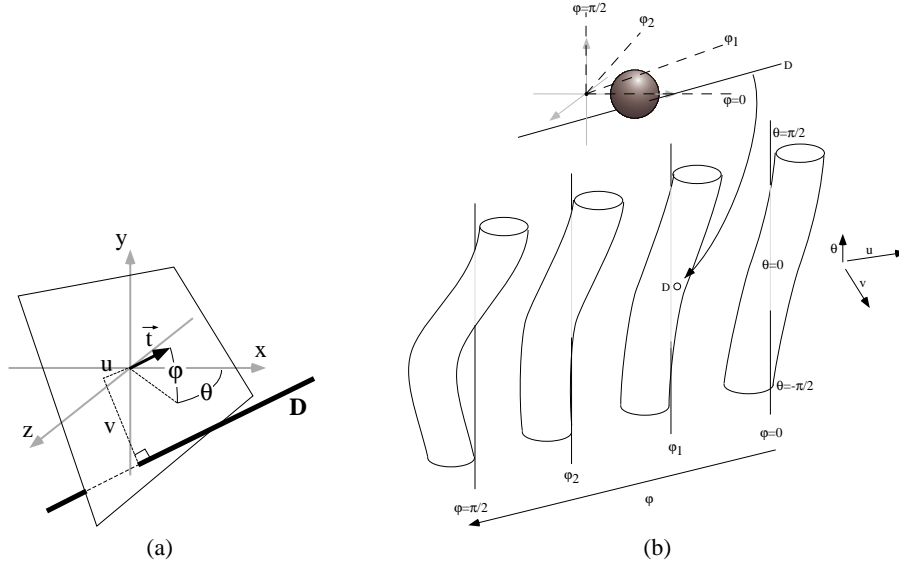


Fig. 1. (a) Duality (b) Tangency Volume of a sphere. The θ axis ($u = 0, v = 0$) is shown for each φ -slice providing a better 3D visualization. In the left-hand φ -slice, which corresponds to the discontinuity in the duality for $\varphi = \frac{\pi}{2}$, the “cylinder” just turns around the θ axis. The line D intersects the object and has its dual inside the tangency volume.

Figure 1b shows a representation of the tangency volume of a sphere. For each φ -slice, the set of tangents is a sort of 2D “cylinder”, forming a 3D structure in the 4D dual space. If we consider a 2D $\theta\varphi$ -slice (horizontal in figure 1b) the set of tangents sharing that direction is a circle in the dual space. This is general: because of the definition of u and v , the set of tangents to one object in one direction is the outline of the object in this direction.

If a line has its dual on the tangency volume, it is tangent to the object. If the dual is inside the 4D set bounded by the tangency volume, it intersects the object, similarly to line D on figure 1b.

Segment Visibility Let us now consider visibility with occlusion. A line which intersects the object is collinear to at least two segments, one before and one after the object.

Consider a $\theta\varphi$ -slice such as that on the lower left of figure 2. The sets of lines that intersect and that do not intersect the object are bounded by the outline of the object. For segment visibility we have to consider the segments that see the front of the object and those that see its back. Since such segments are collinear to the same line, they are projected on the same point in the 4D line dual space. Consequently the set of segments that see the front and the set of segments that see the back of the object are projected onto the same position of the 4D dual space as shown in the right of figure 2. The outline, which is the set of tangents to the object for the chosen θ and φ , is incident to the three sets (front, back and no intersection). This means that a segment tangent to the object has topological neighbours that do not intersect the objects, some that see the front, and some that see the back.

To differentiate the segments, we add a pseudo-dimension. It is not a continuous dimension since we just have to sort all the collinear segments. If we impose $\theta = ct$, $\varphi = ct$ and $v = ct$, the sets of segments can be represented by a graph³ shown on the lower right. Each tangent corresponds to a vertex of the graph. This graph is a 1D structure embedded in 2D. Similarly, for a $\theta\varphi$ -slice, the sets of segments are represented by a 2D structure embedded into 3D. We call the partition of the segments of direction (θ, φ) according to their visibility the *auxiliary complex* for (θ, φ) (see also figure 4).

In a similar manner, a φ -slice is in fact a 3D structure embedded into 4D, and the sets of segments is a 4D space embedded into 5D.

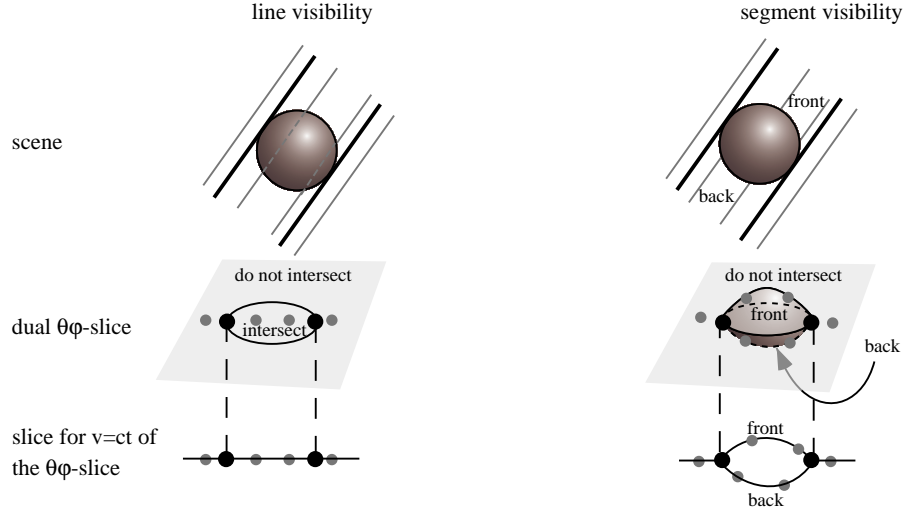


Fig. 2. Visibility for $\theta = ct$ and $\varphi = ct$. If we consider lines (on the left), visibility can be described by a planar structure (below). But if we consider segments (on the right) we have different levels on this plane depending on the side of the object. The set of segments which do not intersect and the sets of those that intersect the front or the back of the object share the same boundary, the tangents to the object which correspond to its outline. Recall that the Auxiliary Complex shown on the lower right is a 2D structure embedded into 3D, i.e. it is “empty”, since the points outside the surfaces have no meaning.

2.3 Bitangents

Line Visibility Now consider two objects. If a line has its associated dual point inside the tangency volumes of both objects, it intersects them both. The tangency volumes give us a partition of the dual space of the 3D lines according to the objects they intersect. We call this partition the *dual arrangement*. Its faces are 4D sets of lines which intersect the same objects. They are bounded by portions of the tangency volumes which are 3D. The intersection of two tangency volumes is a 2D set corresponding to the lines tangent to the two objects (bitangents).

For a φ -slice the set of bitangents is a space curve (shown as dashed line in figure 3 on the two φ -slices on the right). It corresponds to the intersection of the two “cylinders”

³ It is in fact an embedding of a graph since the points on the edges also have a meaning

which are the φ -slices of the tangency volumes. The slice of a 4D face is a volume corresponding to the intersection of the inside of the two cylinders.

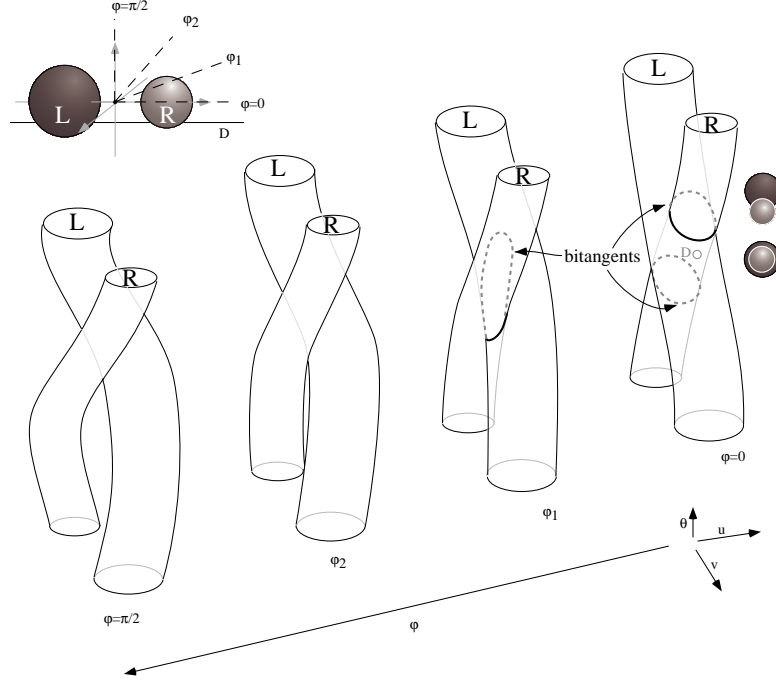


Fig. 3. Dual arrangement for two spheres.

Segment visibility An auxiliary complex for two objects is shown on figure 4 for a given direction. It is still delimited by the outline of the objects, but for example the outline of the upper sphere has no influence on the set B of segments that see the back of the lower sphere. Note that the two bitangents (shown in fat black lines) are incident to all faces.

Figure 5 is a φ -slice for $\varphi = 0$ of all the faces of the scene composed of two spheres of figure 3. The view in a given direction is shown on the left of the cylinders, and we consider the associated auxiliary complex shown six times on the top of the schema. Each time, a face is hatched and a volume is drawn below which corresponds to the φ -slice of the face of the visibility complex at $\varphi = 0$. Note that the union of these volumes is more than the entire 3D space, since a φ -slice of the complex is a 3D structure embedded into 4D.

2.4 Tritangents

Consider now a scene of three objects. A line tangent to the three objects has its dual at the intersection of the three tangency volumes. A set of connected tritangents is a 1D set in the 4D dual space. Its projection on a φ -slice is a point. The set of tritangents can be also interpreted as the intersection of the three sets of bitangents.

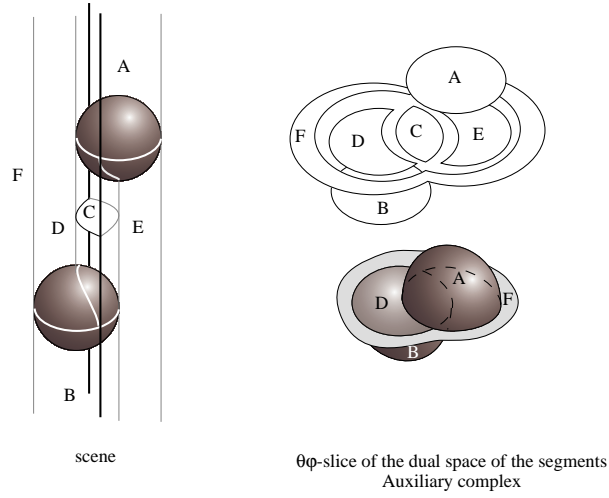


Fig. 4. Auxiliary Complex for two spheres. Recall that the auxiliary complex is a 2D structure embedded in 3D. In the lower representation, only the points on the surfaces represented are associated with segments. In the upper view, the faces of the auxiliary complex have been moved out to make their incidences easier to understand.

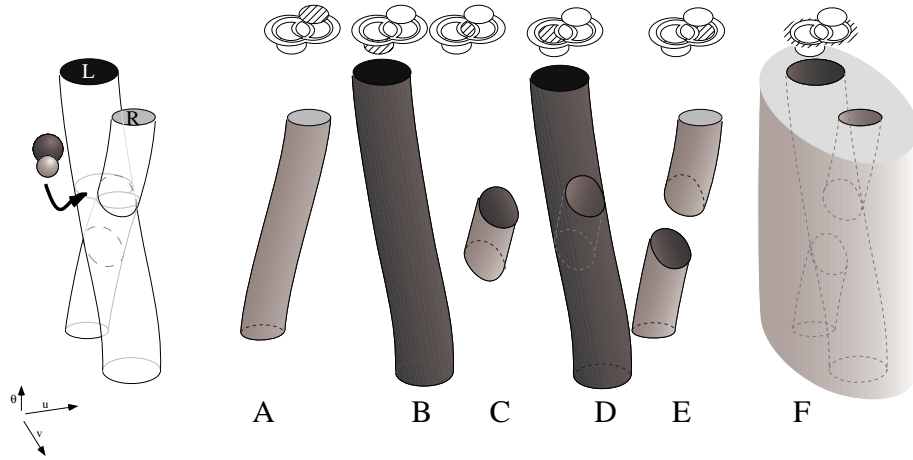


Fig. 5. φ -slice for $\varphi = 0$ of the faces of the visibility complex of the previous scene. A is the set of segments that see the front of R , B is the set of segments that see the back of L . C is the set of segments between L and R . It can be interpreted as the intersection of set of lines that see L and the set of lines that see R , and in the dual space it has the shape of $A \cap B$. D is the set of segments that see the front of L . Since the visibility is occluded by R in this direction, D has the shape of $B - A$. Similarly, E is the set of segments that see the back of R . Finally, F is the set of segments that see none of the two spheres. It is the complement of $A \cup B$.

Figure 6 shows part of the visibility complex of a scene of three spheres. On the φ -slice $\varphi = 0$ two orthographic views of the scene for $\theta = 0$ (View 0) and for $\theta = \theta_2$ (View 2) are drawn next to the corresponding θ in the φ -slice. The set F of segments that see the spheres R and B is shown by its two slices F_0 and $F_{\varphi 1}$. Note that it is the intersection of the tangency volume of R and B minus the tangency volume of G . The tritangents are the points in white. Note also that because of the occlusion by the sphere G , lines that are bitangents of the R and B do not correspond to bitangent segments. This is shown in figure 7 which is a zoomed view of the φ -slice $\varphi = 0$. The set of bitangents B_0 is cut because bitangent lines such as D intersect G and correspond to no bitangent segment. We can thus see that the tritangent T_0 and T'_0 are the intersection of the φ -slices of the three tangency volumes, and are also incident to the three sets of bitangents B_0 , B'_0 and B''_0 .

Note that a scene does not necessarily contain tritangents in the general case.

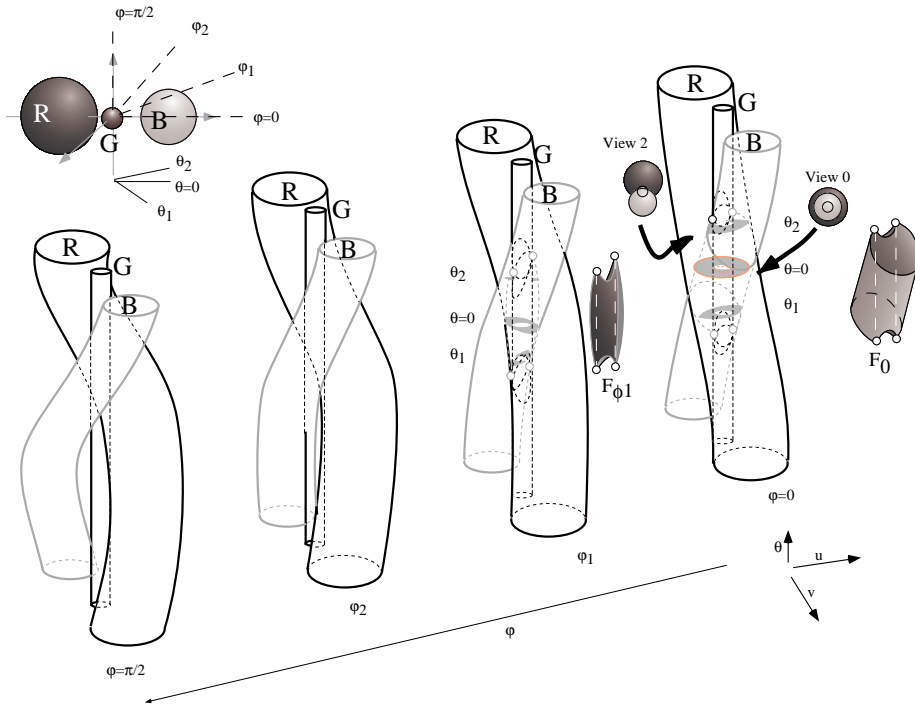


Fig. 6. Visibility Complex of a scene of three spheres.

3 Data Structure and Storage Complexity

3.1 Overview of the Data Structure

We have defined the dual arrangement which is the partition of the lines of the 3D space into connected components according to the objects they intersect. It is a 4D structure.

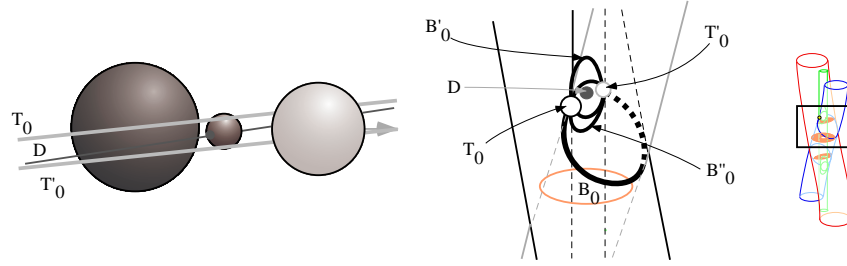


Fig. 7. Zoomed view of the φ -slice $\varphi = 0$.

Similarly, the 3D visibility complex is the partition of the maximal free segments of 3D space into connected components according to the objects they touch. It is a 4D structure embedded into 5D. The dimensions and incidences of the boundaries of the faces are summarised in table 3.1.

Note that the elements of the visibility complex and those of the dual arrangement are not the same. A line can be tangent to two objects and correspond to no bitangent segment because of occlusions.

In the general case, a scene can have a degenerate visibility complex with no vertex and no tritangency edge.

Dim	Scene configuration	φ -slice in the dual space	Name
4			face
3			tangency face
2			bitangency face
1			tritangency edge
0			vertex

Table 1. Elements of the visibility complex

3.2 Polygonal case

In the case of polygonal scenes, the outlines of the objects can be decomposed into edges and vertices. Consequently the tangency volumes of a polygon can be divided into sets of lines going through the edges which are 3D sets, and sets of lines going through the vertices which are 2D sets. A 2D component of the complex corresponds to a segment touching two edges, or to a segment touching one vertex of a polygon. In the same manner, the 1-faces of the complex correspond to segments going through three edges (the *EEE* events of the aspect graphs or of the discontinuity mesh) or to segments going through an edge and a vertex (the *EV* events). Vertices of the complex can be *EEEE* or *EEV* or *VV* events. In particular a line (or a segment) going through the vertex of a polygon can be interpreted as being tangent to the two edges incident to this vertex.

In the polygonal case the visibility complex is always non-degenerate since there are always VV vertices and EV 1-faces.

3.3 Complexity

In the general case, there exist convex objects for which the number of faces of the complex is unbounded. However, in the polygonal case, the storage complexity of the visibility complex is $O(n^4)$, where n is the number of edges of polygons. This complexity depends strongly on the configuration of the scene. We show below that the proposed construction algorithm is $O(n \log n)$.

As mentioned in the introduction, practical experience with discontinuity meshing has shown that the scenes studied in computer graphics tend to have more optimistic visibility complexity than that predicted by the theoretical worst case [3].

4 Applications of the approach

4.1 View computation

A view around a point is defined by the extremities of the set of segments going through this point. The set of segments going through a point is a 2D surface in the dual space (u and v can be expressed with $\sin(\theta)$ and $\sin(\varphi)$). The view can be expressed as the intersection of the visibility complex with this surface. Each face intersected corresponds to an object seen. An intersection with a tangency volume corresponds to an outline in the image. The ray-tracing algorithm is equivalent to a sampling of such a surface.

In figure 8, the surface described by the lines going through viewpoint V is represented by its φ -slices which are curves. The intersections of these curves with the tangency volumes are the points of the view on the outline of the objects, such as D_1 , D_2 , D_3 , D_4 and D_5 . However, all the intersections do not necessarily correspond with an outline since the objects are not transparent, and points such as D' must not be taken into account. Consider the φ -slice $\varphi = 0$ and the slice V_0 of the lines going through V with $\varphi = 0$. Figure 9 shows the φ -slices of the faces of the visibility complex and their traversal. We traverse the visibility complex up and down along V_0 . Initially, the segments see nothing, since we are in the face F . At D_1 , we leave face F and have to choose between face A and E . Since V lies in the front of the sphere R , we now traverse A from D_1 to D_2 . D' lies on no boundary of face A and is thus not considered. We then traverse face D and finally face F again. Once the φ -slice has been traversed, the intersections with the boundaries of the faces are maintained while φ is swept. Visibility changes will appear when V_φ meets a bitangency edge or a new tangency volume.

For a walkthrough, the view can be maintained since the events where the visibility changes correspond to intersections of the surface described by V with the 1-faces of the visibility complex. This approach is similar to the one described in [2] where conservative visibility events are lazily computed.

4.2 Form-Factors

The form factor F_{ij} ⁴ involved in radiosity computation is the proportion of light that leaves patch i which arrives at patch j . It can be expressed as the measure of lines which intersect i and j divided by the measure of lines which intersect i . In the dual space, it is the measure of the face F_{ij} divided by the measure of the inside of the tangency volume of i . See [9] [4] for the equivalent interpretation of the form factors with the 2D visibility complex.

⁴ The same notation is used for the form factor and for the face between i and j though the form factor is a scalar and the face is a set of segments.

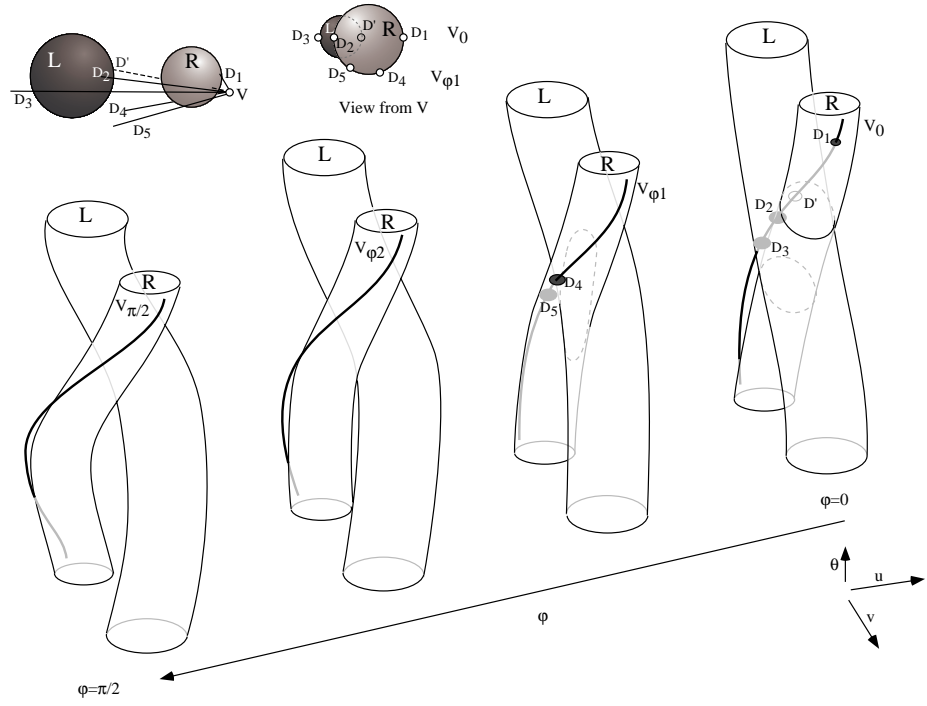


Fig. 8. The View around a point is the intersection of the visibility complex and the surface described by the set of segments going through this point.

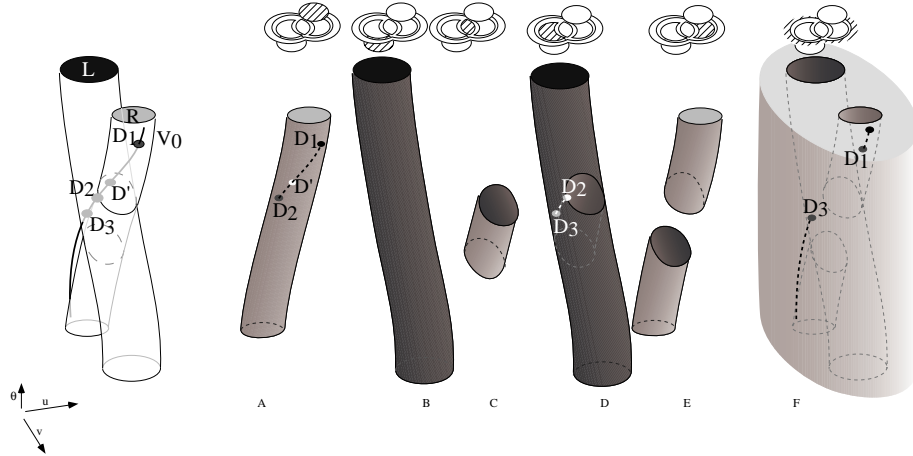


Fig. 9. Traversal of the ϕ -slice $\phi = 0$ of the complex to compute the view around point V .

4.3 Other applications

The 1-faces of the visibility complex correspond to the visibility events of the aspect graph. The complex can thus help in its construction. The complexity of the aspect graph is $O(n^6)$ though the visibility complex is “only” $O(n^4)$ because the aspect graph is an arrangement of the $O(n^3)$ 1-faces of the complex.

In the same way, the 1-faces inside the tangency volume of the scene correspond to the discontinuity surfaces of the discontinuity meshing methods. The visibility complex gives all the events to compute a discontinuity mesh where all the objects are considered as sources.

In the context of hierarchical radiosity, whenever a link between two objects i and j is to be refined the boundary of the face F_{ij} of the visibility complex provides all the visibility information pertinent to this energy exchange. This information can be used to effect progressive discontinuity meshing and to improve the quality of the form-factor calculation.

5 Implementation

We give here a general outline for the implementation of this data structure for scenes of polygons. The development of the actual implementation will present technical difficulties which we have not yet addressed. We simply sketch an outline of the form the data structure will have and give a general idea of how the construction will proceed.

5.1 Data Structure

To represent the 3D visibility complex, we can use a polytope structure. Each k -face has pointers to its boundaries (faces of a lower dimension) and to the faces of a larger dimension it is adjacent to. A tangency face has for example a list of the bitangency face of its boundary, and three pointers to its adjacent faces. For each k -face we also store the two objects it can see and the objects to which its segments are tangent.

5.2 Algorithm

We present here the outline of an algorithm to build the visibility complex. It consists of a direct enumeration of the vertices of the complex inspired by [6], and then a sweep of these vertices.

All the potential $O(n^3)$ EV and EEE events are first enumerated, and we then compute the intersection of the corresponding discontinuity surfaces with the n objects of the scene. This gives us all the vertices of the visibility complex which are then sorted in φ and stored in a priority queue.

We then maintain a φ -slice of the complex during the sweep of the vertices. For each vertex swept we link all the k -faces incident to this vertex.

The algorithm presented is $O(n^4 \log n)$, but experience in the field of discontinuity meshing and backprojections has shown that the cost can be much reduced thanks to accelerations techniques [3]; the number of EEE actually considered is usually far less than n^3 .

6 Conclusions

We have presented a new approach for visibility computation and described a powerful data-structure which encapsulates all the visibility information in a 3D scene. The dual space used affords a better understanding of the visibility events, which have been presented in detail. Moreover, this representation gives all the relations of adjacency between these events.

The 3D visibility complex is a very promising data structure for numerous computer graphics applications: we have briefly outlined its potential use for the visibility computation of a view, its use in form-factor computations and discontinuity meshing as well as the computation of aspects or backprojections.

We have presented a first outline of the data structure and a construction algorithm. Current work focuses on the completion of the algorithm and the data structure and its subsequent implementation for polygonal scenes.

It is nonetheless evident that when applied to large scenes, the 3D visibility complex will suffer from combinatorial growth in storage. To cope with this combinatorial complexity, two strategies will be explored. Lazy construction can allow the computation of only the most important visibility events and faces of the visibility complex when they are actually needed by the application. A hierarchical extension of the 3D visibility complex will be studied.

Finally the visibility complex, like its 2D equivalent, seems very promising for dynamic environments due to its inherently coherent construction.

References

1. Arvo J, Kirk D. Fast ray-tracing by ray classification. SIGGRAPH 87
2. Coorg S, Teller S. Temporally coherent conservative visibility. Proc. of the 12th Ann. ACM Symp. on Computational Geometry, May 1996.
3. Drettakis G, Fiume E. A fast shadow algorithm for area light sources using backprojections. SIGGRAPH 1994.
4. Durand F, Orti R, Rivire S, Puech C. Radiosity in flatland made visibly simple. video of the 12th Ann. ACM Symp. on Computational Geometry, May 1996.
5. Durand F, Puech C. The Visibility Complex made visibly simple, video of the 11th Ann. ACM Symp. on Computational Geometry, June 1995.
6. Gigus Z, Canny J, Seidel R. Efficiently computing and representing aspect graphs of polyhedral objects. IEEE Trans. on Pattern Analysis and Machine Intelligence, June 1991.
7. Gigus Z, Malik J. Computing the Aspect graph for line drawings of polyhedral objects. IEEE Trans on Pattern Analysis and Machine Intelligence, vol. 12, February 1990.
8. Lischinski D, Tampieri F, Greenberg D. Combining hierarchical radiosity and discontinuity meshing. SIGGRAPH 1993.
9. Orti R, Rivire S, Durand F, Puech C. Radiosity for dynamic scenes in flatland with the visibility complex. Eurographics 1996.
10. Pocchiola M, Vegter G. The visibility Complex, To appear in the special issue of Internat. J. Comput. Geom. Appl. devoted to the 9th Annu. ACM Sympos. on Comput. Geometry, held in San Diego, June 1993.
11. Sbert M. An integral geometry based method for fast form-factor computation. Eurographics 1993.
12. Stewart J, Ghali S. Fast computation of shadow boundaries using spatial coherence and backprojections. SIGGRAPH 1994.
13. Teller S. Computing the antipenumbra cast by an area light source. SIGGRAPH 1992.
14. Teller S. Visibility computation in densely occluded polyhedral environments. PhD thesis UC Berkeley, 1992
15. Teller S, Hanrahan P. Global visibility algorithm for illumination computations. SIGGRAPH 1993.

2.4.6 The Visibility Skeleton : A Powerful and Efficient Multi-Purpose Global Visibility Tool (SIGGRAPH'97)

Auteurs : Frédo Durand, George Drettakis et Claude Puech

Actes : SIGGRAPH '97

Date : août 1997

The Visibility Skeleton: A Powerful And Efficient Multi-Purpose Global Visibility Tool

Frédo Durand, George Drettakis and Claude Puech

iMAGIS[†]- GRAVIR/IMAG - INRIA

Abstract

Many problems in computer graphics and computer vision require accurate *global* visibility information. Previous approaches have typically been complicated to implement and numerically unstable, and often too expensive in storage or computation. The Visibility Skeleton is a new powerful utility which can efficiently and accurately answer visibility queries for the entire scene. The Visibility Skeleton is a *multi-purpose* tool, which can solve numerous different problems. A simple construction algorithm is presented which only requires the use of well known computer graphics algorithmic components such as ray-casting and line/plane intersections. We provide an exhaustive catalogue of visual events which completely encode all possible visibility changes of a polygonal scene into a graph structure. The nodes of the graph are extremal stabbing lines, and the arcs are critical line swaths. Our implementation demonstrates the construction of the Visibility Skeleton for scenes of over a thousand polygons. We also show its use to compute exact visible boundaries of a vertex with respect to any polygon in the scene, the computation of global or on-the-fly discontinuity meshes by considering any scene polygon as a source, as well as the extraction of the exact blocker list between any polygon pair. The algorithm is shown to be manageable for the scenes tested, both in storage and in computation time. To address the potential complexity problems for large scenes, on-demand or lazy construction is presented, its implementation showing encouraging first results.

Keywords: Visibility, Global Visibility, Extremal Stabbing Lines, Aspect Graph, Global Illumination, Form Factor Calculation, Discontinuity Meshing, View Calculation.

1 INTRODUCTION

Ever since the early days of computer graphics, the problems of determining visibility have been central to most computations required to generate synthetic images. Initially the problems addressed concerned the determination of visibility of a scene with respect to a given point of view. With the advent of interactive walkthrough systems and lighting calculations, the need for *global* visibility queries has become much more common. Many examples of such requirements exist, and are not limited to the domain

of computer graphics. When walking through a complex building, real-time visualization algorithms require the information of which objects are visible to limit the number of primitives rendered, and thus achieve better frame rates. In global illumination computations, the dominant part of any calculation concerns the determination of the proportion of light leaving surface s and arriving at surface r . This determination depends heavily on the relative occlusion of the two objects, requiring the calculation of which parts of s are visible from r . All such applications need detailed data structures which completely encode global visibility information; previous approaches have fallen short of this goal.

1.1 Motivation

The goal of the research presented here is to show that it is possible to construct a data structure encompassing all global visibility information and to show that our new structure is useful for a number of different applications. We expect the structure we present to be of capital importance for any application which requires detailed visibility information: the calculation and maintenance of the view around a point in a scene, the calculation of exact form-factors between vertices and surfaces, the computation of discontinuity meshes between any two pairs of objects in a scene as well as applications in other domains such as aspect graph calculations for computer vision etc.

Previous algorithms have been unable to provide efficient and robust data structures which can answer global visibility queries for typical graphics scenes. In what follows we present a new data structure which can provide *exact* global visibility information. Our structure, called the *Visibility Skeleton*, is easy to build, since its construction is based exclusively on standard computer graphics algorithms, i.e., ray casting and line-plane intersections. It is a *multi-purpose* tool, since it can be used to solve numerous different problems which require global visibility information; and finally it is well-adapted to *on-demand* or *lazy* construction, due to the locality of the construction algorithm and the data structure itself. This is particularly important in the case of complex geometries.

The central component of the Visibility Skeleton are *critical lines* and *extremal stabbing lines*, which, as will be explained in detail in what follows, are the foci of all visibility changes in a scene. All modifications of visibility in a polygonal scene can be described by these critical lines, and a set of *line swaths* which are necessarily adjacent to these lines. In this paper we present the construction of the Skeleton, and the implementations of several applications. As an example, consider Fig. 1(a), which is a scene of 1500 polygons. After the construction of the skeleton, many different queries can be answered efficiently. We show the view from the green selected point to the left wall which only required 1.4 ms to compute; in Fig. 1(b), the complete discontinuity mesh on the right wall is generated by considering the screen of the computer as an emitter which required 8.1 ms.

After a brief overview of previous work (Section 1.2), we will provide a complete description of all possible nodes, and all the adjacent line swaths in Section 2. In Section 3 the construction algorithm and the actual data structure are described in detail. The results of our implementation are then presented in Section 4, giving the complete construction of the Visibility Skeleton for a suite

[†]iMAGIS is a joint research project of CNRS/INRIA/UJF/INPG. iMAGIS/GRAVIR, BP 53, F-38041 Grenoble Cedex 09 France. E-mail: {Frederic.Durand|George.Drettakis|Claude.Puech}@imag.fr <http://www-imagis.imag.fr/>

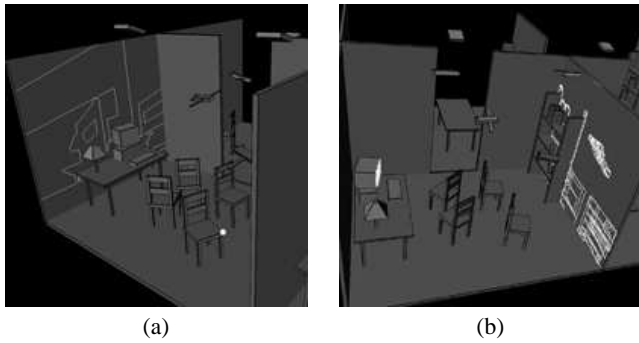


Figure 1: (a) Exact computation of the part of the left wall as seen by the green vertex. (b) Complete discontinuity mesh on the right wall when considering the computer screen as source.

of test scenes. We show how the Skeleton is then used to provide exact point-to-surface visibility information for any vertex in the scene, to calculate the complete discontinuity mesh between any two surfaces in the scene, extract exact blocker lists between two objects, and compute all visibility interactions of one object with all other objects in a scene, which could be used for dynamic illumination updates in scenes with moving objects. Section 5 addresses the issues arising when treating more complex scenes, and in particular we present a first attempt at on-demand construction. The results of the implementation show that this allows significant speedup compared to the complete algorithm. In Section 6 we sketch how the structure can be extended to environments in which objects move, as well as other potential extensions, and we conclude.

1.2 Previous Work

Many researchers in computer graphics, computational geometry and computer vision have addressed the issue of calculating global visibility. We present here a quick overview of closely related previous work, which is of course far from exhaustive.

Interest in visibility structures in computer graphics was expressed by Teller [26], when presenting an algorithm for the calculation of anti-penumbra. This work was in part inspired by the wealth of research in computer vision related to the *aspect graph* (e.g., [21, 10, 9]). The work of Teller is closely related to the development of discontinuity meshing algorithms (pioneered by [14, 17]). These algorithms lead to structures closely resembling the aspect graph which contain visibility information (*back-projections*) with respect to a light source [5, 24]. Discontinuity meshes have been used in computer graphics to calculate visibility and improve meshing for global illumination calculations [18, 6]. Nonetheless, these structures have always been severely limited by their inability to treat visibility between objects other than the primary light sources. This is caused by the fact that the calculation of the discontinuity mesh with respect to a source is expensive and prone to numerical robustness problems.

An alternative approach to calculating visibility between two patches for global illumination has been proposed by Teller and Hanrahan [27]. In this work a conservative algorithm is presented which answers queries concerning visibility between any two patches in the scene but does not provide exact visibility information. In addition, this approach provides tight blocker lists of potential occluders between a patch pair. Information on the potential occluders between a patch pair is central in the design of any refinement strategy for hierarchical radiosity [12]. The ability to determine analytic visibility information between two arbitrary patches would render practical the error bound refinement strategy of [16], which requires this information.

In computational geometry, the problem of visibility has been extensively studied in two dimensions. The visibility complex [22] provides all the information necessary to compute global visibility. This was successfully used in a 2D study of the problem applied to radiosity [19]. A similar structure in 3D, called the *asp*, has been presented in computer vision by Plantinga and Dyer [21], to allow the computation of aspect graphs. This structure provides the information necessary to compute exact visibility information. A related, but more efficient structure called the 3D visibility complex [7] has been proposed. Both structures have remained at the theoretical level for the full 3D perspective case which is the only case of interest for 3D computer graphics, despite partial implementations of orthographic and other limited cases for the *asp* [21]. Other related work in a computational geometry framework can be found in [15, 20].

Moreover, most of the work done on static visibility does not easily extend to dynamic environments. Most of the time, motion volumes enclosing all the positions of the moving objects are built [3, 8, 23].

2 THE VISIBILITY SKELETON

The new structure we will present addresses many of the shortcomings of previous work in global visibility. As mentioned earlier, the emphasis is on the development of a multi-purpose tool which can be easily used to resolve many different visibility problems, a structure which is easy and stable to build and which lends itself to on-demand construction and dynamic updates.

In what follows, we will consider only the case of polygonal scenes.

2.1 Visual Events

In previous global visibility algorithms, in particular those relating to aspect graph computations (e.g., [21, 10, 9]), and to antipenumbra [26] or discontinuity meshing [5, 24], visibility changes have been characterized by *critical lines sets* or *line swaths* and by *extremal stabbing lines*.

Following [20] and [26], we define an extremal stabbing line to be incident on four polygon edges. There are several types of extremal stabbing lines, including vertex-vertex (or *VV*) lines, vertex-edge-edge (or *VEE*) lines, and quadruple edge (or *E4*) lines. As explained in Section 2.3.1, we will also consider here extremal lines associated to faces of polyhedral objects.

A *swath* is the surface swept by extremal stabbing lines when they are moved after relaxing exactly one of the four edge constraints defining the line. The swath can either be planar (if the line remains tight on a vertex) or a regulus, whose three generator lines embed three polygon edges.

We call *generator elements* the vertices and edges participating in the definition of an extremal stabbing line.

We start with an example: after traversing an *EV* line swath from left to right as shown in Figure 2(a), the vertex as seen from the observer will lie upon the polygon adjacent to the edge and no longer upon the floor. This is a visibility change (often called visibility event). The topology of the view is modified whenever the vertex and the edge are aligned, that is, when there is a line from the eye going through both e and v .

This *EV* line swath is a one dimensional (1D) set of lines, passing through the vertex v and the edge e_1 , thus it has one degree of freedom (varying for example over the edge e). When two such *EV* surfaces meet as in Figure 2(b) a unique line is defined by the intersection of the two planes defined by the *EV* surfaces. This line is an extremal stabbing line; it has zero degrees of freedom.

In what follows we will develop the concepts necessary to avoid any direct treatment of the line swaths themselves since sets of lines

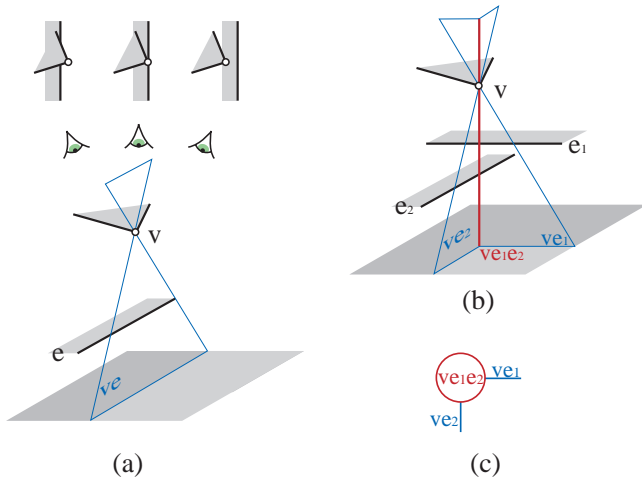


Figure 2: (a) While the eye traverses the line swath VE , the vertex v passes over the edge e . (b) Two line swaths meet at an extremal stabbing line (c) and induce a graph structure

or the surfaces described by these sets are difficult to handle, in part because they can be ruled quadrics. All computations will be performed by line – or ray – casting in the scene.

We will be using the extremal stabbing lines to encode all visibility information, by storing a list of all line swaths adjacent to each extremal stabbing line. In our first example of Figure 2(b), the VEE line ve_1e_2 is adjacent to the two 1D elements ve_1 and ve_2 described above; i.e., the swaths ve_1 and ve_2 . Additional adjacencies for the VEE line ve_1e_2 are implied by the interaction of ve_2 and e_1 (Fig 3(a)).

To complete the adjacencies of a VEE line, we need to consider the EEE line swaths related to the edges e_4 and e_2 , and the two edges e_4 and e_3 which are adjacent to the vertex v (Fig. 3(b) and (c)).

The simple construction shown above introduces the fundamental idea of the Visibility Skeleton: by determining all the appropriate extremal stabbing lines in the scene, and by attaching all adjacent line swaths, we can completely describe all possible visibility relationships in a 3D scene. They will be encoded in a graph structure as shown on Fig.3, to be explained in Section 2.3.2. Consider the example shown in Fig.3(a): The node associated to extremal stabbing line ve_1e_2 is adjacent in the graph structure to the arcs associated with line swaths ve_1 , ve_1' and ve_2 .

2.2 The 3D Visibility Complex, the Asp and the Visibility Skeleton

The *Visibility Complex* [7], is a structure which also contains all relevant visibility information for a 3 dimensional scene. It is also based on the adjacencies between visibility events and considers sets of maximal free segments of the scene (these are lines limited by intersections with objects).

The zero and one-dimensional components of the visibility complex are in effect the same as those introduced above, which we will be using for the construction of the Visibility Skeleton. Similar constructions were presented (but not implemented to our knowledge for the complete perspective case) for the *asp* structure [21] for aspect graph construction.

In both cases, higher dimensional line sets are built. For the visibility complex in particular, faces of 2, 3 and 4 dimensions are considered. For example, the set of lines tangent to two objects has 2 degrees of freedom, those tangent to one object 3 degrees of

freedom, etc. (see [7] for details).

These sets and their adjacencies could theoretically be useful for some specific queries such as view computation or dynamic updates, for example in some specific worst cases such as scenes composed of grids aligned and slightly rotated. In such cases, almost all objects occlude each other and the high number of line swaths and extremal stabbing lines makes the grouping of lines into higher dimensional sets worthwhile.

The *Visibility Complex* and *asp* are intricate data-structures with complicated construction algorithms since they require the construction of a 4D subdivision. In addition they are difficult to traverse due to the multiple levels of adjacencies. Our approach is different: we have developed a data structure which is easy to implement and easy to use.

These facts also explain the name *Visibility Skeleton*, since our new structure can be thought of as the skeleton of the complete Visibility Complex.

2.3 Catalogue of Visual Events and their Adjacencies

The Visibility Skeleton is a graph structure. The nodes of the graph are the extremal stabbing lines and the arcs correspond to line swaths. In this section (and in Appendix 7.1) we present an exhaustive list of all possible types of arcs and nodes of the Visibility Skeleton.

2.3.1 1D Elements: Arcs of the Visibility Skeleton

In Figure 4, we see the four possible types of 1D elements: an EV line swath (shown in blue), an EEE line swath (shown in purple) and two line swaths relating a polygonal face (F) to one of its vertices (Fv) or an edge of another polygon (FE) (both are shown in blue). In the upper part of the figure we show the view (with changes in visibility), as seen from a viewpoint located above the scene and, from left to right in front of, on, or behind the line swath.

Note that the interaction of an edge e and a vertex v can correspond to many ve arcs of the skeleton. These arcs are separated by nodes. Consider, for example, arcs ve_1 and ve_1' adjacent to node ve_1e_2 in Fig. 3(a).

2.3.2 0D Elements: Nodes of the Visibility Skeleton

As explained in Section 2.1, two line swaths which meet define an extremal stabbing line, which in the Visibility Skeleton is the node at which the arcs meet. This section presents a list of the configurations creating nodes and their corresponding adjacencies. A figure is given in each case.

The simplest node corresponds to the interaction of two vertices shown in Figure 5(a).

The interaction of a vertex v and two edges e_1 and e_2 can result in two configurations, depending on the relative position of the vertex with respect to the edges. The first node was presented previously in Figure 3 and the second is shown in Figure 5(b).

The interaction of four edges is presented in Figure 6, together with the six corresponding adjacent EEE arcs. Face related nodes are given in detail in the appendix: EFE , FEE , FF , E and Fvv (see Fig. 18 to 19).

3 DATA STRUCTURE AND CONSTRUCTION ALGORITHM

Given the catalogues of nodes and arcs presented in the previous section, we can present the details of a suitable data structure to

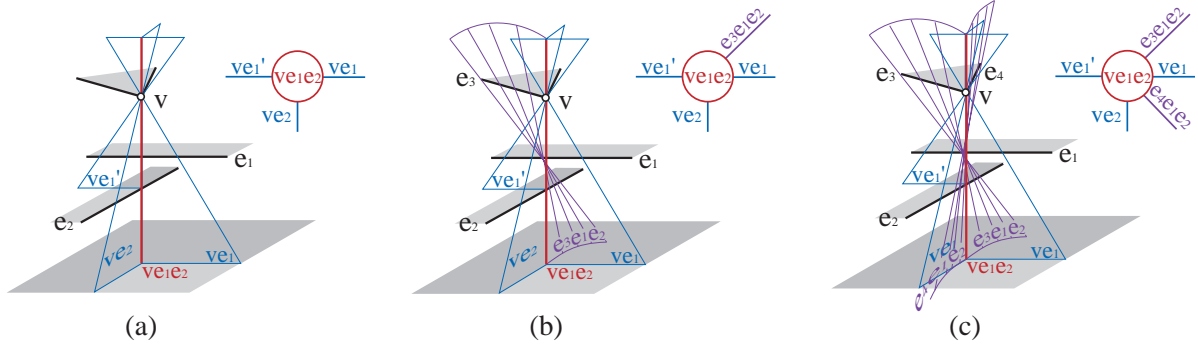


Figure 3: (a) An additional *EV* line swath is adjacent to the extremal stabbing line, (b) (c) and two *EEE* line swaths

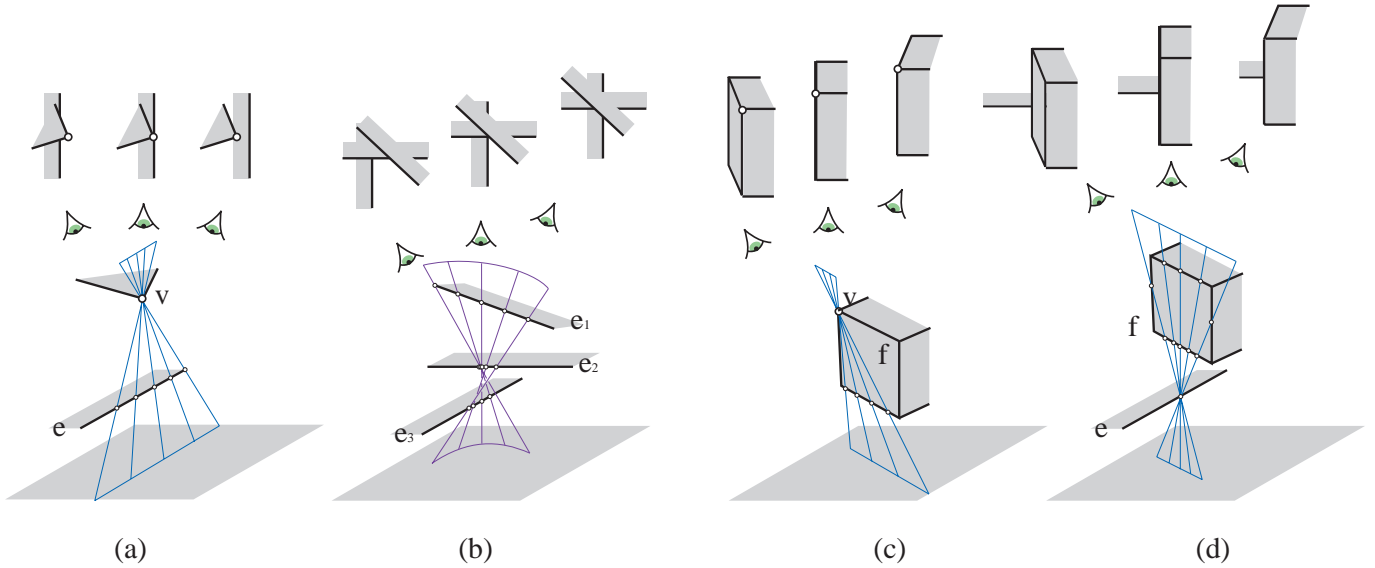


Figure 4: (a) Same as Fig. 1(a). (b) In front of the *EEE* line swath the edge e_2 is visible, on the swath the edges meet at a point and behind e_2 is hidden. (c) In front of the *FV* we see the front side of F , on the swath we see a line and behind we see the other side of F . (d) The *FE* swath is similar to the *FV* case.

represent the Visibility Skeleton graph structure, as well as the algorithm to construct it.

Preliminaries: Our scene model provides the adjacencies between vertices, edges and faces. Before processing the scene, we traverse all vertices, edges and faces, and assign a unique number to each. This allows us to index these elements easily. In addition, we consider all edges to be uniquely oriented. This operation is arbitrary (i.e., the orientation does not depend on the normal of one of the two faces attached to the edge), and facilitates consistency in the calculations we will be performing.

3.1 Data Structure

The simplest element of the structure is the node. The *Node* structure contains a list of arcs, and pointers to the polygonal faces F_{up} and F_{down} (possibly void) which block the corresponding extremal stabbing line at its endpoints P_{up} and P_{down} .

The structure for an *Arc* is visualized in the Fig.7(a). The arc represented here (swath shown in blue) is an *EV* line set. There are two adjacent nodes N_{start} , N_{end} , represented as red lines. All the adjacency information is stored with the arc. Details of the structures *Node* and *Arc* are given in Fig. 7(b).

To access the arc and node information, we maintain arrays of

balanced binary search trees corresponding to the different type of swaths considered. For example, we maintain an array ev of trees of *EV* arcs (see Fig.7(b)). These arrays are indexed by the unique identifiers of the endpoints of the arcs. These can be faces, vertices or edges (if the swath is interior, that is if the lines traverse the polyhedron).

This array structure allows us to efficiently query the arc information when inserting new nodes and when performing visibility queries. The balanced binary search tree used to implement the query structure is ordered by the identifiers of the generators and by the value of t_{start} .

3.2 Finding Nodes

Before presenting the actual construction of each type of node, we briefly discuss the issue of “local visibility”. As has been presented in other work (e.g., [10]), for any edge adjacent to two faces of a polyhedron, the negative half-space of a polygonal face is locally invisible. Thus when considering interactions of an edge e , we do not need to process any other edge e' which is “behind” the faces adjacent to e . This results in the culling of a large number of potential events.

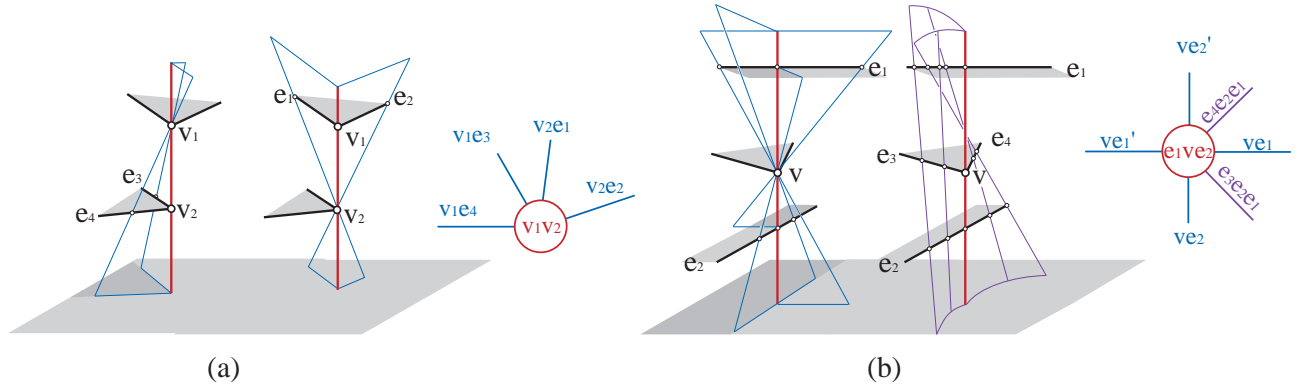


Figure 5: (a) A VV node v_1v_2 is adjacent to four EV arcs defined by a vertex and an edges of the other vertex: v_1e_3 and v_1e_4 and e_1v_2 and e_2v_2 . (b) An EVE node e_1ve_2 : each edge defines two EV arcs with v depending on the polygon at the extremity, and to two EEE defined by e_1 , e_2 and the two edges adjacent to v .

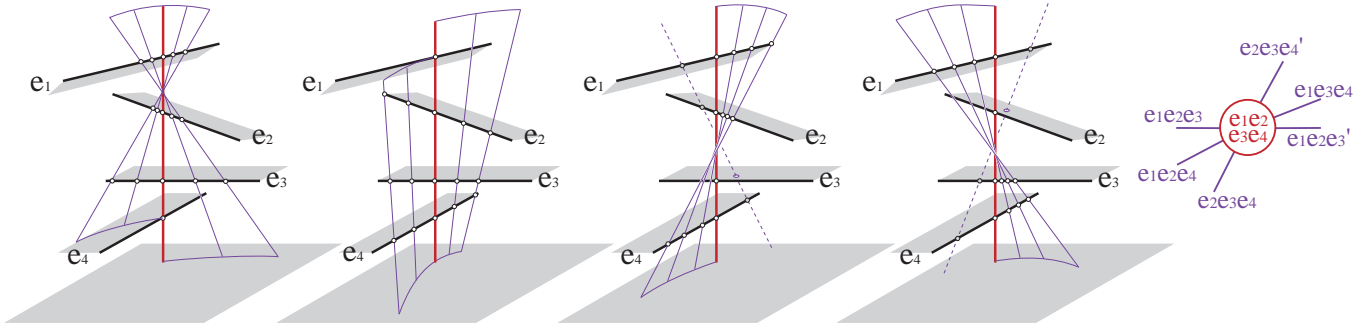
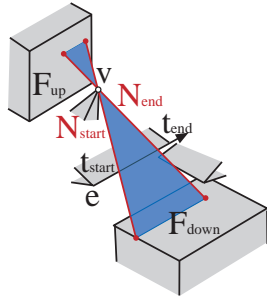


Figure 6: An $E4$ node is adjacent to six EEE arcs.



```

class Node {
    List<Arcs> adjacentArcs
    Face  $F_{up}$ ,  $F_{down}$ 
    Point3D  $P_{up}$ ,  $P_{down}$ 
}

class Arc {
    Node  $N_{start}$ ,  $N_{end}$ 
    float  $t_{start}$ ,  $t_{end}$ 
    Face  $F_{up}$ ,  $F_{down}$ 
}

class EV : child of Arc {
    Edge  $e$ 
    Vertex  $v$ 
}

class VisibilitySkeleton {
    tree<EV>  $ev[F_{up}][F_{down}]$ 
    tree<EEE>  $eee[F_{up}][F_{down}]$ 
    ...
}
    
```

(a)

(b)

Figure 7: Basic Visibility Skeleton Structure.

3.2.1 Trivial Nodes

The simplest nodes are the VV , Fvv and Fe nodes. For these, we simply loop over the appropriate scene elements (vertices, edges and faces). The appropriate lines are then intersected with the scene using a traditional ray-caster to determine if there is an occluding object between the related scene elements, in which case no extremal stabbing line is reported. Otherwise it gives the elements

and points at the extremities of the lines, and thus the appropriate location in the overall arc tree array.

3.2.2 VEE and EEEE Nodes

We consider two edges of the scene e_i and e_j . All the lines going through two segments are within an extended tetrahedron (or double wedge) shown in Fig. 8, defined by four planes. Each one of these planes is defined by one of the edges and an endpoint of the other.

To determine the vertices of the scene which can potentially generate a VEE or EVE stabbing line, we need only consider vertices within the wedge. If a vertex of the scene is inside the double wedge, there is a potential VEE or EVE event.

We next consider a third edge e_k of the scene. If e_k cuts a plane of the wedge, a VEE or EVE node is created. If edge e_k of the scene intersects the plane of the double wedge defined by edge e_i and vertex v of e_j , there is a ve_ie_k or $e_i ve_k$ event (Fig. 8(a)).

We next proceed to the definition of the $E4$ nodes. The intersections of e_k and the planes of the double wedge *restrict* the third edge e_k . To compute a line going through e_i , e_j , e_k we need only consider the restriction of e_k to the double wedge defined by e_i and e_j . This process is re-applied to restrict a fourth edge e_l by the wedge of e_i and e_j , by that of e_i and e_k and by that of e_j and e_k . This multiple restriction process eliminates a large number of candidates.

Once the restriction is completed, we have two EEE line sets, those passing through e_l , e_i and e_j and those passing through e_l , e_i and e_k . A simple binary search is applied to find the point on e_l (if it exists) which defines the $E4$ node. We perform this search for a point P of e_l by searching for the root of the angle formed by

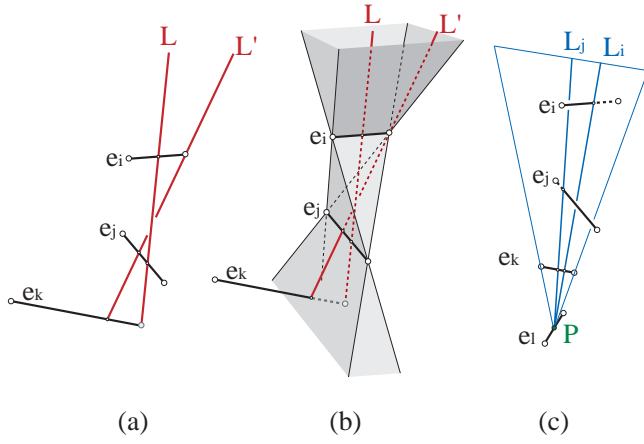


Figure 8: (a) (b) VEE enumeration and EEE restriction. (c) E4 computation: find the root of the angle of the lines going through $e_j e_k e_l$ and that through $e_l e_k e_i$.

the two lines defined by the intersection of the plane (P, e_i) with e_j and with e_k . This is shown on Fig.8(c).

A more robust algorithm such as the one given in [28] could be used, but the simpler algorithm presented here seems to perform well in practice. This is true mainly because we are not searching for infinite stabbing lines, but for restricted edge line segments. The potential VEE and E4 enumeration algorithm is given in Fig. 9.

We have developed an acceleration scheme to avoid the enumeration of all the triples of edges. For each pair of edges, we reject very quickly most of the third potential edges using a regular grid. Instead of checking if each cell of the grid intersects the extended tetrahedron, we use the projection on the three axis-aligned planes. For each such plane, we project the extended tetrahedron (which gives us an hourglass shape), and we perform the actual edge-tetrahedron intersection only for the edges contained in the cells whose three projections intersect the three pixelized hourglasses.

3.2.3 Non-Trivial Face Nodes

To calculate the non-trivial face-related nodes, we start by intersecting the plane of each face f_1 with every edge of the scene. For edges intersecting the face we attempt to create an FvE node (Fig.18).

For each pair of intersections, we search for a FEF node. To do this we determine if the line joining the two intersections intersects the face f_1 . The last operation required is the verification of the existence of an FF node. This case occurs if the faces adjacent to the edge of the intersection cause an FF . The construction for the FEF and FF nodes is described in Fig. 10 (a).

3.3 Creating the Arcs

The creation of the arcs of the Visibility Skeleton is performed simultaneously with the detection of the nodes. When inserting a new node, we create all the adjacent arcs from the corresponding catalogue presented in Section 2.3.2. For each of these arcs a we calculate the arc parameter t corresponding to the node to be inserted, and proceed as explained in Fig.12. We then access the list of arcs in the Skeleton with the same extremities (thus in the same list of the array) and which have the same generator elements (*vertices and edges*) as the arc a . If the value of t indicates that the node is contained in the arc, we determine whether this node is the start of the end node of the arc. This is explained in more detail in the following paragraph. If this position is already occupied we split

the arc, else we assign the node the corresponding extremity of the arc. This process is summarized in Fig. 11.

We have seen above that each time an arc adjacent to a node is considered, we have to know if it is its *start node* or its *end node*. In some cases this operation is trivial, for example for a $v_1 v_2$ node and one if its adjacent $v_1 e$ arcs, we simply determine if v_2 is the starting vertex of e . In other cases, this can be more involved, especially for the E4 case. This case and the necessary criteria for the other cases are summarized in Table 2 in the Appendix.

In Fig. 12, we illustrate the construction algorithm. Initially a trivial vv_e node is created. The second node identified is vfe , which is adjacent the arc ve . Thus the arc ve is adjacent to both vv_e and vfe . The third node to be created is vee_3 . When this node is inserted, we realize that the start node for ve already exists, and we thus split the ve arc. This splitting operation will leave the end of the ve arc connected to vv_e undefined. The final insertion shown is $ve_2 e$ which will fill an undefined node previously generated.

4 IMPLEMENTATION AND FIRST APPLICATIONS

We have completed a first implementation of the data structure described. We have run the system on a set of test scenes, with varying visibility properties. In its current form, we have successfully computed the Visibility Skeleton for scenes up to 1500 polygons.

In what follows we first present Visibility Skeleton construction statistics for the different test scenes used. We then proceed to demonstrate the flexible nature of our construction, by presenting the use of our data structure to efficiently answer several different global visibility queries.

4.1 Implementation and Construction Statistics

Our current implementation requires convex polyhedra as input. However, this is not a limitation of the approach since we use polyhedral adjacencies simply for convenience when performing local visibility tests.

We treat touching objects by detecting this occurrence and slightly modifying the ray-casting operation. We also reject coplanar edge triples. Other degeneracies such as intersecting edges are not yet treated by the current implementation.

We present statistics on the size of our structure and construction time in Table 1. Evidently, these tests can only be taken as an indication of the asymptotic behavior of our algorithm. As such, we see that our test suite indicates quadratic growth of the memory requirements and super-quadratic growth of the running time. In particular, for the test suite used, the running time increases with $n^{2.4}$ on average, where n is the number of polygons.

The VEE nodes are the most numerous. There are approximately a hundred times fewer E4 nodes, even though theoretically there should be an order of magnitude more.

We believe that the memory requirements could be greatly decreased by an improved implementation of the arrays of trees. Currently, a large percentage of the memory required is used by these arrays (e.g. for scene (d) of Table 1., the arrays need 53.7Mb out of a total 135Mb). Since these arrays are very sparse (e.g. 99.3% empty for scene (d)), it is clear that storage requirements can be greatly reduced.

In the case of densely occluded scenes, the memory requirements grow at a slower rate, on average much closer to linear than quadratic with respect to the number of polygons. As an example, we replicated scene (a) 2, 4 and 8 times, thus resulting in isolated rooms containing a single chair each. The memory requirements (excluding the quadratic cost of the arrays) are 1.2Mb,

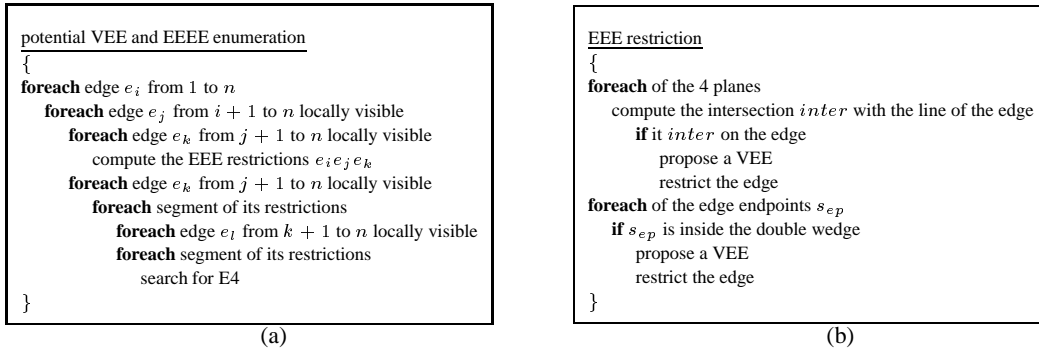


Figure 9: Enumeration of Potential VEE and E4 Nodes.

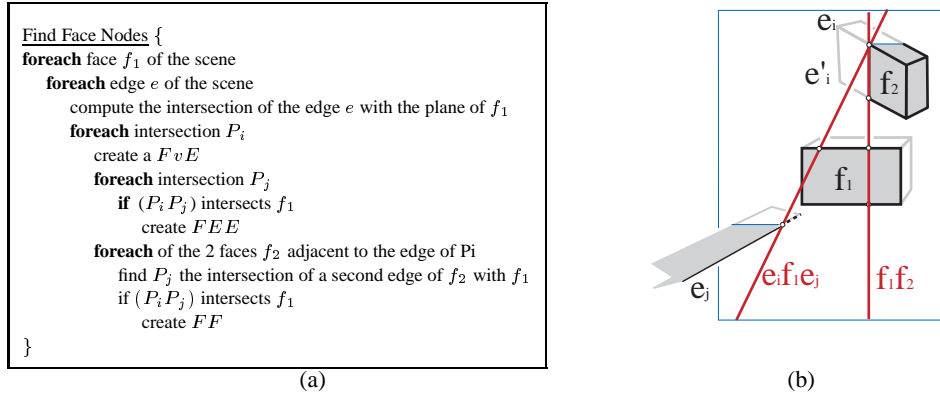


Figure 10: Finding Face Nodes.

2.8Mb, 8.6Mb and 17.3Mb, for respectively 78, 150, 300 and 600 polygons.

The theoretical upper bounds are very pessimistic, $O(n^4)$ in size because every edge quadruple can have two lines going through it [28], and $O(n^5)$ in time because such potential extremal stabbing lines have to be ray-cast with the whole scene. But such bounds occur only in uncommon worst case scenes such as grids aligned and rotated or infinite lines. It is clear that our construction algorithm would be very inefficient for such cases. More efficient construction algorithms are possible, but these approaches suffer from all the problems described previously in Section 2.2.

In what concerns the robustness of the computation, previous aspect graph and discontinuity meshing algorithms depend heavily on the construction of the arrangement (of the mesh or aspect graph “cells”), as the algorithm progresses. In the construction presented here, this is not the case since all operations are completely local. Since we perform ray-casting and line-plane intersections, the number of potential numerical problems is limited. Degeneracies can occasionally cause some problems, but due to the locality, this does not effect the construction of the Skeleton elsewhere. More efficient sweep-based algorithms are particularly sensitive to such instabilities, since an error in one position in space can render the rest of the construction completely incorrect and inconsistent.

4.2 Point-to-Area Form-Factor for Vertices

The calculation of point-to-area form factors has become central in many radiosity calculations. In most radiosity systems, point-to-area calculations are used to approximate area-to-area calculations [4, 2], and in others the actually point-to-area value is computed at the vertices [29].

In both theoretical [16] and experimental [6] studies, previous

research has shown that error of the visibility calculation is a predominant source of inaccuracies. This is typically the case when ray casting is used. Lischinski et al. [16] have developed a very promising approach to bounding the error committed during light transfer for hierarchical radiosity. For it to be useful for general environments, access is required to the exact visibility information between a point on one element with respect to the polygon face it is linked to. This information is inherently global, since a pair of linked elements can contain *any* two surface elements of the scene.

The Visibility Skeleton in its initial form can answer this query exactly and efficiently for the original vertices of the input scene.

To calculate the view of a polygonal face from a vertex v , with respect to a face f , we first access all the EV arcs of the skeleton related to the face f . This is simply the traversal of the line of our global two-dimensional array of arcs, indexed by f . For each entry of this list (many of which are empty), we search for the EV arcs related to v . These EV arcs are exactly the visible boundary of f seen from v .

An example is shown in Fig. 13(a) and (b) For scene (b), containing 312 and 1488 polygons, the extraction of the point-to-area boundary takes respectively 1.2 ms and 1.5 ms (all query time are given without displaying the result).

4.3 Global and On-The-Fly Discontinuity Meshing

In radiosity calculations, it is often very beneficial to subdivide the mesh of a surface by following some [14, 18], or all [6] of the discontinuity surfaces between two surfaces which exchange energy. The partial [14, 17] or complete [5, 24] construction of such meshes has in the past been restricted to the discontinuity mesh between a source (which is typically a small polygon) and the receivers (which are the larger polygons of the scene). For all other interactions be-

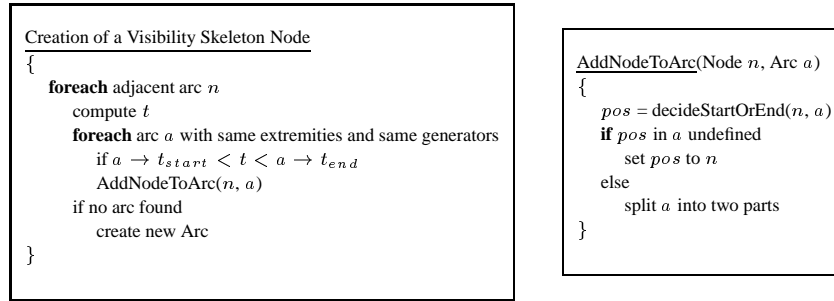
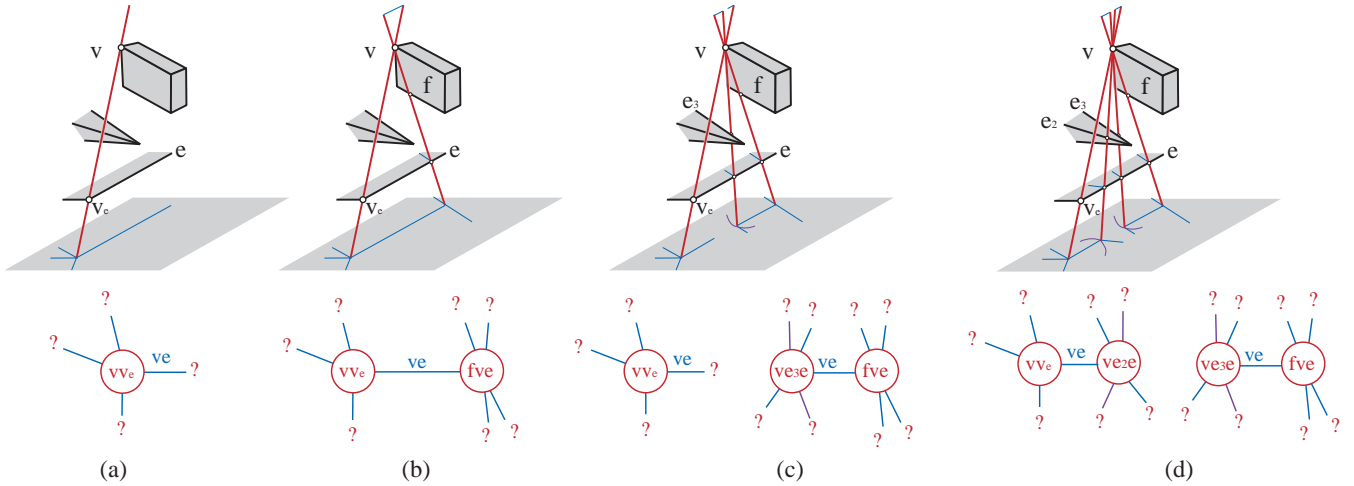


Figure 11: Node Creation

Figure 12: Example of node insertions: (a) Insertion node vv_e . (b) Insertion of node fve . Arc ve has now two ending nodes. (c) Insertion of node ve_3e . Arc ve is split. (d) Insertion of node ve_4e , the two arcs ve have their actual adjacent nodes.

tween surfaces of scenes, the algorithmic complexity and the inherent robustness problems related to the construction of these structures has not permitted their use [25].

For many secondary transfers in an environment, the construction of a global discontinuity mesh (i.e., from any surface (emitting/reflecting) to any other receiving surface in a scene), can aid in the accuracy of the global visibility computation. This was shown in the discontinuity driven subdivision used by Hardt and Teller [13]. In their case, the discontinuity surfaces are simply intersected with the scene polygons, and thus visibility on the line swath is not computed. With the Visibility Skeleton, the complete global discontinuity mesh between two surfaces can be efficiently computed.

To efficiently perform this query, we add an additional two-dimensional array $DM(i, j)$, storing all the arcs from face f_i to f_j . Insertion into this array of lists and well as subsequent access is performed in constant time. To extract the discontinuity mesh between to surfaces f_i and f_j we simply access the entry $DM(i, j)$, and traverse the corresponding list. In Fig. 14(a), the complete discontinuity mesh between the source and the floor is extracted in 28.6 ms. The mesh caused by the small lamp on the table in Fig 14(b) was extracted in 1.3 ms (note that the arrangement is not built).

The resulting information is a set of arcs. These arcs can be used as in Hardt and Teller to guide subdivision, or to construct the arrangement of the discontinuity mesh on-the-fly, to be used as in [6] for the construction of a subdivision which follows the discontinuities. The adjacency information available in the Skeleton arcs and nodes should permit a robust construction of the mesh arrangement.

4.4 Exact Blocker Lists, Occlusion Detection and Efficient Initial Linking

When considering the interaction between two surfaces, it is often the case that we wish to have access to the exact list of blocker surfaces hiding one surface from the other. This is useful in the context of blocker list maintenance approaches such as that presented by Teller and Hanrahan [27].

The Visibility Skeleton can again answer this query exactly and efficiently. In particular, we use the global array $DM(i, j)$, and we traverse the related arcs. All the polygons related to the intervening arcs are blockers. It is important to note that this solution results in the *exact* blocker list, in contrast with all previous methods. Consider the example shown in Fig. 13(c) where we compute the occluders between the left ceiling lamp and the floor in 4 ms.

The *shaft* structure [11] would report all objects on the table though they are hidden by the table. In this case the Visibility Skeleton reports the exact set of blockers.

When constructing the Visibility Skeleton, we compute all the mutually visible objects of the scene: if two object see each other, there will be at least one extremal stabbing line which touches them or their edges and vertices. This is fundamental for hierarchical radiosity algorithms since it avoids the consideration of the interaction of mutually visible objects in the initial linking stage.

Similarly, the Skeleton allows for the detection of the occlusions caused by an object. This can be very useful for the case of a moving object m allowing the detection of the form factors to be recomputed. To detect if the form factor F_{ij} has to be recomputed we perform a query similar to the discontinuity mesh between two polygons: we traverse $DM(i, j)$ and search for an arc caused by



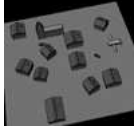




	a	b	c	d	e	f	g
Scene							
Polygons	84	168	312	432	756	1056	1488
Nodes ($\times 10^3$)	7	37	69	199	445	753	1266
Arcs ($\times 10^3$)	16	91	165	476	1074	1836	3087
Construction	1 s 71 s	12 s 74	37 s 07	1 min 39 s	5 min 36 s	14 min 36 s	31 min 59
Memory (Mb)	1.8	9	21	55	135	242	416

Table 1: Construction statistics (all times on a 195Mhz R10000 SGI Onyx 2). Storage is scene dependent and can be greatly reduced.

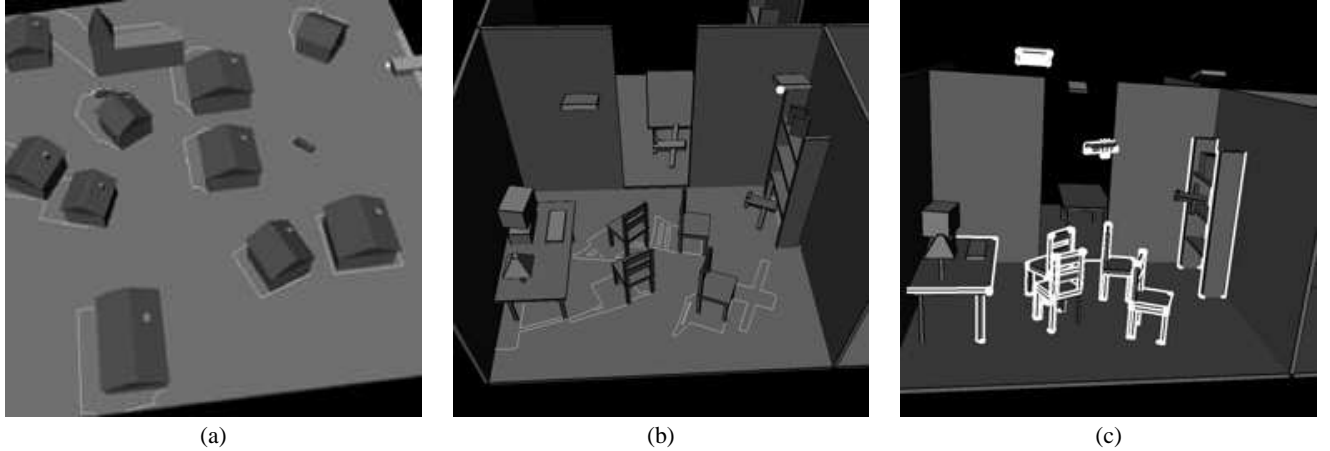


Figure 13: (a) Part of the scene visible from a vertex of the airplane. (b) Part of the floor seen by a vertex of the right-hand light source. (c) List of occluding blockers between the left light source and the floor. Note that the objects on the table that are invisible from the floor are not reported as blockers.

an element (vertex, edge or face) of m . This gives us the limits of occlusions of m between f_i and f_j . Moreover, by considering all the arcs of the skeleton, we report all the form factors to be recomputed, and not a superset. Fig 14(c) shows the occlusions caused by the body of the plane between the screen and the right wall. This computation required 1.3 ms.

5 DEALING WITH SPATIAL COMPLEXITY: ON-DEMAND CONSTRUCTION

We propose here an on-demand or *lazy* scheme to compute visibility information only where and when needed. For example, if we want the discontinuity mesh between two surfaces, we just need to compute the arcs of the complex related to these two faces, and for this we only need to detect the nodes between these two faces.

The key for this approach is the locality of the Visibility Skeleton construction algorithm. We only compute the nodes of the complex where needed. The fact that some arcs might have missing nodes causes no problem since no queries will be made on them. Later on, other queries can appropriately link the missing nodes with those arcs.

Two problems must be solved: determination of what is to be computed, and determination of what has already been computed.

We propose two approaches: a source driven computation, and an adaptive subdivision of ray-space in the spirit of [1].

In the context of global illumination, the information related to “sources” (emitters or reflectors) is crucial. Thus the part of the visibility skeleton we compute in an on-demand construction is related

to lines cutting the sources. The event detection has to be modified: every time a double wedge or a face does not cut the source, the pair of edges or the face is discarded, and if a potential node is detected, the ray-casting is performed only if the corresponding critical line cuts the source.

We use our grid-acceleration scheme here too: for each first edge, an edge pair is formed only for the edges that lie inside the hourglass defined by the source and the first edge.

When considering many sources one after the other, we also have to detect nodes already computed. If the sources are small, it is not worth rejecting double wedges, and only the final ray-casting and node insertion can be avoided (in our implementation they account for a third of the running time). We can perform a “final computation” if we want all the nodes that have not yet been computed: we just test before ray-casting if the critical line cuts one of the sources.

For scene (g) of Table 1, the part of the Visibility Skeleton with respect to one of the sources is computed in 4 min. 15 s. instead of 31 min. 59 s. for the entire scene.

When the number of sources becomes large, most of the time would be spent in checking if lines intersect the sources or if they have already been subdivided. If we need visibility information only between two objects, not between an object and the whole scene, we propose the use of ray classification of [1] together with the notions of dual space of [7] to build the visibility skeleton only where and when needed. The idea (which is not currently implemented) is to parameterize the lines of the 3D space (which is a set in 4D space), for example by their direction and projection on a plane or by their intersections with two parallel planes. We then perform a subdivision of the space of lines with a simple scheme

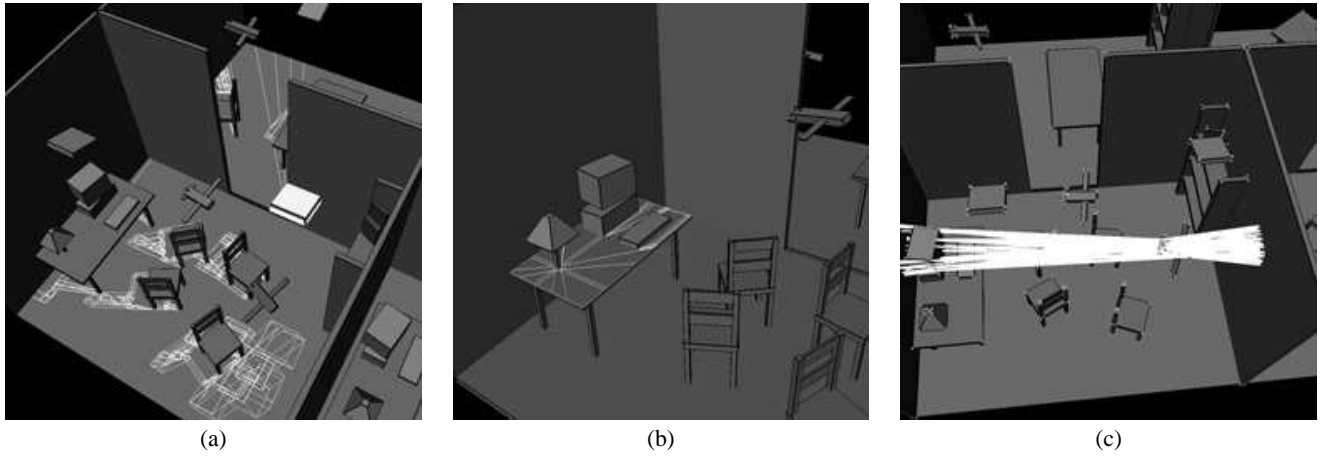


Figure 14: (a) The complete discontinuity mesh with respect to the right source. (b) Discontinuity mesh between the lamp and the table. (c) Limits of the occlusions caused by a part of the plane between the computer screen and the right wall.

(e.g., grid, hierarchical subdivision) and compute the nodes of the complex located inside a given cell of this subdivision.

6 CONCLUSIONS AND FUTURE WORK

We have presented a new data structure, called the *Visibility Skeleton*, which encodes all global visibility information for polygonal scenes. The data structure is a graph, whose nodes are the extremal stabbing lines generated by the interaction of edges and vertices in the scene. These lines can be found using standard computer graphics algorithms, notably ray-casting and line-plane intersections. The arcs of the graph are critical line sets or swaths which are adjacent to nodes. The key idea for simplicity was to treat the nodes and deduce the arcs using the full catalogues of all possible nodes and adjacent arcs we have presented for polygonal scenes. A full construction algorithm was then given, detailing insertion of nodes and arcs into the Skeleton.

We presented an implementation of the construction algorithm and several applications. In particular, we have used the Skeleton to calculate the visible boundary of a polygonal face with respect to a scene vertex, the discontinuity mesh between any two polygons of the scene, the exact list of blockers between any two polygons, as well as the complete list of all interactions of a polygon with all other polygons of the scene.

The implementation shows that despite unfavorable asymptotic complexity bounds, the algorithm is manageable for the test suite used, both in storage and in computation time. In addition, we have developed and implemented a first approach to on-demand or lazy construction which opens the way to hierarchical and progressive construction techniques for the Skeleton.

The use of our implemented system shows the great wealth of information provided by the Visibility Skeleton. Only a few of the many potential applications were presented here, and we believe that there are many computer graphics (and potentially computer vision) domains which can exploit the capacities of the Skeleton.

In future work many issues remain to be investigated. From a theoretical point of view, the most challenging problems are the development of a hierarchical approach so that the Visibility Skeleton can be used for very complex scenes as well as the resolution of all theoretical issues for the treatment of dynamic scenes. Some of the problems for the dynamic solution are sketched in Fig 15. Adapting the algorithm to curved objects requires the enumeration of all relevant events and definitely has many applications.

Finally the field of applications must be extended: exact point to

area form-factor from any point on a face, aspect graph construction, and incorporation into a global illumination algorithm.

Acknowledgements

We would like to thank Jean-Dominique Gascuel for his AVL-tree code and Seth Teller for the very fruitful discussions we had and for all the suggestions he gave on conservative, lazy and practical approaches.

7 Appendix

7.1 Complete Catalogue of FACE Adjacencies

Face related events are adjacent to FE elements Fv elements as well as EEE arcs when two non-coplanar edges are involved.

The interaction of a face with two edges is shown in Fig. 16, the interaction of a face a vertex and an edge is shown in Fig. 18 and finally the interaction of two faces is shown in Fig. 17.

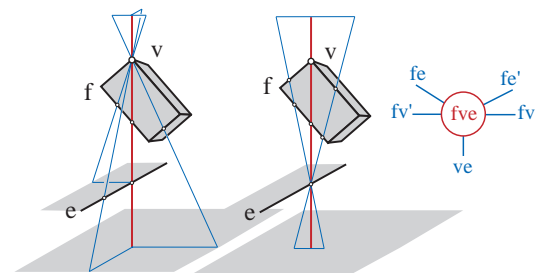


Figure 18: A FvE node.

7.2 Details of the Construction to find the Orientation of Arcs

Finding the correct extremity of an arc when inserting a node is crucial for the construction algorithm to function correctly. We present here the most complex case, which is the insertion of an $E4$ node.

Consider the node $e_1e_2e_3e_4$ shown in Fig. 19, and the adjacent arc $e_1e_2e_3$. The question that needs to be answered is whether the

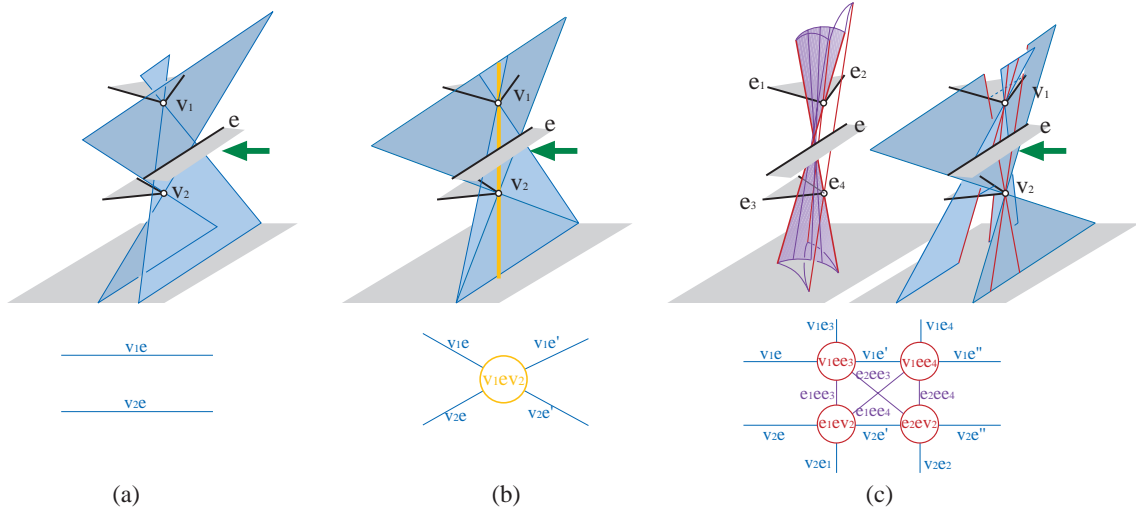


Figure 15: The edge moving from right to left causes a VEV temporal visibility event which is the meeting of two EV with the the two same extremities and with a common element (here the edge e). Four nodes are created, the EV arcs are split into three parts and eight arcs are created. These events and the topological visibility changes are local in the visibility skeleton.

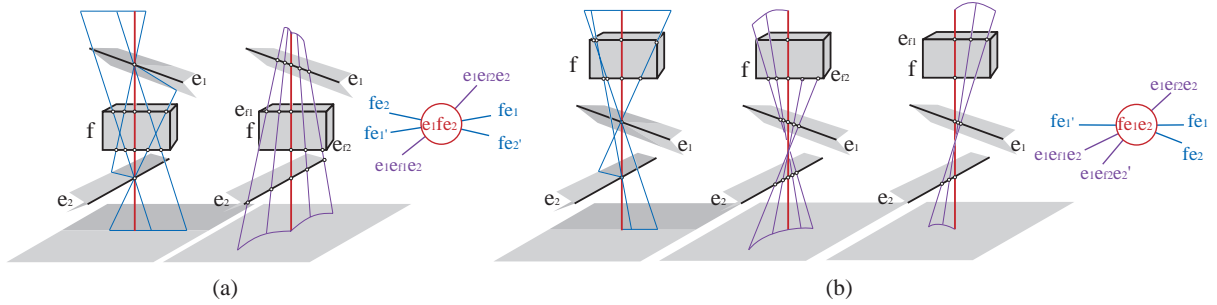


Figure 16: An EFE node.

node $e_1e_2e_3e_4$ is the start or the end node of this arc. To answer this query, we examine the movement of the line l going through e_1 , e_2 and e_3 , when moving on e_1 . The side of e_4 to which we move will determine whether we are a start or an end node.

Consider the infinitesimal motion $d\vec{e}_1$ on e_1 . The corresponding point of e_3 on the EEE will lie on the intersection of the plane defined by e_2 and the defining point on e_1 . The motion of $d\vec{e}_1$ on e_1 corresponds to a rotation of $\alpha = \frac{\vec{e}_1 \cdot \vec{n}}{d_1}$ of the plane around e_2 . Symmetrically, this rotation corresponds to the motion $d\vec{e}_3$ on e_3 and we have $\alpha = \frac{d\vec{e}_3 \cdot \vec{n}}{d_3}$, by angle equality. Thus, $d\vec{e}_3 = \vec{e}_3 \frac{d_3 d\vec{e}_1 \cdot \vec{n}}{d_1 e_3 \cdot \vec{n}}$.

Now we want to obtain $d\vec{e}_4$, the infinitesimal motion of the line going through the three edges around e_4 . We consider the line as being defined by its origin on e_1 and by its unnormalized direction vector \vec{dir} from e_1 to e_3 . For the motion $d\vec{e}_1$ of the origin, the direction vector of moves by $d\vec{e}_3 - d\vec{e}_1$, and thus $d\vec{e}_4 = d\vec{e}_1 + \frac{d_4}{d_3 - d_1} (d\vec{e}_3 - d\vec{e}_1)$.

The sign of $(\vec{e}_4 \times \vec{e}_4) \cdot \vec{node}$ determines on which side of e_4 the line l will move.

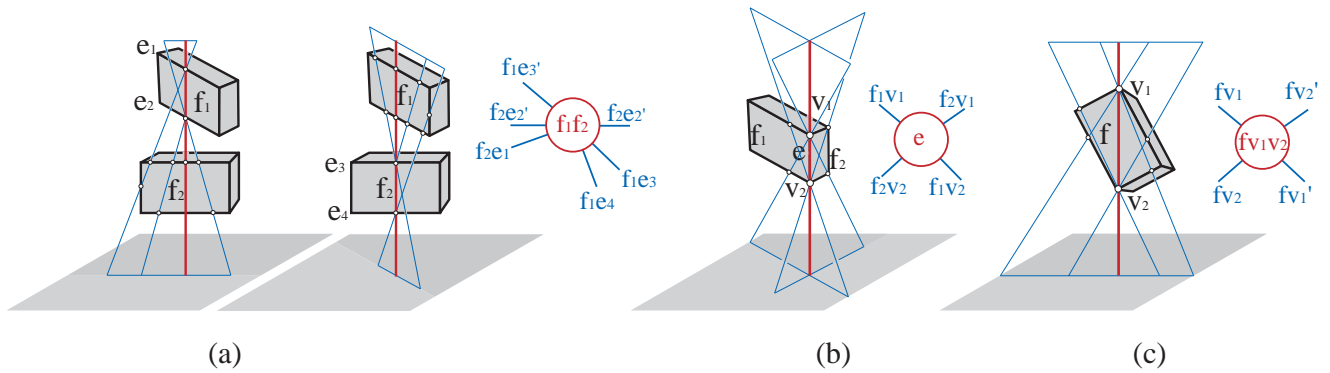
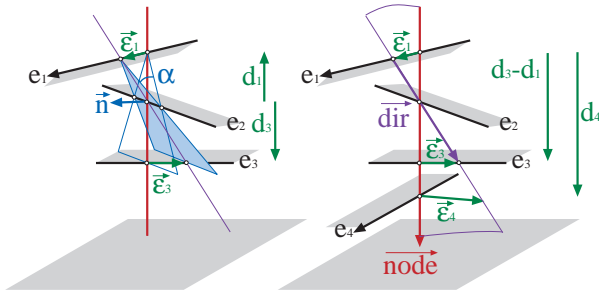
The adjacencies also depend on the face related to the edges which are visible from the other edges. The other cases are simpler and summarized in Table 2.

Node	Adjacent arc	Start or End Criterion
v_1v_2	v_1e_3	$v_1 == startV(e)$
ve_1e_2	ve_2 $e_3e_1e_2$ $e_2e_1e_3$	$(\vec{e}_2 \times \vec{e}_1) \cdot \vec{node} > 0$ $v == startV(e_3)$ $\vec{n} = \vec{normal}(v, \vec{e}_2)$ $\vec{e}_3 \cdot \vec{n} * \vec{e}_1 \cdot \vec{n} > 0$
e_1ve_2	ve_2 $e_2e_1e_3$	$(\vec{e}_1 \times \vec{e}_2) \cdot \vec{node} > 0$ $\vec{n} = \vec{normal}(v, \vec{e}_2)$ $\vec{e}_3 \cdot \vec{n} * \vec{e}_2 \cdot \vec{n} > 0$
$e_1e_2e_3e_4$	$e_1e_2e_3$	$\vec{n} = \vec{normal}(\vec{e}_2, \vec{node})$; $\vec{e}_3 = \vec{e}_3 \frac{d_3 \vec{e}_1 \cdot \vec{n}}{d_1 \vec{e}_3 \cdot \vec{n}}$ $\vec{e}_4 = \vec{e}_1 + \frac{d_4}{d_3 - d_1} (\vec{e}_3 - \vec{e}_1)$ $(\vec{e}_4 \times \vec{e}_4) \cdot \vec{node} > 0$
e_1fe_2	fe_1 $e_1e_{f1}e_2$	$\vec{e}_2 \cdot \vec{normal}(f) > 0$ $\vec{e}_1 \cdot \vec{normal}(f) > 0$
fe_1e_2	fe_2 $e_2e_1e_{f1}$	$\vec{e}_1 \cdot \vec{normal}(f) > 0$ $\vec{n} = \vec{normal}(\vec{node}, \vec{e}_1)$ $\vec{n} \cdot \vec{e}_2 * \vec{n} \cdot \vec{e}_{f1} > 0$
fve	fv ve	$\vec{e} \cdot \vec{normal}(f) > 0$ $\vec{e} \cdot \vec{normal}(f) > 0$

Table 2: for each arc adjacent to a created node, there is a criterion that tells if it is a start node or an ending node.

References

- [1] James Arvo and David B. Kirk. Fast ray tracing by ray classification. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, vol-

Figure 17: (a) A FF node, (b) an Fe node and (c) and Fvv node.Figure 19: Determining the direction of an $E4$ node insertion.

- ume 21, pages 55–64, July 1987.
- [2] Daniel R. Baum, Holly E. Rushmeier, and James M. Winget. Improving radiosity solutions through the use of analytically determined form-factors. *Computer Graphics*, 23(3):325–334, July 1989. Proceedings SIGGRAPH '89 in Boston, USA.
 - [3] Daniel R. Baum, John R. Wallace, Michael F. Cohen, and Donald P. Greenberg. The back-buffer algorithm : an extension of the radiosity method to dynamic environments. *The Visual Computer*, 2:298–306, 1986.
 - [4] Michael F. Cohen and Donald P. Greenberg. The hemi-cube : A radiosity solution for complex environments. *Computer Graphics*, 19(3):31–40, July 1985. Proceedings SIGGRAPH '85 in San Francisco (USA).
 - [5] George Drettakis and Eugene Fiume. A fast shadow algorithm for area light sources using back projection. In Andrew Glassner, editor, *SIGGRAPH 94 Conference Proceedings (Orlando, FL)*, Annual Conference Series, pages 223–230. ACM SIGGRAPH, July 1994.
 - [6] George Drettakis and Francois Sillion. Accurate visibility and meshing calculations for hierarchical radiosity. In X. Pueyo and P. Schröder, editors, *Rendering Techniques '96*, pages 269–279. Springer Verlag, June 1996. Proc. 7th EG Workshop on Rendering in Porto.
 - [7] Frédo Durand, George Drettakis, and Claude Puech. The 3d visibility complex, a new approach to the problems of accurate visibility. In X. Pueyo and P. Schröder, editors, *Rendering Techniques '96*, pages 245–257. Springer Verlag, June 1996. Proc. 7th EG Workshop on Rendering in Porto.
 - [8] David W. George, François Sillion, and Donald P. Greenberg. Radiosity redistribution for dynamic environments. *IEEE Computer Graphics and Applications*, 10(4), July 1990.
 - [9] Ziv Gigus, John Canny, and Raimund Seidel. Efficiently computing and representing aspect graphs of polyhedral objects. *IEEE Trans. on Pat. Matching & Mach. Intelligence*, 13(6), June 1991.
 - [10] Ziv Gigus and Jitendra Malik. Computing the aspect graph for the line drawings of polyhedral objects. *IEEE Trans. on Pat. Matching & Mach. Intelligence*, 12(2), February 1990.
 - [11] Eric A. Haines. Shaft culling for efficient ray-traced radiosity. In Brunet and Jansen, editors, *Photorealistic Rendering in Comp. Graphics*, pages 122–138. Springer Verlag, 1993. Proc. 2nd EG Workshop on Rendering (Barcelona, 1991).
 - [12] Pat Hanrahan, David Saltzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. *Computer Graphics*, 25(4):197–206, August 1991. SIGGRAPH '91 Las Vegas.
 - [13] Stephen Hardt and Seth Teller. High-fidelity radiosity rendering at interactive rates. In X. Pueyo and P. Schröder, editors, *Rendering Techniques '96*. Springer Verlag, June 1996. Proc. 7th EG Workshop on Rendering in Porto.
 - [14] Paul Heckbert. Discontinuity meshing for radiosity. *Third Eurographics Workshop on Rendering*, pages 203–226, May 1992.
 - [15] Michael Mc Kenna and Joseph O'Rourke. Arrangements of lines in space: A data structure with applications. In *Proc. 4th Annu. ACM Sympos. Comput. Geom.*, pages 371–380, 1988.
 - [16] Dani Lischinski, Brian Smits, and Donald P. Greenberg. Bounds and error estimates for radiosity. In Andrew S. Glassner, editor, *SIGGRAPH 94 Conference Proceedings (Orlando, FL)*, Annual Conference Series, pages 67–74. ACM SIGGRAPH, July 1994.
 - [17] Dani Lischinski, Filippo Tampieri, and Donald P. Greenberg. Discontinuity meshing for accurate radiosity. *IEEE Computer Graphics and Applications*, 12(6):25–39, November 1992.
 - [18] Dani Lischinski, Filippo Tampieri, and Donald P. Greenberg. Combining hierarchical radiosity and discontinuity meshing. In Jim Kajiya, editor, *SIGGRAPH 93 Conference Proceedings (Anaheim, CA)*, Annual Conference Series, pages 199–208. ACM SIGGRAPH, August 1993.
 - [19] Rachel Orti, Stéphane Rivière, Frédo Durand, and Claude Puech. Radiosity for dynamic scenes in flatland with the visibility complex. In Jarek Rossignac and François Sillion, editors, *Computer Graphics Forum (Proc. of Eurographics '96)*, volume 16, pages 237–249. Poitiers, France, September 1996.
 - [20] M. Pellegrini. Stabbing and ray shooting in 3-dimensional space. In *Proc. 6th Annu. ACM Sympos. Comput. Geom.*, pages 177–186, 1990.
 - [21] H. Plantinga and C. R. Dyer. Visibility, occlusion, and the aspect graph. *Internat. J. Comput. Vision*, 5(2):137–160, 1990.
 - [22] M. Pocchiola and G. Vegter. The visibility complex. 1996. special issue devoted to ACM-SoCG'93.
 - [23] Erin Shaw. Hierarchical radiosity for dynamic environments. Master's thesis, Cornell University, Ithaca, NY, August 1994.
 - [24] A. James Stewart and Sherif Ghali. Fast computation of shadow boundaries using spatial coherence and backprojections. In Andrew Glassner, editor, *SIGGRAPH 94 Conference Proceedings (Orlando, FL)*, Annual Conference Series, pages 231–238. ACM SIGGRAPH, July 1994.
 - [25] Filippo Tampieri. *Discontinuity Meshing for Radiosity Image Synthesis*. PhD thesis, Department of Computer Science, Cornell University, Ithaca, New York, 1993. PhD Thesis.
 - [26] Seth J. Teller. Computing the antipenumbra of an area light source. *Computer Graphics*, 26(4):139–148, July 1992. Proc. SIGGRAPH '92 in Chicago.
 - [27] Seth J. Teller and Patrick M. Hanrahan. Global visibility algorithms for illumination computations. In J. Kajiya, editor, *SIGGRAPH 93 Conf. Proc. (Anaheim)*, Annual Conf. Series, pages 239–246. ACM SIGGRAPH, August 1993.
 - [28] Seth J. Teller and Michael E. Hohmeyer. Computing the lines piercing four lines. Technical report, CS Dpt. UC Berkeley, 1991.
 - [29] John R. Wallace, Kells A. Elmquist, and Eric A. Haines. A ray tracing algorithm for progressive radiosity. *Computer Graphics*, 23(3):315–324, July 1989. Proceedings SIGGRAPH '89 in Boston.

2.4.7 The 3D Visibility Complex : a unified data-structure for global visibility of scenes of polygons and smooth objects (CCCG'97)

Auteurs : Frédo Durand, George Drettakis et Claude Puech

Actes : 9th Canadian Conference on Computational Geometry

Date : août 1997

The 3D Visibility Complex: a unified data-structure for global visibility of scenes of polygons and smooth objects

Frédo Durand, George Drettakis and Claude Puech
iMAGIS-GRAVIR/INRIA *

Abstract

In this paper we describe a unified data-structure, the *3D Visibility Complex* which encodes the visibility information of a 3D scene of polygons and smooth convex objects. This data-structure is a partition of the maximal free segments and is based on the characterization of the topological changes of visibility along critical line sets. We show that the size k of the complex is $\Omega(n)$ and $O(n^4)$ and we give an output sensitive algorithm to build it in time $O((n^3 + k) \log n)$.

This theoretical work has already been used to define a practical data-structure, the *Visibility Skeleton* described in a companion paper.

1 Introduction

Visibility is a crucial issue; motion planning in robotics, object recognition in computer vision, lighting simulation or view maintenance in computer graphics are some examples where global visibility computations are required. The notion of "coherence" is often cited as the key to treat these problems efficiently and not restart every computation from scratch, but its characterization is not straightforward.

The usual space-subdivision methods do not translate the line nature of visibility, since a line of sight intersects many cells of any subdivision.

Computational geometers have characterized sets of lines in space by using Plücker duality. It is an oriented projective 5D dual space in which lines of space are naturally and linearly embedded (lines intersecting a given line are associated with hyperplanes). Its main drawback is the necessity of an intersection with the Plücker hypersurface [CEG⁺96, Pel90]. The scenes considered have always been polygonal and are mainly restricted to isothetic or c-oriented polygons. (In fact there exists a few results on ray-shooting with spheres involving parametric search without Plücker

coordinates [MS97]). These techniques have been used by Teller in computer graphics to compute the antipenumbra cast by an area light source through polygonal portals [Tel92]. The problem with these methods is that intersections of lines with the entire scene are considered; occlusion is not really treated.

In computer vision, the *aspect graph* has been developed to characterize the viewpoints from which the scene has the same topological aspect. The viewing space (S^2 for orthographic projection, R^3 for perspective projection) is partitioned along *visual events*. Construction algorithms have been developed for polygons and algebraic objects, both for orthographic and perspective projection, and some of them have been implemented; see [EBD92] for a good survey. A main drawback of aspect graphs is their size: $O(n^6)$ for orthographic projection, and $O(n^9)$ for perspective projection.

To build the aspect graph, Plantinga and Dyer [PD90] defined an intermediate data structure called the *asp*. For the orthographic case, it is a partition of the 4D space of oriented lines of space according to the first object they hit, and for the perspective case it is a partition of the 5D space of oriented half lines (rays). This approach has been limited to polygonal scenes. It was applied to maintain views, but the degrees of freedom allowed by the implementation were limited to rotation along a predefined axis [PDS90].

In lighting simulation, researchers have computed the discontinuities of the lighting function (which correspond to the limits of umbra and penumbra) also called *discontinuity meshes*. This characterizes the visibility of a light source. Initially only a subset of discontinuities were computed (e.g., [LTG93]), followed by algorithms computing all the discontinuities, together with a structure, the *back-projection*, which encodes the topological aspect of the light source [DF94, SG94]. These approaches are nonetheless restricted to a single light-source at a time.

Recently, a data-structure which encodes all the visibility information of a 2D scene called the *Visibility Complex* has been defined [PV96]. This structure is a partition of the set of maximal free segments according to the object they touch. Optimal construction algorithms have been developed for smooth convex objects [PV96] as well as polygons [Riv97] and used for lighting simulation [ORDP96].

* Laboratoire GRAVIR / IMAG. iMAGIS is a joint research project of CNRS/INRIA/INPG/UJF. Postal address: B.P. 53, F-38041 Grenoble Cedex 9, France. Contact E-mail: Frederic.Durand@imag.fr. <http://www-imagis.imag.fr>

In [DDP96] we introduced the *3D Visibility Complex* for scenes of convex smooth objects (the polygonal case was simply mentioned). An $O(n^4 \log n)$ brute-force algorithm was roughly sketched, and applications for lighting simulation, walkthroughs and aspect graph computation were proposed.

In this paper, we present a unified version of the 3D visibility complex for scenes of polygons and smooth convex objects. It is based on a complete catalogue of critical line sets which are lines where visibility changes. We derive bounds for the size of the complex and present an output sensitive construction algorithm.

Moreover, the formalism described in this article has been used to develop and implement a global visibility data-structure called the *Visibility Skeleton* [DDP97]. It is a simplified version of the 3D visibility complex for polygonal scenes built using a brute-force algorithm.

2 Scenes and maximal free segments

We consider scenes of polygons and algebraic smooth convex objects. Concave objects and piecewise smooth objects are beyond the scope of this article but could be handled by considering other critical line sets described by the theory of singularity [PPK92, Rie87]. The algebraic objects are assumed to have bounded degree. In what follows, n represents the overall complexity of the scene which is the total number of objects, polygons, edges and vertices. The objects are assumed to be in general position; degeneracy issues are not addressed in this paper.

In this work we do not consider lines but maximal free segments to take occlusion efficiently into account. Intuitively, a segment represents a class of rays, and we want to group the rays that “see” the same objects. Since many segments can be collinear, we need a fifth dimension to distinguish them. But it is not a continuous dimension: there is only a finite number of segments collinear to one line. See figure 1(a) where a 2d equivalent is shown. The segments a and b are collinear, t is tangent to the object and is adjacent to segments above and below the object. Topologically we have a branching structure represented in fig. 1 for parallel segments. Note that almost everywhere the graph is locally 1-dimensional. Similarly in 3D, the segment space is a 4D space embedded in 5D. This can be seen as a unification of the spaces used by Plantinga and Dyer [PD90]: in the orthographic case they deal with a 4D space and in the perspective case with a 5D space.

We use the same parameterization for lines as [PD90, DDP96]: they are represented by two coordinates of direction, the angles θ (azimuth) and φ (elevation) which are the spherical coordinates of the director vector, and the coordinates (u, v) of the projection onto the plane perpendicular to the line and going through the origin (the axes of the plane are chosen such as u is orthogonal to both the director vector and the vertical). See figure 1(b). Note that if φ

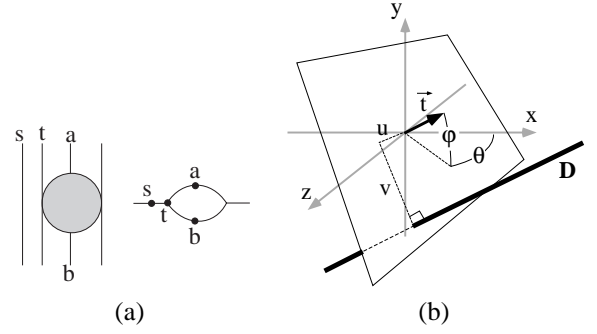


Figure 1: (a) 2D equivalent of the segment space: Parallel segments in the scene and local topology with branchings. (b) Parameterization of lines in space.

is fixed we obtain all the lines contained in a set of parallel planes. We call it a φ -slice. If we also fix v , we obtain the lines of a plane in which θ and u are the polar coordinates. We call it a φv -slice.

3 Critical segments

We define a segment to be in general position if it touches objects only at its extremities. A segment that touches objects in its interior will be called *critical*. At such an intersection there is a *local event*. If a segment touches more than one object in its interior, we call this a *multilocal event*. Critical segments are grouped into *critical segment sets*. The dimension of such a set can be seen as the number of degrees of freedom a segment has to keep the events. We can also refer to the codimension of such a set, which is the complement to the dimension of the space (the number of fixed degrees of freedom).

For the class of scenes we consider, there are two kinds of local events: tangency events and vertex events. The object or the vertex are called the *generators* of the event. To stay tangent to an object, a segment has three degrees of freedom. It is of codimension 1. It is of course the same when a segment goes through the edge of a polygon. We call this a T event from tangency (also referred to as E from edge in the aspect graph or discontinuity meshing literature which deals with polygonal scenes). A segment that goes through a vertex has two degrees of freedom (rotation), and thus has codimension 2. We call it a V event.

The combination of many local events causes a multilocal event, and the codimensions are added. We use the notation $+$ to describe such a combination. For example, a segment that is tangent to an object and that goes through a vertex belongs to a $T+V$ critical line set of codimension $1+2=3$ (it is a 1D set).

There is also a different kind of multilocal event that was not described in [DDP96]. A segment can be tangent to two objects and belong to one of their common tangent planes. In this case, the common tangent plane adds one codimen-

sion and we use the notation $++$. For example $T + ++T$ critical segment sets have codimension $1 + 1 + 1 = 3$ (1D set). (One may think of the example of two parallel cylinders and notice that lines contained in a bitangent plane have two degrees of freedom. This case is not considered here because it is degenerated.) These events are crucial for dynamic maintenance of views, aspect graphs and discontinuity meshes. For example a sphere hidden behind another sphere will appear when their outlines are tangent, that is when the viewpoint lies on a $T + ++T$ segment.

Each local event corresponds to an algebraic equation: a line tangent to an algebraic object or going through a vertex. A set of critical segments can thus be associated with the connected set of lines verifying the corresponding set of equations.

Events caused by faces are considered as $T + T$ events since they involve two edges. In the same way, segments going through an edge are $V + V$ events. The reason why the case of vertices (which could be seen as two edges events) is distinguished is that they introduce “discontinuities” at the end of edges and require a specific treatment as we shall see in section 5.4.

4 The 3D Visibility Complex

The *3D visibility complex* is the partition of maximal free segments of 3-space into connected components according to the objects they touch. Its faces of dimension 4 are maximal connected components of segments in general position with the two same objects at their extremities.

The different faces of lower dimension correspond to critical segments as summarized in table 1.

Theorem 1 *The size of the 3D visibility complex is $\Omega(n)$ and $O(n^4)$ where n is the complexity of the scene.*

Proof (sketched)

The number of $(k+1)$ -faces adjacent to a k -face is bounded. For example a 1-face $T_1 + T_2 + T_3$ is adjacent to five 2-faces: two faces $T_1 + T_2$ (there are two different faces because one extremity of the segments can lie on the object tangent at T_2 or not. See [DDP96]), $T_1 + T_3$ and two $T_2 + T_3$.

Each 4-face is adjacent to at least one 3-face, a 3-face to at least one 2-face, and a 2-face to at least one 1-face. We just sketch the demonstration. For a given face F of the complex, we consider the associated critical line set S . This set of lines contains a line set S' with one more codimension (one of the lines tangent to one object is also tangent to a second object, one of the lines tangent to two objects belongs to one of their common tangent plane, and one line going through a vertex is tangent to an object). Consider a continuous path from the line associated with a segment s of F to one of S' , and the corresponding continuous path over the segments. If all the segments of this path have the

Dimension	Type	Configuration
3	T	
2	T+T V	
1	T+T+T T++T T+V	
0	T+T+T+T T++T+T T+T+V V+V	

Table 1: faces of the visibility complex.

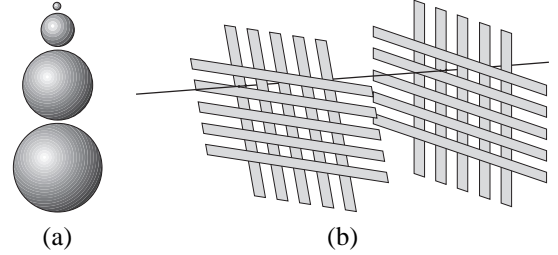


Figure 2: (a) Scene with an $O(n)$ Visibility Complex (b) Scene with an $O(n^4)$ Visibility Complex (an example of $T + T + T + T$ critical line is shown).

same extremities, F is adjacent to the face with one more codimension associated with S , otherwise when the extremity changes there is a tangency local event and one more codimension.

Note that a 1-face may be adjacent to no 0-face (we give an example below of a scene without a 0-face).

So the size of the complex is bounded by the number of 1-faces which are not adjacent to a 0-face plus the number of 0-faces. For each kind of events, the number of possible systems of algebraic equations depends on the number of objects implicated, the $T + T + T + T$ critical line sets are thus the most numerous with $O(n^4)$.

We show in figure 2(a) an example of a scene with a visibility complex of size $O(n)$: there is one $T + ++T$ face for each pair of neighbour spheres. Note there is no 0-face in that case. The scene in figure 2(b) is the same as in [PD90] and has an $O(n^4)$ visibility complex. There are two “grids”, each one composed of two very slightly distant orthogonal sets of $\frac{n}{4}$ parallel rectangles (this is also valid with thin ellipsoids). Consider a rectangle in each of the four sets: there

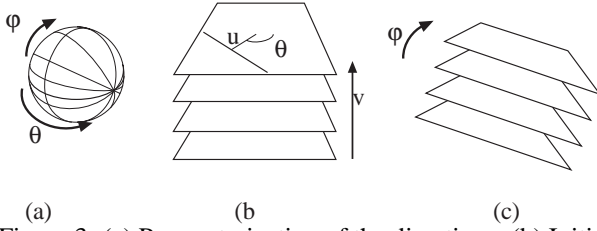


Figure 3: (a) Parameterization of the directions. (b) Initial v sweep (c) φ sweep.

is always a $T + T + T + T$ critical segment.

Visual events considered in the aspect graph literature [EBD92, PD90] correspond to the 1-faces of the visibility complex. For example the topology of a view changes when a vertex and an edge are aligned from the viewpoint. The aspect graph is in fact the arrangement of those events in the viewing space. This explains its size: $O(n^6)$ in the orthographic case where the viewing space is S^2 and $O(n^9)$ in the perspective case where the viewing space is R^3 .

In [DDP97] we presented a data-structure called the *Visibility Skeleton* which corresponds to the graph of the 0 and 1-faces of the visibility complex. First experiments with a few typical computer-graphics scenes show that the number of these faces (and thus the size of the complex) is about quadratic in the number of input polygons.

5 Output-sensitive sweep

Our algorithm is a double sweep with a preprocessing phase. First the scene is swept by a horizontal plane and a 2D Visibility Complex [PV96] of the φv -slice is maintained (figure 3(b)). We then sweep φ (figure 3(c)), but some 0-faces can not be detected during this sweep and have to be preprocessed.

5.1 Sweeping the initial slice

To build the initial φ -slice, we first maintain a φv -slice of the 3D visibility complex which corresponds to the 2D visibility complex [PV96] of the sweeping plane. We briefly review the 2D visibility complex. It is the partition of the segments of the planes according to the objects they touch. Its 2D faces are connected components of segments touching the same objects (they are φv -slices of the 4-faces of the 3D visibility complex). They are bounded by edges which correspond to segments tangent to one object (φv slices of the 3-faces T) and vertices which are free bitangents of the 2D scene (φv -slices of 2-faces $T + T$). Since a view around a point corresponds to the extremities of the segments going through this point, it corresponds to the traversal of the 2D visibility complex along the 1D path of these segments. The object seen changes when the path traverses a new face, which occurs at an edge of the 2D complex. In the case

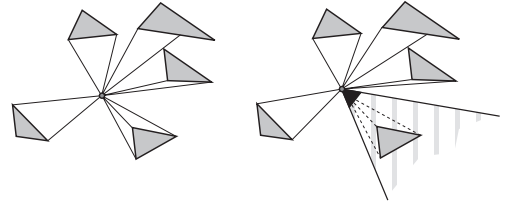


Figure 4: When the first vertex of a polyhedron is swept, the 2D view is computed in the sweeping plane and is restricted for each edge adjacent to the vertex by considering the angle formed by the direction of the two adjacent polygons.

of a polygon, the chain of edges of the 2D complex going through one of its vertices is the view around this vertex.

The 2D visibility complex has to be updated when the sweeping plane is tangent to an object or contains a vertex and when three 2D slices of objects share a tangent.

When the sweeping plane starts intersecting an object, we have to “insert” this object in our 2D complex. This is done by computing a view around the point of tangency or around the vertex using the current 2D visibility complex. This can be done in $O(v \log n)$ where v is the size of the view using the techniques described in [Riv97]. When the path of this view crosses an edge of the 2D complex it corresponds to a new $T + T$ or $V + T$ face of our 3D complex. In the case of the first vertex of a polyhedron, the view has to be restricted for each edge of the polyhedron, corresponding to the view seen by a vertex of the 2D slice (see figure 4).

Symmetrically, when an object stops intersecting the sweeping plane, the corresponding faces of the 2D visibility complex are collapsed. These faces are those along the chains of edges corresponding to segments tangent to this object. Their removal can be done in $O(v)$ where v is again the size of the view.

When a vertex in the middle of a polyhedron is encountered the 2D views around the points corresponding to the edges under the vertex have to be merged, and then the view around this vertex has to be restricted for each edge above the vertex, in the same manner as first vertex sweep-events, see figure 5. Each operation is linear in the size of each view.

As the plane moves, three slices of objects can share a tangent (corresponding to a $T + T + T$ face of the 3D complex), in which case the 2D visibility complex is updated using the technique of [Riv97]. Basically, for each bitangent we compute the value of v where it will become tangent to a third object and store these sweep-events in our queue which requires time $O(\log n)$ whenever a bitangent is created.

Finally, a bitangent of the 2D complex can correspond to a common tangent plane. For each bitangent, we compute the value of v for which it will lie on a bitangent plane and insert this sweep-event in the queue. Of course, these sweep-events have to be discarded if the bitangent is collapsed before.

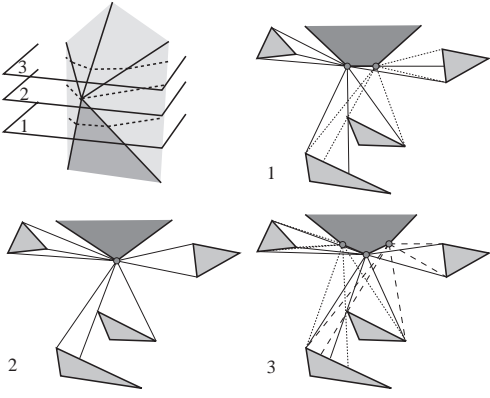


Figure 5: Fusion-restriction of a view around edges when a vertex is swept

5.2 Principle of the φ sweep

We now have computed a φ -slice of the 3D visibility complex. It is the partition of the segments contained in the set of horizontal planes. In this φ -slice, 1-faces of the complex have dimension 0, 2-faces have dimension 1, and so on.

During the φ -sweep (fig. 8(c)) we maintain this φ -slice as well as a priority queue of sweep-events. In what follows, we will only describe the update of the 1-faces of the visibility complex, the update of the upper dimensional is done at each sweep-event using a catalogue of adjacencies of the 1-faces which for reason of place cannot be given here. As stated before, the number of adjacent upper-dimensional faces is bounded; their update does not affect the complexity.

We first prove that some sweep-events are regular: a 1D component of the φ -slice is collapsed as its two extremities merge. These sweep-events can be detected by computed for each 1D component of the φ -slice the value of φ for which it will collapse. We will then study the case of irregular sweep-events.

5.3 Regular 0-faces

Consider a $T_1 + T_2 + T_3 + T_4$ segments with extremities O_0 and O_5 and elevation angle φ_0 (fig. 6). Consider the 1D critical line set $T_1 + T_2 + T_3$. We locally parameterize it by φ and call it $l(\varphi)$. The ruled surface described by $l(\varphi)$ cuts O_4 at φ_0 . Two 1-faces of the complex are associated with $l(\varphi)$, one for $\varphi < \varphi_0$ and one for $\varphi > \varphi_0$; one has O_5 at its extremity, the other O_4 . It is the same for $T_2 + T_3 + T_4$. Moreover the two 1-faces before φ_0 are adjacent to a 2-face $T_2 + T_3$. In the φ -slice, this 2-face is a 1D set bounded by the slices of $T_1 + T_2 + T_3$ and $T_2 + T_3 + T_4$. This 1D set collapses at φ_0 , it is thus a regular sweep-event. It can be detected by considering the adjacent $T + T + T$ faces in the φ -slice and maintaining a priority queue.

The $T + +T + T$ faces can be handled the same way because they are adjacent to a pair of $T + +T$ and a pair of

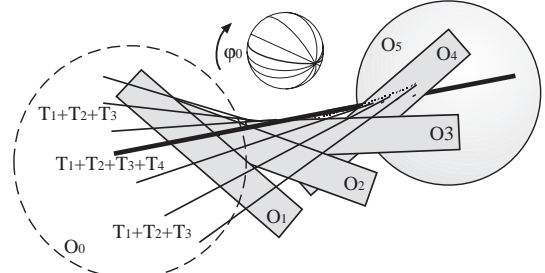


Figure 6: $T + T + T$ critical line set adjacent to a $T + T + T$ critical line.

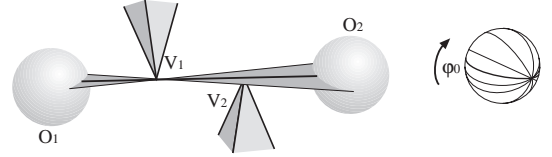


Figure 7: None of the $T + V$ critical segment sets adjacent to this $V + V$ critical segment exist before φ_0

$T + T + T$ 1-faces, and the faces of a pair are associated with the same line set.

5.4 Irregular 0-faces

Unfortunately, all the 0-faces are not regular sweep-events. The $T + T + V$ and $V + V$ events cannot be detected in this way. The main reason is that vertices represent discontinuities at the end of edges, and we have no guarantee that a 1-face adjacent to such a 0-face exists for $\varphi < \varphi_0$. See figure 7 where the four $T + V$ faces appear at φ_0 ; this corresponds in the dual space to situation (b) of fig. 8.

These events thus have to be preprocessed by considering all the VV pairs and all the Object-Object- V triplets.

Fortunately, at least one slice of an adjacent 2-face exists before such 0-faces appear (face V_1 in fig 7). The proof is omitted from this version. This face is found using a search structure over the 1D components of the φ -slice ordered by their generators. The 0-face is then tested for occlusion: we test if the generators (V_2 here) lies between the extremities (O_1 and O_2) of the 2-face. It can then be inserted.

5.5 Non monotonic 1-faces

There is another kind of irregular sweep-event. A 1-face of the complex can appear during the sweep without a 0-face event. This is obviously the case for $T + +T$ events since they can be adjacent to no 0-face, but this can also be the case for $T + T + T$ events. Consider the associated line set, it is not necessarily monotonic with respect to φ (see fig. 8(c)). These sweep-events also have to be preprocessed and inserted in the φ -slice with a search over the 1D components.

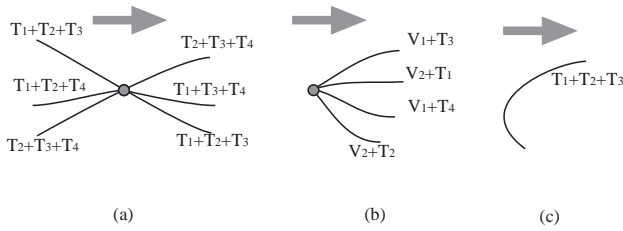


Figure 8: Different sweep-events represented in the dual space. The $T + T + T + T$ event (a) is regular, but the $V + V$ event (b) has to be preprocessed as well as the null derivative with respect to φ of the $T + T + T$ events (c).

5.6 Complexity of the algorithm

Theorem 2 *The visibility complex can be built in time $O((k + n^3) \log n)$ where n is the complexity of the scene, and k the number of 0-faces of the complex.*

During the initial v sweep, each view computation requires time $O(v \log n)$ where v is the size of the view. A view corresponds to the number of 3-faces of the 3d visibility complex adjacent to the appearing/disappearing 2 faces. The total cost is thus bounded by $O(k \log n)$. Each tritangent event requires time $O(\log n)$, here again the cost is bounded by $O(k \log n)$.

During the φ sweep, each regular event requires $O(\log n)$ to maintain the priority queue.

The preprocessing of the other 0-faces and non-monotonic 1-faces requires the enumeration of all the triplets of objects and the insertion of the computed faces in the priority queue, it is therefore $O(n^3 \log n)$.

The output-sensitive nature of this algorithm is very important since experiments on a few polygonal scenes [DDP97] have shown that the number of $T + T + T + T$ segments which is responsible of the theoretical $O(n^4)$ is in fact much less than the number of $T + T + V$ segments.

6 Conclusions and future work

We have introduced a unified data structure, the 3D visibility complex, which encodes the global visibility informations for 3D scenes of polygons and convex smooth objects. Its size k is $\Omega(n)$ and $O(n^4)$ and we have presented an output-sensitive algorithm to build the structure in time $O((n^3 + k) \log n)$.

Future work includes the use of the visibility complex to maintain views around a moving viewpoint, a study of the events involved by concave and piecewise smooth objects, the development of a better construction algorithm, and the incremental update of the visibility complex when an object is moved, added or removed.

Acknowledgments

The authors would like to thank Seth Teller, Michel Pocchiola and Sylvain Petitjean for very fruitful and inspiring discussions.

References

- [CEG⁺96] B. Chazelle, H. Edelsbrunner, L. J. Guibas, M. Sharir, and J. Stolfi. Lines in space: combinatorics and algorithms. *Algorithmica*, 15:428–447, 1996.
- [DDP96] F. Durand, G. Drettakis, and C. Puech. The 3d visibility complex, a new approach to the problems of accurate visibility. In *Proc. of 7th Eurographics Workshop on Rendering in Porto, Portugal*, June 1996.
- [DDP97] F. Durand, G. Drettakis, and C. Puech. The visibility skeleton: A powerful and efficient multi-purpose global visibility tool. *Computer Graphics (Siggraph'97 Proceedings)*, 1997.
- [DF94] G. Drettakis and E. Fiume. A fast shadow algorithm for area light sources using back projection. In *Computer Graphics Proceedings, Annual Conference Series: SIGGRAPH '94* (Orlando, FL), July 1994.
- [EBD92] D. Eggert, K. Bowyer, and C. Dyer. Aspect graphs: State-of-the-art and applications in digital photogrammetry. In *Proceedings of the 17th Congress of the Int. Society for Photogrammetry and Remote Sensing*, 1992.
- [LTG93] D. Lischinski, F. Tampieri, and D. Greenberg. Combining hierarchical radiosity and discontinuity meshing. In *Computer Graphics Proceedings, Annual Conference Series: SIGGRAPH '93* (Anaheim, CA, USA), August 1993.
- [MS97] S. Mohaban and M. Sharir. Ray shooting amidst spheres in 3 dimensions and related problems. *to appear in SIAM J. Computing*, 1997.
- [ORDP96] R. Orti, S. Rivière, F. Durand, and C. Puech. Radiosity for dynamic scenes in flatland with the visibility complex. In *Proc. of Eurographics*, Poitiers, France, 1996.
- [PD90] H. Plantinga and C. R. Dyer. Visibility, occlusion, and the aspect graph. *IJCV*, 1990.
- [PDS90] H. Plantinga, C. R. Dyer, and B. Seales. Real-time hidden-line elimination for a rotating polyhedral scene using the aspect representation. In *Proceedings of Graphics Interface '90*, 1990.
- [Pel90] M. Pellegrini. Stabbing and ray shooting in 3-dimensional space. In *Proc. 6th Annu. ACM Sympos. Comput. Geom.*, 1990.
- [PPK92] S. Petitjean, J. Ponce, and D.J. Kriegman. Computing exact aspect graphs of curved objects: Algebraic surfaces. *IJCV*, 1992.
- [PV96] M. Pocchiola and G. Vegter. Topologically sweeping visibility complexes via pseudo-triangulations. *Discrete Comput. Geom.*, December 1996. special issue devoted to ACM-SoCG'95.
- [PV96] M. Pocchiola and G. Vegter. The visibility complex. *Internat. J. Comput. Geom. Appl.*, 96. special issue devoted to ACM-SoCG'93.
- [Rie87] J.H. Rieger. On the classification of views of piecewise smooth objects. *Image and Vision Computing*, 1987.
- [Riv97] S. Rivière. Dynamic visibility in polygonal scenes with the visibility complex. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, 1997.
- [SG94] J. Stewart and S. Ghali. Fast computation of shadow boundaries using spatial coherence and backprojections. In *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 1994)*, Computer Graphics Proceedings, July 1994.
- [Tel92] S. Teller. Computing the antipenumbra of an area light source. *Computer Graphics*, July 1992. Proceedings of SIGGRAPH '92 in Chicago (USA).

2.4.8 Fast and Accurate Hierarchical Radiosity Using Global Visibility (ACM TOG'99)

Auteurs : Frédo Durand, George Drettakis et Claude Puech

Revue : ACM Transactions on Graphics

Date : mars 1999

Fast and Accurate Hierarchical Radiosity Using Global Visibility

Frédo Durand, George Drettakis and Claude Puech
iMAGIS - GRAVIR / IMAG - INRIA

Recent hierarchical global illumination algorithms permit the generation of images with a high degree of realism. Nonetheless, appropriate refinement of light transfers, high quality meshing and accurate visibility calculation can be challenging tasks. This is particularly true for scenes containing multiple light sources and scenes lit mainly by indirect light. We present solutions to these problems by extending a global visibility data structure, the Visibility Skeleton. This extension allows us to calculate exact point-to-polygon form-factors at vertices created by subdivision. The structure also provides visibility information for all light interactions, allowing intelligent refinement strategies. High-quality meshing is effected based on a perceptually-based ranking strategy which results in appropriate insertions of discontinuity curves into the meshes representing illumination. We introduce a hierarchy of triangulations which allows the generation of a hierarchical radiosity solution using accurate visibility and meshing. Results of our implementation show that our new algorithm produces high quality view-independent lighting solutions for direct illumination, for scenes with multiple lights and also scenes lit mainly by indirect illumination.

Categories and Subject Descriptors: I.3.7 [Computing Methodologies]: Computer Graphics—*Three-Dimensional Graphics and Realism*

General Terms: Global Illumination, Global Visibility

Additional Key Words and Phrases: Hierarchical Radiosity, Form Factor Calculation, Discontinuity Meshing, Hierarchical Triangulation, Perception

1. INTRODUCTION AND PREVIOUS WORK

Recent advances in global illumination, such as hierarchical radiosity [Hanrahan et al. 1991] and its combination with discontinuity meshing [Lischinski et al. 1993] have resulted in high quality lighting simulations. These lighting simulations are *view independent* and are suitable for walkthroughs. The quality of the resulting illumination is important everywhere in the scene, since the user can, for example, approach a shadow of an object and see its details.

Despite the high quality of existing techniques, certain aspects of these algorithms are still suboptimal. In particular, deciding when a light-transfer is *refined* appropriately, and thus computed with higher precision is a hard decision; current algorithms ([Hanrahan et al. 1991; Lischinski et al. 1994; Gibson and Hubbard 1996] etc.) include methods based on error bounds which in many cases prove insufficient. Creating a *mesh* to represent lighting variations accurately (notably for shadows) is hard; discontinuity meshing approaches [Lischinski et al. 1993; Drettakis and Sillion 1996] have proposed some solutions for these

iMAGIS is a joint research project of CNRS/INRIA/UJF/INPG.

Address: iMAGIS/GRAVIR, BP 53, F-38041 Grenoble Cedex 09 France

{Fredo.Durand|George.Drettakis|Claude.Puech}@imag.fr

<http://www-imagis.imag.fr/>

issues which are however often limited in their applicability. Recent approaches (e.g., [Christensen et al. 1996; Ureña and Torres 1997]) avoid this problem by performing a view-dependent, ray-casting “final gather”; view-independence and the capacity for interactive display and walkthroughs are thus sacrificed. Accurate *visibility* calculation is also fundamentally hard, since we have to consider the potential interaction between all polygons in the scene for global illumination.

The above three problems, *visibility*, *refinement* and *meshing* are accentuated in the following two lighting configurations: scenes lit by multiple sources and scenes lit mainly by indirect illumination. In this paper we present a new algorithm which addresses the three shortcomings mentioned above. For all three problems, refinement, meshing and visibility previous approaches lack information on accurate *global visibility* relationships in the scene. This information is provided by the *Visibility Skeleton* [Durand et al. 1997]. To achieve our goal, we first extend the Skeleton to provide visibility information at vertices resulting from subdivision of the original input surfaces. The extended Skeleton allows the fast computation of exact point-to-polygon form-factors for any point-polygon pair in the scene. In addition, all visibility information (blockers and all discontinuity surfaces) is available for any polygon-polygon pair.

This global visibility information allows us to develop an intelligent refinement strategy, since we have knowledge of visibility information for *all* light transfers from the outset. We can *rank* discontinuity surfaces between any two hierarchical elements (polygons or patches resulting from their subdivision), using perceptually-based techniques [Gibson and Hubbard 1997]; thus only discontinuities which are visually important are considered. An appropriate mesh is created using these discontinuities; illumination is represented very accurately resulting in high-quality, view-independent meshes. To achieve this in the context of a hierarchical radiosity algorithm, we have introduced a hierarchy of triangulations data structure. Radiosity is gathered and stored at vertices, since the extended Skeleton provides us with the exact vertex-to-polygon form-factor. An appropriate multi-resolution push-pull procedure is introduced. The high-quality mesh, the exact form-factor calculation and the hierarchical triangulation result in lighting simulation with accurate visibility.

Our approach is particularly well-suited for the case of multiple sources since the discontinuity ranking operates simultaneously on *all* light energy arriving at a receiver. Indirect illumination is also handled very well, since visibility information, and thus the refinement and meshing strategies as well as the form-factor computation apply equally well to all interactions, i.e., both direct (from the sources) and indirect (reflected light). Examples of these two cases are shown in Fig. 1. In Fig. 1(a) we see a scene lit by 10 separate light sources, where the multiple shadows are visible but the mesh complexity is reasonable (see Table 2, in Section 7.2). In Fig. 1(b) we see a room lit mainly by indirect lighting; notice the high quality shadows created entirely by indirect light (e.g., on the far wall from the books and lamp).

1.1 Previous Work

The new algorithm we present here is in a certain sense an extension of hierarchical radiosity, using visibility structures, advanced meshing techniques and perceptually-based subdivision. We briefly review hierarchical radiosity methods, accurate visibility techniques and related visibility-based refinement for lighting algorithms and finally perceptually-based refinement for illumination.

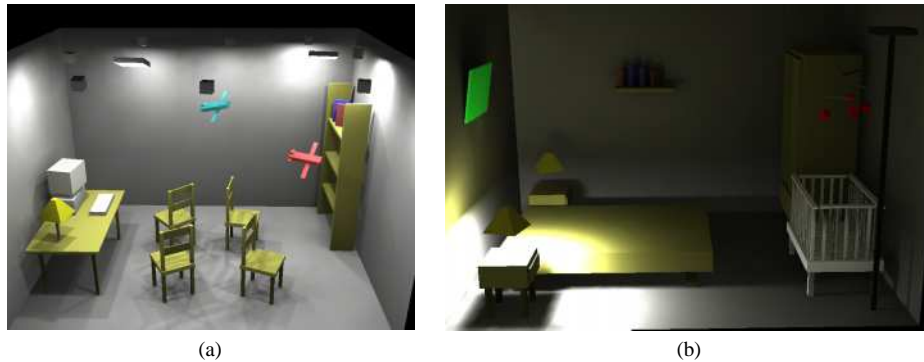


Fig. 1. Images computed using our new hierarchical radiosity algorithm based on the extended Visibility Skeleton and hierarchical triangulations. (a) A scene with multiple sources. The skeleton construction took 2min 23s and the lighting simulation 8min. (b) A scene mainly lit by indirect light. The skeleton construction took 4min 12s and the lighting simulation 6min 58s. Note the shadows caused by indirect illumination, cast by the books on the back wall.

1.1.1 Hierarchical radiosity. The hierarchical radiosity algorithm [Hanrahan et al. 1991] allows efficient calculation of global illumination. Lighting calculations are limited to a user-specified level of accuracy, by means of hierarchically subdividing the polygons in the scene into a quadtree, and creating light-transfer “links” at the appropriate levels of the hierarchy. In the original hierarchical radiosity solution [Hanrahan et al. 1991], radiosity is considered constant over each quadtree element. The rectangular nature of the quadtree, and the constant reconstruction result in the need for very fine subdivision for high quality image generation (high quality shadows etc.).

Higher-order (non-constant) methods have also been introduced, notably in the context of wavelet-based solutions [Gortler et al. 1993]. The wavelet-based radiosity solutions presented to date typically operate on discontinuous bases, resulting in visible discontinuities if the solution is displayed directly (e.g., [Christensen et al. 1996]). Zatz [Zatz 1993] used a Galerkin-type method and shadow masks to improve the quality of the shadows generated. To avoid the problem of discontinuous representations the “final gather” step was introduced by [Reichert 1992] and used for wavelet solutions (e.g., [Christensen et al. 1996]). A final gather step consists of creating a ray-cast image, by querying the object-space visibility and lighting information to calculate illumination at each pixel [Ureña and Torres 1997]. This approach allows the generation of high quality images from a coarse lighting simulation, at an additional (frequently high) cost. The solution thus becomes view-dependent, and interactive display and walkthrough capability are lost.

More recently, Bekaert *et al.* have presented an efficient algorithm which combines hierarchical radiosity and Monte-Carlo radiosity [Bekaert et al. 1998]. However, the stochastic nature of the algorithm makes it difficult to refine along shadow boundaries.

1.1.2 Accurate Visibility and Image Quality. The accurate calculation of visibility in a lighting simulation is essential: both the numerical quality of the simulation and the visual quality of the resulting image depend on it. The exact computation of visibility between two patches in a scene or between a patch and a point requires the treatment of

visual events. Visual events are caused at boundaries in a scene where the visibility of one object changes with respect to a point of view. Such events occur at visibility boundaries generated by the interaction between vertices and edges of the environment (see Section 2). In the case of the view of a light source, these boundaries correspond to the limits of umbra and penumbra. By choosing certain of these boundaries and using them to guide the (irregular) mesh structure, *discontinuity meshing* lighting algorithms have been introduced resulting in more visually accurate images (e.g., [Heckbert 1992; Lischinski et al. 1992]).

In the vision literature, visual events have been extensively studied [Plantinga and Dyer 1990; Gigus et al. 1991]. The *aspect graph* structure completely encodes all visibility events in a scene. The determination of the visible part of an area light source in computer graphics is exactly the calculation of the aspect of the light at a given point. Algorithms performing this operation by building the complete discontinuity mesh and the *backprojection* data structure (encoding the source aspect) have been presented (e.g., [Teller 1992; Drettakis and Fiume 1994; Stewart and Ghali 1994]). The full discontinuity mesh and backprojection allows the computation of the exact point-to-area form-factor with respect to an area light source. Nonetheless, these methods suffer from numerical problems due to the required intersections between the discontinuity surfaces and the scene polygons, complicated data-structures to represent the highly irregular meshes and excessive computational requirements. The Visibility Complex [Durand et al. 1996] and its simplification, the Visibility Skeleton [Durand et al. 1997], present complete, *global* visibility information between any pair of polygons. We have chosen to use the Visibility Skeleton because of its flexibility, relative robustness (compared to discontinuity meshing) and ease-of-use. A review of necessary machinery from the Skeleton used here is presented in Section 2.

1.2 Visibility-Based Refinement Strategies for Radiosity

In the Hierarchical Radiosity algorithm, mesh subdivision is effected through link refinement. The original algorithm used a *BFV* criterion (radiosity times form-factor modulated by a visibility factor V for partially occluded links). The resulting meshes are often too fine in unoccluded regions, and do not always represent fine shadow details well.

Refinement strategies based on error bounds [Lischinski et al. 1994; Gibson and Hubbard 1996] have improved the quality of the meshes and the simulation compared to the *BFV* criterion. Conservative visibility determination in architectural scenes [Teller and Hanrahan 1993], accurately characterises links as *visible*, *invisible* or *partially visible*. This triage guides subdivision, allowing finer subdivision in partially illuminated regions.

Discontinuity meshing clearly improves the visual quality of images generated by lighting simulation [Lischinski et al. 1992; Heckbert 1992; Drettakis and Fiume 1994], since the mesh used to represent illumination follows the actual shadow boundaries, instead of finely subdividing a quadtree which attempts to approximate the boundary. One problem is the extremely large number of discontinuities. Tampieri [Tampieri 1993] attempted to limit the number of discontinuity lines inserted, by sampling the illumination along the discontinuities and only inserting those with radiosity values differing more than a predefined threshold. In the context of progressive refinement radiosity, Stuerzlinger [Stuerzlinger 1994], only inserted discontinuities at a second level of a regular quadrilateral adaptive subdivision, once a ray-casting step has classified the region as important. The only discontinuities inserted were those due to the blocker identified by the ray caster.

Several methods combining discontinuity meshing with hierarchical radiosity have been presented [Lischinski et al. 1993; Drettakis and Sillion 1996; Hardt and Teller 1996; Boua-

touch and Pattanaik 1995]. Hardt and Teller [Hardt and Teller 1996] present an approach in which potential discontinuities from all surfaces are considered, without actually intersecting them with the blockers. Potential discontinuities are ranked, and those deemed most important are inserted and the lowest level of the quadtree. In [Drettakis and Sillion 1996] the backprojection information is used in the complete discontinuity mesh creating a large number of small triangles. Exact form-factors of the primary source are then computed at the vertices of these triangles. The triangles are then clustered into a hierarchy. Standard [Hanrahan et al. 1991] constant-element hierarchical radiosity follows. The previously cited problems of discontinuity meshing, the expensive clustering step and the fact that the inner nodes of the hierarchy often overlap, limit the applicability of this approach to small models. The only other hierarchical radiosity method with gathering at vertices is that of Martin *et al.* [Martin et al. 1997], which requires a radiosity value at each vertex, and a complex push procedure.

The algorithm of Lischinski *et al.* [Lischinski et al. 1993] is much more complete and relevant to our work. The basis of this approach is to separate the light simulation and rendering steps. This idea is similar in spirit to the use of a “final gather” step. Lischinski *et al.* first compute a “global pass” by creating 2D BSP trees on scene polygons subdivided by choosing important discontinuities exclusively due to the primary sources. The 2D BSP tree often incurs long splits and consequently long or thin triangles, which are inappropriate for high quality lighting simulation. The second, view independent, “local pass” recomputes illumination at the vertices of a triangulated subdivision of the leaf elements of the BSP tree. To achieve high quality images, the cost of triangulation and shading (light recomputation at vertices using “method D” [Lischinski et al. 1993]), is higher than that of the actual lighting simulation (if we ignore the initial linking step).

1.3 Perceptually Based Refinement

Recently, perceptually-based error metrics have been used to reduce the number of elements required to accurately represent illumination (*e.g.*, [Gibson and Hubbold 1997; Hedley et al. 1997]). Tone-reproduction approaches [Tumblin and Rushmeier 1993; Ward 1994] are used to map calculated radiosity values to display values which convey a perceptual effect closer to that perceived by a real world viewer. Since display devices have limited dynamic range compared to real world luminance values, the choice of this mapping is very important. The tone reproduction mappings of [Tumblin and Rushmeier 1993; Ward 1994] depend on two parameters: a world adaptation level which corresponds loosely to the brightness level at which a hypothetical observer’s eye has adapted, and a display adaptation level which corresponds to the brightness displayed on the screen. Choice of these parameters affects what will be displayed, and, more importantly, which differences in radiosities will actually be perceptible in the final image. Most notably, one can define a “just noticeable difference” using this mapping. In the context of lighting, a just noticeable difference would correspond to the smallest difference in radiosity values, which once transformed via tone reproduction, will be visible to the viewer of the display. Display adaptation is typically a fixed value (*e.g.*, half the maximum display luminance [Ward 1994]), while the world adaptation level can be chosen in a number of different ways. Using static adaptation [Gibson and Hubbold 1997], one uses an average which is independent of where the observer is looking, while dynamic adaptation (which is closer to reality) changes depending on where the observer is looking. Gibson and Hubbold [Gibson and Hubbold 1997] use tone reproduction to guide subdivision in a progressive refinement

radiosity approach, thus allowing a subdivision only if the result will be “just noticeable”.

Hedley *et al.* [Hedley et al. 1997], in a similar spirit, use a tone mapping operator to determine whether a discontinuity should be inserted into a lighting simulation mesh. This is performed by sampling across the discontinuity (in a manner similar to that of Tampieri [Tampieri 1993]), but also orthogonally across the discontinuities. This results in an important reduction of discontinuities without loss of visual quality.

1.4 Paper Overview

Previous algorithms surveyed above provide view-independent lighting simulations which are acceptable for many situations. In particular, quadtree based hierarchical radiosity provides fast solutions of moderate quality, even for scenes mainly lit indirectly. Nonetheless, in walkthroughs the observer often approaches regions of shadow, and in these cases the lack of shadow precision is objectionable. Previous approaches based on discontinuity meshing alleviate this problem for direct lighting, but rapidly become impractical for scenes with many lights, or for which indirect lighting is dominant. Their limitations are due to the sheer number of discontinuity surfaces that need to be considered when computing indirect illumination and the complexity of the meshes which result. These issues are discussed in [Lischinski et al. 1992] and [Tampieri 1993]. The solutions adopted to date have restricted the use of discontinuity information to those from primary light sources [Lischinski et al. 1993; Drettakis and Sillion 1996]; for subsequent light bounces (secondary, tertiary etc.), approximate ray-casting approaches are used for visibility computations in light transfer.

The new algorithm presented here allows the generation of accurate shadows for a more general class of scenes, including those with dominant indirect illumination. To achieve this goal we extend the Visibility Skeleton to support view calculation at vertices resulting from subdivision, and to use a link-based storage mechanism which is more adapted to a hierarchical radiosity approach. This extended structure is presented in Section 2. The resulting structure allows us to select and insert discontinuity lines for all light transfers, and to calculate exact point-to-area form-factors rapidly, using the visibility information provided. These choices required us to develop a new hierarchical radiosity algorithm, with gathering at vertices, based on embedded *hierarchical triangulations* allowing the mesh to follow discontinuity lines. The details of this structure, and the novel push-pull algorithm are presented in Section 3. In Section 4 we present the new hierarchical radiosity algorithm using accurate global visibility and we present the new point-polygon and polygon-polygon link data structures. In Section 5 we present the corresponding refinement processes and the visibility updates required for their use. In Section 6 polygon subdivision and the perceptually-based refinement criterion are described. We then present results of our implementation, as well as a discussion of relative limitations and advantages of our approach, and we conclude.

2. THE VISIBILITY SKELETON

The Visibility Skeleton [Durand et al. 1997] is a data structure encoding all the global visibility relationships in a 3D scene. It is based on the notion of *visibility events*.

A visibility event is the locus of a topological change in visibility. An example is shown in Fig. 2(a) (taken from [Durand et al. 1997]). When the viewpoint moves from left to right, vertex v as seen from the observer will no longer lie on the floor, and will now be on the polygon adjacent to edge e . We say that there is a topological change in the view from

the eyepoint.

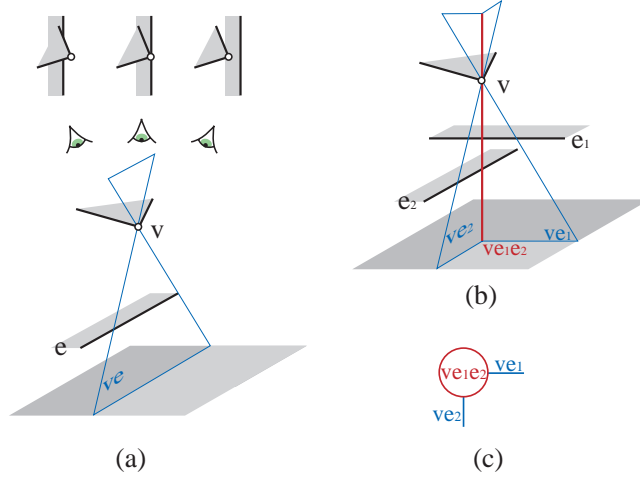


Fig. 2. (a) An EV line swath, (b) The VEE node is adjacent to two line swaths (c) The graph structure induced (Figure taken from Durand *et al.* [Durand et al. 1997])

A visibility event is a 1D set of lines: in Fig. 2(a) the EV event is the set of lines going through v and e ; it can be parameterized by the abscissa on e . We call the surfaces swept by such sets of lines *line swaths*. Such swaths are caused by the interaction of an edge and a vertex (EV swaths) or three edges (EEE) swaths.

The extremities of these 1D line sets are lines with no degrees of freedom: the *extremal stabbing lines*. These are lines passing through four edges of the scene. Examples are vertex-vertex (VV) lines passing through two vertices, vertex-edge-edge (VEE) lines passing through two edges and a vertex (see for example Figure 2(b)) or 4-edge ($E4$) lines passing through four edges.

This construction naturally defines a graph in line-space. The nodes are the extremal stabbing lines and the arcs are the line swaths. The nodes of the graph are adjacent to a certain number of arcs (swaths), which are defined in a catalogue for each type of node [Durand et al. 1997]. Fig. 2(b) shows the adjacencies of a VEE extremal stabbing line and two EV critical line swaths. The graph structure induced, consisting of a node and the two arcs, is shown in Fig. 2(c).

Efficient access to the visibility information is provided by means of an n^2 array (where n is the number of objects in the scene) indexed by the polygons at the extremities of the swaths: A cell of the array indexed by polygons (P, Q) stores in a search tree all the line swaths whose extremities lie on P and Q .

The *visibility skeleton* is the graph of line swaths and extremal stabbing lines together with the array of search trees.

The construction of the Skeleton proceeds as a set of nested loops over the edges and vertices of the scene to determine the nodes (extremal stabbing lines). First simple cases (e.g., VV) are found, and subsequent loops over the edges allow the identification of VEE and $E4$ nodes.

Once a potential extremal stabbing line is detected, it is tested for occlusion using ray-casting. If an object lies between its generators, it is discarded. Otherwise a node is created. The ray-casting operation also provides the extremities of the node.

Once a node is created, its neighbourhood in the graph is updated using the catalogue of adjacent arcs. If an adjacent arc has already been created (because of its other extremity) it is just linked to the new node; otherwise it is created and linked. The array of search trees is used for efficient search of the existing arcs.

It is important to note that the line swaths are not actually constructed geometrically: only the extremal stabbing lines are involved in the geometric construction. This makes the algorithm more robust, since only ray-casting is needed, as opposed to traditional discontinuity meshing which requires complicated swath-polygon intersections [Drettakis and Fiume 1994].

2.1 Extensions to the Visibility Skeleton

2.1.1 Memory requirements. The storage of the arcs of the Skeleton in a two dimensional array incurs an $O(n^2)$ cost in memory. In the scenes presented in [Durand et al. 1997] half of the memory was used for the array, in which more than 95% of the cells were empty! It is even more problematic when the scene is highly occluded such as in the case of a building where each room sees a only fixed number of other rooms: the number of arcs is only $O(n)$. Moreover, for our lighting simulation, we will need to subdivide the initial polygons into sub-patches and incrementally compute visibility information between some pairs of sub-patches, but not all.

For these reasons we store the set of critical line swaths between two polygons on the polygons themselves. Each polygon P stores a balanced binary tree; each node of this tree contains the set of arcs between P and another polygon Q . This set is itself organized in a search tree (see Figure 3). Each set of arcs is referenced twice, once on P and once on Q . In the same manner, each vertex V has a search tree containing the sets of arcs between V and a polygon Q (which represents the view of Q from V). These sets of arcs are closely related to the notion of *links* in hierarchical radiosity as we will see in Section 4.2.

For the scenes presented [Durand et al. 1997], this approach results in an average memory saving of about 30%. Moreover, we have ran our modified version of the visibility skeleton on a set of scenes consisting of a room replicated 2, 4, and 8 times, showing roughly linear memory growth for the skeleton. Using a binary tree instead of an array incurs an additional $O(\log n)$ time access cost, but this was not noticeable in our tests.

2.1.2 Visibility Information at Vertices from Subdivision. To permit the subdivision of surfaces required to represent visual detail (shadows etc.) on scene polygons, visibility skeleton information must be calculated on the triangles created by the subdivision and the corresponding interior vertices. The process is presented in detail in Section 5.

Since the visibility information is now stored on polygons and vertices (instead of in a 2D array), the generalization to subdivided polygons is straightforward. On each sub-patch or sub-vertex, we store the visibility information only for the patches it interacts with.

2.2 Treating Degenerate Configurations

Computational geometry often makes the assumption that the scenes considered are in a “general configuration”. Unfortunately, computer graphics scenes are very often highly degenerate: many points are aligned, segments or faces are parallel or coplanar and objects

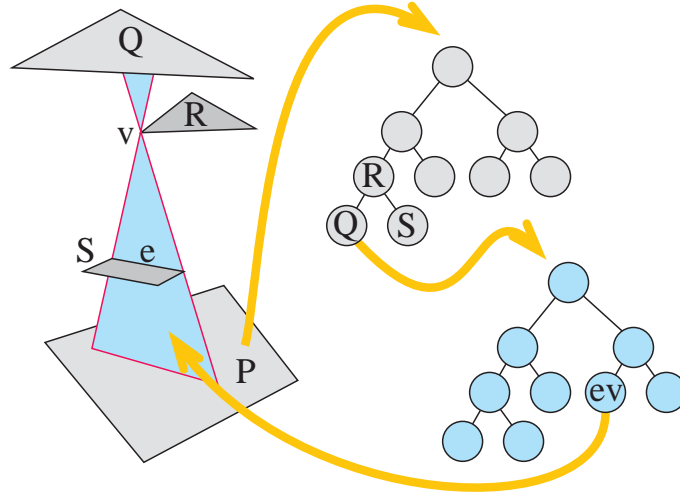


Fig. 3. Summary of the visibility skeleton structure. Each polygon stores a search tree indexed by the polygons it can see. For each pair of polygons, a search tree of visibility events is stored.

touch each other.

This results in degenerate visibility events; *e.g.*, *VVV* extremal stabbing lines passing through three aligned vertices, or *E5* stabbing lines going through five edges.

These degenerate configuration cause duplicate line swaths and result in numerical instabilities in the occlusion test of a potential extremal stabbing line. This line may then be randomly discarded. Inconsistencies can thus appear in the neighbourhood of the corresponding nodes of the graph. A consistent policy has to be chosen to include these nodes and their adjacent arcs or not.

We first have to identify the occurrence of these problems. When a potential extremal stabbing line is tested for occlusion, we also check for grazing objects. This requires a simple modification to the point-in-polygon test used for the ray-casting occlusion test of the potential extremal stabbing lines. We thus detect the intersection with a silhouette edge or vertex.

We also have to deal with the aforementioned degenerate extremal stabbing lines. A first possibility is to explicitly create a catalogue of all these degeneracies. This approach however quickly makes the implementation intractable because of the large number of different cases. We have chosen to always consider the simplest configuration, that is the one in which we have the smallest number of visual events. For example, if four edges E_1 , E_2 , E_3 and E_4 are parallel in that order, we consider that E_2 occludes E_1 and then E_3 occludes E_2 etc. The configurations to be treated are thus simpler and correspond to the standard Skeleton catalogue of events. The problems of numerical precision are treated using a consistent ϵ threshold for equality and zero tests.

3. IRREGULAR HIERARCHICAL TRIANGULATIONS FOR ILLUMINATION

In previous work (*e.g.*, [Heckbert 1992; Lischinski et al. 1992]) it has been shown that the creation of a mesh well adapted to the discontinuities in illumination results in images

of high visual quality. Incorporating such irregular meshes into a hierarchical radiosity algorithm presents an important challenge. As mentioned in Section 1.1.2, most previous algorithms [Lischinski et al. 1993; Drettakis and Sillion 1996] addressing this issue have restricted the treatment of discontinuities to those due to direct (primary) illumination.

The core of the problem is that two conflicting goals are being addressed: that of a simple regular hierarchy, permitting straightforward manipulations and neighbor finding and that of an essentially irregular mesh, required to represent the discontinuity information. The first goal is typically achieved using a traditional quadtree structure [Hanrahan et al. 1991] and the second typically by a BSP-type approach [Lischinski et al. 1993].

In previous approaches, discontinuity information and accurate visibility were incorporated into constant-element hierarchical radiosity algorithms. In the case of the Skeleton, this would be wasteful, since we have all the necessary information to compute *exact* form-factor from any polygon in the scene to any vertex (see Section 5 to see how this is also true for vertices resulting from subdivision). Gathering to vertices introduces one important complication: contrary to elements whose level in the hierarchy is clearly defined, vertices are shared between hierarchy levels.

As a solution to the above issues, we introduce hierarchical triangulations for hierarchical radiosity. Our approach has two major advantages over previous hierarchical radiosity methods: (i) it adapts well to completely irregular meshes and this in a local fashion (triangulations contained in triangulations), avoiding the artifacts produced by splitting edges of a 2D BSP tree and (ii) it allows gathering to vertices by a “lazy wavelets”-type (or sub-sampling) construction (see the book by Stollnitz *et al.* [Stollnitz et al. 1996] pp 102–104 and 152–154). It preserves a linear approximation to radiosity during the gather and the push process of the solution.

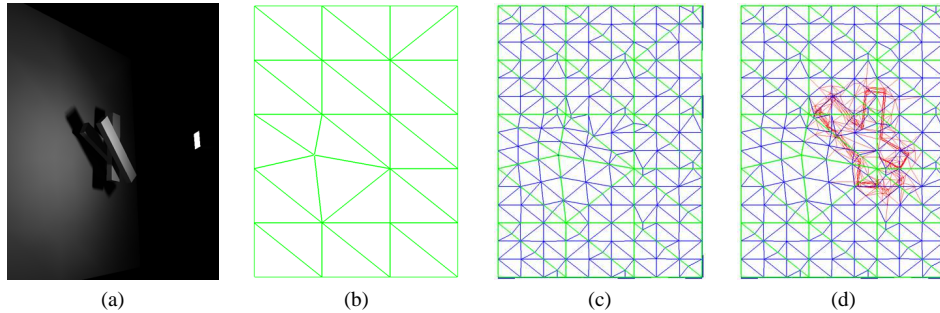


Fig. 4. Hierarchical Triangulation Construction. Notice how the triangles are overall well shaped but also well adapted to local detail. (a) Scene geometry: the leftmost polygon is illuminated by the area source on the right pointing leftwards. (b) First level of subdivision for the leftmost polygon (green). (c) Second level (blue). (d) third level (red).

3.1 Hierarchical Triangulation Construction

Our hierarchical triangulation construction has been inspired by that of de Floriani and Puppo [De Floriani and Puppo 1995]. As in their work we start with an initial triangulation, which is a constrained Delaunay triangulation (CDT). The CDT allows the insertion of constrained edges into the triangulation, which are not modified to satisfy the Delaunay

property and thus remain “as is”. Each triangle of the initial triangulation can be subdivided into a sub-triangulation, and so on recursively. At each level, a CDT is maintained.

An example of such a construction is shown in Figure 4, clearly showing the first advantage mentioned above. As we can see, the triangulation maintains well-shaped triangles everywhere in the plane, while providing fine details in the regions where this is necessary. The representation of such detail induces irregular subdivision at the finer levels.

Our hierarchical triangulation is “matching”, in the sense that edges split across two levels of a triangulation are done so at the same point on the edge. At the end of each subdivision step an “anchoring” operation is performed by adding the missing points in the neighboring triangles, thus resulting in a conforming triangulation across levels required for the push phase of hierarchical radiosity.

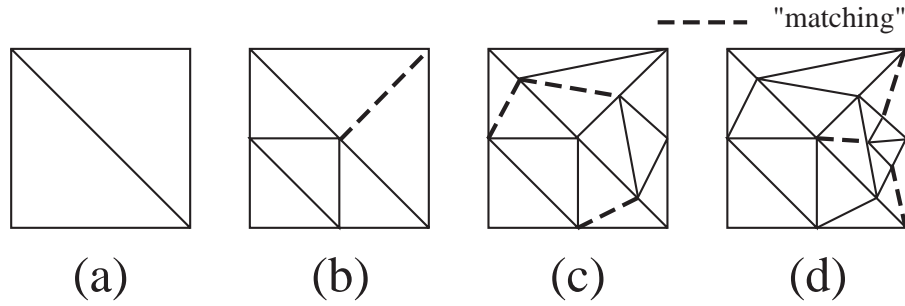


Fig. 5. The “matching” constraint for the Hierarchical Triangulation. The sequence shows subsequent segment insertions. The dashed lines show the insertions performed to enforce the “matching” constraint.

As mentioned above, vertices are shared between different levels of the triangulation. The initial level of a triangulation is an *HPolygon*, which contains an *HTriangulation* child once subdivided, where the prefix *H* represents the hierarchical nature of the construction.

To transmit neighborhood information between levels (for the matching operation), we use a special *HEdge* structure. An *HEdge* is shared between hierarchy levels by all edges which correspond to the same segment. It contains pointers to sub *HEdge*’s when it is subdivided. To perform a matching operation we determine whether the edge on which we insert a point p has already been split. We then add the new points corresponding to the previously split vertices, and split the *HEdge* at the point p . The neighbouring triangle can thus identify the newly inserted sub-*HEdge*’s from the shared *HEdge*. For example, in Figure 5, after the subdivision of the lower left triangle in Fig. 5(a), the *HEdge* shared between the two triangles notifies the upper right triangle that the edge has been split, and facilitates the matching operation as shown in Fig. 5(b).

3.2 Linear Reconstruction of Illumination using Hierarchical Triangles

The second advantage, that of linear reconstruction of illumination across irregular meshes, requires the use of a “lazy-wavelet” or “sub-sampling” type construction. Lazy wavelets provide an elegant formalism for a simple approach: a piecewise linear approximation is refined through the addition of new sampling points [Stollnitz et al. 1996].

As mentioned above, in our hierarchical triangulation representation of radiosity, vertices will be shared between hierarchy levels. As a consequence, traditional push-pull procedures [Hanrahan et al. 1991] cannot be directly applied.

To understand why, consider the 1D example shown in Fig. 6. Segment $v_a v_b$ is illuminated by two light sources S_1 and S_2 . Assume that initially both light transfers are refined, and vertex v_1 is added. This results in the configuration of level 1 (Fig. 6). The light transfer with S_2 is further refined with the addition of v_2 on the right, thus splitting segment $v_1 v_b$. Finally, the light transfer from S_1 is refined on the left, with the addition of v_3 .

To determine the light contribution of S_1 in the interval $[v_1, v_b]$ we interpolate between the values transferred by $S_1 \rightarrow v_1$ and $S_1 \rightarrow v_b$, which are “represented” at level 1. However, for light S_2 , we must interpolate in the subinterval $[v_1, v_2]$ using the transfers determined by $S_2 \rightarrow v_1$ and $S_2 \rightarrow v_2$, and in the subinterval $[v_2, v_b]$ using the values determined by $S_2 \rightarrow v_2, S_2 \rightarrow v_b$, all of which are “represented” at level 2. Thus v_1 is shared between level 1 and level 2. As a consequence traditional push-pull procedures with gathering at elements rather than vertices cannot work, since they require that an element clearly belong to a certain hierarchy level.

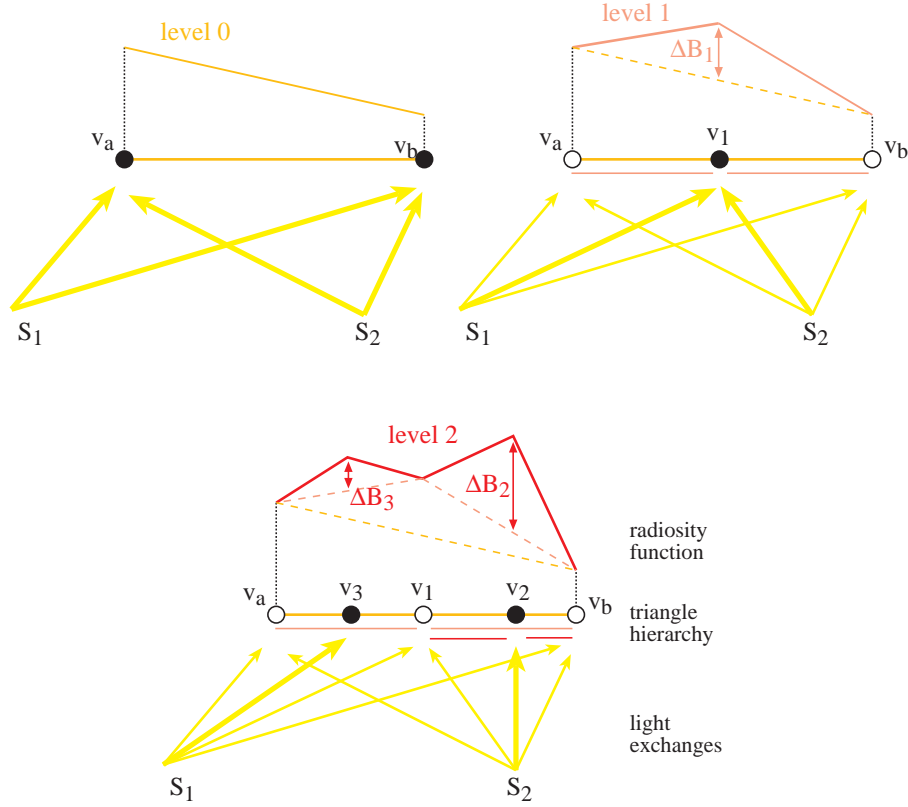


Fig. 6. Consistent multiresolution representation with lazy wavelets. Instead of storing radiosity values, we store the difference of the radiosity values at refined vertices.

A naive solution would be to duplicate vertex v_1 to differentiate exchanges simulated at different levels of the hierarchy. This however is not sufficient, since it is unclear how to perform the push operation. In particular, assume that we had one representation of v_1 for level 1 and one for level 2. It is unclear where the transfers $S_1 \rightarrow v_1$ and $S_2 \rightarrow v_1$ should be stored. If $S_1 \rightarrow v_1$ is stored at level 1, we can interpolate correctly in the interval $[v_1, v_b]$ to perform the push onto vertex v_2 . However the value will no longer be available at level 2 to enable the interpolation between v_3 and v_1 . In a symmetrical manner, we need $S_2 \rightarrow v_1$ to perform the interpolation at level 1 for the interval $[v_a, v_1]$ and the push on v_3 , and at level 2 for the interpolation in the interval $[v_1, v_2]$. With gathering at vertices and linear interpolation, it no longer makes sense to speak of a transfer at a given level of the hierarchy.

We use lazy wavelets to provide a solution to these problems. Instead of storing the actual radiosity value, at refined vertices we store the *radiosity difference* as shown in Fig. 6. This is the difference between the radiosity value at the current level and the interpolated value of the immediate ancestor. This provides a multi-resolution representation, since certain light transfers are refined more in the appropriate regions with the addition of new links.

The push procedure is then straightforward: To compute the total radiosity at a vertex, we interpolate the value of its ancestor, and add the radiosity difference. We obtain the total value at this vertex, which is thus recursively pushed down the hierarchy in a breadth-first manner.

This construction is directly applicable to the 2D case, by using barycentric coordinates (or bilinear for quadrilaterals) for the interpolation. We thus can simply perform a push operation on a hierarchical triangulation with gathering at the vertices.

Note however that it is slightly more involved to compute the difference of a light transfer than the total light transfer. Section 4.2.2 will deal with this problem through the use of “negative” links.

The pull computation is simpler, since we pull values to the triangles. At each triangle leaf, the value given is simply the average of values at the vertices (after the push). An intermediate node receives as a value the area-weighted average of its children triangles, as in standard hierarchical radiosity.

The advantages of this approach are that we can now create a consistent multi-resolution representation of radiosity over the hierarchical triangulation, while gathering at vertices. In addition, the push operation maintains a linear reconstruction of the radiosity function down to the leaf level.

4. VISIBILITY-DRIVEN HIERARCHICAL RADIOSITY: ALGORITHM AND DATA STRUCTURES

The hierarchical triangulation structure is one of the tools required to effect visibility-driven hierarchical radiosity. In particular, we can efficiently represent the irregular lighting discontinuities in a hierarchical structure. In addition, the information contained in the extended Visibility Skeleton provides *exact* and *global* visibility information. As a consequence, we can compute exact (analytical) area-to-point form factors for any light transfer, direct (primary) or indirect. The information contained in the Skeleton arcs (i.e. the visibility events affecting any light transfer) also allows the development of intelligent refinement criteria, again for any exchange of light.

In what follows we present our new algorithm which uses the extended Skeleton and the

hierarchical triangulations for efficient refinement and accurate light transfer.

4.1 Algorithm Outline

Our new algorithm is outlined in Fig. 7. It begins with the creation of the Visibility Skeleton for the given scene, using the improved link-based approach (Section 2.1.1). After this step, we have all the information available to calculate form-factors from each polygon to each (initial model) vertex in the scene. In addition, polygon-polygon visibility relationships are available directly from the skeleton, thus obviating the need for initial linking (i.e. only necessary links are created). After computing the form-factors of the initial polygons to the initial vertices, a “gather” step is performed to the vertices, followed by a “push-pull” process. In practice we perform a fixed number of iterations; however it would be possible to iterate to convergence, since these iterations are not computationally expensive.

Note that even at this very initial phase, the form-factors at the vertices of the scene are exact. To bootstrap subdivision, we first insert the maxima of the light source illumination functions into large receiver polygons (procedure *insertMaxima()*, see also Section 6.2).

```

visibilityDrivenHR
{
  computeSkeleton()      // compute the Visibility Skeleton
  computeCoarseLighting() // 3 gather push-pull
  insertMaxima()         // insert the maxima of light sources into meshes
  while( !converged() ) do
    subdividePolygons()  // Refine the polygons using visibility info
    refineLinks()        // Refine the links using visibility info
    gatherAtVertices()   // Gather at the vertices of the Hierarchical Triangulation
    pushPull()
  endwhile
}

```

Fig. 7. Visibility Driven Hierarchical Radiosity

Once the system has been initialized in this manner, we begin discontinuity based subdivision (*subdividePolygons()*) and link refinement (*refineLinks()*). Using the global visibility information, we are capable of subdividing surfaces by following “important” discontinuities. After the completion of each subdivision/refinement step, a gather/push-pull operation is performed, resulting in a consistent multi-resolution representation of light in the scene.

In the following discussion we use the terms “source” and “receiver” for clarity. A source is any polygon in the scene which emits or reflects light. For secondary or tertiary illumination, for example, “sources” will be polygons other than the primary light sources (e.g., the walls, ceiling or floor of a room).

In the rest of this section, we present the link data structures and discuss issues related to form-factor calculation and multi-resolution link representation. In Section 5 we describe the refinement process for links, and the details of visibility updates; in Section 6 we

present the polygon subdivision strategy and the perceptually-based refinement criterion used to effectively perform the subdivision.

4.2 Link Data Structures and Form-Factors

The central data structures used for our lighting solution are the links used to perform subdivision and light transfers. In contrast to previous hierarchical radiosity methods, two distinct link types are defined: point-polygon links which are used to gather illumination at vertices, and polygon-polygon links, which are used to make refinement decisions and to maintain visibility information while subdividing.

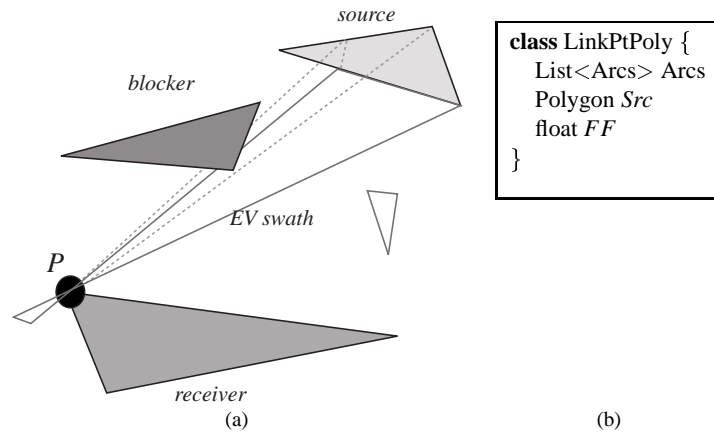


Fig. 8. (a) A point-polygon link used to gather illumination at vertex P . Note that all the arcs of the skeleton between P and the polygon *source* are stored with the link, e.g., the *EV swath* shown. (b) The corresponding data structure.

4.2.1 Point-Polygon Links. As mentioned above, the skeleton provides all the information required to calculate the exact area-to-point form-factor from any polygon in the scene to any vertex. By updating the view information as shall be discussed below (Section 5.3), we extend this capacity to new vertices created by subdivision.

There are numerous advantages to calculating illumination at vertices. When computing radiosity at patch centers, the result can be displayed as flat shaded polygons. To provide a more visually pleasing result, the radiosity values are usually first *extrapolated* to the patch vertices and then interpolated. Inevitably, this introduces many artifacts in the approximation of the original radiosity function. In addition, it is much cheaper and simpler to compute exact polygon-to-vertex form-factors than polygon-to-polygon form-factors. Computing radiosity at vertices was first introduced by Wallace *et al.* [Wallace et al. 1989] in the context of progressive refinement radiosity. For hierarchical radiosity, the fact that vertices can be shared between different levels renders gathering at vertices more complicated.

A point-polygon link and the corresponding data structure are shown in Fig. 8. The point-polygon links are stored at each vertex of the hierarchical triangulations. A point-polygon link stores the form-factor calculated, as well as the arcs of the visibility skeleton

(visibility events of the view) between the point and the polygon. An example is shown in Fig. 8, where the *EV* swath is stored with the link between point *P* and the source polygon.

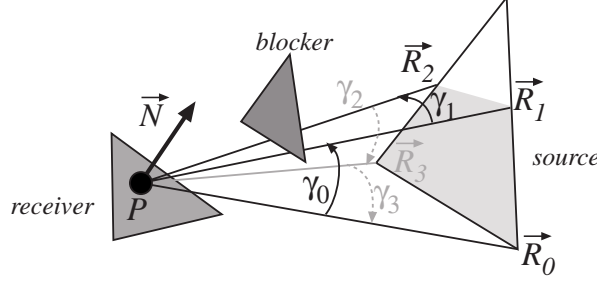


Fig. 9. Geometry for the calculation of a form factor

The point-area form factor is computed analytically using the formula in *e.g.* [Baum et al. 1989]. Consider Fig. 9.

$$F_{P,source} = \frac{1}{2\pi} \vec{N} \cdot \sum \gamma_i \frac{\vec{R}_i \times \vec{R}_{i+1}}{\|\vec{R}_i \times \vec{R}_{i+1}\|}$$

The sum is evaluated using the arcs of the skeleton stored in the point-polygon link. \vec{R}_i and \vec{R}_{i+1} correspond to the two nodes (extremal stabbing lines) of the arc.

Fig 10 shows an example of form-factor computation with the Visibility Skeleton; the computation is exact. For comparison, the average (relative) error is given using ray-casting and a jittered grid sampling on the source (both the kernel and visibility are evaluated by Monte-Carlo). Note that 36 rays are needed to have a mean error of 10%; numerical error on the form-factor is being measured. As expected from stratified sampling the convergence rate is about $O(n^{-\frac{3}{4}})$ [Mitchell 1996], since the function to be integrated is only piecewise continuous because of the visibility term. In Section 7.2.1 we will show the effect of this accuracy on the image quality.

4.2.2 Multi-Resolution Link Representation. To maintain the multi-resolution representation of radiosity in the hierarchical triangulation, we require the representation of ΔB as described in Section 3.2, for the push phase of the push-pull procedure.

When a new vertex is inserted into a receiver polygon, “negative links” are created, from the source to the three vertices of the triangle containing the newly inserted vertex[†]. These links allow the direct computation of ΔB as follows:

$$\Delta B = B_l - \sum_{i=0..2} c_i B_{nl}^i, \quad (1)$$

where B_l is the radiosity gathered from the positive link, B_{nl}^i is the radiosity gathered from the negative links and c_i are the barycentric coordinates of vertex P_i . An example of negative links is shown in Fig. 11(b).

[†]In practice, these are simply pointers to the previously existing links.

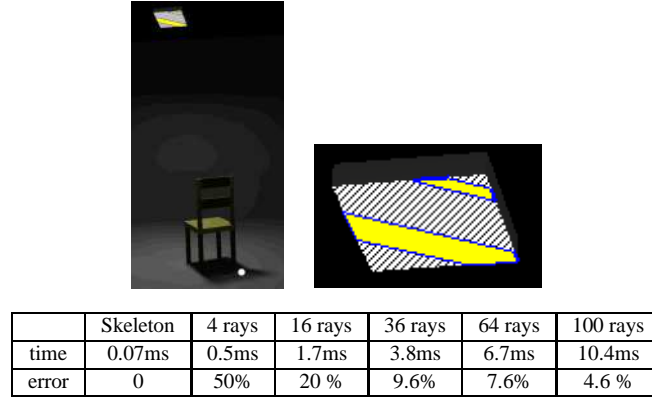


Fig. 10. Example of Form-Factor computation from the white point on the floor to the area light source using the visibility skeleton and ray-casting with jittered sampling. The hidden part of the source is hatched. The Visibility skeleton timing does not include the visibility update (about 0.13ms per link on average for this image).

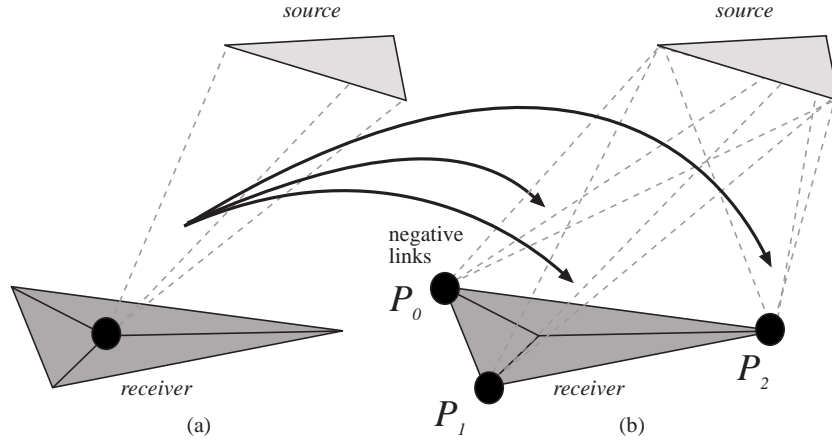


Fig. 11. (a) A newly inserted point (in black) and the point-polygon link to the source; the vertex points to the receiver. (b) three new negative links to the source used for the ΔB representation.

The entire gather/push-pull process is illustrated in Fig. 12. Note that the field *child* of an *HPolygon* is the associated triangulation.

4.2.3 Polygon-Polygon Links. The polygon-polygon link is used mainly to determine how well the light transfer is represented, in a manner similar to that of the links in previous hierarchical radiosity algorithms. This information is subsequently used in the refinement process as described below.

A polygon-polygon link stores visibility information via pointers to the point-polygon links (two sets of three links for a polygons pair) between each polygon and the vertices of the other polygon. A polygon-polygon link is illustrated in Fig. 13, with the corresponding point-polygon links from the source to the receiver.


```

Gather ()
{
  for each vertex  $v$ 
     $\Delta B_v = \text{gather}(\text{positive links}_v) - \text{gather}(\text{negative links}_v)$ 
}
Push (HPolygon poly)
{
  if child(poly) == NULL return
  for each vertex  $v$  in triangulation child(poly)
     $B_v = \Delta B_v + \text{interpolation}(\text{poly}, v)$ 
  for each triangle  $t$  in triangulation child(poly)
    Push( $t$ )
}
Pull (HPolygon poly)
{
  if child(poly) == NULL
     $B_{\text{poly}} = \text{average of } B_v, \text{ for all } v \text{ vertex of poly}$ 
    return
  for each triangle  $t$  in triangulation child(poly)
    Pull( $t$ )
   $B_{\text{poly}} = \text{average of } B_t, \text{ for all } t \text{ in child(poly)}$ 
}

```

Fig. 12. Gather and push-pull

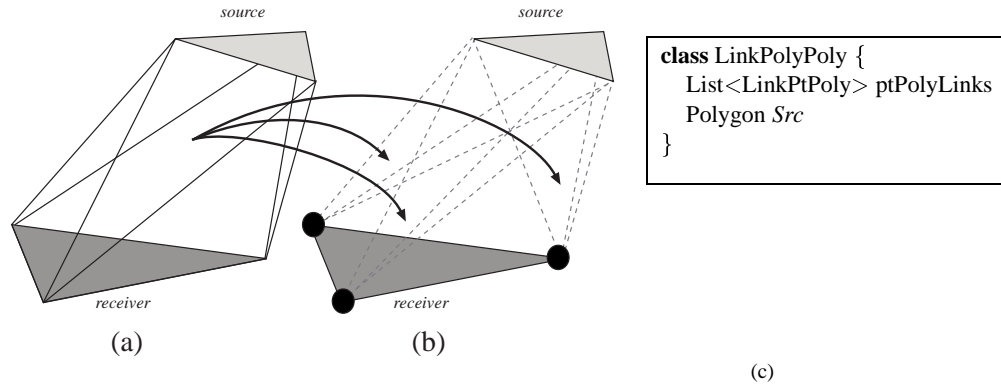


Fig. 13. (a) A polygon-polygon link used to estimate illumination transfer between two polygons (b) The polygon-polygon links store pointers to the 6 corresponding point-polygon links: 3 of them (*source* \rightarrow *receiver*) are shown here (c) The corresponding data structure.

Note that in the case of a subdivided polygon, all the neighboring triangles of a vertex v share all the point-polygon links related to v .

5. LINK REFINEMENT

Now that the link data structures have been described in detail, we can present the link refinement algorithm. Note that the process is slightly more involved than in the case of

standard hierarchical radiosity (e.g., [Hanrahan et al. 1991]), because the subdivision is not regular and the existence of the two link types requires some care to ensure that all updates are performed correctly. This section describes how the links are actually refined.

5.1 Refinement overview

Consider a light exchange from a source polygon to a receiver polygon. Because we gather radiosity from the polygon at the vertices, two kinds of refinement can be necessary.

- Source-refinement* if the radiosity variation over the source polygon is too high,
- Receiver-refinement* if the sampling on the receiver is too coarse.

The link refinement algorithm is straightforward: for each polygon and each of its polygon-polygon links, the link is tested for refinement. If the test fails, the link is refined and the new point-polygon and polygon-polygon links are created. Finally, the visibility of the link is updated. The refinement test uses a perceptually-based refinement criterion, based on the visibility information contained in the extended Skeleton (see Section 6.4). Note that since we compute exact point-to-area form factors, the source refinement cannot be caused by the inaccuracy of the form-factor computation. It can only happen because the radiosity of the source is not uniform, i.e., if a receiver-refinement has occurred on the source in another exchange.

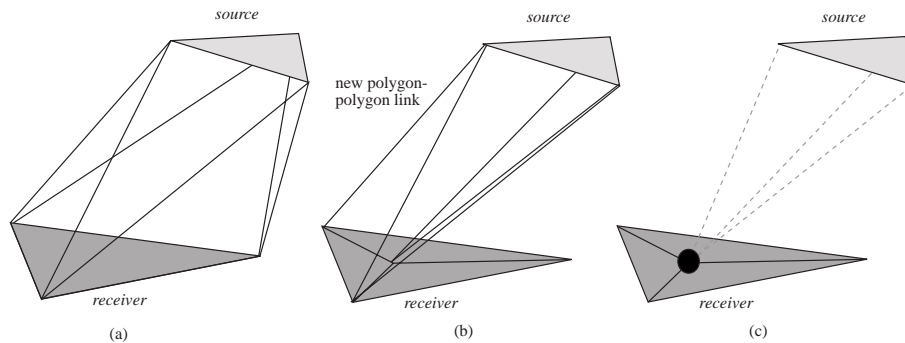


Fig. 14. Receiver refinement: (a) Original polygon-polygon link (b) Insertion of a point on the receiver and one of the three new polygon-polygon links created (c) The additional point-polygon link to the source.

5.2 Source and Receiver Refinement

The first type of refinement is that of a source. If the representation of radiosity across the source is considered insufficient for the given transfer (i.e., the variation of radiosity is too high across the source), the link will be refined. Note that the geometric subdivision of the source has occurred at a previous iteration, typically due to shadowing. New polygon-polygon links are created between the original receiver and the sub-triangles of the source. New point-polygon links are created for each vertex and each source sub-triangle, and the corresponding visibility data is correctly updated (see Section 5.3).

The second type of refinement is that of the receiver. For example, in Fig. 14, a point is added to the receiver. As a consequence, the triangulation is updated and three new

polygon-polygon links are added. One of these is shown in Fig. 14(b). In addition, a new point-polygon link is created, from the point added on the receiver to the source (Fig. 14(c)).

5.3 Visibility Updates

Each refinement operation requires an equivalent update in the visibility information contained in the point-polygon links. We again distinguish the two main cases, source refinement and receiver refinement.

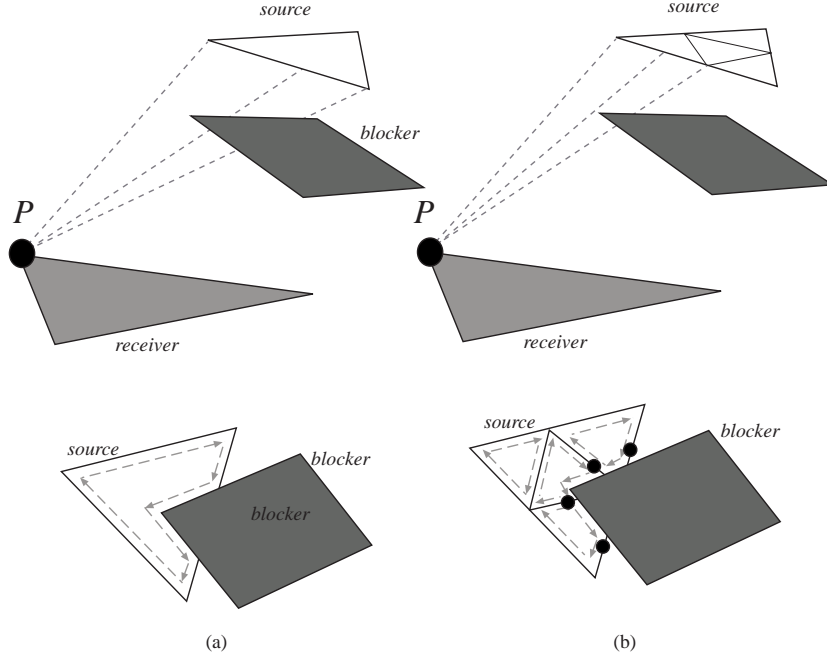


Fig. 15. Source visibility updates. The dashed arrows (lower part) represent the limits of the visible part of the source used to compute the form-factors. (a) The point-polygon link before subdivision and below the corresponding view of the source (b) One of the 4 new point-polygon links due to subdivision and the four new views. The black circles correspond to new nodes of the Skeleton.

In the case of source refinement we need to update the existing visibility information contained in the new point-polygon links. Since the visibility information of such a link can be represented by the view of the source from the receiver point of the link, all that needs to be done is the update of the link with respect to the new source sub-triangles. For example, in Fig. 15(a) the original view from point P is shown in the lower part of the figure. Once the polygon-polygon link is subdivided, four new views are computed, shown in the lower part of Fig. 15(b). The new point-polygon links now contain the references to the skeleton arcs (swaths), corresponding to the parts of the view affected. For example the leftmost source sub-triangle is completely unoccluded from P and thus no arcs are stored. For the others, the intersections of the previously existing arcs and the source sub-triangles result in new skeleton nodes (corresponding to the black circles in Fig. 15(b)). The

corresponding arcs are then subdivided. The new nodes are adjacent to these subdivided arcs. Note that all visibility/view updates are performed in 2D.

Refining a receiver is more involved. When adding a point to a polygon, a new view needs to be computed. We use the algorithm of the Skeleton construction which is robust. The only difference is that we use the blocker lists defined by the arcs stored in the initial point-polygon links instead of the entire model. Since the number of polygons in any given blocker list is relatively small, the cost of computing the new view is low. An example is shown in Fig. 16(a)-(b), where the point P is added to the receiver. In Fig. 16(b) we see the point and the new point-polygon link, and in Fig. 16(c) the newly calculated view is illustrated. The black circles correspond to newly created nodes of the skeleton.

The case of full visibility is detected using the information contained in the polygon-polygon links. The visibility update is then optimized: no new arc is computed and the unoccluded form-factor is used, thus saving time and memory.

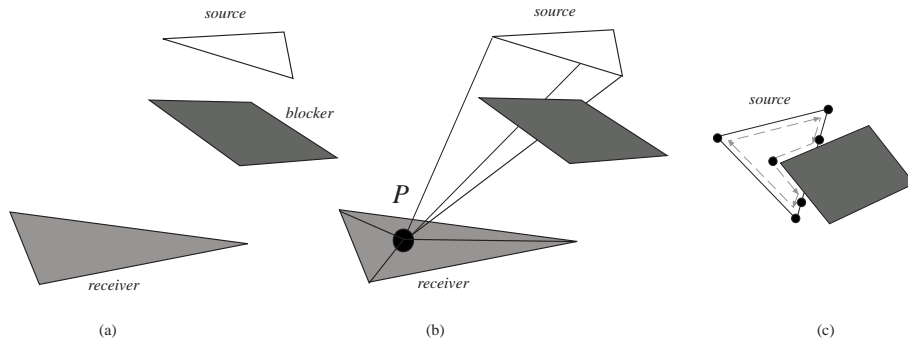


Fig. 16. Receiver visibility updates: (a) The initial configuration. Blocker information is contained in the source-receiver link. (b) A point is added to the receiver, creating a new point-polygon link. (c) The new view of the source computed at P . The blocker lists are updated using this computation.

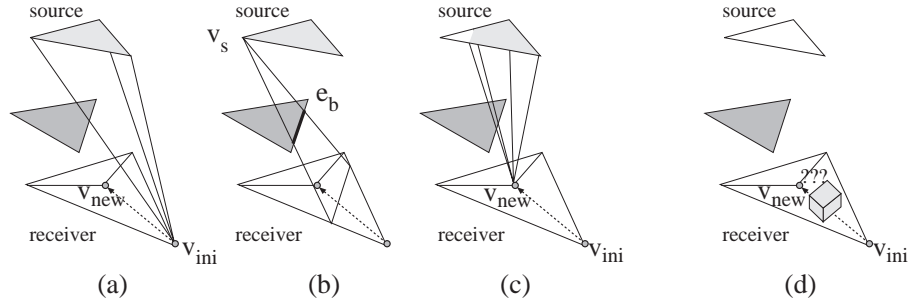


Fig. 17. Receiver refinement with visibility events (this solution was not implemented). (a) We start with a view at one of the initial vertices. (b) We walk across the receiver to the new vertex. Here we cross a $v_s e_b$ event. v_s begins to be hidden by the blocker. (c) We obtain the view at the new vertex. (d) In the case of touching objects, no information can be kept while crossing the interface between the two objects.

5.4 Alternative Visibility Updates

Visibility updates could be performed using the information encoded in the Skeleton, starting from the view at one of the initial vertices, then walking to the new vertex and updating the view each time a visibility event is crossed (see Fig. 17). This method is however hard to implement, and suffers from robustness problems if the visibility events are not crossed in a coherent (if not exact) order. Moreover, the case of touching objects complicates the problem furthermore since no information can be kept while walking “under” a touching object.

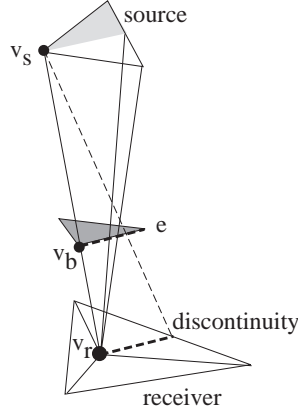


Fig. 18. Degeneracy due to discontinuity meshing. The receiver is split along discontinuity $v_s e$, causing a degenerate $v_s v_b v_r$ extremal stabbing line.

5.5 Treating Degeneracies

Subdividing along discontinuities induces degenerate viewpoints. For example in Fig. 18, we subdivide the receiver along the discontinuity $v_s e$. The view from v_r has a degenerate $v_s v_b v_r$ extremal stabbing line. To treat it coherently, we store with each vertex of the triangulation the extremal stabbing line which caused it (which is possibly null). This is a simpler and more robust alternative to the ray-casting modified for grazing objects described in Section 2.2. The treatment of the degeneracy then proceeds in the same manner.

A different alternative would have been to slightly perturb the point position to avoid those degeneracies. Two reasons have prevented us from doing so. First, discontinuity meshing allows us to delimit regions of umbra (full occlusions), regions of full visibility, and regions of penumbra. The two first region types require coarser subdivision than the latter. If we perturb the point position, some regions which should have been totally in the umbra will have a very small part in the penumbra, and need more subdivision. Second, point perturbation would cause numerical precision problems.

6. POLYGON SUBDIVISION

We have now seen how link and visibility information is updated during the light propagation process. Evidently, link updates are a consequence of a *refinement* decision, based on an appropriate criterion. We have chosen to use a *perceptually-based* refinement criterion.

In what follows, we first review basic concepts of perceptual mapping which we use for our refinement criteria. We next present the polygon subdivision process, and then detail the perceptually based refinement criterion which we have used for our algorithm.

6.1 Perceptual Just Noticeable Difference

The work in the field of perception provides us with two important features. First, it permits the conversion of radiometric quantities into displayable colors while preserving the subjective impression a viewer would have when observing the real scene. Second, it allows us to use error thresholds related to the error an observer is able to perceive, which are thus easy to set.

The human eye can deal with a very high dynamic range, while computer displays are usually limited to a 1 to $100\text{cd}/\text{m}^2$ range [Gibson and Hubbard 1997]. The eye adapts itself according to the luminosity of the scene being looked at. This explains why we are able to see dark night scenes as well as very luminous sunny scenes. The tone mapping operation deals with the transformation of high range radiometric quantities into low range display colors, while trying to provide the viewer with the same impression as the real scene. One obvious and simple method is to divide all quantities by the maximum radiosity of the scene. The problem with this approach is that if the light source intensity is halved, the scene will look exactly the same, though we would expect it to seem darker.

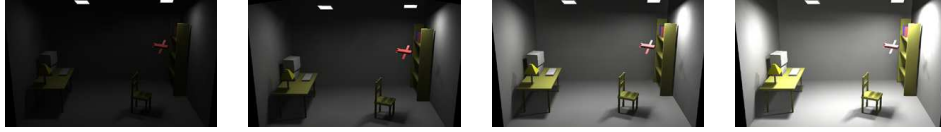


Fig. 19. Effect of Ward's tone-mapping on the same scene with different light source intensities. Note how details remain perceptible while the impression of darkness or luminosity is preserved.

Ward's contrast preserving tone mapping operator [Ward 1994] deals with this problem. A simple scaling factor sf is used for the whole scene which depends on the maximal displayable luminance L_{dmax} and the world adaptation level L_{wa} which is usually the logarithmic average of the scene luminosity without primary light sources. In what follows, all intensities are expressed in *candelas/meter*² (a *candela* is a *lumen/steradian* [Ward 1994]). The scaling factor sf is then given by

$$sf = \frac{1}{L_{dmax}} \left[\frac{1.219 + (L_{dmax}/2)^{0.4}}{1.219 + L_{wa}^{0.4}} \right]^{2.5}$$

Fig. 19 demonstrates the effect of this operator on a given scene with different source intensities.

We use a technique similar to that of Gibson *et al.* to compute the adaptation level [Gibson and Hubbard 1997]. We use a static adaptation level which is the average radiosity of the scene. However, since we use hierarchical radiosity as opposed to progressive radiosity, we do not have to rely on an estimate involving the average luminance and reflectance. Instead, at any step we use the average radiosity value of the polygons of the scene. This

is why we start the radiosity computation with several gather steps to compute a coarse estimate of the light distribution.

The use of a global static adaptation level is only a coarse approximation of the human visual system adaptation. As shown by Gibson *et al.* it gives a fairly good estimate of the dynamic adaptation level [Gibson and Hubbard 1997]. More elaborate solutions could be explored, such as the use of local adaptation levels computed using the average radiosity in the neighbourhood of an object, and more involved tone-mapping operators could also be used [Tumblin and Rushmeier 1993; Ferwerda et al. 1996] but this is beyond the scope of this article.

Once the tone mapping operation has been applied, the admissible error can be set as a given percentage of the maximum displayable intensity L_{dmax} . Psychovisual studies [Gibson and Hubbard 1997; Murch 1987] have shown that the human eye is able to distinguish a difference of 2%: this is the *just noticeable difference*. We will call the allowed error ϵ_{percep} , and in practice we will use $\epsilon_{percep} = 2\%$ for all our refinement criteria.

```

subdividePolygons() {
  for each polygon  $r$  and each poly-poly link  $s$  to  $r$ 
    if shouldRefine  $Link(s, r)$ 
      refine source
  for each polygon  $r$  and each poly-poly link  $s$  to  $r$ 
    if shouldRefine  $Link(s, r)$ 
      if iteration < 3
        regularSubdivision(  $r$  ) // perform grid-like subdivision
      else
        find and insert discontinuities in  $r$ 
      complete subdivision at this level // create sub-triangles in meshes
}

```

Fig. 20. Polygon Subdivision

6.2 Polygon Subdivision

Our experiments have shown that subdividing along the discontinuities during the first few subdivisions results in the creation of triangles with poor aspect ratios, inducing very visible artifacts. For this reason, subdivision of the polygons is performed using a two step strategy:

- During the first two subdivisions:* The polygons are subdivided in a regular grid-like manner. In particular, a regular grid is created as a function of size of the polygon being subdivided.
- During the third and subsequent subdivisions:* Insert shadow discontinuities or other illumination detail. Discontinuities are added as constrained edges, and result in a modified triangulation.

This approach is similar in spirit to the approaches of Stuerzlinger [Sturzlinger 1994] and Hardt and Teller [Hardt and Teller 1996] where the discontinuity meshing is, however, used for display purposes only. The polygon subdivision algorithm is outlined in Figure

20. In contrast to standard hierarchical radiosity, we cannot subdivide the polygons on-the-fly when a link needs subdivision, because polygon subdivision is not uniform and has to be performed along discontinuity curves. For this reason, we first consider all the polygon-polygon links for a given polygon to decide if it requires subdivision, and to determine which discontinuities will be inserted. After all discontinuities for a given receiver have been inserted, the CDT is completed.

6.3 Maxima insertion

If we consider the radiosity of a light source as a function defined over a receiver, it has been shown that subdividing a mesh used to represent illumination at the maximum of the function can increase the accuracy of the radiosity solution [Drettakis and Fiume 1993]. We thus first compute the maxima of the unoccluded radiosity functions of the light sources before the first refinement.

The maxima are computed only for important light-transfers (estimated using the disk-disk formula [Hanrahan et al. 1991] and the perceptual metric). Given a receiver and a polygon considered as a source, we use a gradient-descent algorithm to locate the maximum. Once the maximum is found, we compute the contribution of the source at this point; if it is above ϵ_{percep} the maximum is stored to be subsequently inserted in the mesh. The radiosity of the receiver polygon is updated to take this maximum into account. That is, a gather is performed at the maximum (before a link is created from it) to obtain a better estimate of the light distribution that will be used for the first refinement.

The maxima are inserted as a separate initial step during the first subdivision. The points of the regular subdivision which are too close to a maximum are not inserted. An example was shown in Fig. 4(b), where the maximum corresponds to the point on the lower left which is not exactly on the grid. We thus obtain nearly regular meshes with well shaped triangles.

The maxima-search process is applied iteratively to take indirect illumination into account. The insertion of maxima of indirect sources is very important for example in Fig. 23, where the table (illuminated by the lamp) is the most important light source for the upper part of the left wall.

6.4 Refinement Criterion

We distinguish two refinement criteria (or *oracles*): a radiometric criterion which accounts for the variation of the unoccluded radiosity, and a visibility (discontinuity) criterion. Moreover, the discontinuity criterion also guides the choice of the discontinuity curves to be inserted.

The radiometric oracle estimates if the linear interpolation of the light transfer is “accurate enough”. We sample the unoccluded form-factor (see [Baum et al. 1989] and Section 4.2.1) at the center of the patch and at the edge mid-points, and compare this to the linearly interpolated value. If the perceptually transformed difference is larger than ϵ_{percep} , we proceed with subdivision.

The principle of our visibility oracle is to estimate (as a percentage of the maximum displayable intensity) the “shadow amount” cast by the blockers, that is the radiosity that would be transferred without the blocker. Our refinement criterion thus has three steps: unoccluded estimate, “shadow amount” estimate and “shadow sharpness” estimate.

Consider a receiver and a source. Recall that a source is any polygon in the scene, considered as a source at this step of the refinement process. In what follows we refer to

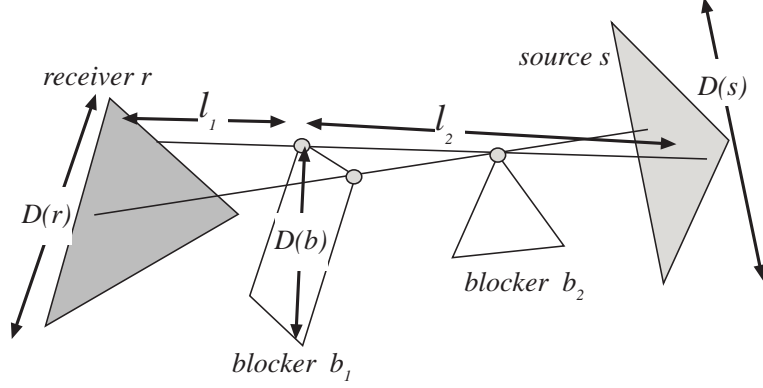


Fig. 21. Refinement criterion geometry

Fig. 21, for the definition of all geometric quantities.

First, we compute an estimate of the unoccluded light transfer B_{unoc} using the disk-disk formula [Hanrahan et al. 1991]. As above, if the estimate is less than ϵ_{percep} , the link will not be subdivided.

Second, we consider each visibility event between the source and the receiver, and estimate the “shadow amount”. To do this, we estimate the part of the source potentially hidden by the blocker by using the projected diameter of the blocker on the source to estimate its projected umbra:

$$D_{proj}(b) = D(b) * \frac{l_2}{l_1}$$

The estimated percentage of occlusion is then:

$$occlu = \frac{\frac{\pi}{4} D_{proj}(b)^2}{Area_{source}}$$

(clamped to 1). The “shadow amount” is:

$$shadow = B_{unoc} * occlu$$

If $shadow$ is below ϵ_{percep} , the visibility event is ignored.

Third, we estimate the sharpness of the shadow. The extent of the zone of penumbra is approximated by projecting the diameter of the source onto the receiver:

$$D(penumbra) = D(s) * \frac{l_1}{l_2}$$

If the size of the receiver is bigger than the zone of penumbra, then the receiver may contain regions where the source is completely visible, and regions where the blocker

projects entirely on the source. In the latter case, the fraction of occlusion is maximal and approximated by *occlu*. The variation of radiosity on the receiver is thus equal to *shadow*. Otherwise, we make the approximation that the radiosity varies linearly in the penumbra, and the variation of radiosity on the receiver is then:

$$\Delta(B) = \begin{cases} \textit{shadow} & \text{if } D(\textit{penumbra}) > D(r) \\ \textit{shadow} * D(r) / D(\textit{penumbra}) & \text{otherwise} \end{cases}$$

All the links containing visibility events with $\Delta(B) > \epsilon_{\textit{percep}}$ will be subdivided. As explained above, in the first two iterations subdivision will be regular. During the third and later iterations, subdivision is performed by inserting the discontinuities with the highest $\Delta(B)$. This is the *ranking* phase of our algorithm, similar in spirit to that of [Hardt and Teller 1996].

$\Delta(B)$ is computed using l_1 and l_2 at the two extremities of the visibility event, and taking the maximum. Note that the evaluation of these oracles is very rapid since the links and events are pruned as soon as we can decide that they will not cause subdivision.

7. IMPLEMENTATION AND RESULTS

7.1 Implementation

We have used the C++ Visibility Skeleton implementation of [Durand et al. 1997], with the extensions and changes described in Section 2. The scene polyhedra are represented using a winged-edge data-structure and a pre-processing step is performed to detect touching objects, which is a necessary step for the treatment of degeneracies.

The hierarchical triangulation has been incorporated into the same system. We use the public domain implementation of [Guibas and Stolfi 1985] by Dani Lischinski [Lischinski 1994] for the constrained Delaunay triangulation. Each subdivided polygon contains a triangulation QuadMesh.

On our test scenes, the algorithm spends most of its time on the visibility update, especially the calculation of the views at new vertices for the receiver refinement. For example, for the Desk scene of Fig. 22, for the last iteration, the computation of the criterion and the refinement of the mesh took 15 seconds, updating the visibility took 64 seconds, and the gather/push-pull took 2.5 seconds.

We have chosen not to use textures in our examples since they usually hide the accuracy of the lighting simulation.

7.2 Results

We present results for four different scenes. The first scene is a simple “Desk” scene, containing 438 polygons and two large, powerful light sources (see Fig. 22). This scene is used to illustrate the general functionality of our algorithm. The second scene contains the same geometry, but with 8 additional small, powerful light sources. The two large light sources have been turned down in intensity; we call this scene “Many Lights” (see Fig. 23). This scene shows how our approach treats the case of multiple light sources effectively. The third scene has been chosen to demonstrate the performance of our algorithm for mainly indirect lighting. We chose a common example of a bedroom lit exclusively by a small downwards-pointing, bed-side lamp. Most of the room is lit indirectly; this scene is called “Indirect” (see Fig. 25). Finally, simply in the interest of showing a completely different type of scene, we show the result of our approach on a “Village” scene, containing

buildings and cars. The scene is lit overhead by a rectangular light (see Fig. 28). In what follows we present various performance statistics as well as an informal comparison with hierarchical radiosity using quadtree subdivision, but with improved refinement and error bound strategies ([Gibson and Hubbard 1996; Lischinski et al. 1994]).

All times presented are in seconds on an R10000 195 MHz Silicon Graphics Onyx 2 workstation.

Before presenting the results for the complete algorithm, we present some interesting statistics concerning the importance of accurate visibility for form-factor computation.


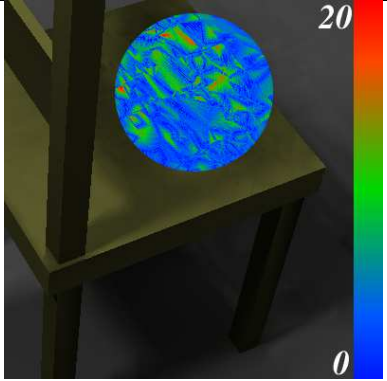
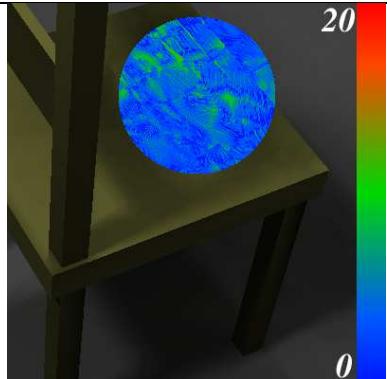
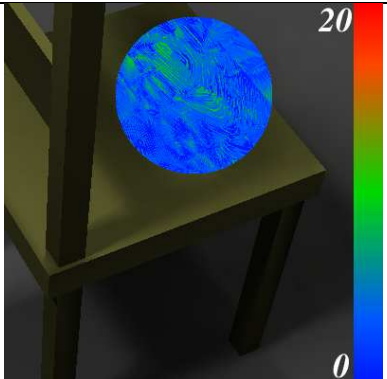
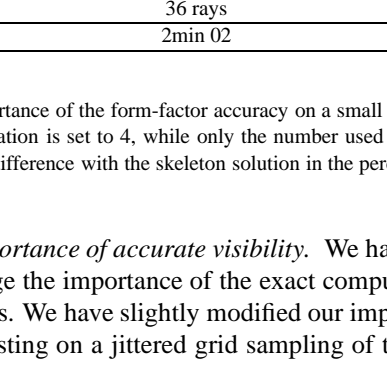
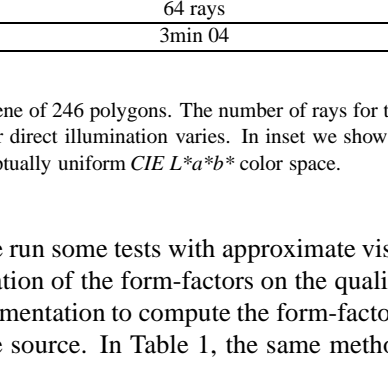


Image		
	Method	exact (Skeleton)
	Total time	1min 19
Image		
	Method	16 rays
	Total time	1min 17
Image		
	Method	36 rays
	Total time	2min 02
Image		
	Method	64 rays
	Total time	3min 04

Table 1. Importance of the form-factor accuracy on a small scene of 246 polygons. The number of rays for the indirect illumination is set to 4, while only the number used for direct illumination varies. In inset we show in false color the difference with the skeleton solution in the perceptually uniform $CIE L^*a^*b^*$ color space.

7.2.1 Importance of accurate visibility. We have run some tests with approximate visibility to judge the importance of the exact computation of the form-factors on the quality of the images. We have slightly modified our implementation to compute the form-factors using ray-casting on a jittered grid sampling of the source. In Table 1, the same method

is used for discontinuity-based mesh subdivision for all cases shown. It is performed using the Skeleton, and the cost of the skeleton construction and update is not included in the ray-casting timings, which report exclusively the cost of form-factor computation. As mentioned before (Fig. 10), inexact form-factor computation introduces significant error. The visual consequences of this can be seen in Table 1. Moreover, the computation overhead is significant if high quality is required because at least 64 rays are needed per form-factor.

Note that the effect on the images is particularly dramatic, because the subdivision induced by the discontinuity meshing is not uniform. The thin triangles introduce very visible artifacts. These results confirm those observed in [Drettakis and Sillion 1996].

<i>Scene</i>	<i>Pol</i>	<i>Skel</i>	<i>1st</i>	<i>2nd</i>	<i>3rd</i>	<i>Total</i>	<i>Mem(ini/tot)</i>	<i>links</i>	<i>tris</i>
Desk	444	2min 08	22s	16s	1min 14	4min	40/200MB	378K	46K
Many	492	2min 23	2min 38	55s	4min 27	10min 23	47/365MB	1546K	104K
Bed	534	4min 12	1min 25	58s	4min 35	11min 10	56/400MB	383K	43K
Village	312	45s	12s	7s	24s	1min 28	15/43MB	134K	28K

Table 2. Timing and memory results for the test scenes. The memory statistics shown are the initial memory usage for the skeleton *before* any subdivision, and the total memory used after the subdivision for lighting.

7.2.2 General Solution. The images of Fig. 22 show the initial steps of the algorithm as described previously. Fig. 22(a) is the result of three gather steps on the initial unsubdivided scene. Note that at this point we already have a very crude approximation of the global distribution of illumination in the scene, since the form-factors at the vertices are exact. In Fig. 22(b) we see the first step which is a regular grid together with the maxima of the light sources inserted into the mesh. Fig. 22(c) and (d) show the evolution of the algorithm after two iterations. The shaded images without the meshes are shown in (d). In (e) we show the discontinuities actually inserted. Note that these include discontinuities for all light transfers (direct and indirect) and that their number is much lower than that for a discontinuity meshing type approach (about 40% of the discontinuities caused by direct sources have been inserted).

In Table 2 we show the statistics of scenes computed using our method. For the “Desk” scene, we see that the total solution, including illumination, requires 4 minutes of computation. The quality of the solution is very high, including well-defined shadows on all surfaces. Note high quality shadows on the chairs and the table. The total number of point-polygon links is 378,746, and the number of leaf triangles is 46,058.

7.2.3 Treating Many Lights. One scene type for which our approach performs particularly well is that of multiple sources. This is demonstrated by our second test scene containing 10 lights and the same geometry as “Desk”. Fig. 23(a) shows an overview of the scene as rendered by our new approach, and Fig. 23(c) shows a closeup of the floor. The shadows due to the multiple sources are well represented in the areas when appropriate. The perceptually based ranking algorithm has correctly chosen the discontinuities that are of importance, since the combined influence of all sources is taken into account. This is shown by the small number of discontinuities present on the floor in Fig. 23(d). From Table 2 we see that 1.5 million links were used in this scene and the total computation time was 10 minutes 23 seconds. Only 10% of the direct discontinuity segments have been inserted.

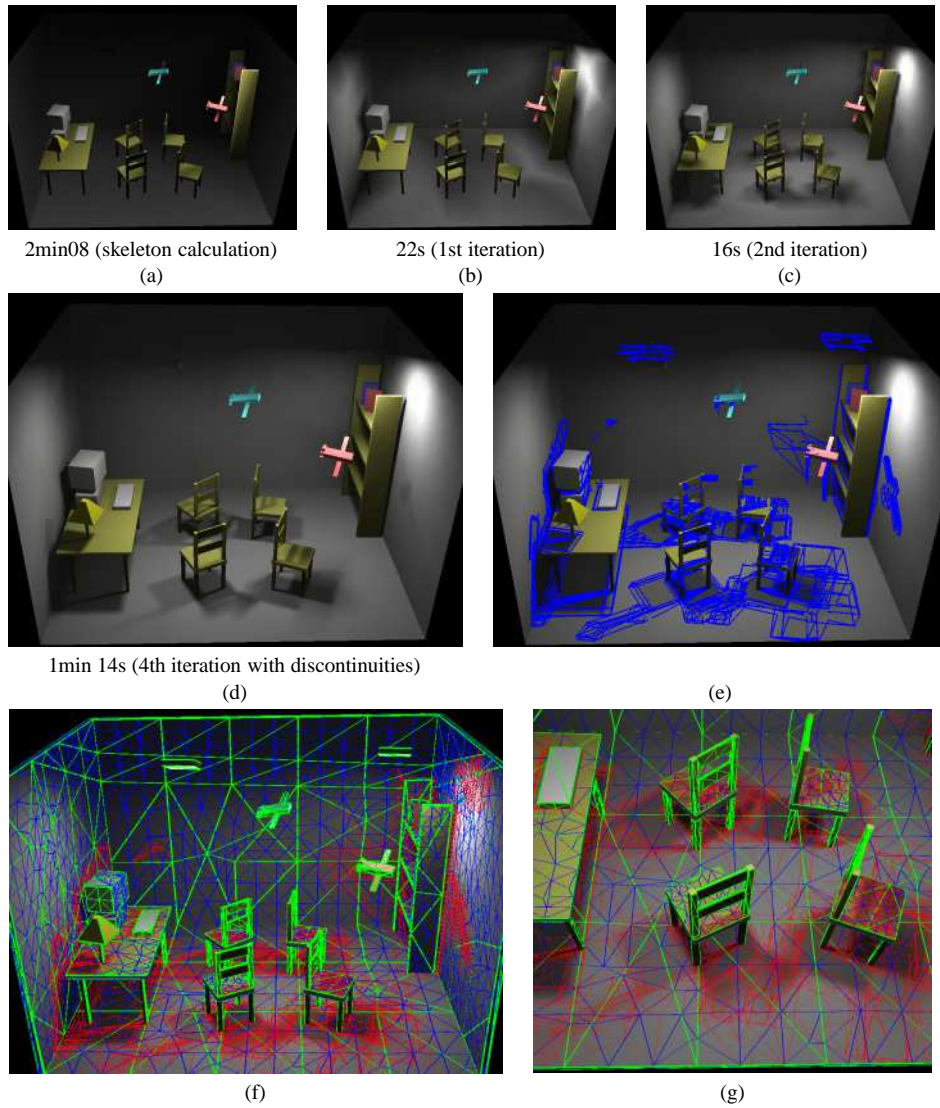


Fig. 22. Initial Desk Scene. In (a) we show the initial, unsubdivided scene. In (b) we show the first step which includes the grid and the maxima, in (c) we show the second iteration and (d) show the results of the third iteration which includes the discontinuity meshing. (e) shows the discontinuities actually inserted. (f) and (g) show the hierarchical triangular mesh (first level in green, second in blue, and third in red).

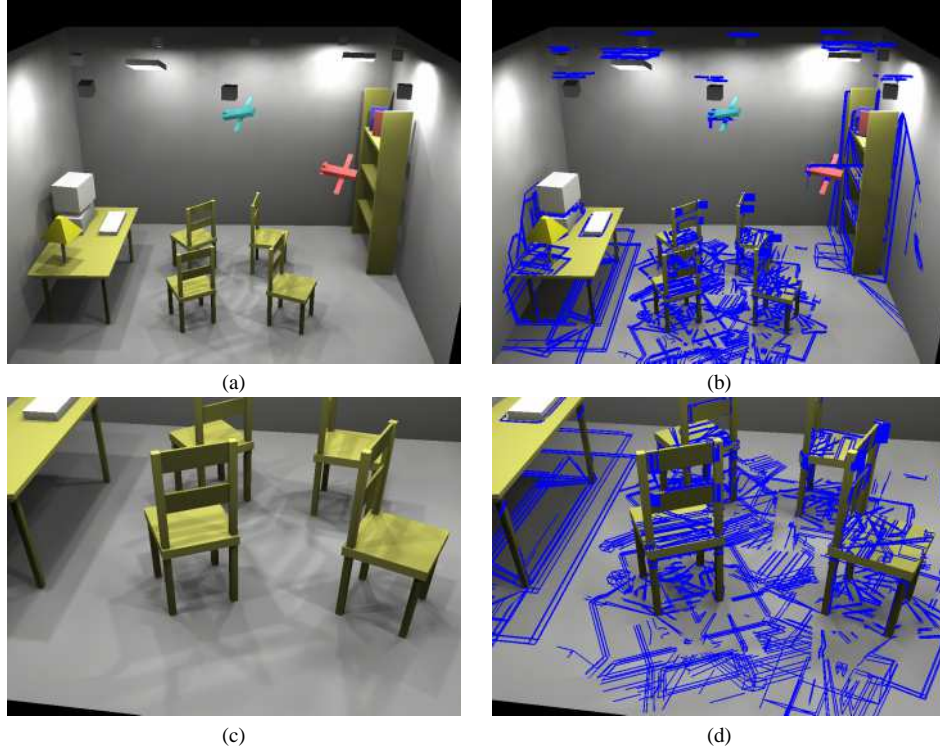


Fig. 23. Many Lights scene: (a) the final image, (b) the discontinuities actually inserted. (c) and (d) a closeup view of the floor.

As an informal comparison, we have compared to an implementation of hierarchical radiosity with clustering with the refinement proposed by Gibson and Hubbard [Gibson and Hubbard 1996] using the error bound propagation of Lischinski *et al.* [Lischinski et al. 1994]. For the Many lights scene computation with 1 million links, the computation time is almost 2 hours (Table 3). In addition, the quality of the results is lower, since the multiple shadows are much less sharp, or even missing (see Fig. 24). A much larger number of links would be necessary to compute an image of similar quality to Fig. 23 using hierarchical radiosity. Note that despite the fact that this method uses approximately the same number of elements (110K *vs.* 104K for our method), the quality of the resulting images is much lower.

<i>Scene</i>	<i>Pol</i>	<i>1st</i>	<i>2nd</i>	<i>3rd</i>	<i>4th</i>	<i>Total</i>	<i>Mem</i>	<i>links</i>	<i>elems</i>
Many	492	1 hr 25	22 min	10 min	-	1 hr 57	147 Mb	1098K	110K
Bed	534	11 min	37 min	6 min	25s.	54 min	94 Mb	903K	32K

Table 3. Comparative Timing and memory results for the test scenes using Hierarchical Radiosity with error bounds [Lischinski et al. 1994] and Gibson and Hubbard [Gibson and Hubbard 1996] refinement.

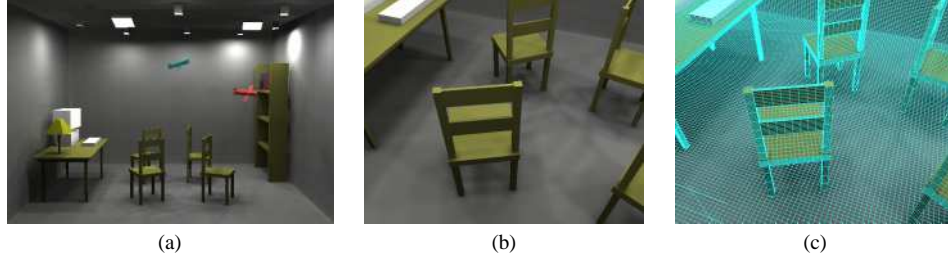


Fig. 24. Hierarchical Radiosity comparative results for Many Lights scene: (a) the final mesh, (b) a general view of the rendered scene (c) a closeup view of the floor.

7.2.4 Indirect Illumination. Accurate and efficient computation for indirect lighting is another challenge for our approach. It is for this type of scene that we see the power of our accurate form-factor and discontinuity ranking method. Previous approaches require significantly longer computation time to achieve this level of precision for secondary illumination.

This is illustrated with our third test scene (Fig. 25 and 26), in which light arrives from the bedside lamp which is pointing downwards only (no light leaves from the sides or the top of the lamp). Thus everything in the room above the level of the lamp is lit indirectly.

The algorithm uses a relatively small number of point-polygon links (383,715), and manages to represent shadows generated by secondary illumination. Notice for example the shadows of the right hand lamp or the books on the far wall in Fig. 25(d); these are caused by illumination of light bouncing off the bedside table and the bed.

Another informal comparison is presented, using the same algorithm as described above (based on [Gibson and Hubbard 1996; Lischinski et al. 1994]). Using almost a million links, hierarchical radiosity takes a slightly less than one hour, and produces lower quality results (see Fig. 27(a) and (b)).

Moreover, the advantages of our linear lazy-wavelet representation are well illustrated on the overall view of the hierarchical radiosity solution. The left part of the back wall is much lighter than the right part, with a strong discontinuity inbetween revealing the quadtree nature of the mesh. This is because interpolation is applied as a post-process at the finest level of subdivision; exchanges simulated at higher level are thus not correctly interpolated.

7.2.5 Village Scene. A final scene of a village is shown in Fig. 28, to show that the algorithm can be used for different scene types. Here the scene is lit overhead by a rectangle and also by the head and rear lights of the cars.

8. DISCUSSION

Our new approach shows promising results in what concerns the representation of accurate shadows, in particular for the cases of multiple sources and indirect lighting. However, the method presented is not without limitations. We believe that it is worthwhile to review what we consider to be the most important limitations and drawbacks as well as the most important advantages and contributions of our approach.

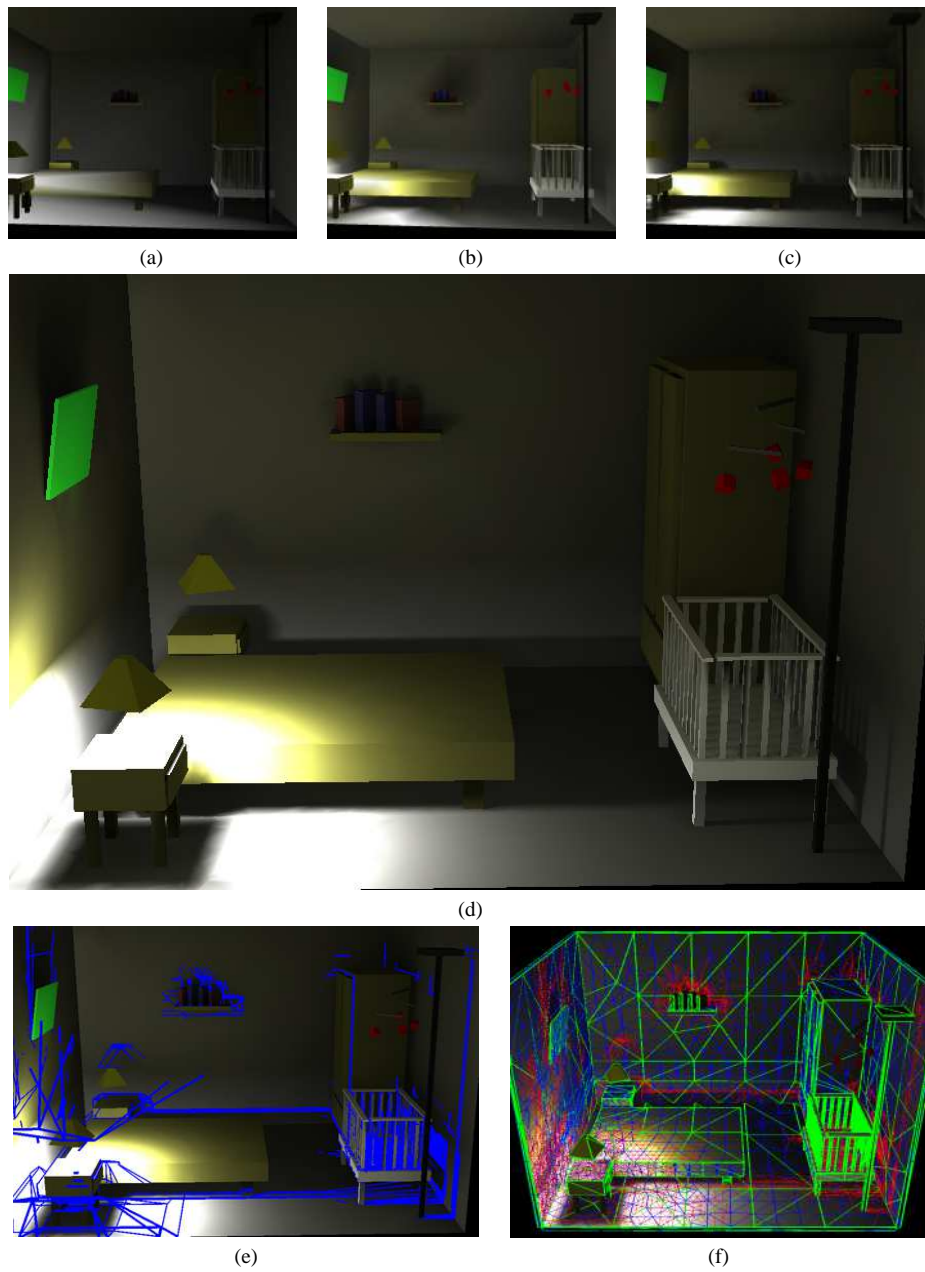


Fig. 25. Indirect lighting scene: (a) Initial solution, (b) first iteration (c) second iteration (d) final image (e) discontinuities inserted (the discontinuities inserted on the front wall are represented though this wall is backface-culled) (f) hierarchical triangulation.

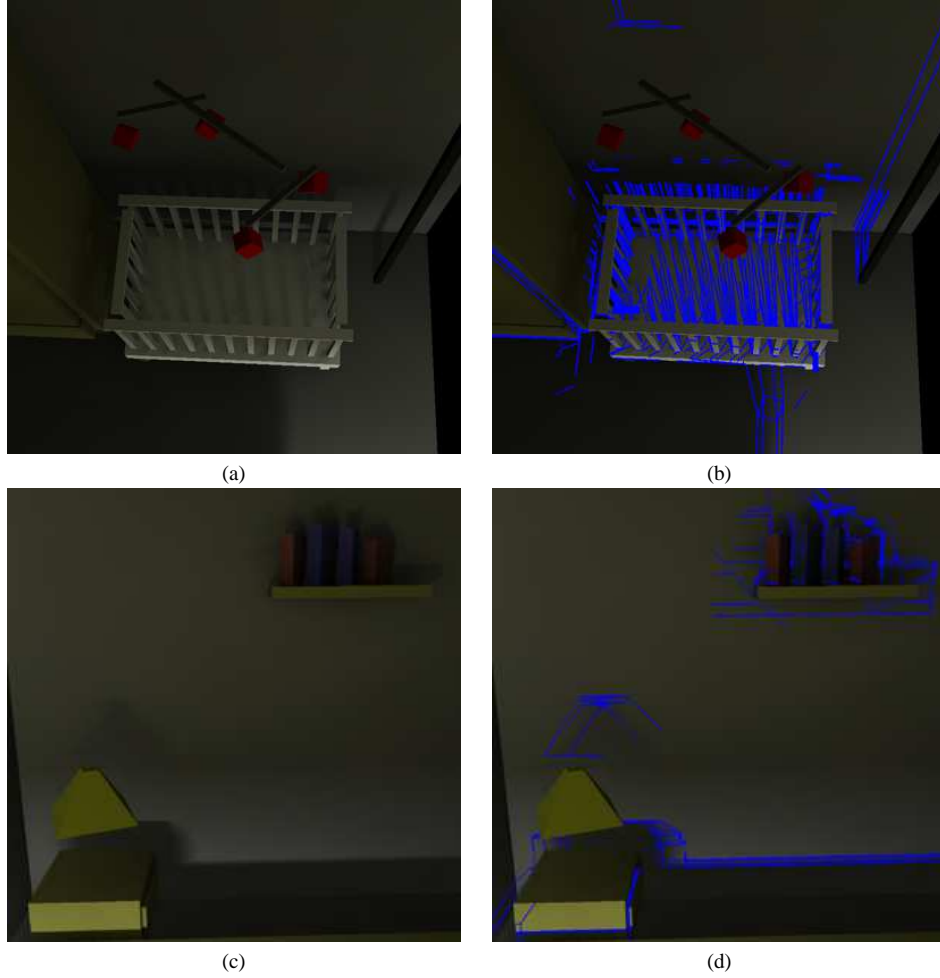


Fig. 26. Indirect lighting scene: (a) and (b) closeup of the right wall (c) and (d) closeup of the back wall. The lower part of the wall is directly illuminated by the left lamp (which is not visible on this image), while the upper part is indirectly illuminated by the left table. Note the indirect shadows cast by the books and the right lamp.

8.1 Limitations

Two major limitations of this work can be identified, the first is high memory consumption and the second is numerical robustness problems of the algorithms used.

The memory usage of the skeleton data structure is high, and can often have quadratic growth in the number of input polygons, depending on the how complex the visibility relations are between polygons. Even for simple environments, our method uses very large amounts of memory (see Table 2). To make our approach practical for large scenes, it is evident that we need to adopt one or a combination of the following strategies: lazy or on-demand skeleton construction, divide-and-conquer strategies (similar to *e.g.*, [Hardt and Teller 1996]) or a clustering approach allowing a multi-resolution representation. Some

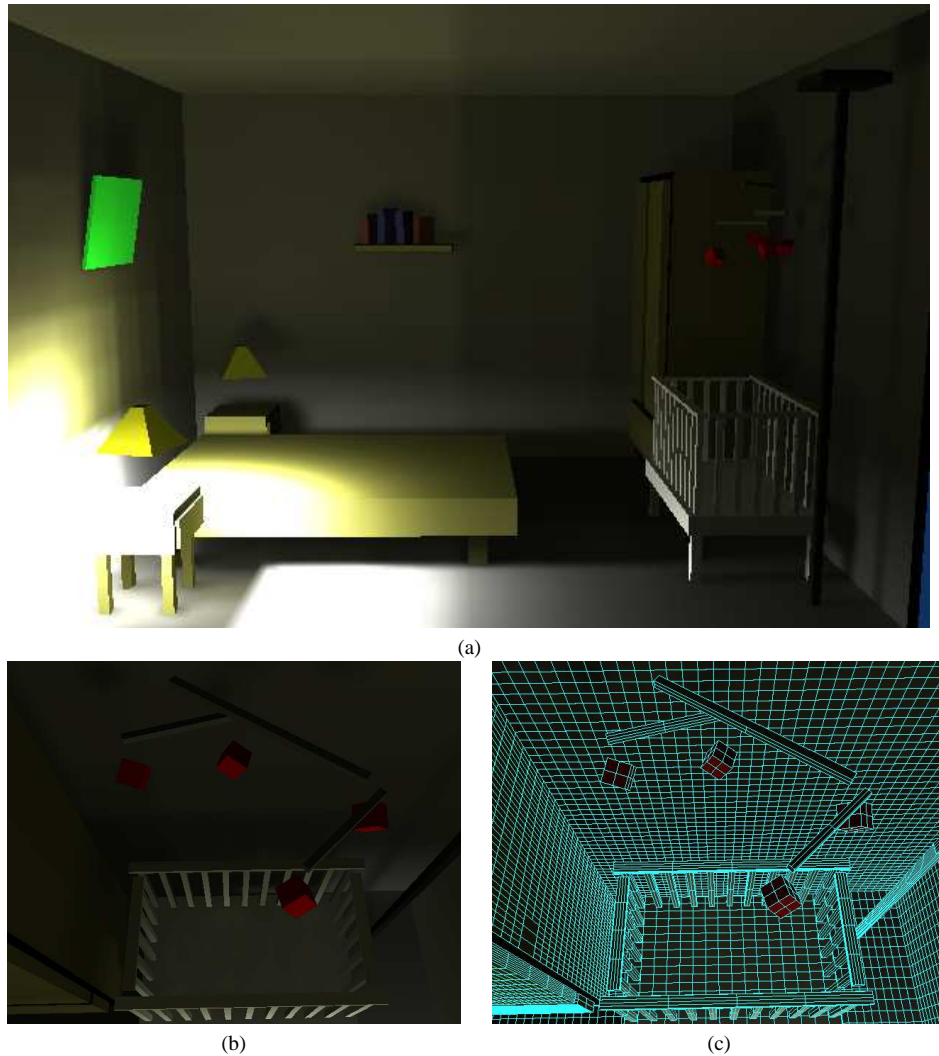


Fig. 27. Hierarchical Radiosity comparative results for Indirect Lighting scene: (a) the final image, (b) and (c) a closeup view of the right-hand wall.

ideas in these direction can be found in the Section 9 on future work.

Numerical robustness and the treatment of degenerate cases are important issues. Despite the simplicity of the construction algorithm which is based on ray-casting for node determination, degenerate cases can cause problems. As discussed in Section 2.2 we have been able to reliably treat most of these. Nonetheless, in the case of subdivision, many visual events coincide, causing problems of coherence both for the ray-tracing step (for node creation) and the adjacency determination. These problems are particularly evident in the case of view updates. A coherent and consistent treatment of degeneracies is planned, but is a research topic in itself and beyond the scope of this paper (see Section 9). Insertion

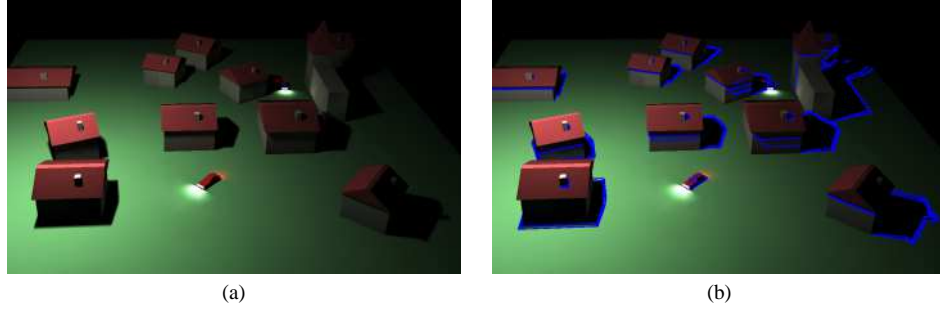


Fig. 28. Village scene (a) final image (b) discontinuities actually inserted

of points in the mesh also causes problems, especially during subdivision due to numerical imprecision. Symbolic calculations instead of numerical intersections could potentially resolve most of these problems.

8.2 Advantages

The visibility-driven hierarchical radiosity algorithm introduced here has many advantages. First we achieve visually accurate shadows using discontinuities and exact point-to-polygon form-factors, for both direct and indirect illumination. The new hierarchy of triangulations data-structure, the novel two link types and the multi-resolution point-area link representation allow accurate linear reconstruction of radiosity over irregular meshes. The global treatment of visibility and discontinuities permits the definition of an efficient refinement oracle. Using a perceptually based method to estimate shadow importance, our refinement algorithm has proven to be very efficient for previously hard-to-handle scenes such as scenes lit with multiple light sources and scenes lit mainly by indirect light. As part of an informal comparison, we have seen that Hierarchical Radiosity uses more computation time to produce much lower quality results, as would be expected.

Approaches such as that of [Lischinski et al. 1993] based on discontinuity meshing have difficulty with large numbers of light sources, since the number of discontinuities becomes unmanageable very quickly. This has consequences both on computation time and on robustness in the construction of the discontinuity mesh. For similar reasons, no discontinuity-based *hierarchical* lighting algorithm has been proposed previously in which discontinuities are treated for indirect light transfers.

9. CONCLUSION AND FUTURE WORK

We have presented a new hierarchical radiosity algorithm using the extended Visibility Skeleton. We have extended the Skeleton by replacing the n^2 table representation of the nodes and arcs by a structure of hierarchical links from polygons to polygons (and vertices to polygons). We have introduced update algorithms permitting the maintenance of consistent views at vertices added to a polygon due to subdivision, as well as the resulting sub-faces.

These extensions result in a powerful data structure which permits the computation of exact point-to-polygon form-factors for any vertex/polygon pair in the scene, and which provides detailed visibility information between any (sub)polygon-(sub)polygon pair.

We have introduced a novel hierarchical radiosity algorithm using this structure, based on a “lazy wavelet” or “sub-sampling” type multi-resolution representation. The basic data structure used is a non-uniform hierarchical triangulation, which consists of a hierarchy of embedded constrained Delaunay triangulations. By maintaining radiosity differences at subdivided vertices, we introduce a linear “push” step, resulting in higher quality radiosity reconstruction at the leaves. A new, perceptually-based, discontinuity driven refinement criterion has also been introduced, resulting in hierarchical subdivision of surfaces well adapted to shadow variations. The results of our implementation show that we can generate accurate high-quality, view-independent solutions efficiently. The results also show that our approach is particularly well suited to previously hard-to-handle cases such as multiple light sources and scenes lit almost entirely by indirect illumination.

Future Work

As was the case with the initial Skeleton work [Durand et al. 1997], memory usage remains the major limitation of the visibility skeleton. It is clear that a clustering-type approach is required, which will allow us to apply our algorithm to the parts of the scene where it is required. The idea would be to compute a visibility skeleton inside each cluster and approximate visibility skeletons between clusters. The challenge is to define this approximate skeleton, since clusters are not opaque objects.

Moreover, we believe that this clustering approach is a promising way of solving the robustness problem. If the objects are grouped into clusters of a given size, it is easier to set an epsilon for the computations inside this cluster and decide which error is acceptable. In addition, since the number of objects would be almost constant inside clusters, specific verification algorithms could be applied.

The advantage of the skeleton construction is that it is local, and thus can be built in a “lazy” or even “on-demand” fashion. Using to-be-defined criteria, we could compute only the parts of the visibility skeleton related to “important” light transfers. This information could be deleted once used, thus dramatically reducing the memory requirements.

The skeleton could also be used for Monte-Carlo methods. In the case of standard Monte-Carlo techniques, the inherent random nature of the sampling makes it hard to take coherence into account. However, more recent approaches such as quasi Monte-Carlo radiosity [Keller 1996], photon maps [Jensen 1996] or Metropolis light transport [Veach and Guibas 1997] could be coupled with the skeleton for a better exploration of the path space.

Extending the skeleton and the resulting illumination algorithm to dynamic scenes is another promising research direction. The notion of visual events can be extended to temporal visual events, for example when one line goes through five edges of the scene.

ACKNOWLEDGMENTS

We wish to thank Seth Teller for the discussions we had about visibility, Dani Lischinski for his insights on radiosity and discontinuity meshing, and François Sillion and Cyril Soler for all their answers about hierarchical radiosity. This research was funded in part by the European Union ARCADE Reactive LTR project #24944.

References

- BAUM, D. R., RUSHMEIER, H. E., AND WINGET, J. M. 1989. Improving radiosity solutions through the use of analytically determined form-factors. In J. LANE

- Ed., *Computer Graphics (SIGGRAPH '89 Proceedings)*, Volume 23 (July 1989), pp. 325–334.
- BEKAERT, P., NEUMANN, L., NEUMANN, A., SBERT, M., AND WILLEMS, Y. 1998. Hierarchical monte carlo radiosity. In G. DRETTAKIS AND N. MAX Eds., *Eurographics Rendering Workshop 1998* (New York City, NY, June 1998). Eurographics: Springer Wein.
- BOUATOUCH, K. AND PATTANAIK, S. N. 1995. Discontinuity Meshing and Hierarchical Multiwavelet Radiosity. In W. A. DAVIS AND P. PRUSINKIEWICZ Eds., *Proceedings of Graphics Interface '95* (San Francisco, CA, May 1995), pp. 109–115. Morgan Kaufmann.
- CHRISTENSEN, P. H., STOLLNITZ, E. J., SALESIN, D. H., AND DEROSE, T. D. 1996. Global illumination of glossy environments using wavelets and importance. *ACM Transactions on Graphics* 15, 1 (Jan.), 37–71. ISSN 0730-0301.
- DE FLORIANI, L. AND PUPPO, E. 1995. Hierarchical triangulation for multiresolution surface description geometric design. *ACM Transactions on Graphics* 14, 4 (Oct.), 363–411. ISSN 0730-0301.
- DRETTAKIS, G. AND FIUME, E. 1993. Accurate and consistent reconstruction of illumination functions using structured sampling. *Computer Graphics Forum (Eurographics '93)* 13, 3 (Sept.), 273–284.
- DRETTAKIS, G. AND FIUME, E. 1994. A fast shadow algorithm for area light sources using backprojection. In A. GLASSNER Ed., *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series (July 1994), pp. 223–230. ACM SIGGRAPH: ACM Press. ISBN 0-89791-667-0.
- DRETTAKIS, G. AND SILLION, F. 1996. Accurate visibility and meshing calculations for hierarchical radiosity. In X. PUEYO AND P. SCHRÖDER Eds., *Eurographics Rendering Workshop 1996* (New York City, NY, June 1996), pp. 269–278. Eurographics: Springer Wein. ISBN 3-211-82883-4.
- DURAND, F., DRETTAKIS, G., AND PUECH, C. 1996. The 3D visibility complex: A new approach to the problems of accurate visibility. In X. PUEYO AND P. SCHRÖDER Eds., *Eurographics Rendering Workshop 1996* (New York City, NY, June 1996), pp. 245–256. Eurographics: Springer Wein. ISBN 3-211-82883-4.
- DURAND, F., DRETTAKIS, G., AND PUECH, C. 1997. The visibility skeleton: A powerful and efficient multi-purpose global visibility tool. In T. WHITTED Ed., *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series (Aug. 1997), pp. 89–100. ACM SIGGRAPH: Addison Wesley. ISBN 0-89791-896-7.
- FERWERDA, J. A., PATTANAIK, S., SHIRLEY, P., AND GREENBERG, D. P. 1996. A model of visual adaptation for realistic image synthesis. In H. RUSHMEIER Ed., *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series (Aug. 1996), pp. 249–258. ACM SIGGRAPH: Addison Wesley. held in New Orleans, Louisiana, 04-09 August 1996.
- GIBSON, S. AND HUBBOLD, R. J. 1996. Efficient hierarchical refinement and clustering for radiosity in complex environments. *Computer Graphics Forum* 15, 5, 297–310. ISSN 0167-7055.
- GIBSON, S. AND HUBBOLD, R. J. 1997. Perceptually-driven radiosity. *Computer Graphics Forum* 16, 2, 129–141. ISSN 0167-7055.
- GIGUS, Z., CANNY, J., AND SEIDEL, R. 1991. Efficiently computing and repre-

- senting aspect graphs of polyhedral objects. *IEEE PAMI* 13, 6, 542–551.
- GORTLER, S. J., SCHRODER, P., COHEN, M. F., AND HANRAHAN, P. 1993. Wavelet radiosity. In *Computer Graphics Proceedings, Annual Conference Series, 1993* (1993), pp. 221–230.
- GUIBAS, L. AND STOLFI, J. 1985. Primitives for the manipulation of general subdivisions and computation of voronoi diagrams. *ACM Transactions on Graphics* 4, 2 (April), 74–123.
- HANRAHAN, P., SALZMAN, D., AND AUPPERLE, L. 1991. A rapid hierarchical radiosity algorithm. In T. W. SEDERBERG Ed., *Computer Graphics (SIGGRAPH '91 Proceedings)*, Volume 25 (July 1991), pp. 197–206.
- HARDT, S. AND TELLER, S. 1996. High-fidelity radiosity rendering at interactive rates. In X. PUEYO AND P. SCHRÖDER Eds., *Eurographics Rendering Workshop 1996* (New York City, NY, June 1996), pp. 71–80. Eurographics: Springer Wein. ISBN 3-211-82883-4.
- HECKBERT, P. 1992. Discontinuity meshing for radiosity. *Third Eurographics Workshop on Rendering*, 203–226.
- HEDLEY, D., WORRALL, A., AND PADDON, D. 1997. Selective culling of discontinuity lines. In J. DORSEY AND P. SLUSALLEK Eds., *Rendering Techniques '97* (8th EG workshop on Rendering, Saint Etienne, France, June 1997), pp. 69–81. Springer Verlag.
- JENSEN, H. W. 1996. Global illumination using photon maps. In X. PUEYO AND P. SCHRÖDER Eds., *Eurographics Rendering Workshop 1996* (New York City, NY, June 1996), pp. 21–30. Eurographics: Springer Wein. ISBN 3-211-82883-4.
- KELLER, A. 1996. Quasi-monte carlo radiosity. In X. PUEYO AND P. SCHRÖDER Eds., *Eurographics Rendering Workshop 1996* (New York City, NY, June 1996), pp. 101–110. Eurographics: Springer Wein. ISBN 3-211-82883-4.
- LISCHINSKI, D. 1994. Incremental delaunay triangulation. In P. S. HECKBERT Ed., *Graphics Gems IV*, pp. 47–59. San Diego, CA: Academic Press Professional.
- LISCHINSKI, D., SMITS, B., AND GREENBERG, D. P. 1994. Bounds and error estimates for radiosity. In A. GLASSNER Ed., *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series (July 1994), pp. 67–74. ACM SIGGRAPH: ACM Press. ISBN 0-89791-667-0.
- LISCHINSKI, D., TAMPIERI, F., AND GREENBERG, D. P. 1992. Discontinuity meshing for accurate radiosity. *IEEE Computer Graphics and Applications* 12, 6 (Nov.), 25–39.
- LISCHINSKI, D., TAMPIERI, F., AND GREENBERG, D. P. 1993. Combining hierarchical radiosity and discontinuity meshing. In *Computer Graphics Proceedings, Annual Conference Series, 1993* (1993), pp. 199–208.
- MARTIN, I., PUEYO, X., AND TOST, D. 1997. An Image Space Refinement Criterion for Linear Hierarchical Radiosity. In *Proceedings of Graphics Interface '97* (San Francisco, CA, May 1997). Morgan Kaufmann.
- MITCHELL, D. P. 1996. Consequences of stratified sampling in graphics. In H. RUSHMEIER Ed., *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series (Aug. 1996), pp. 277–280. ACM SIGGRAPH: Addison Wesley. held in New Orleans, Louisiana, 04-09 August 1996.
- MURCH, G. 1987. Color displays and color science. In H. J. DURRER Ed., *Color and the Computer*, pp. 1–25. Boston: Academic Press.

- PLANTINGA, H. AND DYER, C. 1990. Visibility, occlusion, and the aspect graph. *International Journal of Computer Vision* 5, 2, 137–160.
- REICHERT, M. C. 1992. A two-pass radiosity method driven by lights and viewer position. Master's thesis, Program of Computer Graphics, Cornell University.
- STEWART, A. J. AND GHALI, S. 1994. Fast computation of shadow boundaries using spatial coherence and backprojections. In A. GLASSNER Ed., *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series (July 1994), pp. 231–238. ACM SIGGRAPH: ACM Press. ISBN 0-89791-667-0.
- STOLLNITZ, E. J., DE ROSE, T. D., AND SALESIN, D. H. 1996. *Wavelets for Computer Graphics: Theory and Applications*. Morgan Kaufmann, San Francisco, CA.
- STURZLINGER, W. 1994. Adaptive mesh refinement with discontinuities for the radiosity method. In *Fifth Eurographics Workshop on Rendering* (Darmstadt, Germany, June 1994), pp. 239–248.
- TAMPIERI, F. 1993. *Discontinuity Meshing for Radiosity Image Synthesis*. Ph. D. thesis, Cornell University, Ithaca, NY.
- TELLER, S. AND HANRAHAN, P. 1993. Global visibility algorithms for illumination computations. In *Computer Graphics Proceedings, Annual Conference Series, 1993* (1993), pp. 239–246.
- TELLER, S. J. 1992. Computing the antipenumbra of an area light source. In E. E. CATMULL Ed., *Computer Graphics (SIGGRAPH '92 Proceedings)*, Volume 26 (July 1992), pp. 139–148.
- TUMBLIN, J. AND RUSHMEIER, H. E. 1993. Tone reproduction for realistic images. *IEEE Computer Graphics and Applications* 13, 6 (Nov.), 42–48. also appeared as Tech. Report GIT-GVU-91-13, Graphics, Visualization & Usability Center, Coll. of Computing, Georgia Institute of Tech.
- UREÑA, C. AND TORRES, J. C. 1997. Improved irradiance computation by importance sampling. In J. DORSEY AND P. SLUSALLEK Eds., *Eurographics Rendering Workshop 1997* (New York City, NY, June 1997), pp. 275–284. Eurographics: Springer Wien. ISBN 3-211-83001-4.
- VEACH, E. AND GUIBAS, L. J. 1997. Metropolis light transport. In T. WHITED Ed., *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series (Aug. 1997), pp. 65–76. ACM SIGGRAPH: Addison Wesley. ISBN 0-89791-896-7.
- WALLACE, J. R., ELMQUIST, K. A., AND HAINES, E. A. 1989. A ray tracing algorithm for progressive radiosity. In J. LANE Ed., *Computer Graphics (SIGGRAPH '89 Proceedings)*, Volume 23 (July 1989), pp. 315–324.
- WARD, G. 1994. A contrast-based scalefactor for luminance display. In P. HECKBERT Ed., *Graphics Gems IV*, pp. 415–421. Boston: Academic Press.
- ZATZ, H. R. 1993. Galerkin radiosity: A higher order solution method for global illumination. In *Computer Graphics Proceedings, Annual Conference Series, 1993* (1993), pp. 213–220.

Éclairage pour des scènes de grande complexité

Les algorithmes développés en image de synthèse ne sont réellement applicables que s'ils sont capables de traiter des scènes de grande complexité. Plus nous sommes capables de traiter des scènes complexes, plus il est possible d'atteindre une performance interactive. En ce qui concerne le rendu et la simulation de l'éclairage, la complexité a plusieurs formes : La complexité géométrique, c'est-à-dire le nombre d'objets dans une scène, ou la complexité photométrique, c'est-à-dire la simulation de différents phénomènes d'éclairage, comme la réflexion spéculaire, la réfraction etc, ou les sources non-diffuses comme le soleil.

Nous avons étudié les deux aspects de la complexité pour le rendu, en commençant par la complexité géométrique et ensuite en développant des algorithmes pour l'éclairage non-diffus. Pour la complexité géométrique, nos travaux se distinguent entre le travail sur les structures de données, surtout pour le tracer de rayons, et ensuite sur des travaux sur la radiosité pour les scènes très complexes.

3.1 Structures de Données Hiérarchiques pour le lancer de Rayons

Pour tout algorithme traitant des scènes complexes, il est souvent utile d'avoir une structure de subdivision spatiale, permettant de localiser une opération dans l'espace de façon efficace. Dans cet esprit, en collaboration avec le doctorant Frédéric Cazals et son directeur de thèse Claude Puech, nous avons développé une structure de données adaptée aux environnements très complexes (des centaines de milliers d'objets) pour le lancer de rayons. Dans la nouvelle méthode [CDP95] nous présentons une approche différente des algorithmes précédents en construisant une structure par la segmentation des données en groupes d'objets de même taille, ensuite le regroupement des objets de la même taille qui sont proches et enfin la construction d'une structure hiérarchique de grilles contenant les objets regroupés. Ceci est illustré dans la Figure 3.1(a), où l'on montre une structure de grille récursive « traditionnelle » et dans (b) où nous montrons la nouvelle structure de la hiérarchie de grilles uniformes.

La structure construite donne des résultats très satisfaisants par rapport aux structures précédentes en termes de mémoire utilisée ainsi qu'en temps de calcul. Dans la Figure 3.2 nous montrons les résultats des tests sur des modèles d'un bâtiment composé de plusieurs cuisines (donc contenant des objets avec une grande différence d'échelle). Le nombre des polygones est montré sur l'axe horizontal. Nous voyons que le temps de rendu de la nouvelle structure est au même niveau que la structure de grille récursive, mais en utilisant toujours moins de mémoire.

Enfin, contrairement aux approches précédentes, la nouvelle structure ne nécessite pas la définition de plusieurs paramètres de construction, ce qui facilite son utilisation.

Avec mon doctorant Eric Paquette (co-dirigé en co-tutelle par Pierre Poulin à l'université de Montréal), nous avons développé une nouvelle structure de données pour traiter le rendu par lancer de rayons pour des scènes contenant plusieurs sources de lumière [PPD98]. Cette structure est illustrée dans la Figure 3.3. Une

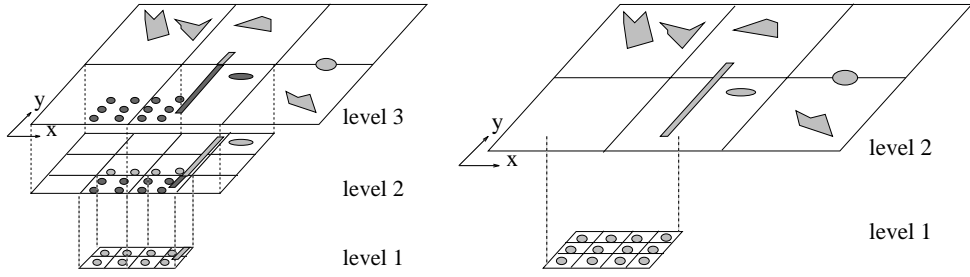


FIG. 3.1: (a) Grille recursive

(b) Hierarchie de Grilles Uniformes

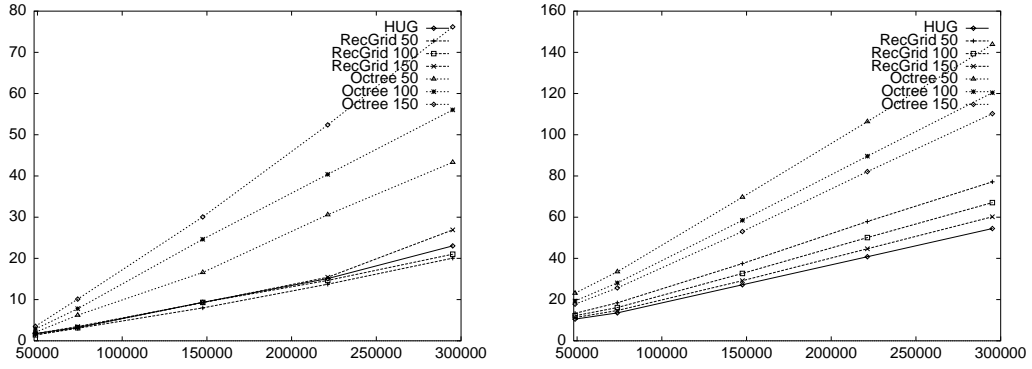


FIG. 3.2: (a) Temps du rendu et (b) mémoire en MB pour le modèle de plusieurs cuisines [CDP95]

structure type « octree » de sources est construite à partir d'un ensemble des sources ponctuelles. Chaque nœud intérieur représente les enfants par une source ponctuelle ayant une intensité et une position égales à la moyenne des enfants. Un critère d'erreur a été développé pour choisir le niveau de la hiérarchie utilisé pour faire le rendu à un point.

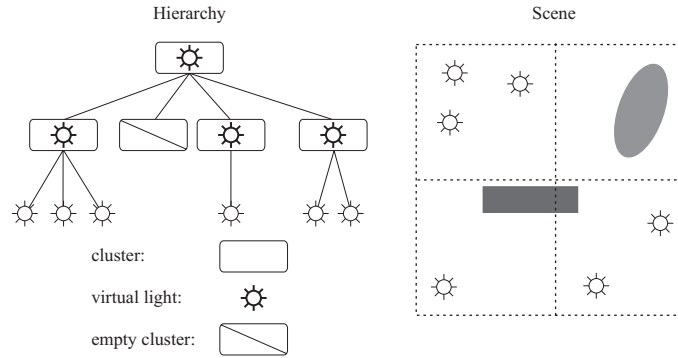


FIG. 3.3: La Hiérarchie de Sources Lumineuses pour le tracer de rayons de scènes contenant des milliers de lumières [PPD98].

La hiérarchie des sources lumineuses est particulièrement bien adaptée pour des scènes contenant plusieurs milliers de sources. Des exemples sont les rues éclairées la nuit, les arbres de Noël etc. Pour des scènes de ce type, nous avons effectué des tests ; les résultats sont présentés dans la Figure 3.4 où nous observons une augmentation de temps de calcul quasi-logarithmique avec le nombre de sources (axe horizontal).

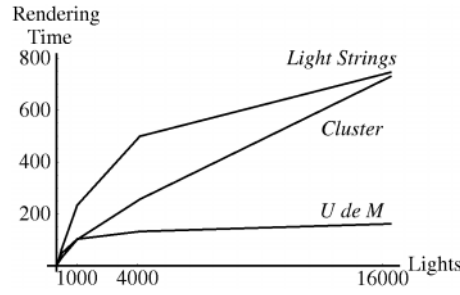


FIG. 3.4: Comportement logarithmique de la méthode de la hiérarchie de sources lumineuses.

3.2 Algorithmes Hiérarchiques d'éclairage

La radiosité hiérarchique introduite en 1991 [HSA91], a marqué un tournant dans les algorithmes d'éclairage, car elle permettait le traitement de l'éclairage global d'une manière beaucoup plus efficace que les approches précédentes [CGIB86, CCWG88]. Dans nos recherches, nous avons abordé des problèmes liés à ces algorithmes, dans le but de leur amélioration en temps de calcul et utilisation de mémoire.

La radiosité hiérarchique comporte plusieurs étapes. D'abord, des *liens* de transfert d'énergie sont établis entre les surfaces de la scène. Ensuite, un critère d'erreur guide ce qu'on appelle le « raffinement » de ces liens ; une fois le niveau de raffinement décidé, la lumière est transportée à travers les liens (« gather » en anglais). En dernière étape, une représentation multi échelle est maintenue, en parcourant la hiérarchie de haut en bas (en « poussant » les valeurs de radiosité vers les feuilles de la hiérarchie) et de bas en haut en moyennant les valeurs de bas en haut (« push-pull »).

3.2.1 Étude et Améliorations de la Radiosité Hiérarchique

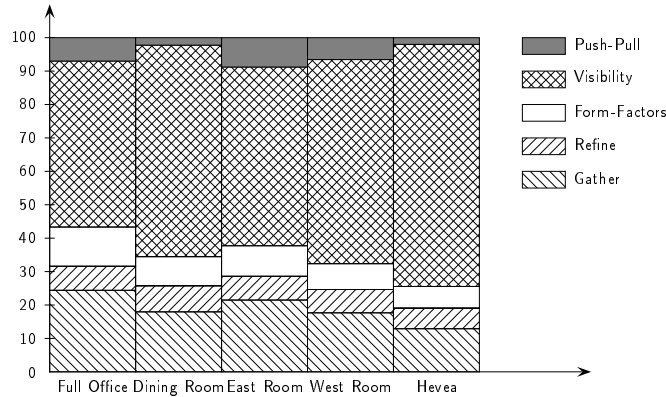


FIG. 3.5: Temps passé dans chaque étape de la radiosité hiérarchique [HSD94].

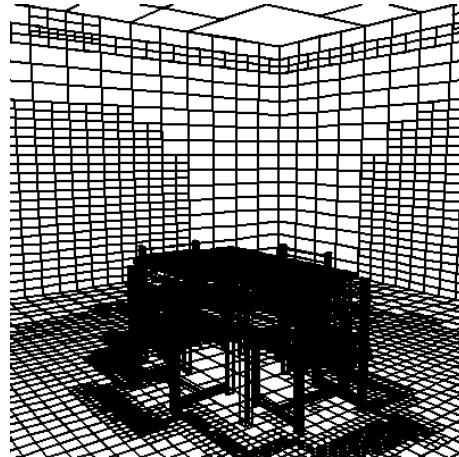
Avec le doctorant Nicolas Holzshuch et son directeur François Sillion nous avons d'abord étudié le comportement de l'algorithme de la radiosité hiérarchique, et ensuite développé un nouvel algorithme qui apporte deux améliorations importantes à la méthode originale de la radiosité hiérarchique [HSD94].

Dans le contexte de l'étude, nous avons constaté que la plupart du temps de la solution est passé dans les calculs de visibilité (voir Figure 3.5).

En ce qui concerne les améliorations, l'étape de complexité $O(n^2)$ d'établissement des liens énergétiques entre chaque paire d'objets est retardée en évitant l'établissement d'un lien jusqu'au moment où cela devient nécessaire. En pratique, de nombreux liens potentiels ne deviennent jamais significatifs, le résultat étant une accélération important du temps de calcul. La deuxième amélioration évite la subdivision inutile du maillage en utilisant un critère de similarité entre facteurs de formes (la mesure d'échange d'énergie entre surfaces). Cette amélioration est illustrée dans la Figure 3.6.



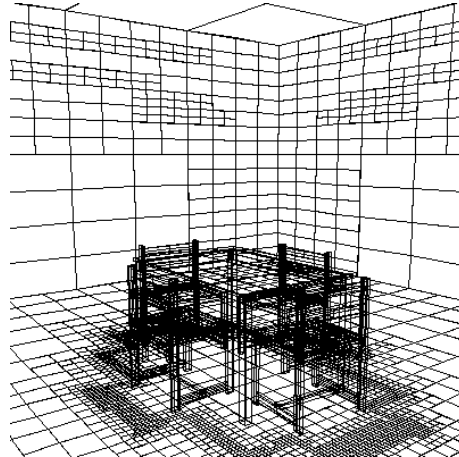
1. Radiosité Hiérarchique



2. Le maillage d'origine



3. Algorithme Amélioré



4. Le maillage réduit

FIG. 3.6: Résultats de l'algorithme de réduction des liens.

Pour éviter complètement l'étape de l'établissement initial des liens, il existe une autre approche : regrouper des objets en *groupes* (« clusters » en anglais) et traiter les échanges énergétiques entre les groupes au lieu de traiter chaque surface ou objet indépendamment [Sil95].

La complexité de cette généralisation est telle, qu'il devient maintenant difficile à comprendre comment déterminer les paramètres nécessaires pour effectuer une simulation d'éclairage. Dans le cadre d'un projet européen (ARCADE ESPRIT #24944), nous avons effectué une étude expérimentale sur des scènes types, tirées d'utilisations industriels [HDSD99].

Cette étude porte sur les différents algorithmes de construction de clusters (KDT et variantes, et des algorithmes basés sur la proximité des objets). Si on montre le graphe du temps de calcul de la solution (en augmentant la qualité requise par l'utilisateur) par rapport à l'erreur qui résulte, nous observons que différentes scènes ont des comportements différents (voir Figure 3.7). Nous n'avons pas pu conclure sur tous les aspects des algorithmes, mais nous avons pu constater que pour deux types de scènes, les scènes avec les objets « organisés », comme les bureaux (scènes OFFICE et VRLAB de la Figure 3.7) et les « soupes de polygones » (scènes AIRCRAFT et CAR de la Figure 3.7), différents algorithmes semblent convenir plus ou moins. Les scènes de type « organisées » sont mieux traitées par les algorithmes examinés. Les scènes de type « organisées » sont mieux traitées par les algorithmes examinés.

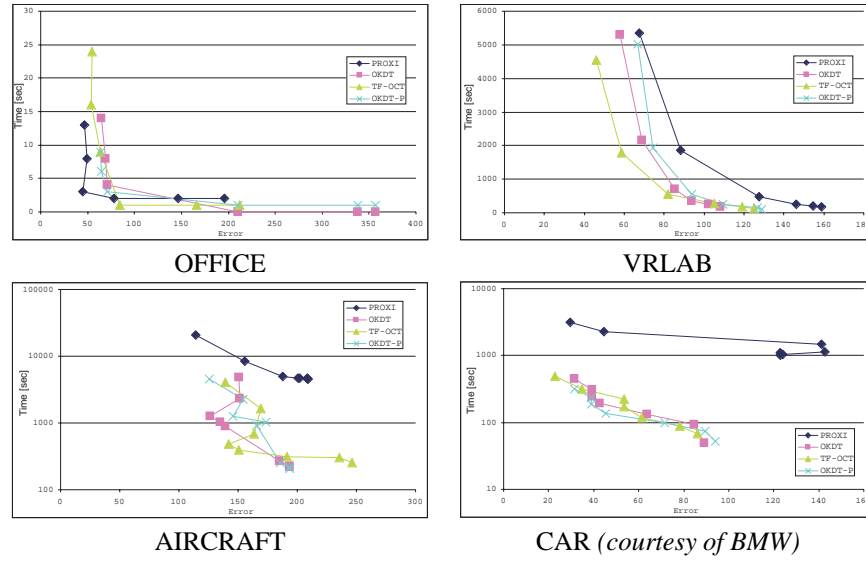


FIG. 3.7: Time-error curves

3.2.2 Visibilité Multi-résolution

Nous avons étendu (en collaboration avec François Sillion) la méthode de « clustering » en développant une nouvelle approche pour la mesure d'erreur pour la simulation de l'éclairage en utilisant des « traits » importants d'une image : les régions d'ombre causées par des objets qui cachent la lumière.

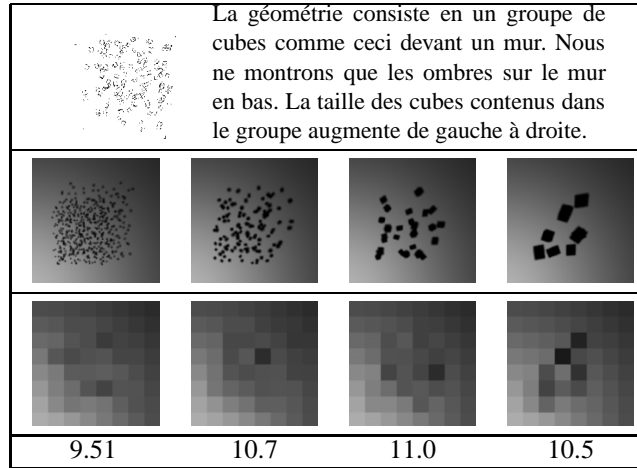


FIG. 3.8: Comparaison de solutions approximatives en utilisant des clusters avec des tailles de cubes différentes. Haut : images de référence (éclairage du mur) Milieu : images approximatives en utilisant un maillage « grossier ». Bas : Erreur dans la norme L^2 . Notez que les quatre images ont une valeur d'erreur L^2 similaire, et qu'elle cache toutes de l'information de l'éclairage. La taille variable des « traits », n'est pas donnée par cette mesure d'erreur.

Les méthodes d'estimation d'erreur précédentes sont incapables de distinguer l'erreur en fonction de la taille de traits. Par exemple, dans la Figure 3.8, nous voyons que la norme L^2 ne donne aucune information sur la taille de traits cachée par l'approximation. Nous avons déterminé un nouveau critère d'erreur qui a conduit au développement d'un nouvel algorithme. Cet algorithme utilise la structure hiérarchique de regroupement pour le calcul de la visibilité : quand l'utilisateur ne veut pas d'ombres plus détaillées pour une taille spécifiée, la transmission moyenne du groupe d'objets est utilisée au lieu de faire un test de visibilité pour chaque objet contenu. La conséquence est une accélération assez importante du temps de

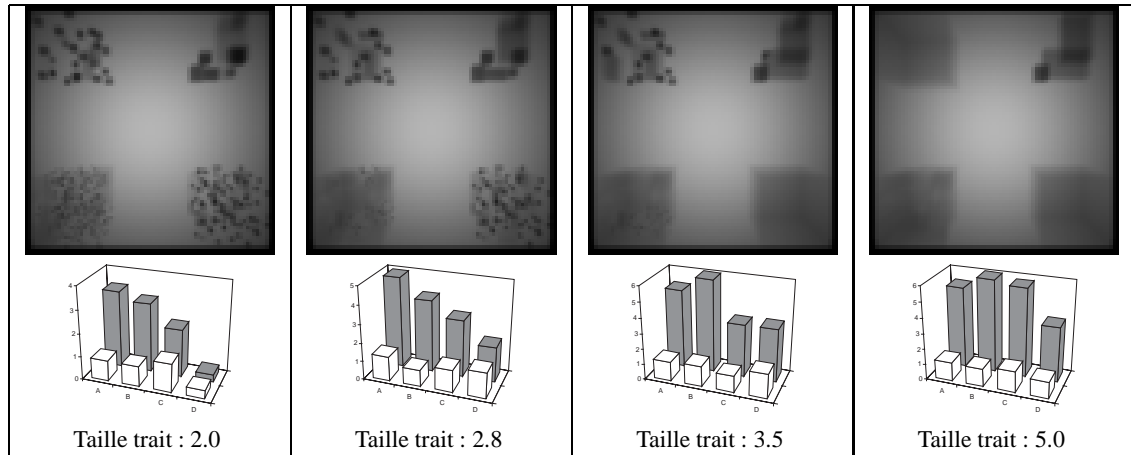


FIG. 3.9: Résultats pour l'algorithme de visibilité multirésolution[SD95].

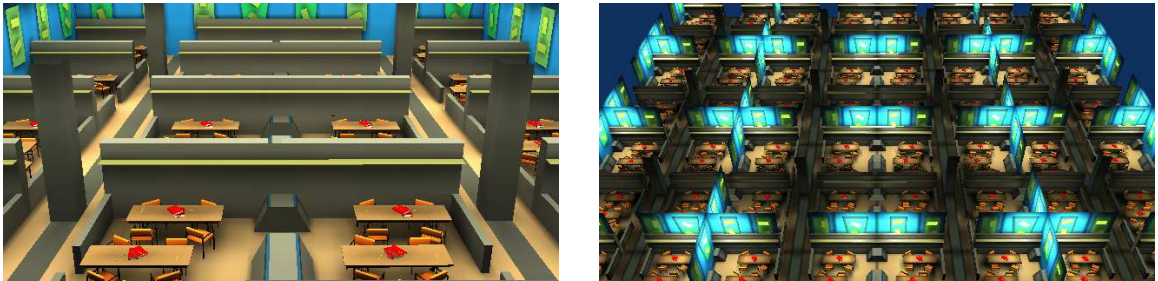


FIG. 3.10: Des scènes de tests utilisées par le nouvel algorithme de réduction de mémoire pour la radiosit  hi rarchique [GD99].

calcul en fonction de la qualit  requise [SD95]. Pour comprendre les r sultats, regardez la Figure 3.9 ; de gauche   droite, nous augmentons la taille de trait que l'utilisateur veut absolument pr server. L'erreur L^2 , montr e en colonnes sombres en arri re plan, donne diff rentes valeurs d'erreur pour chaque groupe de cubes ; la nouvelle mesure d'erreur donne   peu pr s la m me valeur, ce qui permet   l'algorithme de constater que, pour la taille de trait choisi, l'approximation est acceptable pour l'utilisateur.

3.2.3 R duction de la m moire utilis e par la Radiosit  Hi rarchique

Un autre aspect tr s important concernant l'utilisation de la radiosit  hi rarchique, m me avec le clustering, est la consommation m moire. Pour traiter des sc nes typiques d'aujourd'hui, contenant plusieurs centaines de milliers d'objets, les besoins en m moire peuvent  tre trop  lev s par rapport   la capacit  de m moire des ordinateurs disponibles.

La consommation m moire peut se distinguer en deux parties : la m moire utilis e par les liens, et la m moire utilis e par la hi rarchie elle-m me (pointeurs vers les enfants, valeurs de radiosit  etc.). Avec mon doctorant Xavier Granier (que je co-dirige avec Claude Puech), nous avons d velopp  un nouvel algorithme, capable de traiter des sc nes des centaines de milliers d'objets (voir Figure 3.10).

Notre nouvelle approche consiste   d finir un nouveau cadre pour la solution de la radiosit  hi rarchique, en mettant ensemble les  tapes de « refine », « gather », et « push-pull », ce qui permet   la fois    viter le stockage des liens (voir  galement [SSSS98]), et le stockage de la hi rarchie. Un exemple est montr  dans la Figure 3.11, o  l'on voit dans (b) que les enfants du n ud r_1 ne sont pas stock s. Le contenu des enfants de r_1 dans la hi rarchie est remplac  par une texture.

3.3.1 L'éclairage non-diffus

Nous avons développé une première solution [SDS95] de représentation directionnelle pour les objets avec réflectance non-diffus, dans le cadre d'un algorithme de radiosité hiérarchique avec « clustering ».

En particulier, nous stockons l'intensité radiante sortante (voir Figure 3.12), par une représentation d'harmoniques sphériques. La lumière incidente n'est pas stockée au niveau des nœuds intérieurs de la hiérarchie : elle est directement « poussée », vers les feuilles, où elle est immédiatement réfléchie et stockée dans les fonctions directionnelles. Nous avons également développé une solution permettant la représentation directionnelle de la visibilité volumique. Cette représentation permet le changement de l'aspect de l'ombre sur le sol dans la Figure 3.13.

Cette méthode était une première démonstration de la faisabilité d'une telle approche dans le contexte de la radiosité hiérarchique avec clustering. Une approche similaire a été également développée par Christensen et al [CLSS97]. La méthode souffre particulièrement de la consommation mémoire et du temps de calcul, en partie dû au parcours répétitif de la hiérarchie pendant la réflexion immédiate.

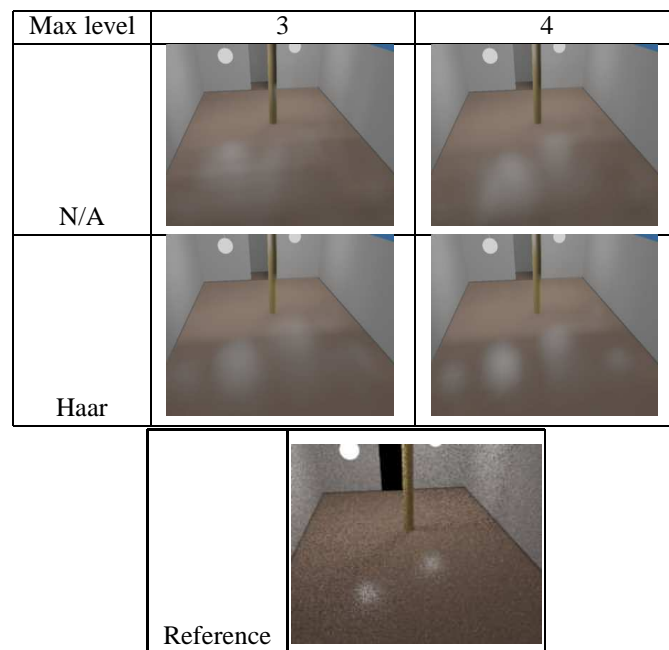


FIG. 3.14: Comparaison de la solution non-adaptative (N/A) et Haar.

Plus récemment, nous nous sommes attaqués à ces problèmes [SSG⁺99], en introduisant deux nouvelles solutions : (i) le stockage directionnel de la lumière incidente par une structure de fonction Dirac appelée « échantillons d'éclairage », (*illumination samples* en anglais), et (ii) l'utilisation d'une base d'ondelettes Haar pour le stockage de l'éclairage sortant. Ces travaux ont été effectués avec nos partenaires européen (M. Stamminger, A. Scheel, F. Perez-Cazorla, et F. Sillion et X. Granier d'iMAGIS) dans le cadre du projet Européen SIMULGEN, dont j'ai été le coordonnateur pour la première phase, et le responsable iMAGIS/GRAVIR pour la deuxième.

Max Level	Distr	Triangles		Time	
		N/A	Haar	N/A	Haar
3	2878	782K	640K	510 s	722 s
4	2878	3127K	1829K	731 s	1125 s

TAB. 3.1: Comparaison de la représentation Non-adaptative (N/A) et Haar montrant le nombre fonctions directionnelles utilisées (DirDistr) et le temps de calcul Max Level et le niveau maximal de subdivision.



FIG. 3.15: Maison éclairée par le soleil et le ciel.

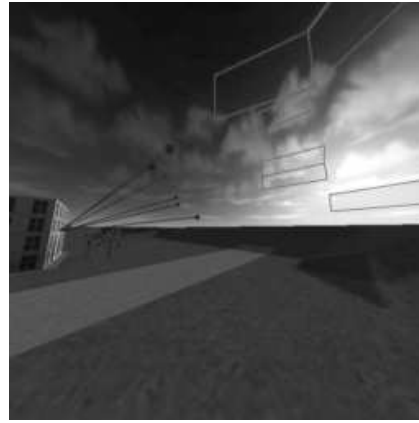


FIG. 3.16: Mains de coté avec visualisation des liens vers le ciel.

Cette nouvelle solution offre une meilleure qualité d'éclairage non-diffus avec un stockage moindre. Par exemple, nous voyons dans la Figure 3.14 qu'une base non-adaptative (dans ce contexte une base constante, mais les harmoniques sphériques sont dans la même catégorie) avec un niveau de subdivision « 3 », n'arrive pas à bien représenter les « taches », de lumière sur le sol, mais la solution Haar y arrive. Il est intéressant de voir le Tableau 3.1, où on voit que les niveaux de subdivision « 3 » utilise à peu près la même mémoire pour les deux approches, mais Haar donne une meilleur image. Au niveau de subdivision « 4 », même si la représentation non-adaptative donne un meilleur résultat que pour le niveau « 3 », elle nécessitent deux fois plus de mémoire que le Haar.

3.3.2 L'éclairage de scènes d'extérieur

Une solution, hiérarchique pour des scènes complexes d'extérieur, éclairées par le soleil et le ciel a également été développée [DSSD97]. Ces travaux ont été effectués durant un stage ERASMUS (dir. F. Sillion) de K. Daubert et H. Schirmacher de l'université d'Erlangen.

L'idée de cette approche était de représenter le soleil comme une source directionnelle, ce qui a nécessité des modifications dans l'algorithme de raffinement, et de représenter le ciel d'une façon hiérarchique. Un exemple de cette représentation hiérarchique est donné dans la Figure 3.16 où l'on voit une visualisation des différents niveaux de liens vers le ciel. L'image Figure 3.15 montre un exemple d'une solution calculée par cette méthode.

3.4 Discussion

Dans ce chapitre nous avons présenté nos travaux sur les méthodes d'éclairage pour les scènes complexes. Ces méthodes sont des méthodes *approximatives* par opposition aux méthodes *analytiques* du chapitre précédent.

Nous avons apporté des améliorations sur les calculs de radioté, à la fois pour la visibilité et pour la mémoire utilisée. Ensuite nous avons étudié les problèmes de la généralisation de la méthode de radioté à des cas non-diffus.

Pour ces derniers cas, nous avons réussi à faire des simulations d'éclairage non-diffuse pour des petites scènes allant jusqu'à quelques milliers de polygones. Nous croyons par contre que cette limitation est inhérente à ces approches (qui peuvent être malgré tout utiles pour certaines applications), car la quantité d'information qui doit être stockée est trop grande. C'est pour ceci que nous croyons que c'est dans les domaines du rendu à base d'images et le rendu par tracer de rayons que se trouvent les solutions pour les phénomènes photométriquement complexes.

3.5 Articles

3.5.1 Filtering, Clustering and Hierarchy Consutrction (EG'95)

Auteurs : Frédéric Cazals, George Drettakis et Claude Puech

Actes : Congrès Eurographics '95

Date : septembre 1995

Filtering, Clustering and Hierarchy Construction: a New Solution for Ray-Tracing Complex Scenes

Frédéric Cazals,¹ George Drettakis,^{1,2} Claude Puech¹

¹iMAGIS-IMAG, BP 53, F-38041 Grenoble, cedex 09 FRANCE. iMAGIS is a joint project of CNRS, INRIA, INPG and UJF. E-MAIL: Frederic.Cazals@imag.fr.

²Department of Software, Technical University of Catalunya, Diagonal 647, 08028 Barcelona, SPAIN.

Abstract

Data structures that handle very complex scenes (hundreds of thousands of objects) have in the past either been laboriously built by hand, or have required the determination of unintuitive parameter values by the user. It is often the case that an incorrect choice of these parameters can result in greedy memory requirements or severely degraded performance. As a remedy to this problem we propose a new data structure which is fully automatic since it does not require the user to determine any input parameters. The structure is built by first filtering the input objects by size, subsequently applying a clustering step to objects of the same size and finally building a hierarchy of uniform grids (HUG). We then show that this data structure can be efficiently constructed. The implementation of the HUG shows that the new structure is stable since its memory requirements grow linearly with the size of the scene, and that it presents a satisfactory compromise between memory usage and computational efficiency. A detailed comparison with previous data structures is also presented in the results.

1. Introduction

Dealing with very complex scenes is one of the major challenges for current computer graphics research. To facilitate manipulation of such environments it is necessary to develop spatial subdivision structures which allow the implementation of efficient searching algorithms for such applications as rendering (with ray-tracing or radiosity-style methods), collision detection in animation and also for interactive environments (virtual reality etc.).

One of the more challenging aspects of creating such structures is the treatment of scenes which contain large changes in scale, for example the model of a building, which at successive levels of scale contains the walls, the rooms, the doors, windows and furniture, and finally the small detail objects such as the knobs on a television or a phone. For such scenes it is often the case that the complexity of the geometry is to be found in the smaller scales (e.g., thousands of polygons for the models of household appliances).

The data structure we introduce in this paper represents a new approach, by first *filtering* the input objects by size, then *clustering* objects of the same size and finally constructing a hierarchy of uniform grids. Our structure is fully automatic since the user does not need to specify any parameters, and is thus capable of treating large scenes efficiently, without the need for time-consuming experimentation to determine unintuitive parameters.

1.1. Previous Structures for Highly Complex Scenes

From the very first presentation of the ray-tracing algorithm¹, it became clear that a spatial subdivision structure is required to cope with the millions of rays which are intersected with the objects in the environment. In the early algorithms hierarchies of bounding boxes were used^{1, 2}. These methods have been extended to more generalised bounding volumes^{3, 4, 5}. BSP trees⁶ and regular subdivisions of space are also popular in dealing with the ray-tracing problems⁷. Octree structures have been used⁸, as well as uniformly subdivided grids^{9, 10}.

In a few of the previously cited papers very large environments have been treated. Most notably Snyder and

Barr¹⁰ handled scenes with millions of polygons, but user intervention was required to construct the hierarchical data structure. Large environments have also been treated by recursively subdividing uniform grids¹¹ and also for modelling¹².

1.2. Complexity analysis

The performance of spatial subdivision data structures is notoriously difficult to analyse. Some previous work has been done however^{13, 14, 15}. In this section we present a more careful look at some of the spatial subdivision structures which in practice have been used with success^{16, 17}. We are particularly interested in the following classes of spatial subdivision structures: *uniform grids*, *recursive grids* and *octrees* (i.e., object-octrees).

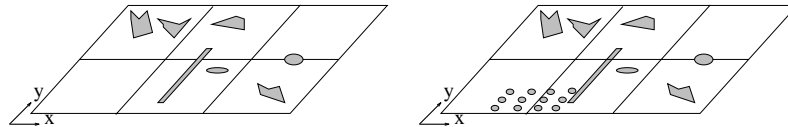


Figure 1: (a) Uniform grid / uniform distribution

(b) Uniform grid / non uniform distribution

1.2.1. Uniform grids

Uniform grids are built by subdividing the sides of the bounding box Δ of the scene, along the x, y, z -axes into n_x, n_y and n_z subdivisions. Each element of this subdivision is called a *voxel*. In each voxel a pointer toward the items intersecting it is stored.

Choosing $n_x = n_y = n_z = \sqrt[3]{n}$ yields n voxels. This results in practically "optimal" performance if the objects of the scene \mathcal{O} are scattered uniformly in Δ . An example of such a situation is shown in Figure 1(a). Nevertheless, if some voxels contain too many objects the uniform grid data structure will fail, since when a ray traverses a highly populated voxel too many ray-objects intersection tests are performed (Figure 1(b)).

A typical remedy used in practice are higher subdivision factors n_x, n_y, n_z , but the time gained by avoiding ray-object intersections is eventually lost by the additional cost of voxel traversals. This approach is also limited by the rapidly increasing memory requirements. In addition arbitrary subdivision parameters are difficult to use since experimentation is necessary to find the best trade-off of "cost of the intersections" vs. "cost of traversal". Another alternative is a recursive data structure, discussed next.

1.2.2. Recursive grids

Jevans¹¹ proposed a solution in which each voxel containing a number of pointers greater than MAXP (where MAXP stands for MAXimum number of Polygons), is recursively subdivided. In the tests presented¹¹, the same subdivision factors are used for every recursive subdivision (e.g. $n_x = n_y = n_z = 10$). As a consequence the memory requirements grow explosively.

Jansen^{16, 17} proposes an adaptive subdivision criterion, which we will call the $\sqrt[3]{n}$ -criterion in what follows. This approach consists in setting $n_x = n_y = n_z = \sqrt[3]{n_i}$ for voxels that contain $n_i > \text{MAXP}$ items. In practice it seems that this choice works quite well. Nonetheless, several questions remain: is there an optimal termination condition MAXP? If such a condition exists, does it avoid the problem of explosive memory growth?

These questions are still unanswered in computer graphics, but have received some attention in the theory of search and sort bucket-like data structures¹⁸. We discuss a brief outline of this work in the following.

Suppose we want to store a set of real numbers $\{x_1 \dots x_n\}$ that are known to belong to the real interval $\Delta = [a, b]$. The usual bucket-like data structures operate as follows: (i) subdivide Δ into n intervals of equal length (ii) in all the buckets where the number of points is $> b$ (with b a constant between 4 and 10 generally) subdivide and iterate recursively. Since the $\{x_i\}$ are assumed to be independent realisations of random variables, theorems assert that (i) for most random variables "Two levels are as good as any"¹⁹ i.e. at depth 2 most of the buckets contain less than b points (ii) the number of buckets necessary to store n is $3Mn$ where M is the maximum of the probability density.

These theoretical results (mainly in one dimension) provide important intuition into the expected behaviour of recursive grids for very complex three-dimensional scenes in which large scale changes occur.

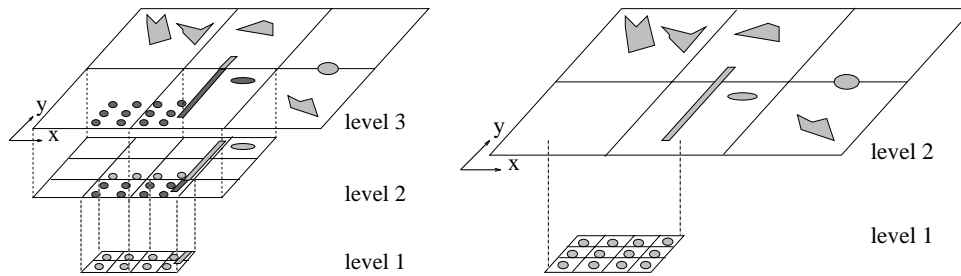


Figure 2: (a) Recursive grid

(b) Hierarchy of Uniform Grids

A first conclusion is that the hierarchy should not be very deep and that it is possible to restrict the memory cost more efficiently than when using a constant subdivision factor. A major difference however is that points in one dimension are not shared between buckets. For the three dimensional case in which the contents of \mathcal{O} are polygons, such overlap may occur. This has two important negative consequences:

- Suppose there are N polygons that are coplanar and the common voxel intersection of which is non-empty. If $N > \text{MAXP}$, the recursion will not terminate.
- The number of references towards an object might be large: see the "long" item on Figure 2(a), that runs across 5 buckets.

1.2.3. Octrees

Octrees⁸ can be viewed as a special case of recursive grids for which subdivision into eight sub-voxels is performed at every step. Their advantage is a natural adaptation to the geometric complexity of a scene and the fact that special optimisation can be performed for ray-traversal. Their main drawbacks are that a hierarchy of large depth may be created, a penalty of traversing the hierarchy is often incurred and that duplication of objects in lists is frequent.

1.3. Towards a better solution

Given the discussion above, we can now put our proposed solution in context. The steps of the new algorithm are summarised as follows:

1. Gather the objects into subsets of similar size.
2. Foreach group of objects of similar size, group the neighbours into *clusters*.
3. Construct a grid with the $\sqrt[3]{}$ criterion for each cluster.
4. Construct a hierarchy of these grids.

The data structure we introduce can thus be seen as a recursive grid where unnecessary intermediate levels and multiple references towards items of different sizes are suppressed. An example of a *HUG* is shown in Figure 2(b) for the set \mathcal{O} of Figure 2(a).

The filtering and clustering steps effect a *bottom-up* construction of the data structure, in the sense we that start with the objects and by grouping them, we construct a hierarchical data structure. This is in strong contrast with all other automatic methods which subdivide their structure adaptively in a *top-down* manner (there do however exist previous methods which construct hierarchies manually^{20, 21}). As we shall see in Section 3 the use of such top-down methods can result in significant additional memory cost.

2. A New Structure: the Hierarchy of Uniform Grids

For the purposes of our construction we consider the *bounding boxes* of objects. In this manner our structure does not depend on the object type (e.g., polygons, bicubic surfaces etc.). In what follows we thus use the words

object and bounding box interchangeably. We define the following properties of objects: (i) the *length* of an object that will be its diameter (ii) the *projection* of an object along an axis that will be a line segment on that axis (iii) a distance function $d(s_1, s_2)$ between segments (iv) a distance function $d(o_1, o_2)$ between objects. For a pair of objects or segments A, B , and a, b points of A and B respectively, the distance function is defined as $d(A, B) = \inf distance(a, b), \forall a \in A \text{ and } \forall b \in B$. Finally, given the lengths $\mathcal{D} = \{d_1, \dots, d_n\}$ of all the objects from \mathcal{O} we will note $d_{inf} = \inf_i length_i$ and $d_{sup} = \sup_i length_i$. Thus d_{inf} is the length of the shortest item, and d_{sup} is the length of the longest item.

2.1. Filtering the Objects by Size

Definition 1 (Filter of lengths) Given a set \mathcal{O} of objects, we call filter \mathcal{F} a strictly increasing sequence of positive real numbers $\{f_1, \dots, f_m\}$ such that $d_{inf} \in [f_1, f_2)$ and $d_{sup} \in [f_{m-1}, f_m)$. A level of the filter \mathcal{F} is an interval $[f_i, f_{i+1})$. Define also (i) $f_k = (f_k + f_{k+1})/2$ the average length of level k (ii) L_k the set of all objects from \mathcal{O} the lengths of which $\in [f_k, f_{k+1})$.

The filter is used as follows: for $o_i \in \mathcal{O}$ we determine the position of $length(o_i)$ in \mathcal{F} by a binary search. The filtering operation costs $O(n \log m)$ and yields at most $m - 1$ non empty levels. If we consider the kernel-estimated density²² $\widehat{f}_{\mathcal{D}}$ from the set \mathcal{D} computing a good filter is a difficult problem. For typical computer graphics scenes this density contains a few peaked modes (because the complexity of the scene lies in the small objects that have similar sizes) and the filter can be determined by searching these modes on a histogram. In practice we subdivide the histogram into a given number of slices (e.g. three), such that each slice contains the same number of points of the histogram (see also Tsai and Chen²³). This results in very satisfactory performance.

To give a precise idea of the filtering process, the following images show the result of filtering the scene *kitchen* which is used in Section 3 for the experimental study (see Figures 10(a) and 10(b)). Because of the intuition "Two levels are as good as any", we computed a three level filter thus gathering objects in groups of three sizes.

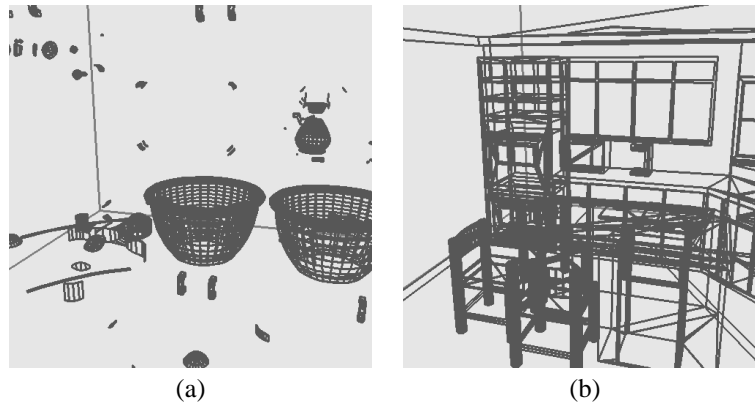


Figure 3: Showing filter levels (a) Level 0 (small objects) (b) Level 2 (big objects)

However, gathering the objects by homogeneous size is not sufficient: consider for example in Figure 10(a) the items corresponding to the bowl on the table and the coffee-pot on the counter: we cannot build a single uniform grid to store these two sets of objects without being faced with the problems mentioned in Section 1.2.1. We therefore wish to isolate two clusters of objects, one for the bowl and one for the coffee-pot.

2.2. Clustering Objects of the Same Size

2.2.1. Projections and clusters

Given a set of objects \mathcal{O} , finding a partition of \mathcal{O} into clusters according to a given criterion is a well studied topic of statistics. Unfortunately, the best clustering methods that do not presuppose the final number of

clusters^{24, 25} are $\Theta(n^2)$ -space and $\Theta(n^2 \log n)$ -time. For very large environments such a cost is unacceptably high. To see why, consider the algorithm step at which a new item is added to the previously computed clusters. This step requires testing this item against all the clusters in order to find the best fit.

As an alternative to this "grouping" method, we choose a "divisive" approach illustrated below. Consider for example Figure 4 where we would like to construct three clusters A , B and C from the set $A \cup B \cup C$. An appropriate necessary condition for two objects to belong to the same cluster is that their projections along the two x, y axis themselves form a cluster. The idea is therefore to successively examine the projections of the items along x, y and z and then to split the initial input into sub-groups that cannot form a cluster. As will be seen, working with the projections enables us to compute clusters which are somewhat more restricted, but in a more efficient manner.

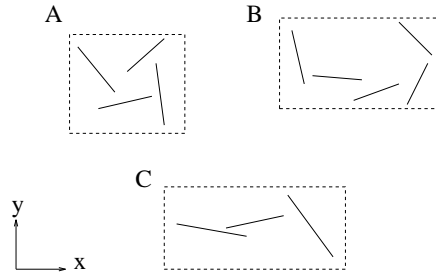


Figure 4: A collection of clusters

Definition 2 (δ -connectivity - δ -cSet) Two objects o_i and o_j are called δ -connected where $\delta \in \mathbb{R}^{*+}$ if their distance $d(o_i, o_j) < \delta$. If they belong to the same level k of the filter \mathcal{F} , let δ_k be the δ associated to level k . We call connectivity coefficient α a strictly positive real number, and we set $\delta_k = \alpha \bar{f}_k$. A set \mathcal{O} is said to be δ -connected or a δ -cSet if $\forall i \in 1..n$ there exists a $j \neq i$ such that $d(o_i, o_j) < \delta$.

Two items belong to the same cluster if the distance separating them is less than a small "percentage" of their relative length (which is nearly the same since the items belong to the same filter level). In practice α is set to a small number (e.g., $\alpha = .01$). The performance of the algorithm is not sensitive to the choice of α . Varying its value between .00125 to .08 resulted in practically no change in the performance of the structure constructed for the test scenes of Section 3.

Definition 3 (Property π_{xyz} , potential cluster and cluster) A set \mathcal{O} is said to be π_x or to have property π_x if the objects' projections along axis x form a δ -cSet. The opposite will be noted $\pi_{\bar{x}}$. Thus a set $\pi_{x\bar{y}z}$ will have the property $\pi_x, \pi_{\bar{y}}$ and π_z . \mathcal{O} is said to be a potential cluster if it has property π along one or two axis, and a cluster if it is π_{xyz} . By CPC we denote either a cluster or a potential cluster.

In Figure 4, we see that neither $A \cup B$ nor $A \cup C$ are clusters since these two sets are respectively $\pi_{\bar{x}y}$ and $\pi_{x\bar{y}}$. It is important to understand that δ -cSets are formed independently for each of the one-dimensional projections allowing the correct isolation of clusters.

2.2.2. The Clustering Algorithm

The clustering algorithm consists of two parts: (a) *Isolate* that finds the subsets of *dataSet* that verify the necessary condition along a *Direction*, and (b) *Cluster3d* which computes the clusters using *Isolate*. The input of the algorithm is an array of objects *dataSet*, and more precisely all the items located between two indices n_1 and n_2 of this array. The information recorded for each cluster or potential cluster (CPC) is: (i) a pair of indices $index_1$ and $index_2$: all the items contained in the array *dataSet* between those indices belong to the same CPC (ii) a flag *dir* indicating the last direction where property π has been checked (1 for x , 2 for y , 3 for z) (iii) a counter *stable* indicating how many directions have property π (thus a CPC is a cluster if *stable* = 3).

The output is a stack *stackOfCPC* of CPC. Recall that the projection of object o_i along a given direction is noted s_i (s for segment). The two endpoints of segment s_i are noted $s_i.left$ and $s_i.right$.

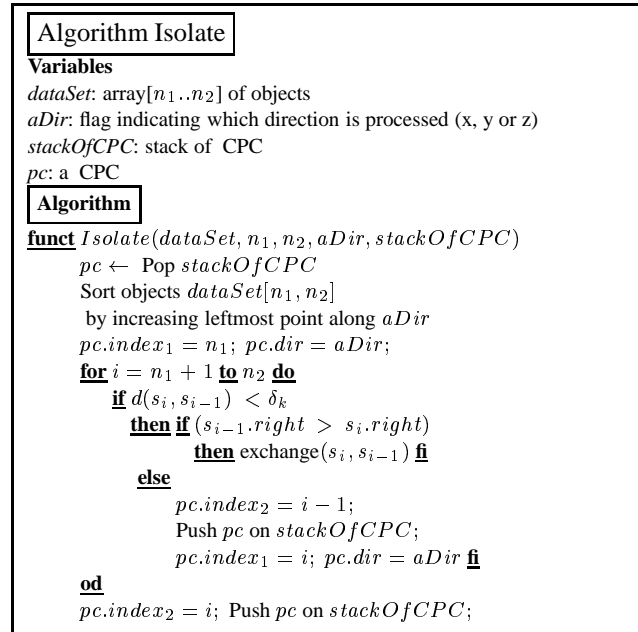


Figure 5: Algorithm Isolate

Proposition 1 Algorithm *Isolate* runs in $O(n \log n)$

PROOF

- The sort step ensures that when a new segment is inserted, the necessary condition has to be checked between the left point of this segment and segments located on its left. In Figure 5 the segments are therefore inserted according to their indices.
- The swap operation **if** $s_{i-1}.right > s_i.right$ is such that when s_{i+1} will be inserted, the rightmost point of the previously inserted segments will be the right point of the segment on its left. For example (Figure 5) when 4 is inserted, the test is performed with 1 and not with 3.
- These two conditions guarantee $O(1)$ operations per insertion and the algorithm therefore runs in $O(n \log n)$ because of the sort step.

△

Given the algorithm *Isolate*, we can present the complete clustering algorithm in three dimensions: *Isolate* is successively applied in the direction of the three axes until sets of items *stable* (i.e. not modified by successive calls to *Isolate* along x , y and z) are found. Since we know we will get at most n clusters, we use an array of size n as a "double stack" as follows: the bottom part of the array contains the stack of potential clusters that are computed by *Isolate*, and the top part contains the stack of final clusters. In other words, once a potential cluster is found to be π_{xyz} , it is popped from the bottom stack and pushed on the top stack. These two stacks grow in opposite directions, but we are sure they never overlap since we use an array of size n and at any time we know there are at most n clusters (potential and final).

Algorithm Cluster3d**Environment**

stackOfCPC: array used as a "double" stack storing the potential clusters and the final clusters

Algorithm

$pc.index_1 = 1$; $pc.index_2 = n$; $pc.dir = 0$;

Push pc on bottom stack

while bottom stack is not empty **do**

begin

$pc \leftarrow$ Pop a potential cluster on bottom stack;

$aDir = (pc.dir + 1) \% 3$;

$Isolate(dataSet, pc.index_1, pc.index_2, aDir, stackOfCPC)$;

if top cluster on bottom stack is stable

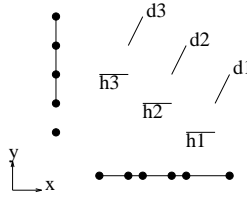
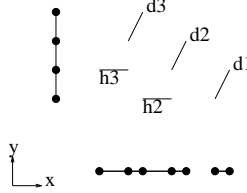
then begin

 Pop this cluster;

 Push it on the top stack

end fi

end

Figure 6: S_0 Figure 7: S_1

Proposition 2 The complexity of algorithm *Cluster3d* is output sensitive and the best and worst case complexities are $\Omega(n \log n)$ and $O(n^2 \log n)$, respectively.

PROOF

To see the lower bound, suppose *Isolate* along the first direction, having processed x , finds that no two items can be put into the same cluster. *Cluster3d* then stops and returns n clusters of one item.

The upper bound is somewhat more involved. The worst case will occur for a set S , such that each application of *Isolate* along each direction results in the removal of one item from a cluster. This can occur at most $O(card(S))$ times, and thus the cost is $\sum_{i=1}^{n-1} O(i \log i) = O(n^2 \log n)$. Δ

As an example consider Figures 6 and 7 where the set S is composed of two kinds of segments, horizontal and diagonal. The circles show the projection of the endpoints along each axis.

- The set $S_0 = \{h_1, h_2, h_3, d_1, d_2, d_3\}$ is π_x .
- *Isolate* along y removes h_1 . The resulting new set $S_1 = S_0 - h_1$ is π_{xy} .
- *Isolate* along x removes d_1 from S_1 and S_2 is π_{xy} .
- As a consequence h_2, d_2, h_3 are successively removed.

For all our experiments, the ratio $\sum_{All\ the\ sorts\ steps} (N_i \log N_i) / n \log n$ was computed (with N_i the cardinal of subsets treated by *Isolate*). For these test cases the value of the ratio belongs to the interval $[3., 5.]$, thus providing a strong indication that the running time is close to the lower bound complexity. Figure 8 shows some clusters computed from the level containing the smallest objects of the scene *kitchen*.

2.2.3. Gridding the clusters

Once n objects of the same size have been gathered in a cluster, we have to construct a uniform grid to store the cluster. Taking $\sqrt[3]{n}$ subdivisions along each axis gives n buckets but does not guarantee $O(1)$ objects per bucket. An alternative choice of subdivision parameter would be $n_x = \lambda(\cup_{i=1..n} P_x(o_i)) / \overline{\lambda_{i=1..n}(P_x(o_i))}$, where P_x is the projection of o_i on the x -axis and λ is the Lebesgue measure (in this case length). This is the measure of the overlappings along an axis, i.e., the length of the union of the projections divided by the average length of the projections. Unfortunately, for some sets \mathcal{O} this can give $n_x = n_y = n_z = O(n)$ resulting in $O(n^3)$ buckets. Such memory cost would render the structure unusable in practice. The previously described $\sqrt[3]{n}$ criterion is therefore used: the gridding of a cluster of n objects contains exactly n voxels. An exception is made for cluster containing the largest object where experiments have shown that $2n$ voxels lead to a substantial improvement. Determining optimal n_x, n_y, n_z remains an open problem.

2.3. Creating the Hierarchy of Grids

Suppose that we have now filtered the items of \mathcal{O} and the clusters of all the non-empty levels have been computed. In addition for each cluster the subdivision parameters have been determined. We also create a single cluster, called *World*, the bounding box of which contains the objects of the higher level of the filter (large objects) and which also covers all the bounding boxes of lower levels clusters. The construction of the \mathcal{HUG} is given below, performed top-down for the filter levels. This must not be confused with the top-down adaptive subdivision of recursive grids and octrees, which completely determines the form of those data structures. For the \mathcal{HUG} , the bottom-up filtering and clustering define the form of the structure.

2.3.1. Algorithm

<p>Algorithm</p> <pre> proc CreateHUG create the highest level cluster gridding and store its objects for all the others filter levels, in decreasing order do for each cluster of the level do create the cluster gridding and store its objects recursively insert this gridding in the hierarchy od od.</pre>

Let us consider the following example where the filter has 3 levels, levels 1, 2 and 3 respectively containing the clusters 1a and 1b, 2 and 3 (the grids are represented with continuous lines, and the items stored in each grid in dashed lines), shown in Figure 9.

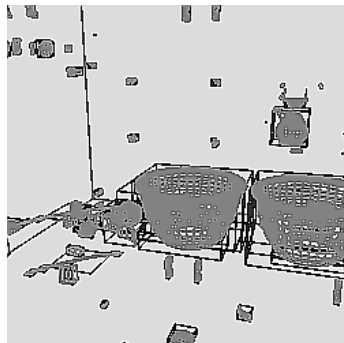


Figure 8: Some level 0 clusters in kitchen

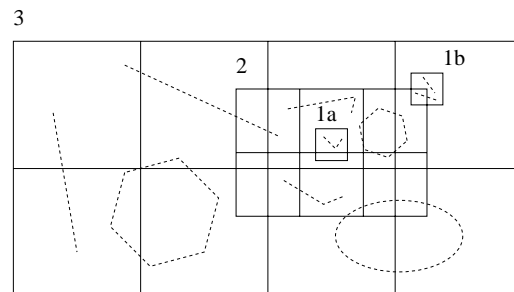


Figure 9: \mathcal{HUG} construction

- We first create grid 3, which contains the whole scene.
- Next we create grid 2. A pointer towards grid 2 is inserted in each voxel it intersects.
- Next grid 1a. Since it intersects voxel (3,2) of grid 3, which contains a sub-grid, we try to insert grid 1a into the subgrid (grid 2). Since grid 1a is contained in grid 2, we store a pointer towards 1a in voxels (2,1) and (2,2) of grid 2 and return.
- Finally grid 1b. Since voxel (4,2) of grid 3 already contains a sub-grid, we try the recursive insertion. But grid 2 does not contain grid 1b and we set a pointer towards 1b in voxel (3,2) of grid 2 and voxel (4,2) of grid 3.

2.3.2. Traversal of the structure for ray-casting

The traversal of the \mathcal{HUG} is performed as follows. When a ray enters the grid *World*, the starting voxel is found in the same manner as for uniform grids⁹. First we intersect the ray with every object contained in this voxel, and we then recursively visit each subgrid referenced in this voxel. After returning from the recursion we visit the next voxel along the ray path. Note that if an intersection is found the ray-traversal will stop in the same manner as for uniform grids.

3. Implementation, Results and Comparative Study

We have fully implemented the filtering, clustering and hierarchical construction of the HUG . In addition we have implemented recursive grids and simulated octrees in the same manner as Jevans and Wyvill¹¹. The environment used was *OORT*, which is a public domain ray-tracer²⁶, using the uniform grid implementation of Amanatides and Woo⁹ adapted to C++. It must be noted that once uniform grids were adapted to C++, the implementation of recursive grids and octrees was performed simply by creating a subclass. The complete filtering and clustering algorithms are also relatively simple to implement, with the entire source for these operations being less than 1500 lines of C++. Thus, given access to an existing ray-tracer already containing regular grids, implementing the HUG is a relatively simple endeavour.

Finally, since the same code base was used for the three kinds of hierarchies (recursive grids, octrees, HUG), the performance comparisons are "fair".

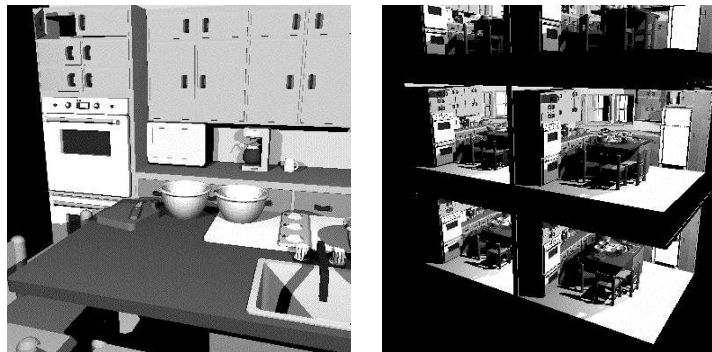


Figure 10: (a) A close-up view of the kitchen scene (b) 6 kitchens

3.1. Experiments

We used two sets of test scenes, constructed from a main model of kitchen:

- (1) A kitchen with additional detail objects (bowls, teapots and stacks of plates) added to augment the complexity of the scene (the basic model is shown in Figure 10(a)).
- (2) A set of rooms of increasing complexity (2, 4, 6 and 8 rooms) obtained by replicating the kitchen, in what amounts to a building model (see Figure 10(b)). Even though a replicated model is not particularly realistic, we believe that the essential geometric characteristics of a typical building model are well represented, since in each room a large amount of detail exists. Even though it may appear that rotating the kitchens with respect to each other may have generated a more interesting distribution of objects, the resulting clusters would not change significantly because of our use of minimum distance (see the beginning of Section 2) in the clustering of bounding boxes. The differences in scale are quite significant, since object lengths vary from .001m to 40m.

The data structures used for comparison with the HUG were the recursive grid and the octree with 3 different values for MAXP (50, 100 and 150) and the HUG . We do not mention the uniform grid, the rendering time of which quickly becomes 10 times worse than those of the other data structures. Lastly, the experiments were run for a 200x200 ray-cast image (no reflections), and the parameters of interest shown above are the rendering time in seconds, the megabytes used by the data structures, and the ratio $\nu_1 = \text{number of voxels} / \text{number of items}$. The memory usage in megabytes was measured using the *sbrk* system call. The preprocessing times (construction of the structures), were of the same order for all the structures.

3.2. Analysis

The experimental results are shown in Figures 11-13. The horizontal axes display the number of polygons in the scenes used. We first examine the performance of octrees, then recursive grids and finally the HUG .

Octrees. As one can see (3 upper curves of Figures 11(a)-(b)) the rendering times for octrees are worse than the other structures, while the number of voxels are much smaller than the other data structures. The total amount of

memory used is however larger. This apparent contradiction is due to severe memory fragmentation problems. These problems result from the straightforward implementation of octrees in our testbed. It is possible that the fragmentation could be avoided to a large extent by using a stack of static memory to allocate the lists. In addition the number of voxels is lower than for other data structures since the percentage of empty voxels is much smaller.

On the other hand, despite these considerations concerning memory management, the cause of inferior performance is due to the the depth of the hierarchy: from 10 levels for the "one kitchen" to about 15-20 for the "many kitchens."

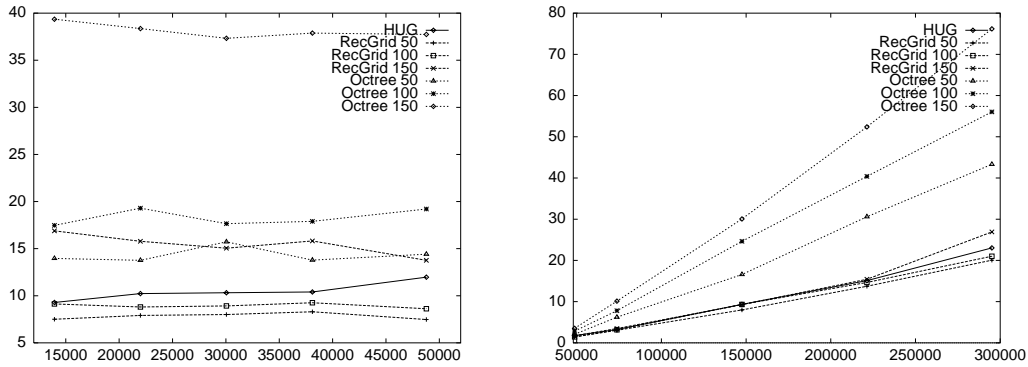


Figure 11: Render time - (a) One kitchen Render time - (b) Many kitchens.

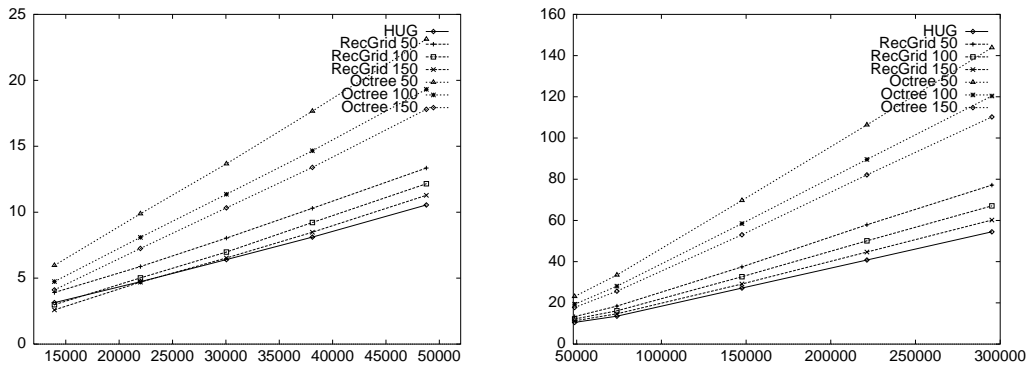


Figure 12: MBytes - (a) One kitchen and (b) Many kitchens.

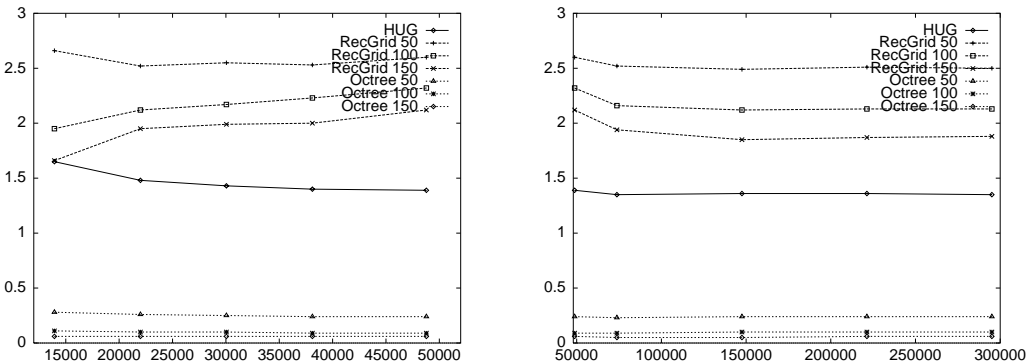


Figure 13: (a) v_1 - One kitchen (b) v_1 - Many kitchens.

Recursive grids. The intuition given by the one dimensional "bucket"-like data structures (section 1.2.2) appears to be verified for the recursive grids: the depth is low (in fact at most 5 in our case). This leads to moderately

high memory requirements but good computational performance. The value of MAXP affects both memory and running time: for the first set of scenes (Figure 11(a)), MAXP=50 (*recGrid 50*) leads to a rendering time 3.5 times faster than MAXP=150 (*recGrid 150*). For the second set of scenes, the choice of MAXP=50 which also gives the best rendering time, requires 30% more memory than MAXP=150.

HUG. The *HUG* performs well on both sets of scenes, with running times that are between those of *recGrid 50* and *recGrid 150*, while always using less memory. As an example, for the largest scene of 300,000 polygons, the *HUG* uses 40 MBytes and *recGrid 50* uses 60 MBytes. It is interesting to note that on a workstation with 180 Mb of real memory, swapping began to occur for all test runs which are reported to use more than 50Mb. Thus for the largest of the scenes tested, the only practically usable structure was the *HUG*. Both octrees and recursive grids vary significantly depending on the choice of the MAXP parameter, which often leads them to use significantly more memory and also to display degraded performance.

In summary, we can say the *HUG* displays stable behaviour both in terms of computational performance and memory, for both sets of scenes tested. Its memory requirements are the lowest of the 3 data structures studied (within the limitations of the comparative octree implementation), and its performance is similar to that of the recursive grids, depending on the choice of the termination condition. Thus the overall benefit of the new structure is the fact that automatic construction without the need for user-defined parameters provides consistent, satisfactory performance.

4. Conclusions and Future Work

We believe that the new approach we have presented is the first step in the development of data structures which have predictable behaviour for very complex scenes without the need for user-tunable parameters. We first discuss the limitations of the approach as presented and some ideas for future work, and we then summarise and conclude.

4.1. *HUG* restrictions and Future Work

Suppose the initial scene \mathcal{O} is composed of two kinds of objects: some big objects that will give the structure of the biggest grid, and a great many "small" objects that are so far from one another that the clustering algorithm will return as many clusters (up to a constant factor) as small objects. In this case, each voxel of the big grid will contain many references towards small clusters: the cluster and its gridding have been devised for the items themselves, not for the sub-grids, and if this number is too high, the direct access to the grid structure is lost. A more general clustering algorithm should therefore be used, for instance computing the Voronoi diagram of the small objects, a cluster being in this case a connected set of Voronoi cells having approximately the same volume. More generally, whenever the density of objects and the density of sub-grids in a cluster are significantly different, two different data structures should be used. It must be noted however that a small number of uniformly distributed small objects will not affect the performance of the grid *World*.

In addition, finding an optimal (or at least a provably good) value for MAXP is an extremely interesting avenue of research. The question is however difficult, and will require analysis of scenes based on statistical properties. It may also be possible to use the *HUG*, which provides automatic clustering of the input data-set, in animation and in "multi-precision" rendering.

4.2. Summary and Conclusion

In this paper we have introduced a novel approach to the construction of a spatial data structure for very complex scenes. We first group objects of the same size, and then create clusters of neighbouring objects in the same size group. An appropriately subdivided uniform grid is then placed around each cluster and a *Hierarchy of Uniform Grids* is constructed. We have shown that this construction can be performed efficiently.

The main advantages of the new structure are that: (a) it is fully automatic since it is constructed bottom-up by grouping objects and thus does not require the determination of any user-tunable parameters or termination conditions (b) it adapts well to very complex scenes without problems of memory usage other structures have when built for complex scenes and (c) it provides a good compromise between speed and memory usage since it uses less memory overall than all other structures (given the limitations of the comparative implementation) and performs satisfactorily in terms of execution speed.

Acknowledgements

The authors wish to thank the reviewers for their insightful comments and suggestions, Prof. Don Greenberg at the Cornell University Program for Computer Graphics for the model of the *kitchen* scene, and the Dynamic Graphics Project of the University of Toronto for use

of the grid traversal code. We also wish to thank Andrew Woo of Alias Research for his many comments and bibliography help. George Drettakis is an ERCIM postdoctoral fellow and is currently financed by the Commission of the European Communities.

References

1. T. Whitted, "An improved illumination model for shaded display", *Communications of the ACM*, **23**(6), pp. 343–349 (1980).
2. S. M. Rubin and T. Whitted, "A 3-dimensional representation for fast rendering of complex scenes", *Computer Graphics (SIGGRAPH '80)*, **14**(3), pp. 110–116 (1980).
3. H. Weghorst, G. Hooper, and D. P. Greenberg, "Improved computational methods for ray tracing", *ACM Transactions on Graphics*, **3**(1), pp. 52–69 (1984).
4. T. L. Kay and J. T. Kajiya, "Ray tracing complex scenes", in *Computer Graphics (SIGGRAPH '86)* (D. C. Evans and R. J. Athay, eds.), vol. 20, pp. 269–278, (Aug. 1986).
5. J. Arvo and D. B. Kirk, "Fast ray tracing by ray classification", in *Computer Graphics (SIGGRAPH '87)* (M. C. Stone, ed.), vol. 21, pp. 55–64, (July 1987).
6. H. Fuchs, Z. M. Kedem, and B. F. Naylor, "On visible surface generation by a priori tree structures", in *Computer Graphics (SIGGRAPH '80)*, vol. 14, pp. 124–133, (July 1980).
7. A. Fujimoto, T. Tanaka, and K. Iwata, "Arts: Accelerated ray-tracing system", *IEEE C. G. & A.*, pp. 16–26 (1986).
8. A. S. Glassner, "Space subdivision for fast ray tracing", *IEEE C.G. & A.*, **4**(10), pp. 15–22 (1984).
9. J. Amanatides and A. Woo, "A fast voxel traversal algorithm for ray tracing", in *Eurographics '87*, pp. 3–10, North-Holland, (Aug. 1987).
10. J. M. Snyder and A. H. Barr, "Ray tracing complex models containing surface tessellations", in *Computer Graphics (SIGGRAPH '87)* (M. C. Stone, ed.), vol. 21, pp. 119–128, (July 1987).
11. D. Jevans and B. Wyvill, "Adaptive voxel subdivision for ray tracing", in *Proc. of Graphics Interface '89*, pp. 164–72, (June 1989).
12. M. Mantyla and M. Tamminen, "Localized set operations for solid modeling", in *Computer Graphics (SIGGRAPH '83)*, vol. 17, pp. 279–288, (July 1983).
13. J. Goldsmith and J. Salmon, "Automatic creation of object hierarchies for ray tracing", *IEEE C. G. & A.*, **7**(5), pp. 14–20 (1987).
14. K. R. Subramanian and D. S. Fussell, "Automatic termination criteria for ray tracing hierarchies", in *Proc. of Graphics Interface '91*, pp. 93–100, (June 1991).
15. J. Cleary and G. Wyvill, "Analysis of an algorithm for fast ray tracing using uniform space subdivision", *The Visual Computer*, **4**, pp. 65–83 (1988).
16. E. Jansen and W. de Leeuw, "Recursive ray traversal", *Ray Tracing News (available at nic.funet.fi:pub/graphics/misc/RTNews)*, **5**(1), (1992).
17. E. Jansen, "Comparison of ray traversal methods", *Ray Tracing News (available at nic.funet.fi:pub/graphics/misc/RTNews)*, **7**(2), (1994).
18. L. Devroye, *Lectures notes on bucket algorithms*. Birkhauser, (1986).
19. M. Tamminen, "Two levels are as good as any", *Journal of algorithms*, **6**, pp. 138–144 (1985).
20. A. S. Glassner, "Spacetime ray tracing for animation", *IEEE C. G. & A.*, **8**(2), pp. 60–70 (1988).
21. D. Kirk and J. Arvo, "The ray tracing kernel", in *Proc. of Ausgraph '88*, pp. 75–82, (1988).
22. B. Silverman, *Density Estimation*. Chapman and Hall, (1986).
23. D.-M. Tsai and Y.-H. Chen, "A fast histogram-clustering approach for multi-level thresholding", *Pattern Recognition Letters*, **13**, pp. 245–252 (1992).
24. A. Gordon, *Classification*. Chapman and Hall, (1981).
25. H. Day and H. Edelsbrunner, "Efficient agglomerative hierarchical clustering methods", *J. of Classification*, **1**, pp. 7–24 (1984).
26. N. Wilt, *OORT: The Object Oriented Ray-Tracer*. (available at gondwana.ecr.mu.oz.au/pub), (1993).

3.5.2 A Light Hierarchy for Fast Rendering of Scenes with Many Lights (EG'98)

Auteurs : Eric Paquette, Pierre Poulin et George Drettakis

Actes : Congrès Eurographics '98

Date : septembre 1998

A Light Hierarchy for Fast Rendering of Scenes with Many Lights

Eric Paquette†, Pierre Poulin†, and George Drettakis‡

† Département d'informatique et de recherche opérationnelle
Université de Montréal. E-mail: {paquette|poulin}@iro.umontreal.ca

‡ iMAGIS-GRAVIR/IMAG-INRIA§
BP 53, F-38041 Grenoble Cedex 9, FRANCE. E-mail: George.Drettakis@imag.fr

Abstract

We introduce a new data structure in the form of a light hierarchy for efficiently ray-tracing scenes with many light sources. An octree is constructed with the point light sources in a scene. Each node represents all the light sources it contains by means of a virtual light source. We determine bounds on the error committed with this approximation to shade a point, both for the cases of diffuse and specular reflections. These bounds are then used to guide a hierarchical shading algorithm. If the current level of the light hierarchy provides shading of sufficient quality, the approximation is used, thus avoiding the cost of shading for all the light sources contained below this level. Otherwise the descent into the light hierarchy continues.

Our approach has been implemented for scenes without occlusion. The results show important acceleration compared to standard ray-tracing (up to 90 times faster) and an important improvement compared to Ward's adaptive shadow testing.

Keywords: Image synthesis, rendering, ray-tracing, hierarchy, illumination, reflection, Phong, bounds, clustering, octree.

1. Introduction

Realistic rendering has been a major goal of computer graphics from its very outset. Many powerful rendering approaches such as ray-tracing¹, radiosity-based methods^{2, 3}, and stochastic ray-tracing or Monte Carlo methods⁴ have been presented over the last two decades, resulting in images of impressive realism. For most existing commercial rendering systems (for animations, film special effects, post-production, advertising, etc.), ray-tracing remains the rendering algorithm of choice. In such environments, scenes containing a large number of geometric primitives as well as a large number of light sources are common. Everyday scenes with a large number of light sources include shopping malls, chandeliers, city streets at night, etc. In addition, more

complex light sources such as extended area sources, shades over a light source, or special sources to simulate flames are often required; they are typically approximated as a large collection of simpler point light sources.

The rendering of these scenes is a challenge to computer graphics, due to the number of light sources and the complex illumination that results, especially when we consider their combined rather than individual effect. In this paper we address the problem of efficient rendering of such scenes by introducing a new data structure, representing the light sources by a light hierarchy.

1.1. Motivation

Despite advances in the treatment of scenes containing a large number of geometric primitives using spatial subdivision techniques (octrees⁵, grids⁶, hierarchies of bounding volumes⁷, etc.), little has been done to accelerate ray-tracing of scenes with many light sources.

For scenes with hundreds or thousands of light sources, shading and shadowing calculations (rays sent to the light

§ iMAGIS is a joint research project of CNRS/INRIA/UJF/INPG.

sources) quickly become the dominant cost of the rendering process. As a result, lighting designers and other users of rendering systems are forced to crudely approximate interesting and complex lighting behavior with only a very small number of simple light sources.

The work on spatial subdivision and hierarchical algorithms for lighting calculations⁸, hierarchical approaches coupled with clustering^{9, 10}, and spatial subdivision have allowed the acceleration of lighting calculation for scenes containing a large number of polygons. A natural application of the same hierarchical concepts is the development and use of the light hierarchy for ray-tracing which we present in this paper.

1.2. Contributions

The goal of our approach is twofold: to provide efficient ray-tracing of scenes with many light sources with minimum loss of image quality, and to provide an intuitive quality parameter based on consistent error bounds for all the approximations made.

To achieve this goal in a comprehensive manner, we have restricted our attention to direct illumination from point light sources using lambertian (diffuse) and Phong¹¹ (specular) reflection models. Such assumptions are common in most commercial uses of ray-tracing systems.

Our solution involves the development of error bounds to evaluate the maximum potential error produced by the approximation. In this paper, we develop an algorithm which exploits the hierarchy and the bounds for the shading under direct illumination of scenes without occlusion. The results of our algorithm in this context (see Section 5) show that our light hierarchy significantly speeds up shading for scenes with many light sources.

It is interesting to note that in computer graphics research, an initial solution to an illumination problem is often presented for the unoccluded case (e.g., for radiosity¹² or hierarchical radiosity¹³), which then led to complete solutions including shadows (hemi-cube radiosity¹⁴ and hierarchical radiosity with shadows⁸). The study and introduction of a comprehensive solution without occlusion is an important step in the necessary understanding of the problem, leading subsequently to an algorithm including shadows.

For a given 3D point being shaded, traversal of the light hierarchy allows us to determine the effect on shading of intermediate nodes (representing potentially large numbers of light sources contained beneath the current level of the light hierarchy). As a consequence, we can completely avoid shadowing calculations for certain such nodes (i.e., avoiding shadowing for many light sources), or identify those light sources or sets of light sources which have the largest impact on final shading. To better underline the potential of our approach, after presenting the algorithm and the results of

the implementation, we will also discuss our ideas on the treatment of shadows in Section 6.

2. Previous and Related Work

There is an extensive literature on the research dedicated to speeding up shadow calculations using spatial coherence and subdivision.¹⁵ Most of these approaches however are highly dependent on the number of light sources, and are thus unsuitable for scenes with many light sources.

Some algorithms have nonetheless addressed the case of scenes with many light sources. Bergeron¹⁶ defines a sphere of influence around each point light source. The radius of each sphere is related to its light source intensity. Any object outside a sphere of influence can ignore the contribution of this point light source for both shading and shadowing. This method is efficient in many cases, but will fail for numerous light sources of low intensity because of the smaller radius of the spheres. Objects outside these spheres will ignore them all, even though the combined contribution of all light sources together may produce an important illumination effect. Another such situation occurs when several point light sources are used to simulate a complex light source. Improving the approximation typically requires increasing the number of point light sources, each of them individually emitting a smaller proportion of the complex light global intensity. With spheres of influence, the radii would then reduce, and the overall scene illumination would decrease as the complex light representation would be improved. This is illustrated in Fig. 1.

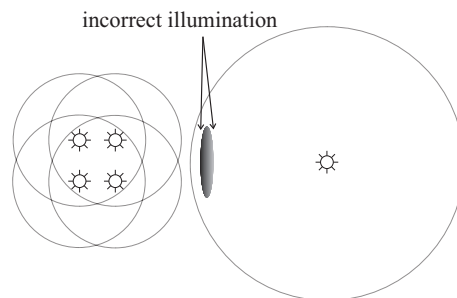


Figure 1: Problem associated with the spheres of influence

Ward¹⁷ presents a different approach where a sorted list of light source contributions is maintained. The main idea is to calculate the potential contribution of each light source at every point to shade (without considering visibility), and to use this estimation to sort the list of light sources. The ordered list is traversed and thus the real contribution (including visibility calculation) of the most important light sources is computed first. If the sum of the potential contributions of the remaining light sources is smaller than a predetermined percentage of the sum of all real contributions computed so far, the traversal stops. This method performs

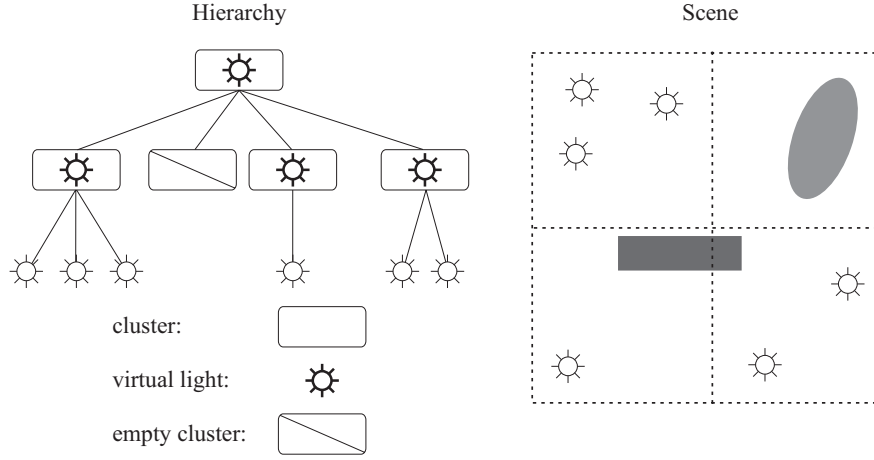


Figure 2: Example of light hierarchy

well for a moderate number of light sources, and is the most suitable algorithm to date for the treatment of scenes with many sources. However as the number of light sources increases, the cost of sorting the contributions (at each pixel) of all these light sources can become an important factor of the total rendering cost for scenes where the geometrical complexity is smaller than the illumination complexity. An extensive comparison of our approach to that of Ward's is presented in Section 5.

Shirley *et al.*⁴ divide light sources into two categories: bright (important) and dim (less important). This selection is performed as a preprocess, and is based on an approach similar to the sphere of influence. A sampling probability is then assigned to each bright light source, and a unique probability is assigned to all the dim light sources. If a large number of rays are shot per pixel, this method can be very effective. However, as with all Monte Carlo approaches, noise due to insufficient sampling can appear in the rendered images. Moreover, since the dim light source to be sampled is chosen randomly, an unsuitable partitioning into dim and bright light sources can greatly increase the amount of noise.

In radiosity-based methods, work has been performed in clustering objects for light-transfer calculations^{9,10}. These methods do not treat light sources separately, since they attempt to treat global illumination, and thus any surface is a light source in later iterations. As a consequence, these methods typically will not perform very well, since (primary) light sources are often clustered with the other objects of the scene, and no light-specific hierarchy is actually built.

Houle and Fiume¹⁸ store an emission map in a multi-resolution structure (quadtree) to efficiently resample a continuously varying emission distribution. However this structure is only valid over a 2D planar surface, and as such cannot easily be extended for independent light sources arbi-

trarily distributed in 3D space without losing its hierarchical nature.

Stam and Fiume¹⁹ simulate flames with a set of multi-resolution particles. To shade a point illuminated by their flames, the illumination is computed at two adjacent levels of the flame hierarchy, starting from the top. When the difference in illumination is larger than a preset threshold, the process continues at the immediately superior resolution. This hierarchical structure is efficient for flames consisting of a large number of particles. An early stop can however occur when the difference between two adjacent levels is small, but would be significant with a higher resolution. This is due to the fact that no bound on the approximation is provided.

3. Hierarchy of Point Light Sources

As mentioned in the introduction, a hierarchical data structure representing the point light sources in the scene is required to achieve our goals of efficient and consistent treatment of scenes with many light sources.

3.1. Octree Light Hierarchy

We have chosen to encode the light sources in an octree structure for the simplicity and compactness of its representation and its hierarchical nature.

The light hierarchy is stored separately from the ray-tracing acceleration structure used for the ray-object intersections (in our case also an octree). This choice has the advantage of creating a tighter bounding box around the light sources. The disadvantage resides in the fact that interaction between objects and light sources has to be treated separately. For the needs of our algorithm, we have found this solution to be satisfactory.

The leaf nodes of the octree store zero, one, or more point

light sources. Intermediate nodes are composed of eight children nodes. Every node keeps an approximate representation of the light sources contained at the current or at lower levels. In particular, *virtual light sources* stored at the nodes represent the set of light sources contained beneath this node. In Fig. 2, we show an example of such a point light hierarchy.

3.2. Light Hierarchy Construction

The creation of the light hierarchy begins by finding the axis-aligned bounding box of all the point light sources in the scene. This box is the root of our hierarchy. The hierarchy is subsequently subdivided in an octree fashion, until each unsubdivided octree cell contains less than a preset maximum number of light sources or a maximum subdivision depth is reached.

Once the octree has been subdivided, we proceed with the calculation of the representation of the virtual light sources. The virtual light source is simply a point light source, placed at a position corresponding to the weighted average of the light sources it represents. The weights are proportional to intensity of each real or virtual point light source over the sum of these intensities.

For tighter bounds on the approximation errors explained in the next section, we compute at each node of the octree the smallest axis-aligned bounding box of all the point light sources it represents. We call it the *minimal bounding box*. Table 1 summarizes this construction.

4. Shading with the Hierarchical Light Structure

Once the light hierarchy is built, the goal is to develop an algorithm allowing us to use the approximate representations (virtual light sources) where appropriate. We will thus avoid the cost of shading (and potentially shadowing) with each light source in the scene. To achieve this goal we need criteria allowing us to choose when the approximate shading gives a satisfactory result, thus permitting us to terminate our descent into the light hierarchy.

In what follows we first present some necessary preliminaries on shading for ray-tracing, and then proceed to describe the error bounds developed for the diffuse and specular cases. These error bounds are then used as the criteria to perform the actual shading. The algorithms for the shading are described for each case.

4.1. Preliminaries

A common shading formulation divides the reflection as a combination of diffuse (pure lambertian) reflection and specular (directional) reflection. One such popular model suggested by Phong¹¹ is shown in Fig. 3, and can be expressed as

$$I = \sum_{i=1}^m S_i f_{att_i} I_i \left(k_d (\vec{N} \cdot \vec{L}_i) + k_s (\vec{R}_i \cdot \vec{E})^n \right) \quad (1)$$

where

- I = light radiance going from the light to the viewer and passing by p ,
- i = i^{th} light source,
- m = number of light sources,
- S = visibility factor of the light source,
- f_{att} = attenuation of light from the light source,
- I_i = intensity of light source i ,
- k_d = diffuse reflection coefficient of the surface,
- \vec{N} = surface normal at p ,
- \vec{L} = vector from p to the light source,
- k_s = specular reflection coefficient of the surface,
- \vec{R} = mirror reflection of the vector \vec{L} at p with respect to \vec{N} ,
- \vec{E} = vector from p towards viewer,
- n = roughness coefficient.

All the vectors are normalized.

We first derive some bounds for the diffuse reflection, and then present the bounds for the specular reflection.

4.2. Diffuse Reflection

We slightly simplify the formulation for clarity, replacing the terms $S = 1$ (unoccluded case) and $f_{att} = 1/d^2$ where d is the distance from the light source to point p . We also replace the dot products using the identities :

$$\begin{aligned} \cos(\theta) &= \vec{N} \cdot \vec{L} \\ \cos(\theta_i) &= \vec{N} \cdot \vec{L}_i \end{aligned}$$

The light reflected by a diffuse surface illuminated by m point light sources can be computed as:

$$I = \sum_{i=1}^m I_i \frac{1}{d_i^2} k_d \cos(\theta_i). \quad (2)$$

By taking a single virtual point light source approximating all m point light sources, we compute:

$$I_{approx} = \left(\sum_{i=1}^m I_i \right) \frac{1}{d^2} k_d \cos(\theta). \quad (3)$$

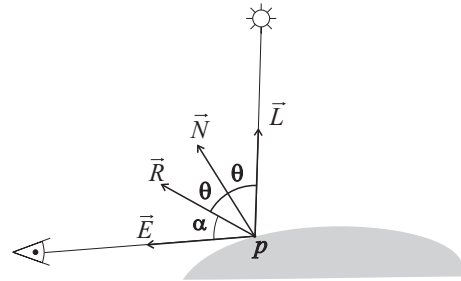


Figure 3: Phong reflection model

Intensity	$I_v = \sum_{i=1}^m I_i$
Position	$P_v = (\sum_{i=1}^m P_i \ I_i\ _1) / (\sum_{i=1}^m \ I_i\ _1)$
Dispersion	$B_{min} = \text{AxisAlignedBox}(\min(Px_i), \min(Py_i), \min(Pz_i), \max(Px_i), \max(Py_i), \max(Pz_i))$

Table 1: Cluster attributes

4.2.1. Error Bound for the Diffuse Model

Given the expressions of the exact and the approximate illumination at a point, we need to develop a bound on the error committed by the approximation. This bound must be tight and efficient to evaluate so it can guide a hierarchical shading algorithm at a reasonable cost. In particular, we are looking for a function ΔI_{abs} such that for any point to shade

$$\Delta I_{abs} \geq |I - I_{approx}|.$$

To derive a suitable expression, we first define a few variables:

$$\begin{aligned} \Delta d_i &= d_i - d \\ \Delta \theta_i &= \theta_i - \theta \\ \Delta d_{max} &= \text{diag} \\ \Delta \theta_{max} &= \arctan\left(\frac{\Delta d_{max}}{d - \Delta d_{max}}\right) \end{aligned}$$

where diag is the largest diagonal of the minimal bounding box at a node of the light octree. These quantities are illustrated in Fig. 4 and 5.

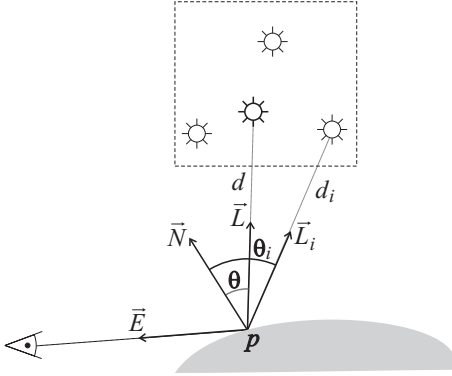


Figure 4: Important quantities for the diffuse reflection model

The following inequalities can then be applied to derive the desired bound:

$$\begin{aligned} 0 &\leq \Delta d_{max} < d \\ (0, 0, 0) &\leq k_d, I_i \\ 0 &\leq \theta, \theta_i \leq \pi/2 \\ 0 &\leq \cos(\theta), \cos(\theta_i) \leq 1. \end{aligned}$$

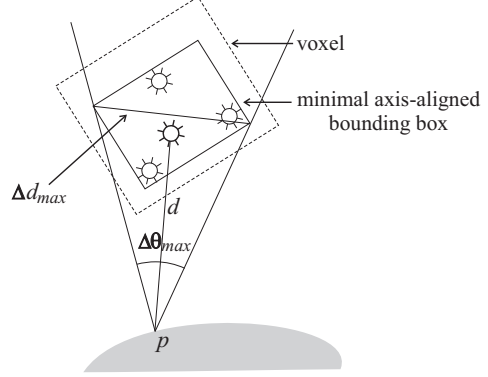


Figure 5: Bounds on the quantities related to a cluster

By bounding the approximate intensity between a minimal and a maximal intensity, I_{min} and I_{max} , ΔI_{abs} becomes:

$$\Delta I_{abs} = \max(I_{approx} - I_{min}, I_{max} - I_{approx}).$$

The derivation of the bound from these equations and inequalities is given in Appendix A. The bound on the error produced by approximating the diffuse reflection at a point p by a single virtual point light source at a node of our octree structure is:

$$\Delta I_{abs} = k_d I_v \max\left(\frac{\cos(\theta)}{d^2} - \frac{\max(0, \cos(\theta) - \Delta \theta_{max})}{(d + \Delta d_{max})^2}, \frac{\min(1, \cos(\theta) + \Delta \theta_{max})}{(d - \Delta d_{max})^2} - \frac{\cos(\theta)}{d^2}\right). \quad (4)$$

The cost of computing this error and the actual illumination using the virtual light source is about the same as that of computing the illumination due to three to four point light sources. The bound is therefore useful when the approximation replaces the evaluation of the illumination from at least three point light sources.

4.2.2. Diffuse Shading using the Light Hierarchy

Equation 4 tells us when a node of the light hierarchy is below the desired threshold. This means the virtual light source associated with this node could replace all the point light sources below this node. However if the traversal of the light hierarchy stops at several nodes, while each individual bound can be below the threshold, their sum might not be.

If the sum of the errors associated with the voxels stored in

the error list is greater than the threshold, we treat the nodes with the highest error at a lower level. We continue this process until the sum of the errors is below the threshold. This way we ensure that our absolute error bound is respected.

The algorithm in Fig. 6 illustrates how the light hierarchy is used to compute the shading.

```

HierarchicalIllumination( Point p, Voxel v )
{
    if v.Empty()
        return( 0 )

    if v.NumberOfSources() ≤ 3
        return( SimpleIllumination( p, v.sources ) )

    Evaluate  $I_v, k_d, d, \Delta d_{max}, \theta, \Delta \theta_{max}$  from p and v.

    if  $\Delta d_{max} > d$ 
        // We cannot evaluate our bound if  $\Delta d_{max} > d$ .
        return( IlluminationAtLowerLevel( p, v ) )

    error = DiffuseBound(  $I_v, k_d, d, \Delta d_{max}, \theta, \Delta \theta_{max}$  )

    // Compare with diffuse error threshold
    if error <  $T_{diffuse}$ 
        AddToErrorList( v, error )
        return( 0 )
    else
        return( IlluminationAtLowerLevel( p, v ) )
}

IlluminationAtLowerLevel( Point p, Voxel v )
{
    if v.Subdivided()
        ∀ v.voxelChild
            sum += HierarchicalIllumination( p, v.voxelChild )
        return( sum )
    else
        return( SimpleIllumination( p, v.sources ) )
}

```

Figure 6: Illumination algorithm using light hierarchy for diffuse surfaces

4.3. Specular Reflection

Specular reflection is mostly associated with highlights on surfaces. The light reflected by a specular surface illuminated by m point light sources can be computed as:

$$I = \sum_{i=1}^m I_i \frac{1}{d_i^2} k_s \cos^n(\alpha_i). \quad (5)$$

Phong specular reflection has the form of a specular lobe $\cos^n(\alpha)$, where n controls the “eccentricity” of the lobe. The larger the value of n , the smaller the simulated roughness of the surface, and the sharper and narrower the highlights.

We consider here specular reflections off surfaces with a

high roughness coefficient (values of $n > 20$). Specular reflections with a low roughness coefficient could be handled with an algorithm similar to the diffuse one.

For specular surfaces with a high roughness coefficient, typically very few light sources will contribute significantly to the illumination at a particular point. We can therefore use the light hierarchy to quickly identify the important light sources. Our approach is based on computing the maximal potential contribution of a voxel of light sources. Only those voxels with a potential contribution greater than a specified specular threshold T_{spec} will be treated at a finer level.

4.3.1. Error Bound for the Specular Model

We need to compute the maximal potential contribution of a voxel. It is derived from the Phong equation in a similar way to that of the diffuse error bound. Again, the complete derivation of this bound is given in Appendix A. The maximal specular contribution corresponds to

$$maxC = I_v k_s \frac{(\min(\cos(\alpha) + \Delta\alpha_{max}, 1))^n}{(d - \Delta d_{max})^2} \quad (6)$$

where $\Delta\alpha_{max}$ is computed in the same way as $\Delta\theta_{max}$ and $\cos(\alpha) = \vec{R} \cdot \vec{E}$.

4.3.2. Specular Shading using the Light Hierarchy

Starting at the root of the light hierarchy, if the maximal potential contribution of a voxel is greater than T_{spec} , we open that voxel and treat its children. If the potential contribution is below T_{spec} , we add the voxel to the list of ignored contributions. After the traversal of the light hierarchy is completed, we check the list of ignored contributions to ensure that its sum is lower than T_{spec} . As for the diffuse reflection, we then recompute the hierarchical illumination of the nodes with the larger maximal contribution at a lower level. This process is repeated until the sum of the ignored contributions is lower than T_{spec} . When the threshold is respected, the maximal error present in the image will be equal or smaller to it. Choosing a very small T_{spec} results in no visual artifacts.

Instead of ignoring the contributions of these voxels, we could adopt an approach similar to Ward¹⁷, and add an approximation of the contribution of these voxels based on the illumination of their virtual light sources. This way, the resulting error would be smaller and we could use a larger threshold resulting in greater speedups. While reducing the error, this would not change the value of the bound. The average error would be smaller, but the maximal error would stay the same.

4.4. Combined Diffuse and Specular Reflections

To handle surfaces with both diffuse and specular reflections, the two algorithms are mainly executed independently. Some calculations as well as the traversal of the hierarchy

are shared, but the main parts of the algorithms are treated separately. The rendering times are a little less than the simple sum of both times, but are still proportional to it.

5. Results

5.1. Basic Test Scenes

We have developed a set of test scenes which allows us to evaluate our approach in several different configurations, while keeping a low intersection cost between rays and the scene made of only a few polygons. The three main scenes treated are shown in the following (the names subsequently used are in quotes): the first is an image of “*Light Strings*” lighting a street (Fig. 7(a)); the second is a scene lit by light sources forming the “*U de M*” logo (Fig. 7(b)); and the last is a simple scene illuminated by a “*Cluster*” of light sources (Fig. 7(c)). The rendering of this last scene is done with the camera on the left, just behind the light sources. In our tests, the rendering of this scene occurs with the light sources being invisible. All the light sources in one scene have equal intensities.

We present results on variations of these scenes containing a number of light sources ranging from 64 to 16384. This is done by increasing the density of the light sources in each case.

For all the specular tests, the roughness factor we use is $n = 200$.

In the result tables presented below, we show the cost of standard ray-tracing with octree acceleration (**RT**), the cost of Ward’s (**Ward**) method¹⁷, the cost of the light hierarchy method (**LH**), and the speedup achieved by our method over the ray-tracing method (**SU**). Our implementation of Ward’s method is slightly different than the one presented in his paper¹⁷ to ensure that, as with our algorithm, it has an absolute error bound.

For each test, the error thresholds (diffuse and specular) we use are 1% RGB or (2.55, 2.55, 2.55) RGB for images quantized in [0..255]. Since the actual error is smaller than the threshold, the test runs result in images that are visually indistinguishable from those computed with the traditional ray-tracing. The maximal observed error at a pixel is (1, 1, 1) RGB for all the images and the average pixel error is negligible since it is less than (0.02, 0.02, 0.02) RGB.

5.2. Diffuse Case

In Table 2 we present the results of our algorithm for diffuse surfaces. Compared to standard ray-tracing, we see that the light hierarchy achieves important speedups (up to 90 times faster). For a high number of light sources (greater than 1024), we are consistently faster than Ward’s method. It should be noted that with Ward’s method, the increase in rendering time does not seem to be logarithmic. In comparison, our method shows a logarithmic increase in time for the

Light Strings and *U de M* scenes, as can be seen in Fig. 8. The more linear behavior of the *Cluster* scene indicates that

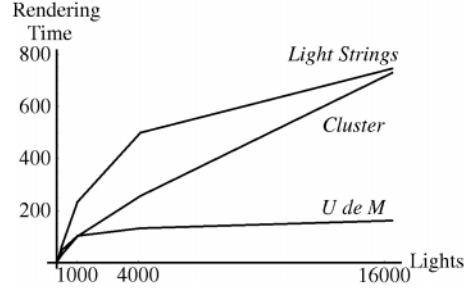


Figure 8: Logarithmic behavior of the light hierarchy method

for this scene, more light sources are required before reaching the “plateau” part of the logarithmic curve.

Scene	RT	Ward	LH	SU
Light Strings				
64 lights	11.4	70.2%	91.2%	1.1
256 lights	87.5	70.3%	73.5%	1.4
1024 lights	550.6	68.3%	42.4%	2.4
4096 lights	2716.3	70.8%	18.4%	5.4
16384 lights	13157.1	72.5%	5.7%	17.6
U de M				
64 lights	14.8	64.9%	85.1%	1.2
256 lights	108.6	60.0%	44.4%	2.3
1024 lights	632.6	57.8%	16.2%	6.2
4096 lights	3114.9	58.7%	4.2%	23.6
16384 lights	14609.3	59.5%	1.1%	90.1
Cluster				
64 lights	8.6	107.0%	95.3%	1.0
256 lights	33.9	112.4%	90.9%	1.1
1024 lights	136.1	119.6%	74.1%	1.3
4096 lights	545.5	127.6%	47.0%	2.1
16384 lights	2179.6	135.6%	33.5%	3.0

Table 2: Results for the algorithm treating diffuse surfaces. All timings in seconds on a R10000 SGI computer. Notice how our method (LH) outperforms Ward’s approach as the number of light sources increases.

5.3. Specular Case

In Table 3 we present the results of our algorithm for the cases above using the specular algorithm. We see that the light hierarchy achieves an impressive speedup compared to simple ray-tracing (at least 6.4 times faster), and that it is always faster than both the ray-tracing and Ward’s method. Compared to Ward’s approach we also achieve significant speedup: up 32 times faster.

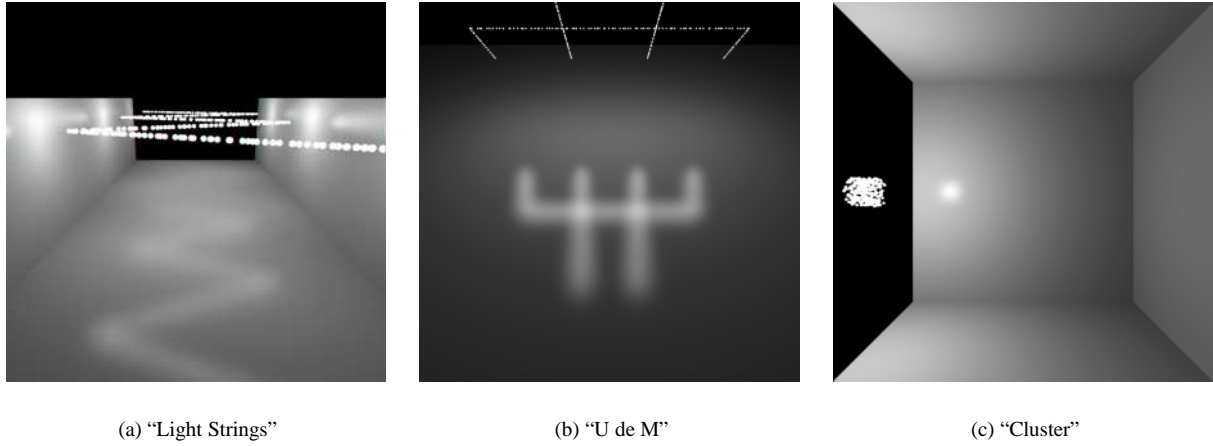


Figure 7: The three different test scenes

Scene	RT	Ward	LH	SU
Light Strings				
64 lights	12.8	26.6%	15.6%	6.4
256 lights	93.5	17.2%	8.3%	12.0
1024 lights	575.4	13.1%	6.3%	15.8
4096 lights	2815.3	12.3%	5.0%	20.1
16384 lights	13549.3	11.5%	3.9%	26.0
U de M				
64 lights	17.1	25.1%	7.0%	14.2
256 lights	117.9	16.1%	3.1%	32.8
1024 lights	670.2	12.5%	1.8%	55.4
4096 lights	3261.5	11.5%	1.3%	76.4
16384 lights	15195.3	11.0%	1.1%	91.6
Cluster				
64 lights	10.4	39.4%	5.8%	17.3
256 lights	41.0	43.7%	3.4%	29.3
1024 lights	164.1	49.7%	2.6%	39.1
4096 lights	656.7	56.0%	2.3%	43.8
16384 lights	2632.3	62.4%	1.9%	51.6

Table 3: Results for the algorithm treating specular surfaces. All timings in seconds on a R10000 SGI computer. Notice how our method (LH) outperforms Ward’s approach everywhere.

5.4. Discussion of Results

Overall, from tables 2 and 3 we see that the light hierarchy can provide a very important speedup in computation times compared to Ward’s approach and to ray-tracing.

The difference in speedup provided by the diffuse and specular algorithms is explained by the fact that, for highly

specular surfaces, no shading computations need to be done for many of the pixels. For the diffuse case, most of the pixels have a large illumination value thus a large error to compensate by using lower levels. Using lower levels demands more work since we need to process more virtual or real light sources.

The *Cluster* scene shows less interesting results than the two other scenes for the diffuse case. As said before, more light sources are needed to reach the “plateau” part of the logarithmic curve. This result seems surprising for a scene with so well placed light sources. In fact, the light sources are not so well placed. They reside in a small cube, but in that cube, they are almost uniformly distributed. This results in a hierarchy that is extremely well balanced, with most of its nodes at a particular level containing the same number of light sources. This obviously results in a hierarchy with fewer levels. When using the hierarchical rendering, we will be often forced to completely calculate the illumination of a node since we do not have a finer representation for it. Uniform distribution of light sources also means that for a given level, we will have many nodes with few light sources compared to many light sources in few (non-empty) nodes if the distribution of light sources is not uniform (as in the *Light Strings* and *U de M* scenes). For the same number of light sources, we will have to sum the potential error of many voxels for the *Cluster* scene compared to the *Light Strings* and *U de M* scenes.

It is interesting to look at what happens when we increase the threshold. As it increases, voxels higher in the hierarchy are used to compute the shading. At a certain point, the resulting illumination shows discontinuities due to the choice of different levels of the hierarchy. Fig. 9 shows the resulting artifacts. To reduce these discontinuities, linear interpolation between the two levels could be used instead. The thresh-

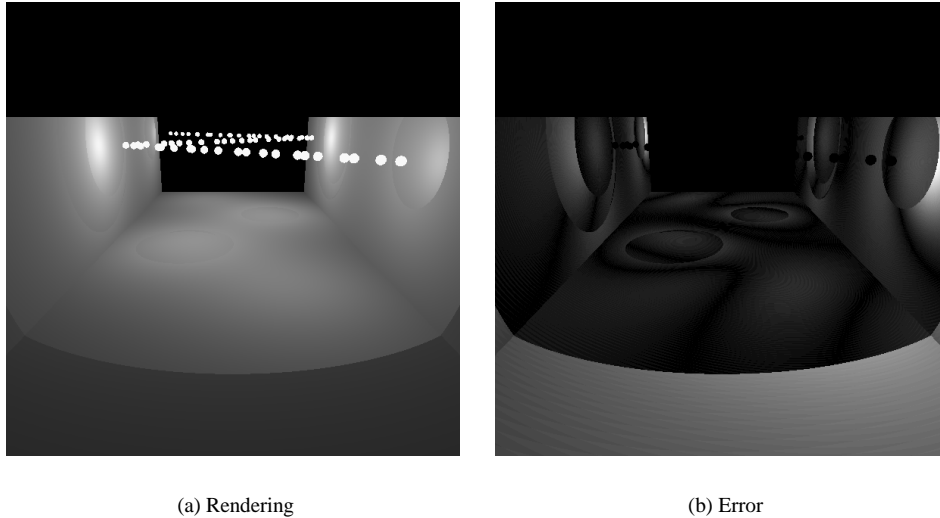


Figure 9: Artifacts related to the increase of the diffuse threshold. (a) is the actual image while (b) is the error image. The intensity and contrast of the error image have been adjusted such that black represents no error and white the maximal error that occurred in this image, (59, 59, 59) RGB.

old we are using for our tests can still be increased without resulting in the artifacts present in Fig. 9. While keeping a maximal error of (1, 1, 1) RGB, we can increase the threshold to cut the rendering time of our tests by half. In fact, our threshold is approximately ten times greater than the actual (maximal) error. This is obviously conservative, but tolerable. Increasing the threshold increases the potential error in the image and the rendering speed. Fig. 9 for example, shows a speed up of ten compared to simple ray tracing with only 64 light sources. This is about 9 times faster than the result of Table 2.

6. Possible Extensions to Handle Occlusions

As mentioned in the introduction, the algorithms and data structures we present here treat the case of unoccluded illumination only. The goal of our paper is thus to present the principles of the light hierarchy and demonstrate, through experimental results, that the potential for speedup is very important. Nonetheless, it is clear that the utility of the light hierarchy will be much more widespread when occlusion is treated completely. For this reason, we present next our first ideas on the treatment of shadows.

We also list other improvements to the algorithms presented as well as interesting new research directions.

6.1. Treating Shadows

The work presented here shows very encouraging results for illumination without occlusion. In the presence of shadows,

one can only hope to have more important gains from the use of the light hierarchy, since the cost of each light ray is multiplied by the cost of the actual ray intersection with the scene.

Even though our error bounds do not include shadow information, the maximum contribution (upper bound) is nonetheless valid, since shadowing can only reduce this contribution. We can thus use our hierarchical illumination algorithm as a basis of a solution for shadows. The problems of determining a lower bound, or at least an estimate of the minimum contribution, as well as the more delicate issue of preservation of shadow shapes have to be treated separately.

6.1.1. Volumetric Soft Shadows

An approach for using the light hierarchy for scenes with shadows could consist in using the hierarchy when a cluster of light sources is entirely visible from a given 3D point to shade. To do this, we need to make a decision on whether a voxel of the light hierarchy at a given level is completely unoccluded, completely occluded, or partially visible from the 3D point. In the first case, we use the algorithms presented above in Section 3; in the second case we do not need to shade at all; for the partially visible case, we must descend into the light hierarchy.

Algorithms for consistent visibility determination have been presented for other problems (e.g., shafts by Haines and Wallace²⁰, or conservative triage by Teller and Hanrahan²¹). What is unclear for these approaches is whether the cost of the visibility determination would make the gains of the light

hierarchy negligible or even useless. This research direction is nonetheless worthy of further investigation.

As an alternative, we can use an approximation to visibility determination by using the ideas presented in the work of Sillion²² on volumetric visibility. Without going into details, we can represent the visibility characteristics of a cluster by a set of extinction coefficients, permitting us to avoid ray-intersections with the contents of the cluster of objects.

6.2. Hierarchy Construction and More General Models

The light hierarchy currently used is constructed very rapidly since we simply subdivide an octree. It is possible that more involved clustering approaches, such as that described in the work of Cazals *et al.*²³, which take into account certain geometric properties of the items being clustered (in our case the light sources), could result in improved performance.

The approach described here is not restricted to the sole use of ray-tracing direct light. The central ideas and concepts of our approach could be applied to improve on the clustering of light sources in radiosity-based algorithms.

7. Conclusions

The introduction of a new data structure in the form of a *light hierarchy* provides an efficient solution to the problem of ray-tracing scenes with many light sources. We have chosen to create an octree hierarchy of the light sources in a scene which is maintained independently of the rest of the geometry. Intermediate nodes of the light hierarchy approximate the illumination due to the light sources contained in the children octree voxels, by means of *virtual light sources*.

Error bounds on the error committed when using the virtual light sources were presented, both for the diffuse and the specular cases. Based on these bounds, the shading calculation at each visible point is performed by a hierarchical descent in the light hierarchy. When the descent ceases at a given hierarchical level, we avoid the cost of shading with all the light sources contained below that level, resulting in significant speedup.

To carefully evaluate the ideas and develop a deeper understanding of the issues involved, we have currently restricted our algorithm to the case of unoccluded scenes. The results for a set of such test scenes show very encouraging speedup, of up to 90 times for both diffuse and specular surfaces.

We believe that the introduction of the light hierarchy for ray-tracing, in conjunction with the error bounds and the hierarchical descent open a very promising research direction for efficient rendering of scenes with many light sources. The development of the complete solution including shadows could lead to significant acceleration of the rendering times when scenes with complex geometry would be used.

Acknowledgments

We acknowledge financial support from NSERC and FCAR. Thanks to the anonymous reviewers for their comments which improved the final version.

Appendix A: Derivation of the bounds

Diffuse bound

We first state few useful rules:

$$0 < y, 0 < \epsilon \Rightarrow \frac{|x|}{y} > \frac{|x|}{y+\epsilon} \quad (7)$$

$$\cos(\phi) - 1|\sigma| \leq \cos(\phi + \sigma) \leq \cos(\phi) + 1|\sigma| \quad (8)$$

$$\cos(\theta_i) = \min(1, \cos(\theta_i)) = \max(0, \cos(\theta_i)). \quad (9)$$

We approximate the minimal diffuse illumination by:

$$\begin{aligned} I &= \sum_{i=1}^m I_i \frac{1}{d_i^2} k_d \cos(\theta_i) \\ &= k_d \sum_{i=1}^m I_i \frac{\cos(\theta + \Delta\theta_i)}{(d + \Delta d_i)^2} \\ &\geq k_d \sum_{i=1}^m I_i \frac{\cos(\theta + \Delta\theta_i)}{(d + \Delta d_{\max})^2} && \text{by 7} \\ &\geq k_d \sum_{i=1}^m I_i \frac{\max(0, \cos(\theta + \Delta\theta_i))}{(d + \Delta d_{\max})^2} && \text{by 9} \\ &\geq k_d \sum_{i=1}^m I_i \frac{\max(0, \cos(\theta) - |\Delta\theta_i|)}{(d + \Delta d_{\max})^2} && \text{by 8} \\ &\geq k_d \sum_{i=1}^m I_i \frac{\max(0, \cos(\theta) - \Delta\theta_{\max})}{(d + \Delta d_{\max})^2} \\ &\geq k_d I_v \frac{\max(0, \cos(\theta) - \Delta\theta_{\max})}{(d + \Delta d_{\max})^2} = I_{\min}. \end{aligned}$$

The same way, we approximate the maximal diffuse illumination by:

$$\begin{aligned} I &= \sum_{i=1}^m I_i \frac{1}{d_i^2} k_d \cos(\theta_i) \\ &= k_d \sum_{i=1}^m I_i \frac{\cos(\theta + \Delta\theta_i)}{(d + \Delta d_i)^2} \\ &\leq k_d \sum_{i=1}^m I_i \frac{\cos(\theta + \Delta\theta_i)}{(d - \Delta d_{\max})^2} && \text{by 7} \\ &\leq k_d \sum_{i=1}^m I_i \frac{\min(1, \cos(\theta + \Delta\theta_i))}{(d - \Delta d_{\max})^2} && \text{by 9} \\ &\leq k_d \sum_{i=1}^m I_i \frac{\min(1, \cos(\theta) + |\Delta\theta_i|)}{(d - \Delta d_{\max})^2} && \text{by 8} \\ &\leq k_d \sum_{i=1}^m I_i \frac{\min(1, \cos(\theta) + \Delta\theta_{\max})}{(d - \Delta d_{\max})^2} \\ &\leq k_d I_v \frac{\min(1, \cos(\theta) + \Delta\theta_{\max})}{(d - \Delta d_{\max})^2} = I_{\max}. \end{aligned}$$

From there, the bound on the error is the maximum between $I_{\text{approx}} - I_{\min}$ and $I_{\max} - I_{\text{approx}}$. Re-arranging few

terms gives:

$$\Delta I_{abs} = k_d I_v \max \left(\frac{\cos(\theta)}{d^2} - \frac{\max(0, \cos(\theta) - \Delta\theta_{max})}{(d + \Delta d_{max})^2}, \frac{\min(1, \cos(\theta) + \Delta\theta_{max})}{(d - \Delta d_{max})^2} - \frac{\cos(\theta)}{d^2} \right). \quad (10)$$

Specular bound

The specular bound is simply an approximation of the maximal specular illumination:

$$\begin{aligned} I &= \sum_{i=1}^m I_i \frac{1}{d_i^2} k_s \cos^n(\alpha_i) \\ &= k_s \sum_{i=1}^m I_i \frac{\cos^n(\alpha + \Delta\alpha_i)}{(d + \Delta d_i)^2} \\ &\leq k_s \sum_{i=1}^m I_i \frac{\cos^n(\alpha + \Delta\alpha_i)}{(d + \Delta d_{max})^2} && \text{by 7} \\ &\leq k_s \sum_{i=1}^m I_i \frac{\min(1, \cos^n(\alpha + \Delta\alpha_i))}{(d + \Delta d_{max})^2} && \text{by 9} \\ &\leq k_s \sum_{i=1}^m I_i \frac{\min(1, \cos^n(\alpha + |\Delta\alpha_i|))}{(d + \Delta d_{max})^2} && \text{by 8} \\ &\leq k_s \sum_{i=1}^m I_i \frac{\min(1, \cos^n(\alpha + \Delta\alpha_{max}))}{(d + \Delta d_{max})^2} \\ &\leq k_s I_v \frac{\min(1, \cos^n(\alpha + \Delta\alpha_{max}))}{(d + \Delta d_{max})^2} = \max C. \end{aligned}$$

References

1. T. Whitted, "An improved illumination model for shaded display", *Communications of the ACM*, **23**(6), pp. 343–349 (1980).
2. M. F. Cohen and J. R. Wallace, *Radiosity and Realistic Image Synthesis*. San Diego, CA: Academic Press Professional, (1993). Excellent book on radiosity algorithms.
3. F. Sillion and C. Puech, *Radiosity and Global Illumination*. San Francisco: Morgan Kaufmann, (1994). excellent coverage of radiosity and global illumination algorithms.
4. P. Shirley, C. Y. Wang, and K. Zimmerman, "Monte carlo techniques for direct lighting calculations", *ACM Transactions on Graphics*, **15**(1), pp. 1–36 (1996). ISSN 0730-0301.
5. A. S. Glassner, "Space subdivision for fast ray tracing", *IEEE Computer Graphics and Applications*, **4**(10), pp. 15–22 (1984).
6. A. Fujimoto, T. Tanaka, and K. Iwata, "Arts: Accelerated ray-tracing system", *IEEE Computer Graphics and Applications*, pp. 16–26 (1986).
7. S. M. Rubin and T. Whitted, "A 3-dimensional representation for fast rendering of complex scenes", *Computer Graphics*, **14**(3), pp. 110–116 (1980).
8. P. Hanrahan, D. Salzman, and L. Aupperle, "A rapid hierarchical radiosity algorithm", in *Computer Graphics (SIGGRAPH '91 Proceedings)* (T. W. Sederberg, ed.), vol. 25, pp. 197–206, (July 1991).
9. B. Smits, J. Arvo, and D. Greenberg, "A clustering algorithm for radiosity in complex environments", in *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)* (A. Glassner, ed.), Computer Graphics Proceedings, Annual Conference Series, pp. 435–442, ACM SIGGRAPH, ACM Press, (July 1994). ISBN 0-89791-667-0.
10. F. Sillion, "Clustering and volume scattering for hierarchical radiosity calculations", in *Fifth Eurographics Workshop on Rendering*, (Darmstadt, Germany), pp. 105–117, (June 1994).
11. B.-T. Phong, "Illumination for computer generated pictures", *Communications of the ACM*, **18**(6), pp. 311–317 (1975).
12. C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Battaile, "Modelling the interaction of light between diffuse surfaces", in *Computer Graphics (SIGGRAPH '84 Proceedings)*, vol. 18, pp. 212–22, (July 1984).
13. P. Hanrahan and D. Salzman, "A rapid hierarchical radiosity algorithm for unoccluded environments", in *Proceedings Eurographics Workshop on Photosimulation, Realism and Physics in Computer Graphics*, (Rennes, France), pp. 151–71, (June 1990).
14. M. F. Cohen and D. P. Greenberg, "The Hemi-Cube: A radiosity solution for complex environments", in *Computer Graphics (SIGGRAPH '85 Proceedings)* (B. A. Barsky, ed.), vol. 19, pp. 31–40, (Aug. 1985).
15. A. Woo, P. Poulin, and A. Fournier, "A survey of shadow algorithms", *IEEE Computer Graphics and Applications*, **10**(6), pp. 13–32 (1990).
16. P. Bergeron, "A general version of crow's shadow volumes", *IEEE Computer Graphics and Applications*, **6**(9), pp. 17–28 (1986).
17. G. Ward, "Adaptive shadow testing for ray tracing", in *Eurographics Workshop on Rendering*, (1991).
18. C. Houle and E. Fiume, "Light-source modeling using pyramidal light maps", *CVGIP: Graphical Models and Image Processing*, **55**(5), pp. 346–358 (1993).
19. J. Stam and E. Fiume, "Depicting fire and other gaseous phenomena using diffusion processes", in *SIGGRAPH 95 Conference Proceedings* (R. Cook, ed.), Annual Conference Series, pp. 129–136, ACM SIGGRAPH, Addison Wesley, (Aug. 1995). held in Los Angeles, California, 06-11 August 1995.
20. E. Haines and J. Wallace, "Shaft culling for efficient ray-traced radiosity", in *Eurographics Workshop on Rendering*, (1991).

21. S. Teller and P. Hanrahan, “Global visibility algorithms for illumination computations”, in *Computer Graphics Proceedings, Annual Conference Series, 1993*, pp. 239–246, (1993).
22. F. X. Sillion, “A unified hierarchical algorithm for global illumination with scattering volumes and object clusters”, *IEEE Transactions on Visualization and Computer Graphics*, **1**(3), pp. 240–254 (1995). ISSN 1077-2626.
23. F. Cazals, G. Drettakis, and C. Puech, “Filtering, clustering and hierarchy construction: a new solution for ray tracing very complex environments”, *Computer Graphics Forum (Proc. of Eurographics '95)*, **15**(3), pp. 371–382 (1995).

3.5.3 An Efficient Progressive Refinement Strategy for Hierarchical Radiosity (EGRW'94)

Auteurs : Nicolas Holzschuch, François Sillion et George Drettakis

Actes : 5th Eurographics Workshop on Rendering

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 results in a quadratic cost with respect to the number of input surfaces, which seriously impairs

* 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.

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 of-*

Table 1. Description of the five test scenes.

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

fice", 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 hierarchical 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 visibility calcula-

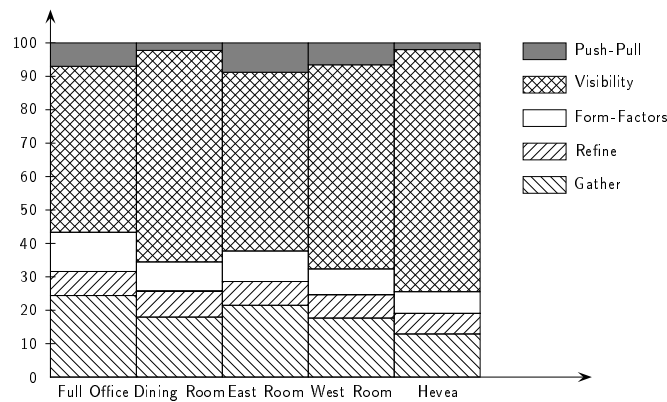


Fig. 1. Relative time spent in each procedure.

tions dominate the computation in the hierarchical algorithm². 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 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.

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
<i>Full office</i>	170	301	5.13
<i>Dining room</i>	402	4824	436
<i>East room</i>	1006	587	194
<i>West room</i>	1647	1017	476
<i>Hevea</i>	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 rely on the original initial linking stage. Thus initial linking tends to become the most expensive step of the algorithm⁴.

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 steps⁵. Figure 2

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.

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.

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.

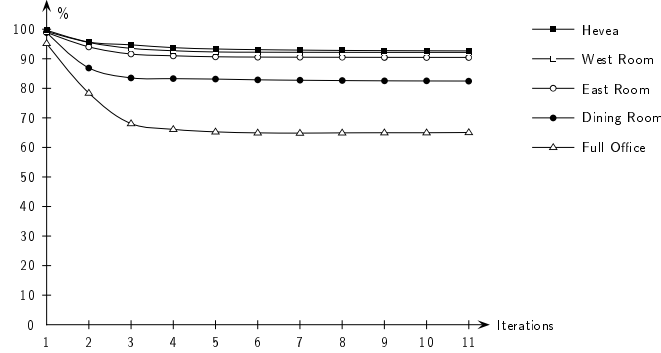


Fig. 2. Percentage of links for which BF does not exceed $\varepsilon_{\text{refine}}$.

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 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 $\text{MinArea} = 0.001$.

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 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.

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 transferred 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 $\varepsilon_{\text{link}} = \varepsilon_{\text{refine}}/5$.

```

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  $B \times F > \epsilon_{\text{refine}}$ 
      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  $B_p > \epsilon_{\text{link}}$  or  $B_q > \epsilon_{\text{link}}$ 
      compute the unoccluded FF
      if  $B \times F > \epsilon_{\text{link}}$ 
        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  $B \times F > \epsilon_{\text{refine}}$ 
        refine  $l$ 
    for each link  $l$  leaving  $p$ 
      gather  $l$ 

```

Fig. 4. Pseudo-code listing for our algorithm

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 cost⁶.

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

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

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 imbalance.

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

$$\mathcal{E}_{E_T} = \sum_p |1 - F_p| B_p A_p \quad (1)$$

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:

$$E_T = \sum_p B_p A_p \quad (2)$$

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].

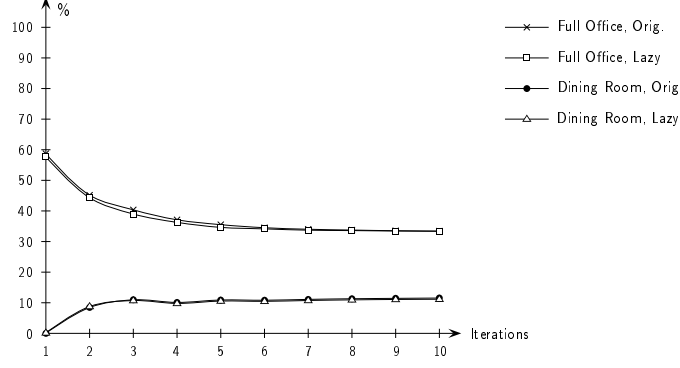


Fig. 5. Incorrect Energy \mathcal{E}_{E_T} / E_T

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-directional 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

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

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

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

$$F_{q,p} = \sum_i F_{q,p_i} \quad (5)$$

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* whether 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:

$$\begin{aligned} F_{p,q}^{\min} &= \min_i (F_{p_i,q}) & F_{q,p}^{\min} &= \min_i (F_{q,p_i}) \\ F_{p,q}^{\max} &= \max_i (F_{p_i,q}) & F_{q,p}^{\max} &= \max_i (F_{q,p_i}) \\ F'_{p,q} &= \frac{1}{A_p} (\sum_i A_i F_{p_i,q}) & F'_{q,p} &= \sum_i F_{q,p_i} \end{aligned}$$

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

$$\begin{aligned} \frac{F_{p,q}^{\max} - F_{p,q}^{\min}}{F_{p,q}^{\max}} &> \varepsilon_{\text{reduce}} & \frac{F_{q,p}^{\max} - F_{q,p}^{\min}}{F_{q,p}^{\max}} &> \varepsilon_{\text{reduce}} \\ \frac{|F'_{p,q} - F_{p,q}|}{F'_{p,q}} &> \varepsilon_{\text{reduce}} & \frac{|F'_{q,p} - F_{q,p}|}{F'_{q,p}} &> \varepsilon_{\text{reduce}} \end{aligned}$$

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

Figure 3 in colour 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.

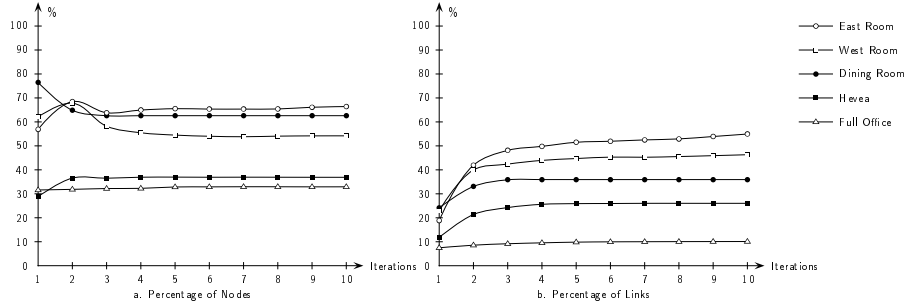


Fig. 6. Percentage of nodes and links left after reduction.

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%.

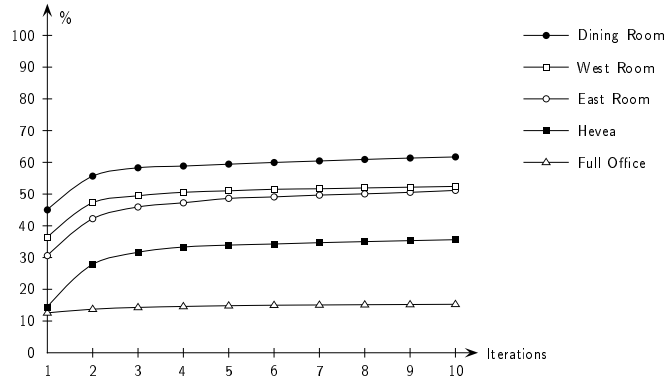


Fig. 7. Percentage of computation time using link reduction.

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 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.

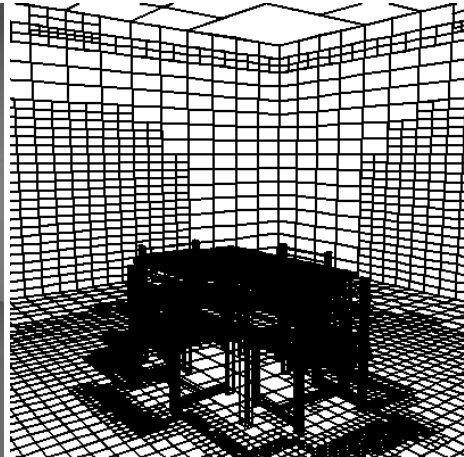
References

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



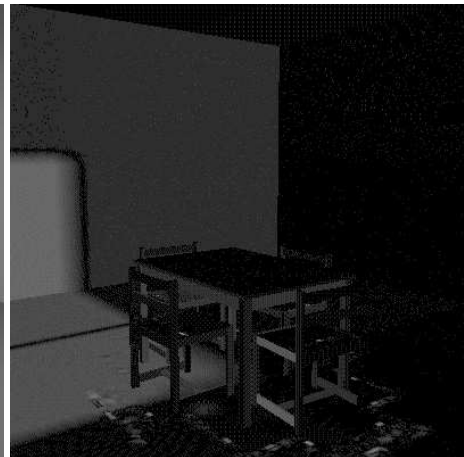
1. The Original Algorithm



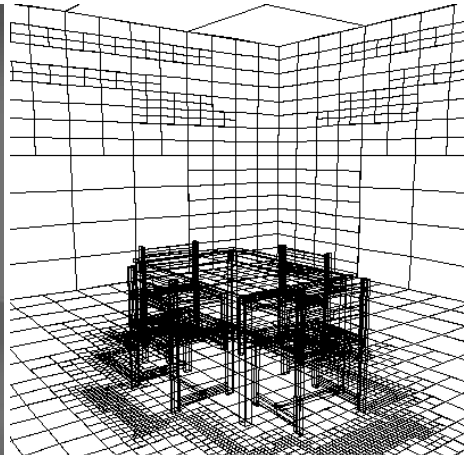
2. The Grid Produced



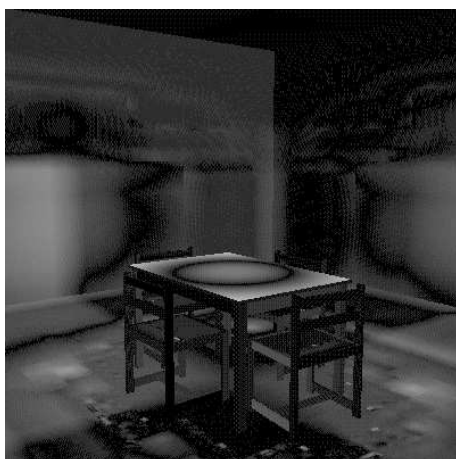
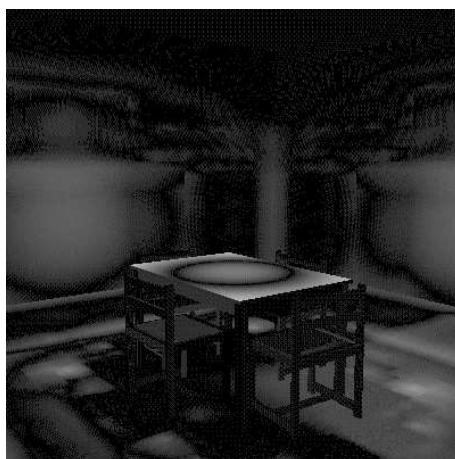
3. With Lazy Linking

4. Diff. Between 1. and 3. ($\times 8$)

5. With Link Reduction



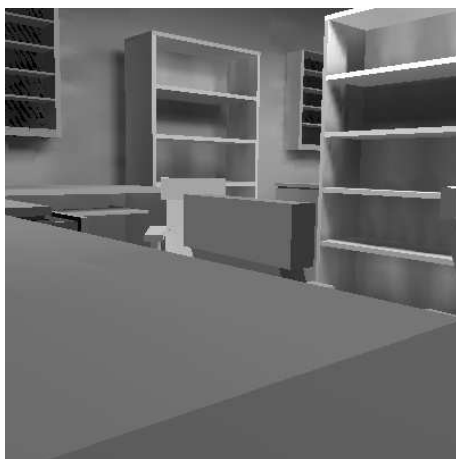
6. The Grid Produced

7. Diff. Between 1. and 5. ($\times 8$)8. Diff. Between 3. and 5. ($\times 8$)

9. Full Office



10. East Room



11. West Room



12. Hevea

3.5.4 Feature-Based Control of Visibility Error : A Multiresolution Clustering Algorithm for Global Illumination'' (SIGGRAPH'95)

Auteurs : François Sillion et George Drettakis

Actes : Congrès SIGGRAPH '95

Date : août 1995

Feature-based Control of Visibility Error: A Multi-resolution Clustering Algorithm for Global Illumination

François Sillion¹ George Drettakis²

¹CNRS ²ERCIM-INRIA
iMAGIS

B.P. 53, 38041 Grenoble Cedex 9, France.

Abstract

In this paper we introduce a new approach to controlling error in hierarchical clustering algorithms for radiosity. The new method ensures that just enough work is done to meet the user's quality criteria. To this end the importance of traditionally ignored visibility error is identified, and the concept of *features* is introduced as a way to evaluate the quality of an image. A methodology to evaluate error based on features is presented, which leads to the development of a *multi-resolution visibility* algorithm. An algorithm to construct a suitable hierarchy for clustering and multi-resolution visibility is also proposed. Results of the implementation show that the multi-resolution approach has the potential of providing significant computational savings depending on the choice of feature size the user is interested in. They also illustrate the relevance of the feature-based error analysis. The proposed algorithms are well suited to the development of interactive lighting simulation systems since they allow more user control. Two additional mechanisms to control the quality of a simulation are presented: The evaluation of internal visibility in a cluster produces more accurate solutions for a given error bound; a progressive multi-gridding approach is introduced for hierarchical radiosity, allowing continuous refinement of a solution in an interactive session.

Keywords: Visibility error, Clustering, Feature-based error metric, Multi-resolution visibility, Hierarchical radiosity, Progressive multi-gridding, Global Illumination.

1 Introduction

Modern global illumination algorithms allow the precise simulation of interreflection effects, penumbras caused by extended light sources, and subtle shading variations caused by complex reflectance properties [2, 15]. Lighting simulation systems operate under very tight and often contradictory constraints: users typically require guaranteed and easily controllable precision levels, with maximum speed for interactive design. An important goal of rendering research is thus to enable the user to reduce the solution error where such reduction is deemed desirable, while at the same time limiting the time spent to achieve this reduction.

Unfortunately, the algorithmic complexity of radiosity methods (quadratic in the number of objects) in effect impairs their use for scenes containing more than a few thousands objects, while Monte-Carlo methods are unable to provide low and medium-quality solutions without too much noise. Therefore means must be found to focus the effort on the most important parts of the calculation.

This paper presents new algorithms and criteria that together allow very fine and efficient user control of the perceived quality of a solution. This is accomplished by first acknowledging the importance of *visibility error*, and using the concept of *features* to evaluate the quality of a solution. This leads to the introduction of *multi-resolution visibility*, which allows precise control of the quality vs. time tradeoff. Additional mechanisms are then discussed to control the quality of a simulation in a working system.

Previous work: error-driven computation and visibility

The introduction of the hierarchical radiosity algorithm [5] was a major step towards the design of practical lighting simulation systems. First, it reduces the overall resource requirements for a given solution. Second, it uses a surface subdivision criterion as an explicit control mechanism. This criterion embodies the priorities used to guide the simulation, as it directs the computational effort to "areas of interest", introducing a natural tool for error estimation.

Hierarchical radiosity (HR) remains quadratic in the number of input objects (since each pair of objects must be linked before hierarchical subdivision begins), and therefore is not suited to large collections of small objects. *Clustering*, the operation of grouping objects together into composite objects that can interact, provides a means to eliminate the quadratic complexity term. Such clustering can be performed manually [11, 7] or automatically [16, 13].

Historically, subdivision criteria for HR first consisted of simple bounds on either the form factor or the exchange of radiosity between two surface patches [5], under the assumption that the error incurred is proportional to the magnitude of the transfer. Using the concept of importance these bounds can be made dependent on the user's interest for each region [17]. However such bounds tend to be quite conservative and thus produce unnecessary subdivision [6].

Recent work has attempted to characterize possible sources of error in global illumination [1], and establish error bounds on radiosity solutions [8]. These error bounds can then be used in the subdivision criterion of a hierarchical algorithm. Since the estimation of the error is decoupled from that of the actual transfer, subdivision can be avoided in regions where significant transfers take place without much error, resulting in better focus of the computational expense.

Existing error controls however typically ignore visibility as a possible source of error, or simply increase the error estimate by a constant factor in situations of partial visibility. Trivial bounds

iMAGIS is a joint research project of CNRS, INRIA, INPG and UJF.
Email: [Francois.Sillion|George.Drettakis]@imag.fr.

of 0 (total occlusion) and 1 (total visibility) are often used. While these bounds are always valid, their use results in unnecessary work being done to narrow down other error bounds by increasing the subdivision. Global visibility algorithms can be used to exploit the structure of architectural scenes and produce guaranteed visibility information [18], but they are not suited to large collections of independent objects. For exchanges between surfaces, *discontinuity meshing* also provides explicit visibility information, and indeed considerably improves the efficiency of HR [10]. However for Monte-Carlo or clustering approaches it is either impossible or impractical to calculate analytic visibility and error bounds must be used. For exchanges between clusters, an approximate visibility estimate can be derived using equivalent volume extinction properties [13], but the error introduced in the process has not yet been analyzed.

Visibility error is admittedly difficult to evaluate, since the computation of visibility itself is a costly process. Still, controlling this source of error is imperative since the quality of shadows plays a significant role in determining the user’s perception of image quality. In complex environments where clustering is most useful, a dominant part of computation time is spent in visibility calculations involving small, geometrically complex objects. Resulting visibility variations produce fine detail shadows, which may be of little interest to the user, or may be lost in the implicit averaging over a surface patch.

Paper overview

The preceding discussion has shown that a key issue in designing efficient lighting simulation systems is to provide adequate control mechanisms to ensure that just enough work is done to meet the user’s quality criteria. It appears that control of visibility error has not yet been attempted, despite its great potential for tightening global bounds and reducing computation costs. The goal of this paper is twofold: first, a new approach to visibility error estimation is proposed, based on *features*, that legitimates the use of a *multi-resolution visibility* algorithm. Second, quality control mechanisms are discussed for interactive simulation systems development.

We begin in Section 2 with the introduction of features to evaluate image quality, and show why existing error metrics are incapable of determining when a given level of detail is satisfactorily represented. A simple metric is then proposed to illustrate how to take into account the user’s interest in a minimal *feature size*. This leads to Section 3 where we explain how to compute *multi-resolution visibility* information using a spatial hierarchy augmented with equivalent *extinction* properties. Selection of a hierarchical level for visibility computation can then be based on the resulting feature size on the receiver. In this paper an application to clustering algorithms is discussed, but multi-resolution visibility is equally promising for Monte Carlo techniques. The construction of a suitable hierarchy is discussed in Section 4. In Section 5 we show that the multi-resolution visibility algorithm successfully generates images (currently for isotropic clusters) in which only selected features sizes are accurately represented, resulting in computational savings. Section 6 presents more quality controls for clustering algorithms, specifically intra-cluster visibility determination in linear time and progressive multi-gridding. We conclude in Section 7.

2 Feature-Based Error Analysis

To a large extent the quality of an image is judged based on how well features of different sizes are represented. It is not easy to characterize what constitutes an illumination feature. For the purposes of this paper, we will consider image features to be the con-

nected regions of varying illumination related to shadows (regions in umbra or penumbra).

2.1 L^p metrics are inadequate for “feature detection”

A major difficulty for accurate lighting simulation is that in general the exact solution is not known at the time of computation. Thus the estimation of the error in a proposed approximation is particularly difficult, and must rely on the computation of error bounds for all algorithmic operations. Even in the case where an exact solution is available, it is not a simple task to define the quality of a given approximation. This is done by choosing a particular error metric to quantify the distance between an approximate solution and the true solution. A “good” metric should therefore convey a sense of the user’s requirements. A central observation in this paper is that when simulating a complex scene, the user is typically interested in capturing illumination variations down to a certain scale. Very small details are not as important, or at least not in all areas of the scene. We strive to define a control mechanism that will avoid any work that would only generate such small details.

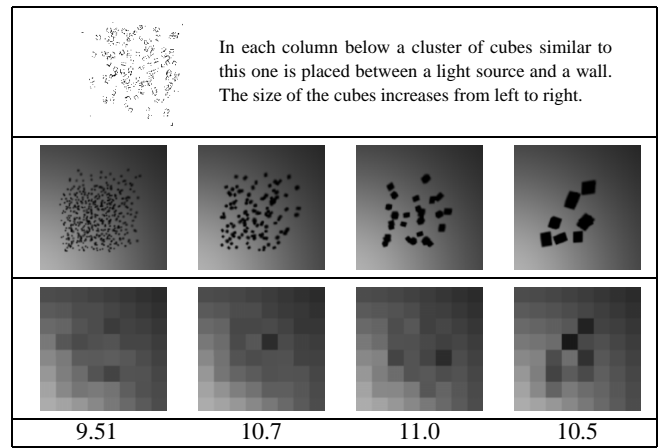


Figure 1: Comparison of approximate illumination solutions using different clusters. Top: reference images (illumination of the wall). Middle: approximate images using a coarse mesh. Bottom: L^2 error norms. Note that the four images have similar L^2 error values, and all hide some illumination information. However the varying size of the missing features cannot be discovered.

Figure 1 illustrates the issue by showing shadows cast on a wall by four different groups of objects. Four approximate images, all computed using the same mesh size, are shown below the “exact” images. Consider a user who is interested in shadows of a specific size, e.g. those of the image on the extreme right, but is satisfied by the averaging of the smaller, detailed shadows on the left¹. The user thus does not wish more work to be done for the detail shadows, but wishes to have a more accurate representation at the larger scale. The subdivision criterion used in a HR algorithm for instance should be capable of halting the subdivision for the left-hand group, while ordering further computation for the group on the right. Thus an error measure should distinguish between the four cases.

Traditional error metrics are incapable of making such a distinction. As an example consider the commonly used family of error metrics expressing the distance between a reference function f and

¹Perhaps a more realistic example would be a situation where a user is viewing an office scene from the doorway, and in which accurate shadows for chairs and desks are important, but averaged, low quality shadows from details such as pens on a desk are satisfactory.

an approximate function \hat{f} as the L^p norm

$$\|\hat{f} - f\|_p = \left(\int |\hat{f}(x) - f(x)|^p dx \right)^{\frac{1}{p}}$$

L^p norms simply add error contributions from all points on a surface (or in an image), and do not take into account higher-level properties of the radiance distributions, such as the size and shape of illumination features. This is illustrated by the similar values obtained for the four groups in Figure 1. Appendix A shows that in fact for a point light source the L^1 or L^2 error introduced by averaging all visibility variations depends only on the average visibility, and not on the size or shape of the shadows.

2.2 A proposal for an error metric based on feature size

Our hypothesis is that illumination features (shadows or bright areas) are important only as far as they have a significant visual impact. Therefore it is possible to define a *feature size* on a receiving surface, and decide that features smaller than that size are “unimportant”: their absence should not contribute to the error.

In the remainder of the paper we refer to the radiosity function over a surface as an “image”. This terminology should not mask the important fact that the entire discussion takes place in three-dimensional object space. In order to demonstrate the relevance of the feature-based approach, we assume for now that we have access to all the information in a reference solution. The multi-resolution visibility technique of Section 3 will show how the ideas developed here can still be used in the absence of such a reference.

A simple error metric based on features is defined by segmenting the image f into two components by means of a *feature mask* $\mathcal{F}^s(f, x)$: a binary function that equals one at points x that belong to a “feature” (of size greater than s) of function f . Computation of feature masks from the reference solution is described in the next section. For points in the mask region we compute an L^p norm of the difference between the approximate function and the reference function. For points outside the feature mask, we are content with an average value (since features present there are smaller than s). Thus in our current implementation we compute average values at each point, for both the approximate and reference functions, using a box filter of size s around the point of interest, and compute an L^p norm of the difference between the averages.

The feature-based error metric (FBEM) is summarized by the following formula, where \bar{f}^s represents the filtered version of f :

$$\begin{aligned} \|\hat{f} - f\|_p^s &= \left(\int |\bar{f}^s(x) - \bar{f}(x)|^p [1 - \mathcal{F}^s(f, x)] dx \right. \\ &\quad \left. + \int |\hat{f}(x) - f(x)|^p \mathcal{F}^s(f, x) dx \right)^{\frac{1}{p}} \end{aligned} \quad (1)$$

2.3 Examples

Table 1 shows the FBEM values computed for the four groups of Figure 1 and different values of the minimum feature size s . The object-space size of typical shadows in these images is respectively 11, 16.5, 22 and 31. For small s values, all FBEM values are high since the metric is equivalent to an L^2 metric in the limit of $s = 0$. As s increases, FBEM values decrease more rapidly for the groups containing smaller objects, as expected. There appears to be a residual error of about 3 due to the mesh size used for the approximate solutions.

Assume the user is interested in clearly seeing features of size 30 or greater, while being content with an average for all features smaller than this size. The extreme right-hand image of Figure 1





Feature size:				
5	14.76	16.34	17.25	17.31
16	9.37	12.24	15.76	15.80
24	4.78	6.50	9.06	14.74
30	4.23	3.16	6.90	13.37
40	3.65	2.33	3.35	6.94

Table 1: Feature-based error metric (FBEM) for the four approximate images of Figure 1 and five different feature sizes. The four measures are equivalent for small feature sizes, and decrease at different rates as a function of s . Images are shown again for clarity.

requires more work since the FBEM value for $s = 30$ is high. The approximation for the other three images is deemed satisfactory since the error is low.

Thus, using the FBEM presented above, it is possible to reveal the presence of features greater than a given threshold in the approximate images, opening the way for selective subdivision based on the user’s minimum feature size of interest. Of course this could not be used *as is* in a subdivision criterion for HR, since it uses a reference solution, but it is useful for a *a posteriori* validation of control mechanisms.

2.4 Computation of feature masks

According to the definition of features given above, computing a feature mask amounts to identifying connected regions of “significant” size. Mathematical morphology provides tools to isolate features based on their size [12]. Consider a binary image, representing for example the characteristic function of an object. We define the action of an *Erosion* operator as follows: all points outside the object (white) are untouched. All points inside the object that have a neighbor outside become white. All other points remain black. An *Expansion* operator is defined similarly by including in the objects all outside points that have a neighbor in the object. Figure 2 shows a reference image and images obtained after a number of erosions (top) or expansions (middle).

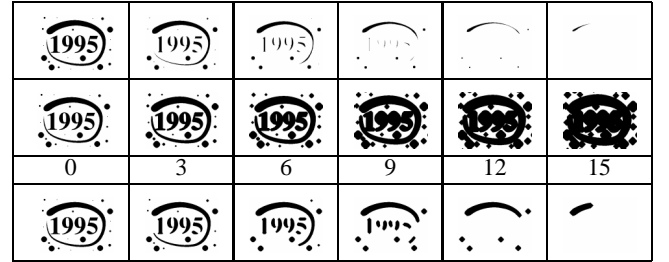


Figure 2: Effect of repeated applications of the erosion (top), expansion (middle) and combined erosions/expansion (bottom) operations on a binary image. The reference image appears in the left column, and the number of applications of the operators increases from left to right. For the bottom row we apply n erosions followed by n expansions.

Clearly an object of diameter $2d$ will disappear after d erosions are applied in sequence. Thus applying a sequence of n erosions followed by n expansions will successfully eliminate all small regions, but keep larger regions (slightly modifying their shape in the process). This process is illustrated in the bottom row of Figure 2.

Computing the effect of the erosion operator on a binary image is straightforward using bitwise operations: the result is the logical

OR of the image and the four translated copies of itself (by one pixel) in the $+x$, $-x$, $+y$ and $-y$ directions. For the expansion operator the logical operator AND is used.

In our examples, the original binary image is computed by recording all areas of the receiver that have a partial or occluded view of the light source. This expensive operation was performed only once during the creation of the reference image. Feature masks are computed by applying the proper number ($p/2$ for a feature size of p) of successive erosions and expansions to eliminate unwanted features. Figure 3 shows some feature masks for the four groups used above.

Original mask	F Size 12	F Size 18	F Size 24

Figure 3: Some feature masks for the images in Figure 1.

3 A Multi-resolution Visibility Algorithm

In the previous section we presented the concept of a *feature size* and introduced an error metric which permits the evaluation of image quality determined by how well illumination features are represented. We now use these fundamental concepts to develop a *multi-resolution* (MR) visibility algorithm. With this algorithm, expensive high quality visibility calculations are only performed when they are expected to help in the accurate representation of features deemed “interesting” by the user.

Hierarchical spatial subdivision structures are often used in the calculation of global illumination algorithms, in particular when form-factor estimation is performed with ray-tracing [19, 4, 5, etc.]. In radiosity clustering algorithms the hierarchy of clusters is also used for radiometric calculations, by letting clusters represent their contents for some energy transfers [13, 16]. The following *multi-resolution visibility* algorithm naturally extends previous clustering approaches by allowing clusters to also represent their contents in some visibility calculations. If a specific feature size s has been chosen, it is unnecessary to consider the contents of a cluster for visibility if these contents will produce features smaller than s .

3.1 Approximate visibility computation between clusters using an extinction model

Let us assume that we have grouped all objects in the scene into a hierarchy of clusters. Approximate visibility calculations can be performed using an analogy between clusters and absorbing volumes [13]. The approximation (asymptotically exact for homogeneous isotropic clusters when the size of the objects goes to zero)

consists of associating an *extinction coefficient* κ with each cluster. The transmittance function between two points P and Q in the scene is then given by

$$T(P, Q) = e^{-\int_{PQ} \kappa(u) du} = e^{-\sum_{i \in \mathcal{C}(PQ)} \kappa_i l_i}$$

where $\mathcal{C}(PQ)$ is the set of clusters traversed by the ray joining P and Q , κ_i is the extinction coefficient of cluster i , and l_i is the length traveled inside cluster i by the ray.

Extinction coefficients express the probability that a random ray is intercepted in the cluster, and are computed as

$$\kappa_i = \frac{\sum_j A_j}{4V_i}$$

where the area of all surface patches contained in cluster i is summed and divided by the cluster’s volume. Since a surface contributes to the extinction of only one cluster, the attenuation due to overlapping clusters is correctly obtained by adding their extinction contributions.

3.2 Multi-Resolution Visibility

In the rest of this section we consider the emitter-blocker-receiver configuration shown in Figure 4, which consists of two surfaces, the emitter E and the receiver R , in two-dimensions. This restriction is for presentation purposes only and is removed later.

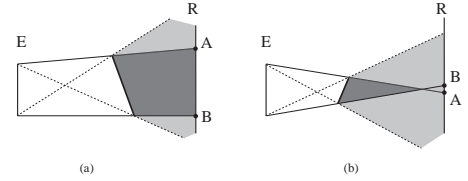


Figure 4: Definition of shadow features created by a blocker. (a) The umbra region is unbounded since the blocker is larger than the emitter: there is always an umbral region on the receiver. (b) For some positions of the blocker the receiver has no umbral region.

If a blocker (which for now we also consider to be a surface) is placed between the emitter and the receiver, *umbra* and *penumbra* regions are created in space. Depending on the position of the blocker, there may or may not be an umbral region on the receiver. (Figure 4). Given the definition discussed above the size of the umbral zone on the receiver $-AB$ in Figure 4(a)–, if it exists, is the *feature size*.

The blocker may actually be a hierarchical representation of a collection of objects (a *cluster*) as pictured in Figure 5(a). In this case, at each level of the hierarchy an extinction coefficient is stored allowing the approximate calculation of the attenuation of a ray if it passes through the cluster, as described previously.

Multi-resolution visibility can be performed by avoiding the descent into the hierarchy after a certain level. When the required conditions are met the extinction coefficient is used instead, thus avoiding the intersection of the ray with all the descendants of this cluster. Evidently, the effect is that visibility is no longer exact, but an average estimation of transmittance. It is here that a large potential gain in computation time can be achieved. In scenes where the small detail objects (e.g., models of phones, keyboards, small objects on a desk etc.), comprise the largest part of the geometric

complexity, the intersection with these objects can quickly become the overwhelming expense of visibility (and overall) computation. By considering the higher level clusters for visibility computation instead of the numerous contents, when such a choice is dictated by the chosen feature size, this expense can be completely avoided.

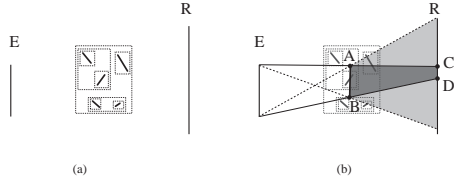


Figure 5: Visibility estimation through a cluster. (a) the blocker is a hierarchy of clusters. (b) an “equivalent blocker” is used to estimate the maximum feature size on the receiver.

Recall the discussion in Section 2 in which the user wishes to accurately represent all features of size greater than s on the receiver. To achieve this, all that is required is to descend sufficiently far into the hierarchy so that the large shadows are accurately calculated, while performing the approximate calculation for small, detail shadows.

To facilitate such a choice each cluster is augmented with a description of the maximum blocker size $BSIZE$ of its contents (we give a precise definition of this in the following section). It then suffices to place a fictitious blocker of size $BSIZE$, at the center of the actual cluster CD in Figure 5(b). The descent in the cluster hierarchy can be terminated if the projected umbral region of the fictitious blocker (AB in Figure 5) is smaller than the chosen feature size s .

Contiguous regions which let light traverse must also be considered as feature creators since a feature can be considered “negative” (umbra in a bright region), or “positive” (lit areas inside a dark region). We thus extend our definition of features from Section 2 by defining $BSIZE$ to be the maximum of the connected regions of light or shadow. This is consistent with the symmetric expression of visibility error with respect to umbra and light presented in Appendix A.

3.3 Characterization of a Cluster for MR Visibility

All that is required in order to apply the preceding algorithm is the determination of $BSIZE$ for each cluster. The restriction to two-dimensions is now lifted, and the treatment for three-dimensional clusters is described. For now clusters are assumed to contain objects placed so that the cluster density can be considered isotropic, and thus does not depend on the direction of incidence of a ray.

The goal is to determine a representative size for a blocking cluster, which will allow the calculation of the maximum feature size given a specific emitter-receiver configuration. At first glance it may seem natural to take $BSIZE$ to be the size of the largest object contained in the cluster. However there is one important consideration: it is the *connected region* of shadow on the receiver which we wish to consider. Furthermore, as discussed above, the regions of light potentially blocked by the contents of the cluster *and* the regions of light which pass through must be considered separately.

A preprocessing step is performed to calculate $BSIZE$ for all clusters in the hierarchy. For each cluster, all the contained objects are orthographically projected into a binary image. This operation is performed for a given cluster and a given direction, resulting in a *view-independent* characterization. The consequence of the isotropic cluster assumption is that a single orthographic projection is sufficient. For non-isotropic clusters the $BSIZE$ parameter is a function of the direction of interest. A simple solution in that case would be to interpolate from a number of sampled directions. Our

current research focuses on more efficient representations for such directional information [14].

The erosion and expansion operators from Section 2.4 are then used to compute the maximum sizes for blockers and free regions inside a cluster. Erosions (respectively expansions) are computed until all objects have disappeared (respectively until all free space has disappeared). The *number* of erosion or expansion operations defines the value of $BSIZE$ for the blocked and free regions respectively. In our implementation we do the projections, erosions and expansions using Graphics hardware.

4 A Hierarchical Structure for Clustering and Multi-Resolution Visibility

Previous automatic clustering approaches have used spatial data structures developed for ray-tracing (hierarchical bounding boxes [3] were used in [16], while in [13] a K-D tree was used). In this section we show that given the calculation of average visibility based on extinction coefficients in the manner of [13], it is beneficial to develop a special-purpose hierarchical data structure, such that the resulting clusters have properties suitable for cluster-based hierarchical radiosity and multi-resolution visibility.

By definition, clusters are constructed to *represent* as accurately as possible the collection of objects they contain. By introducing computation of visibility using extinction coefficients and also multi-resolution visibility, apart from the representation of energy transfer of the contained objects as a whole, the clusters also need to correctly represent the transmission properties of the collection of contained objects.

These two modes of representation place different constraints on the cluster hierarchy. From the point of view of energy exchanges, good clusters allow tight bracketing of radiance or visibility functions (thus surfaces with similar orientation that do not shadow each other are preferred). From the point of view of visibility approximation, good clusters are ones for which the extinction property is plausible (thus homogeneous isotropic clusters are preferred). Given these constraints, we have identified two key properties for clusters: (a) *proximity* and (b) *homogeneity* of the contained objects. Maintaining proximity is a natural way to group objects when the cluster is used to represent radiative transfers. Also, for multi-resolution computation it is important that objects contained in a cluster are close so that the averaging performed does not introduce unacceptable artifacts. Homogeneity here means that we want a cluster to group objects of similar size, and is crucial for the resulting quality of the average visibility computation.

As a simple measure of proximity, we use the percentage of empty space resulting from a clustering operation (i.e., the addition of an object or a cluster to another cluster). Thus we prefer clusters in which the empty space is minimized.

To efficiently group objects of similar size we use a hierarchy of n levels of uniform grids. We start with level 0, which is a single voxel the size of the bounding box of the scene and then at each level i we create a grid which is subdivided into 2^i voxels along each axis. We then insert each object into the level for which its bounding box fits the voxel size.

Once these grids have been constructed, we start at the lowest level n , containing the smallest objects. We group the objects entirely contained in each voxel, by attempting to minimize the empty space, in accordance to the proximity criterion described above. In addition, objects which are very small compared to the grid size are grouped into an appropriate cluster, even if the resulting cluster is largely empty. Once all the voxels of a level have been treated, we attempt to add the objects not entirely contained in a single voxel at this level to the clusters already constructed, again using the same criteria. We then insert the clusters created to the grid of the level

immediately above, and iterate.

Once the cluster hierarchy has been created, the data structure is augmented with average transmission behavior by propagating the average extinction values up the hierarchical structure as in [13]. When multi-resolution visibility is used, the *BSIZE* estimation is also performed for each cluster in the hierarchy in the manner described in Section 3.

Figure 6 presents results obtained with the new hierarchy using first a surface visibility algorithm similar to that of [16], and then the average visibility proposed in [13]. The scene consists of 5380 polygons. It is interesting to observe the significant time gain achievable by the average visibility algorithm given a suitable hierarchy (we observe a factor of 4), while approximate shadows are preserved.



Surface vis: 1 216 s.

Volume vis: 376 s.

Figure 6: Timings (in seconds) using the new hierarchy construction. Throughout the paper all timing information was obtained on an Indigo R4000 computer.

Constructing a suitable hierarchy for cluster-based hierarchical radiosity with extinction and multi-resolution visibility is a difficult problem. The results indicate that the first solution presented here, based on proximity and homogeneity, results in the construction of hierarchies well suited to approximate and multi-resolution visibility calculations.

5 Results of multi-resolution visibility

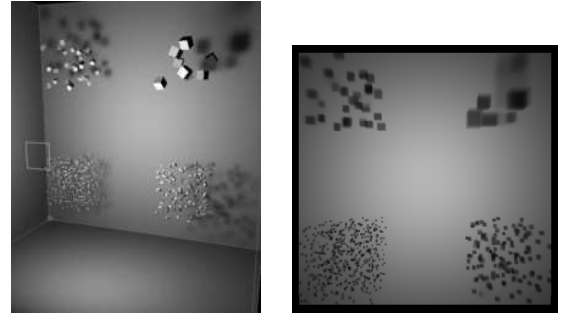
We have implemented the hierarchy construction, the calculation of *BSIZE* and the multi-resolution visibility algorithm in a hierarchical radiosity clustering testbed.

To evaluate the results of the multi-resolution visibility approach we have computed images of test environments using different values for the feature sizes of interest s on a receiver.

The first test scene is shown in Figure 7 (left). It contains the four clusters used in Section 2 and a light source (in yellow). The right-hand image is the illumination obtained on the back wall and serves as a reference image. For all these images visibility was always computed solely using extinction properties (thus we do not attempt to characterize the error introduced by averaged transmission visibility itself).

Figure 8 shows four images, where the desired feature size parameter (see Section 2.2) is changed. For each image the computation time in seconds is given. A very low error threshold was used to ensure that the mesh density was maximal for all images. Thus the decrease in computation time as the desired feature size becomes larger measures the speedup in the visibility calculation.

We next show that multi-resolution visibility is consistent with the feature-based error metric (FBEM) from Section 2.2, by computing the FBEM for the images described above. Although the four clusters have been grouped in a single image for simplicity, we



3D view of test scene.

Reference sol. (2069 s).

Figure 7: Reference image used in the error comparisons.

apply the error metric only on the region of the image corresponding to each cluster, to obtain an FBEM value for each of the four groups.

For the four images, we show for each cluster the value of L^2 error norm (back row) and the value of the FBEM for a feature size s equal to that used in the MR Visibility algorithm. We note that as we increase s the L^2 norm for all clusters increases, as more and more averaging is being performed. Still the increase appears later for larger objects, as expected. The FBEM values are always of similar magnitude, despite the fact that very different levels of averaging are being used in different clusters in a given image. This shows that the multi-resolution visibility algorithm accomplishes its purpose: given a desired feature size, it ensures that the corresponding FBEM remains low while allowing time gains.

Figure 9 shows that even greater speedups can be achieved when a medium error threshold allows MR visibility to reduce the amount of subdivision. The explicit incorporation of MR visibility in refinement criteria is a promising path for further acceleration.

6 Control of Image Quality for Clustering

Recent algorithms separate the computation of high-quality images into two phases: a coarse quality global illumination calculation is first performed using elaborate algorithms such as discontinuity meshing or clustering in a *global pass*. A view-dependent, potentially very expensive *local pass* follows [9, 16]. This local pass is typically a ray-casting operation: at each pixel the energy from all the links is collected, allowing the calculation of high-quality shadows. The cost of this local pass is often many times larger than that of the light-transfer calculation using clusters. In essence this pass may eradicate all computation time benefit achieved by using the clusters in the first place, and exclude any possibility for interactivity with quality and error control.

In contrast, we maintain a “progressive refinement” philosophy, by providing explicit quality controls, allowing computational cost to be focused on desired characteristics of the resulting image. The first component of this approach is the multi-resolution visibility presented above. This technique, coupled with the use of *importance* [17] to assign appropriate feature sizes to different objects could plausibly replace the global/local pass approach while affording more interactivity. We present next two supplementary quality controls: first, the correct treatment of intra-cluster visibility and second, progressive multi-gridding permitting rapid interactive response for hierarchical radiosity.

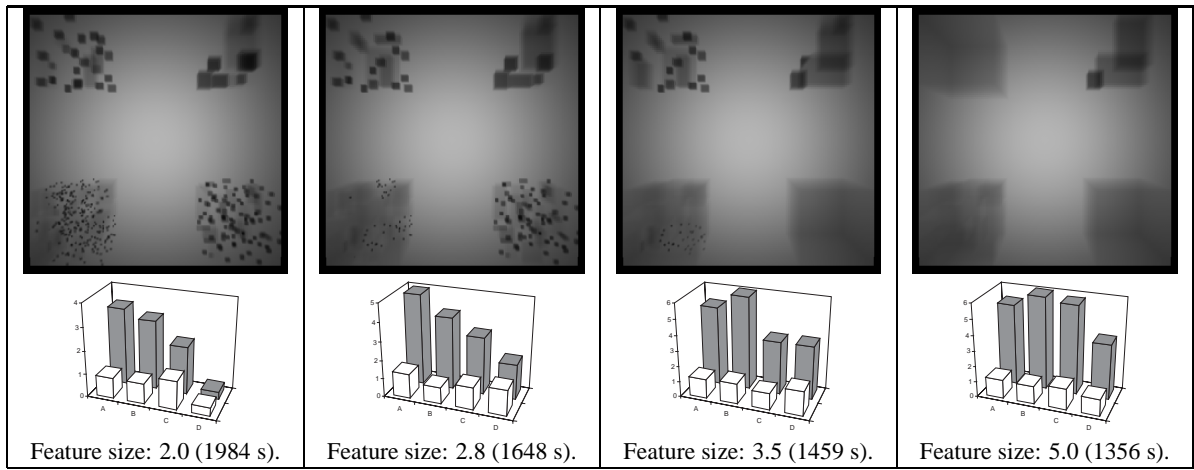


Figure 8: Results for the multiresolution visibility algorithm.

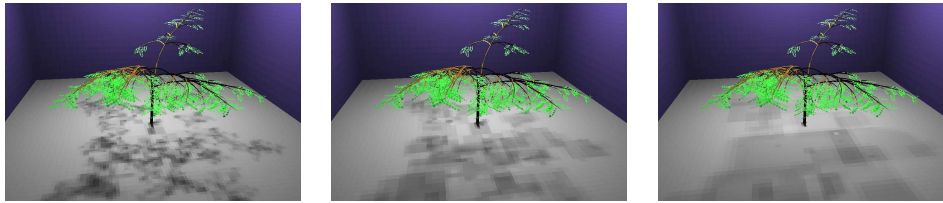


Figure 9: Increasing the desired feature size reduces both the amount of subdivision and the cost of visibility computations. (left) Fsize = 0, 621s. (middle) Fsize = 4, 245s. (right) Fsize = 8, 148s. Tree courtesy of CIRAD, modelled with 7967 polygons using AMAP.

6.1 Intra-cluster visibility

Previous clustering algorithms compute a bound on energy transfer that ignores visibility (bound of 1 on the visibility error), both between the two clusters but also in the distribution of light on each side [16, 13]. This potentially results in light leaks at the scale of the cluster. This behavior is not only visually displeasing but also flawed: since bounds are computed on irradiance values, that irradiance is distributed to many surfaces which should be shadowed, thereby creating energy.

If visibility information inside the cluster with respect to a source cluster can be computed (with some approximation) in time linear in the number of contained objects, the overall time and space complexity of $O(s \log s)$ for clustering is not modified [16].

We propose the use of an item buffer to quickly evaluate this visibility. The cluster's contents are projected in the direction of the light source using a z-buffer to determine visible surfaces from that direction. By counting instances of an item number in the buffer we obtain an estimate of the projected area of each patch visible from the direction of the source. This is used as the projected area in kernel calculations when computing energy bounds. Note that the resolution of the item-buffer can be adapted to the contents of each cluster, provided we know the size of the smallest object in each cluster. Thus the aliasing problems inherent to the item-buffer approach can be reduced. The same technique is also used at the other end of a link, to evaluate the energy leaving a cluster.

In the images of Figure 10 we present an example where a link (shown in purple) has been created from the light source to a cluster of books. Ignoring intra-cluster visibility (left) results in the creation of energy since all books are fully illuminated. Using the visibility buffer to modulate the energy distribution (right), energy is preserved while improving the appearance of the image.

6.2 Progressive multi-gridding

In hierarchical radiosity algorithms subdivision is controlled by an error threshold on individual interactions. A global bound on er-

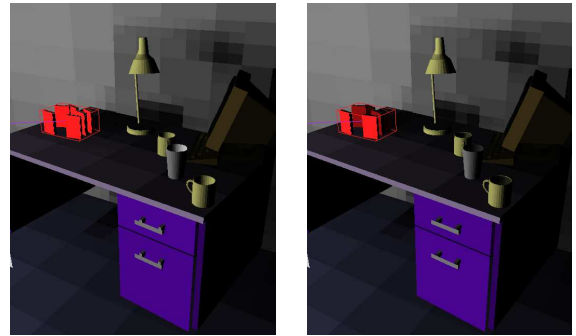


Figure 10: Results of introducing intra-cluster visibility.

ror is difficult to determine and it is consequently difficult for the user to choose an error threshold so as to achieve a certain quality. The problem is exacerbated with clustering, since the subdivision of links is amortized with time, and thus successive iterations may become much more expensive as the allowed error decreases. This sharp and unpredictable increase in iteration time may then destroy interactivity.

As a remedy we develop a *progressive multi-gridding* approach. By analyzing the distribution of error bounds on the links created, we can predict how many of these links would survive a given decrease in the error threshold, and thus estimate the time required for a subsequent iteration with the new error threshold. In a manner more intuitive to the user the amount of computation can be specified (in the form of a maximum number of links to refine) and the system proceeds to deduce the new threshold to use for hierarchical refinement.

This analysis can be performed at a marginal cost by recording a histogram of the distribution of error bounds computed. Specifically, the interval between 0 and the current error threshold ε is divided into a number of bins, each associated with a counter. Every time a link is created (or left unchanged) during the hierarchical

subdivision procedure, we increment the counter for the bin corresponding to the link's error bound. At the start of the next progressive multi-gridding iteration, the new error threshold is chosen such that the sum of all counter for bins with higher error levels is less than a user-specified limit k . This effectively chooses an error threshold such that at most k links are refined. This multi-gridding algorithm does not accelerate the computation but guarantees a continuous update of the simulation.

7 Conclusions

Important advances towards the goal of providing interactive systems capable of treating very complex environments have been made by hierarchical radiosity and clustering algorithms. Nonetheless several important shortcomings of previous approaches were identified in this paper: (a) visibility error is typically ignored, (b) traditional error metrics do not allow the user to specify a desired level of detail and (c) progressive refinement of the simulation is difficult to achieve.

In this paper we introduced a new approach to error estimation based on illumination *features*, which allows the user to choose a level of detail relevant to a given simulation. The quality of a solution then relates to how well features of the user-determined size have been represented.

The principles introduced by the feature-based analysis were used to develop a *multi-resolution visibility algorithm*. The hierarchy constructed for clustering contains transmission information as in [13] and is further augmented with an estimate of the largest equivalent blocker size from its contents. This information is used to limit the cost of visibility calculations. An algorithm which efficiently constructs a suitable hierarchy was also presented. The results of the implementation for isotropic environments show significant computational speedup using MR visibility when the user does not require the accurate representation of small features.

Two additional quality control mechanisms were introduced: *intra-cluster visibility* which corrects potential light-transfer error suffered by previous clustering algorithms, and *progressive multi-gridding* which is essential for interactive clustering systems.

We believe that the introduction of feature-based error and quality evaluation is an important step which will lead to significant acceleration of global illumination algorithms. Multi-resolution visibility is an example of such an achievement.

In future work the extension of our approach to non-isotropic environments must be completely developed. Promising first results in representing directional information for clustering have been obtained [14]. We have not yet addressed the analysis of error caused by the use of extinction coefficients and the effect of visibility correlations between clusters and their contents. Research in these areas is extremely important for the development of reliable quality controls. It will be interesting to observe the results of the application of our approach to Monte Carlo methods. A more in-depth study of feature-based error metrics must be performed. Finally better algorithms for hierarchy construction should be investigated.

8 Acknowledgements

George Drettakis was hosted successively by INRIA (Grenoble, France), UPC (Barcelona, Spain) and GMD (St. Augustin, Germany) as an ERCIM postdoctoral fellow, and was financed in part by the Commission of the European Communities. Pat Hanrahan provided parts of the hierarchical radiosity code; Jean-Daniel Boissonnat suggested the use of erosion/expansion operations. The scene in Figure 6 was assembled using pieces of the Berkeley Soda Hall model; thanks to Seth Teller and all members of the UC Berkeley walkthrough group.

References

- [1] James Arvo, Kenneth Torrance, and Brian Smits. A framework for the analysis of error in global illumination algorithms. In *Computer Graphics Proceedings, Annual Conference Series: SIGGRAPH '94* (Orlando, FL), pages 75–84, 1994.
- [2] Michael F. Cohen and John R. Wallace. *Radiosity and Realistic Image Synthesis*. Academic Press, Boston, 1993.
- [3] J. Goldsmith and J. Salmon. Automatic creation of object hierarchies for ray tracing. *IEEE Computer Graphics and Applications*, 7(5):14–20, May 1987.
- [4] Eric A. Haines. Shaft culling for efficient ray-traced radiosity. In P. Brunet and F.W. Jansen, editors, *Photorealistic Rendering in Computer Graphics*, pages 122–138. Springer Verlag, 1993. Proceedings of the Second Eurographics Workshop on Rendering (Barcelona, Spain, May 1991).
- [5] Pat Hanrahan, David Saltzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. *Computer Graphics*, 25(4):197–206, August 1991. Proceedings SIGGRAPH '91 in Las Vegas.
- [6] Nicolas Holzschuch, François Sillion, and George Drettakis. An efficient progressive refinement strategy for hierarchical radiosity. In *Fifth Eurographics Workshop on Rendering*, Darmstadt, Germany, June 1994.
- [7] Arjan J. F. Kok. Grouping of patches in progressive radiosity. In *Proceedings of Fourth Eurographics Workshop on Rendering*, pages 221–231. Eurographics, June 1993. Technical Report EG 93 RW.
- [8] Dani Lischinski, Brian Smits, and Donald P. Greenberg. Bounds and error estimates for radiosity. In *Computer Graphics Proceedings, Annual Conference Series: SIGGRAPH '94* (Orlando, FL), pages 67–74, July 1994.
- [9] Dani Lischinski, Filippo Tampieri, and Donald P. Greenberg. Discontinuity meshing for accurate radiosity. *IEEE Computer Graphics and Applications*, 12(6):25–39, November 1992.
- [10] Dani Lischinski, Filippo Tampieri, and Donald P. Greenberg. Combining hierarchical radiosity and discontinuity meshing. In *Computer Graphics Proceedings, Annual Conference Series: SIGGRAPH '93* (Anaheim, CA), pages 199–208, August 1993.
- [11] Holly Rushmeier, Charles Patterson, and Aravindan Veerasamy. Geometric simplification for indirect illumination calculations. In *Proceedings Graphics Interface '93*. Morgan Kaufmann, 1993.
- [12] J. Serra. *Image analysis and mathematical morphology : 1*. Academic Press, London, 1982.
- [13] François Sillion. Clustering and volume scattering for hierarchical radiosity calculations. In *Fifth Eurographics Workshop on Rendering*, Darmstadt, Germany, June 1994.
- [14] François Sillion, George Drettakis, and Cyril Soler. A clustering algorithm for radiance calculation in general environments. In *Sixth Eurographics Workshop on Rendering*, Dublin, Ireland, June 1995.
- [15] François Sillion and Claude Puech. *Radiosity and Global Illumination*. Morgan Kaufmann publishers, San Francisco, 1994.
- [16] Brian Smits, James Arvo, and Donald P. Greenberg. A clustering algorithm for radiosity in complex environments. In *Computer Graphics Proceedings, Annual Conference Series: SIGGRAPH '94* (Orlando, FL), pages 435–442, July 1994.
- [17] Brian E. Smits, James R. Arvo, and David H. Salesin. An importance-driven radiosity algorithm. *Computer Graphics*, 26(4):273–282, July 1992. Proceedings of SIGGRAPH '92 in Chicago.
- [18] Seth J. Teller and Patrick M. Hanrahan. Global visibility algorithms for illumination computations. In *Computer Graphics Proceedings, Annual Conference Series: SIGGRAPH '93* (Anaheim, CA), pages 239–246, August 1993.
- [19] John R. Wallace, Kells A. Elmquist, and Eric A. Haines. A ray tracing algorithm for progressive radiosity. *Computer Graphics*, 23(3):315–324, July 1989. Proceedings SIGGRAPH '89 in Boston.

A Visibility error using L^1 and L^2 norms

Consider a patch P illuminated by a point source at point y . To quantify the visibility error on the receiver patch, we compute the L^1 and L^2 norms of the difference between the visibility function $v(x, y)$ defined for $x \in P$, and its average value \bar{v} over patch P . If P^+ is the region of patch P that receives light, \bar{v} is equal to the ratio of the areas of P^+ and P . Separating the integrals into one over P^+ and one over $(P - P^+)$, we find

$$\|v - \bar{v}\|_1 = 2\bar{v}(1 - \bar{v}) \quad (2)$$

and similarly for the L^2 norm

$$\|v - \bar{v}\|_2 = \sqrt{\bar{v}(1 - \bar{v})} \quad (3)$$

Note that both estimates only depend on the average visibility across patch P , not on the distribution of the visibility function. Also note the dependency in $\bar{v}(1 - \bar{v})$, yielding a small error for either almost complete occlusion or almost complete visibility.

3.5.5 Controlling Memory Consumption of Hierarchical Radiosity with Clustering (GI'99)

Auteurs : Xavier Granier et George Drettakis

Actes : Congrès Graphics Interface '99

Date : juin 1999

Controlling Memory Consumption of Hierarchical Radiosity with Clustering

Xavier Granier

George Drettakis

iMAGIS-GRAVIR/IMAG-INRIA, BP 53, F-38041 Grenoble, France.

Email: {Xavier.Granier | George.Drettakis}@imag.fr

iMAGIS is a joint project of CNRS/INRIA/UJF/INPG

Abstract

Memory consumption is a major limitation of current hierarchical radiosity algorithms, including those using clustering. To overcome this drawback we present a new algorithm which reduces the storage required for both the hierarchy of subdivided elements and the links representing light transfers. Our algorithm is based on a link hierarchy, combined with a progressive shooting algorithm. Links are thus stored only when they might transfer energy at subsequent iterations. The push-pull and refine/gather steps of hierarchical radiosity are then combined, allowing the simplification of subtrees of the element hierarchy during refinement. Subdivided polygons replaced by textures and groups of input objects contained in clusters may be deleted. A memory control strategy is then used, forcing links to be established higher in the link hierarchy, limiting the overall memory used. Results of our implementation show significant reduction in memory required for a simulation, without much loss of accuracy or visual quality.

Key words: global illumination, hierarchical radiosity with clustering, memory consumption.

1 Introduction

Global illumination algorithms have made great progress in recent years. With the use of hierarchical radiosity (HR) [8] and clustering [14, 12], global illumination can be simulated for large models, resulting in solutions which are appropriate for real-time walkthroughs. These solutions are typically used for interior design, television and entertainment (virtual sets etc.), and “digital mock-ups”. The widespread use of radiosity solutions results in the need to simulate large environments; scenes of millions of initial (input) polygons are now common, and present a challenge to existing radiosity algorithms.

Despite the impressive advances in computation time and control of the simulation precision, hierarchical radiosity methods still require large amounts of memory. Hierarchical radiosity with clustering requires the use of memory-intensive data-structures: *hierarchical elements*

which contain numerous fields and the *links* which represent the light transfers at different levels of precision. An initial polygonal model which fits in main memory before the simulation may require several times more memory for the global illumination solution, making such a solution infeasible.

Despite important efforts to reduce memory used by global illumination solutions [18, 5, 17], no overall framework has been proposed which allows the control of memory usage for both the element hierarchy and the links, and which maintains the advantage of the representation of global light transfer.

The new approach we present here first develops a novel framework, based on the line-space, or link, hierarchy [4], and a progressive shooting approach. By modifying the link refine, light gather and push-pull steps of hierarchical radiosity, we reduce both the memory used by the links, and the memory of the *element hierarchy* itself. Subdivision on surfaces is replaced by textures, and initial polygons contained in clusters can be removed in a view-dependent manner. A memory control mechanism is applied to the link hierarchy during the modified refine/gather/push-pull: based on user defined memory limits, links are established at higher levels in the link hierarchy, reducing the storage due to links, and permitting the removal of subdivided elements on polygons.

2 Previous Work

The treatment of complex scenes has been a challenge for global illumination since its outset. Progressive refinement radiosity algorithms [3] have produced simulations of complex environments [1]. The main drawback of progressive refinement is the lack of control of global error, since it is impossible to determine whether the global illumination has been properly accounted for.

Hierarchical radiosity [8] is an efficient algorithm which performs light transfers at appropriate levels of accuracy, by subdividing the environment into hierarchical elements and establishing light-transfer *links* between the elements at the appropriate level. The n^2 “initial link-

ing” step (n is the number of polygons in the scene) renders the approach impractical for large scenes. For certain environments (especially building interiors), partitioning schemes based on visibility can allow the treatment of very large databases [18]. The advent of clustering [14, 12, 7] made the treatment of large databases possible in the general case. In particular the largest model treated by a radiosity algorithm to date (to the authors knowledge) is in the order of 258,000 input polygons using hierarchical radiosity with clustering, coupled with a parallel and distributed system [5]. A completely different approach involves the use of stochastic methods. One recent example is [2], which also includes a thorough bibliography. Due to the fundamental differences compared to finite-element approaches, we do not consider this direction further.

The storage of links for hierarchical radiosity is required since at each iteration light is transported across *all* links. Typically, irradiance on all elements is set to zero, and a gather step follows using the radiosity on each element. Irradiance is thus gathered at all levels of the hierarchy from *all* links, and then pushed to the leaves. Radiosity is then pulled up the hierarchy. Willmott and Heckbert [19] observed that wavelet radiosity (which is hierarchical radiosity possibly with higher-order basis functions) uses a prohibitively large amount of memory, especially for the storage of links.

As a response to this observation, Stamminger et al. [17] developed an approach to “get rid of links”. They use a “shooting” approach, which stores an additional “unshot radiosity” variable at each hierarchy element, in the spirit of progressive refinement. The storage of all links is thus no longer required once unshot radiosity is transported across it. A fixed-size cache of links is created, and links are inserted in a sorted manner into the cache based on the energy they transport. If a link is not in the cache, it is recomputed, possibly leading to expensive recursive refinement. In addition, the global representation of all light exchanges is lost. Complex secondary interactions may thus result in significant additional refinement operations.

3 Motivation and Overview

By running “standard” hierarchical radiosity, we have observed that the memory used by the element hierarchy is often on a par or more than that required by the link data structures. This is particularly true for cases in which the link refinement ϵ threshold value is high, and thus many links remain at the level of clusters. In general, the storage of links vs. hierarchy elements is heavily dependent on scene type and the values of the various simulation parameters.

As a consequence, we present a new formulation allowing the reduction of memory for both links and the element hierarchy. The reduction of memory used by links is achieved while maintaining a coherent representation of overall light transfer (in Section 4). This representation, achieved through the line-space hierarchy, enables the introduction of an algorithm which allows us to replace entire subtrees of the element hierarchy during subdivision (Section 5). This allows the replacement of subdivided elements by textures and of entire groups of clustered input elements by a simpler representation. The memory control mechanism is then described (Section 6), which enables our approach to significantly reduce memory used by links and subdivided elements by forcing links to be established high in the link hierarchy. Implementation issues and results are presented in Section 7, and we conclude.

4 Controlling Memory used for Links

As mentioned above (Section 2), links in hierarchical radiosity need to be stored since they are used at each iteration for the gather step. If we use “unshot radiosity”, there is no longer a need for all these links. Intuitively, links from the sources, which are often numerous, do not need to be stored. Once the energy has been transferred from the light sources to the receiver surfaces, their utility ends. For subsequent links, we have two choices: we either can “predict” when links would be useful in the future, and thus not store them, or delete links in the subsequent iteration when we determine that they no longer transport energy. We choose the latter approach, since it is unclear how to achieve reliable prediction of link utility.

To be able to remove unnecessary links, i.e. links from sources and links deleted in subsequent iterations, we will use the line-space hierarchy, which stores the history of link creation. This has the advantage that a full description of light exchanges in the scene always exists, albeit at lower accuracy. This is an important difference between our approach and that of [17], and can be particularly useful in the context of dynamic environments [4] or when using importance [15].

4.1 A “Shooting” Algorithm

To achieve a hierarchical radiosity algorithm in which we can avoid the storage of links, we store an additional “unshot” radiosity field, in the spirit of the progressive refinement algorithm [3]. To initialise the system, we set the radiosity and the unshot radiosity equal to the emittance of each leaf element (i.e. not the clusters). Thus initially only sources have radiosity and unshot radiosity. Unshot radiosities are then pulled up the hierarchy. Finally, radiosities are set to the value of unshot radiosity at

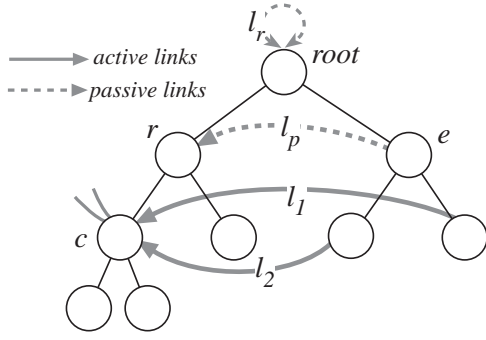


Figure 1: Line-space hierarchy basics

every level.

At each iteration, we gather unshot radiosity into irradiance, and push irradiance down the hierarchy, setting unshot radiosity to zero. Unshot radiosity is then reflected at the leaves, and added into the radiosity values. Unshot radiosities are averaged while pulling, and added to the radiosity values at every level. This algorithm and the error analysis are similar to those developed by Stamminger et al. [17], showing that this algorithm is equivalent to that of “standard” hierarchical radiosity.

4.2 The line-space hierarchy

The line-space hierarchy as defined in [4], is a hierarchy in link space. We use the terms line-space hierarchy and link hierarchy interchangeably, since we are not interested in the shafts attached to links as in [4].

The link hierarchy maintains the history of link subdivision, in the form of *passive* links. For example, in Figure 1 link l_p , between hierarchy elements r and e is subdivided. In a traditional hierarchical radiosity context, this link is deleted. When maintaining a link hierarchy, we store the link l_p , and the links l_1 and l_2 are considered “children links” of l_p . Links transferring energy will be called “active links”. In our example l_1 and l_2 are active links. The passive link l_p no longer corresponds to a real light transfer, but still maintains its form-factor (which required an expensive visibility query to obtain its value), so that if it is re-established in the future as active, it incurs no expense.

4.3 Refinement with Link Removal

Recall that we want to maintain the representation of all energy exchanges and allow the deletion of unnecessary links. To achieve this, we modify the refine and gather steps of traditional hierarchical radiosity. As in previous work [4], we combine refine and gather steps.

In our algorithm, each potential link is first tested against the *refinement criterion*. If we use the ΔBFA criterion, we decide whether the link has enough (unshot) energy to perform the transfer at this level. A new criterion

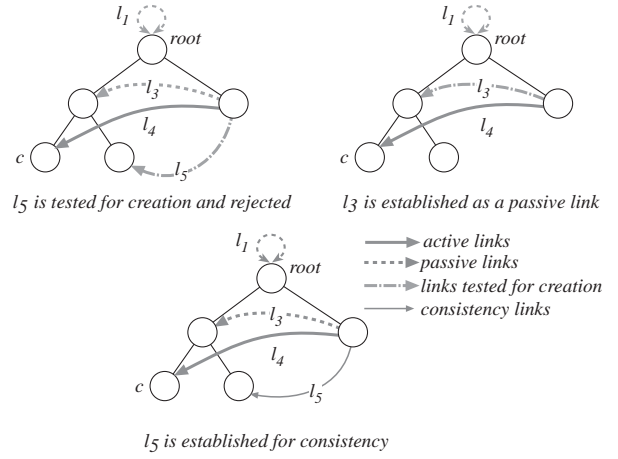


Figure 2: Creating links for consistency at recursion return. l_4 and l_5 are children of the passive link l_3 in the link hierarchy sense.

is added which is the *creation criterion*, deciding whether this link should be created, i.e., stored. Currently, the creation criterion simply does not create links from lights sources. More involved criteria are possible. Using the memory control mechanism discussed later, many other links are also not created, with the light transfer occurring on-the-fly during the recursion.

In addition, we remove links which no longer transfer energy. During refine/gather, instead of traversing the element hierarchy itself, we traverse the line-space hierarchy. While descending this hierarchy, we examine the current link. First we test the link against the refinement criterion. If it does not require further refinement, we gather irradiance across this link. We then test the link against the creation criterion. If it satisfies this test, we establish the link as an *active* link.

If refinement is required, we descend to the children of the link (subdividing when necessary), until the refinement condition test allows us to stop. The important step of our approach is performed when returning from this recursion. If no child link has been created, we test whether the current link L verifies the creation criterion. If it does, L is established as an active link; all links below L , created in previous iterations, are now deleted.

If children links exist, we establish L as a passive link. To maintain consistency, we traverse the immediate children in the line-space hierarchy to ensure that a consistent representation is formed, and create additional links if required. This may be necessary due to link removal. See Figure 2 for an illustration of this case, while the entire algorithm is summarised in Figure 3.

```

RefineAndGather(Link L)
  // Descent
  if L satisfies the refinement criterion then
    // end recursion
    add the irradiance transported to the receiver
    if L verifies creation criterion then
      L is stored as an active link
  else
    for all children l of L do
      RefineAndGather(l)
    // return of the recursion
    calculate Form-Factor of L from its children
    if there is no child link of L then
      if L verifies creation criterion then
        L is stored as an active link
        delete all previous children links
      else
        L is established as a passive link
        children links are established
        as active links for consistency

```

Figure 3: The RefineAndGather Algorithm

Recursive form-factor calculation

An interesting advantage of the **RefineAndGather** algorithm is that we can recursively calculate form-factors, and in particular form-factors of clusters.

We can consider two different cases, depending on whether the receiver r or the emitter e is subdivided:

1. The emitter is subdivided: $F_{re} = \sum_{j \in \text{child}(e)} F_{rj}$
2. The receiver is subdivided: $F_{re} = \frac{1}{A_r} \sum_{j \in \text{child}(r)} A_j F_{je}$,

where child-parent relationships of form-factors are those of corresponding links in the line-space hierarchy sense.

For clusters which do not contain participating media, the following formulation can be applied:

$$F_{rr} = \frac{1}{A_r} \sum_{i \in \text{child}(r)} \sum_{j \in \text{child}(r)} A_i F_{ij}$$

In this case, the self-link corresponding to F_{rr} (see also [12]), will be quite precise, and in certain cases will actually be exact (e.g. polygons which are “backfacing” each other as in the case of a sphere for example). As seen in the algorithm of Figure 3 this calculation can be integrated simply into the **RefineAndGather** algorithm, and thus may significantly improve the precision of computation, especially at the level of clusters.

5 Controlling Memory of Element Hierarchies

As mentioned in the introduction, our goal is to reduce memory for both links and the *element hierarchy*. In addition, since we have removed a large number of links, the actual subdivision of the hierarchy is often no longer

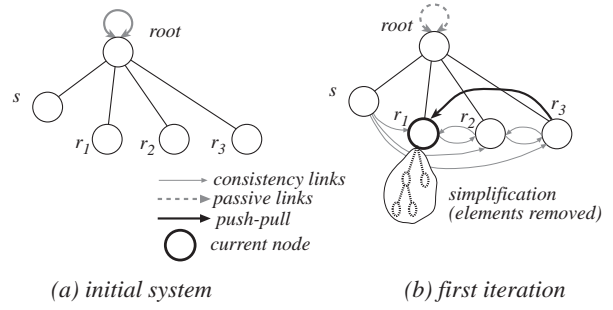


Figure 4: Self-link subdivision: the root self-link is subdivided. Note that some links are not drawn for clarity.

useful for the light simulation itself. It remains nonetheless necessary for the display of the lighting simulation result; however all the information related to the hierarchical representation for the simulation can be discarded and a much cheaper representation can be adopted, which is appropriate for display.

At an abstract level, we require a mechanism allowing the removal of parts of the hierarchy *during* refinement. The alternative, which would involve complete refinement and subdivision followed by the removal of the hierarchy, is inappropriate. Such an approach suffers from the fact that the subdivided element hierarchy may consume all available memory. A new refine-gather-push/pull loop is required, integrating the push-pull step with the **RefineAndGather** approach described previously.

Once the abstract mechanism is in place, we can replace the hierarchy subtree marked as “disposable” with an appropriate representation. In our case, we have chosen to replace the subdivision of surfaces by textures, and, as a first approximation, simplify clusters by removing their contained children.

5.1 Hierarchy Simplification/Refinement Algorithm

We perform an integrated refine-gather-push/pull step with a traversal of link space. We thus start at the root of the link hierarchy, which is the root self-link, and proceed with link refinement.

We need to ensure two basic conditions: (a) that we perform a push-pull operation only *once* at each hierarchy node, and (b) that when we perform a push-pull operation, all links arriving at this node must have been treated.

To do this we identify three different cases which we treat separately: (i) self-links, (ii) receiver subdivision and (iii) emitter subdivision. We use an example to illustrate how each case is treated.

Consider Figure 4(a), showing a scene consisting of sender s and three receivers r_1 - r_3 . At the outset, only the root self-link exists. This link is split, and all pairs of objects are linked for consistency (see discussion above in

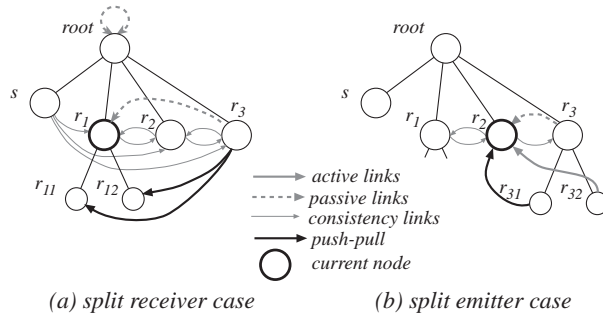


Figure 5: (a) The split receiver case and (b) the split sender case.

Section 4.3). We visit the children links in order; when r_1 is being considered as a receiver, we examine all links arriving at r_1 . A **RefineAndGather** operation will be performed on the link $s \rightarrow r_1$, resulting in the creation of subdivided nodes to represent the illumination.

All other links arriving at r_1 will be treated in order. Assume that the last link is the link $r_3 \rightarrow r_1$, as in Figure 4(b). At this point a push-pull operation is performed on node r_1 and its children. Since no links are created, the subdivision of r_1 is removed and replaced by a suitable representation.

At the second iteration, all receivers r_i have positive unshot radiosity and thus will also act as emitters. Consider the case shown in Figure 5(a). The current node is r_1 , and we are considering the last link arriving at r_1 which transfers light from r_3 . The refinement criterion decides that r_1 should be subdivided. In this case, when we treat the last link $r_3 \rightarrow r_1$ arriving at the receiver r_1 , we perform a push-pull operation on all the children of the current receiver r_1 .

The remaining case is that of a split emitter. In Figure 5(b) the current receiver node is r_2 and the emitter is r_3 . The refinement criterion required r_3 to be subdivided. Assume that the link $r_3 \rightarrow r_2$ is the last link arriving at r_2 (Fig. 5(b)). The push-pull operation will be performed on r_2 when we treat the link $r_{31} \rightarrow r_2$ (i.e., the last link arriving at r_2). Thus all links arriving at r_2 have been treated.

This process is summarised in Figure 6. Note that in practice a temporary variable is used during the return of the pull procedure, so that the current radiosity values are not changed.

This abstract framework allows the replacement of *any* subtree of the hierarchy by an appropriate representation during subdivision. Since the hierarchy of the scene is *complete*, i.e. surfaces and clusters are all contained in a single root cluster, in principle we can replace as much of the hierarchy as we see fit, assuming that this replacement is appropriate.

```

RefineGatherAndPushPull(
    Link L, Element R, Element S, IrradianceDown I)
    if L must be refined then
        choose element to split
        if R is split then
            return RefineGatherAndPushPullRcv(L,R,S,I)
        else if S is split then
            return RefineGatherAndPushPullEmit(L,R,S,I)
        else
            return R  $\rightarrow$  PushPull(I);

RefineGatherAndPushPullRcv(
    Link L, Element R, Element S, IrradianceDown I)
    Radiosity of R is set to Zero();
    for r child of R
        l = potential link from S to r;
        R  $\rightarrow$  Radiosity += r  $\rightarrow$  AreaFactor *
            RefineGatherAndPushPull(l,r,S,I+R  $\rightarrow$  Irradiance);
    R  $\rightarrow$  Radiosity /= R  $\rightarrow$  AreaFactor;
    if R has links but its children do not
        replace subdivision of R
    return R  $\rightarrow$  Radiosity;

RefineGatherAndPushPullEmit(
    Link L, Element R, Element S, IrradianceDown I)
    while c not the last child of S
        l = potential link from c to R;
        RefineAndGather(r,R,c);
    l = potential link from c to R;
    return RefineGatherAndPushPull(l,R,c,I);

```

Figure 6: Refine Gather and Push-Pull algorithm

In practice, we can easily replace the hierarchy which is due to subdivision on a polygon surface, since the underlying geometry remains the same. We will show next how to do this by replacing subdivided elements by textures (recall the case of r_1 in Figure 6(a)). Related ideas have been used in different contexts in [16, 9, 10].

Replacing initial scene geometry (i.e., simplifying clusters and their contents) is much more involved. Many potential solutions can be found, all of which require the use of some geometry simplification technique (i.e. image-based [13], volumetric [11] or multi-resolution representations [6] etc.). Investigating these alternatives is beyond the scope of this paper. Instead, we will present a simple first solution, by representing the cluster as a point for light-transfer, and as a shaded bounded box for display.

5.2 Replacing Polygon Subdivision by Textures

The data structure of a hierarchical element corresponding to an input polygon contains information about its children, radiosity, unshot radiosity and irradiance as well as link information. We create such a node which has a

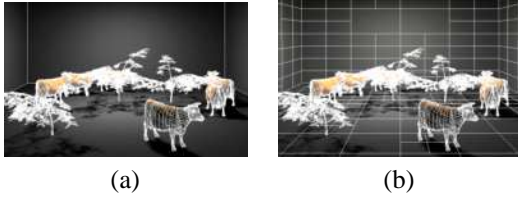


Figure 7: (a) 1st iteration: the illumination detail of the floor and wall polygons is represented by a texture (i.e. no hierarchical subdivision exists on these polygons). (b) 2nd iteration, some re-splitting occurs. The elements share the same texture.

special texture attached to it. This texture corresponds to the values of the subdivided initial polygon.

The texture is a two-dimensional array of the floating point values of radiosity. In addition a second array which represents unshot radiosity is created in the same manner. These two arrays are stored with the input polygon. When displaying the polygon, the floating point array is converted to a texture appropriate for display on the graphics hardware. In subsequent iterations, the polygon may be resubdivided (see Figure 7), but the texture array is not replicated. The sub-elements are simply assigned correct texture (sub)coordinates. Note that we could gain memory by using a more compact representation for these textures.

Irradiance need not be explicitly stored as a texture, since it is only needed during push-pull, and thus is a temporary variable created during the push phase and freed at the end of the pull step for a given subtree of the hierarchy.

The gather and push-pull operations are applied in the normal manner to each entry of the radiosity and unshot radiosity arrays, thus maintaining the same quality solution, without the overhead of the hierarchical data structures. In our system, ignoring the memory required for the original geometry, a quadrilateral hierarchical element costs around 200 bytes, including children pointers, parametric coordinates, radiosity and irradiance fields, list of links pointer, and vertex geometry and color, used for display of intermediate results. Note that our implementation is in C++, incurring additional storage overhead.

5.3 Replacing Cluster Contents

As a first approach, we have chosen a simple replacement strategy for clusters. For the lighting simulation [12], clusters are considered to be a single point sample of radiosity (typically at the center of the cluster). If the **RefineGatherAndPushPull** algorithm decides that a subtree based at a cluster is suitable for simplification, we remove the children surfaces from the cluster. Clusters are simplified as for polygons and if their projection covers less than a pre-defined number of pixels.

The simplified cluster stores a single value for each of radiosity, irradiance and unshot radiosity. For display, we use the bounding box shaded with the colour value of the clusters radiosity. Since the simplified node is a leaf of the element hierarchy, it stores a reflectance value, and irradiance pushed down to it is reflected at the level of the cluster. Finally, we treat the simplified cluster bounding box as a volumetric element, with a (scalar) extinction coefficient in the manner of [12].

In contrast to polygon subdivision replacement, this approach is view-dependent and does not produce exactly the same solution as the “standard” approach. It is presented here simply to demonstrate that the element replacement can be performed at *any* level of the hierarchy. More involved simplification could be used to achieve better results (see Section 8).

6 Memory Control Mechanism

The algorithm presented above will reduce memory used by links and the hierarchy. A certain number of links are however still created (in particular all links for consistency and links transporting indirect light), resulting in a peak in memory usage, typically at the second iteration (recall that no links from sources are stored). It is thus important to be able to control the total amount of memory required as much as possible, in the same spirit as the cache strategy of [17].

Instead of an explicit cache, we simply use the link hierarchy to directly limit the memory used by links. This is done by modifying the link creation criterion (see Section 4.3): before creating a link we test to see whether the link is above a certain level in the link hierarchy. The cut-off level is estimated based on the memory the user wishes to use, and an estimation of the average number of links in the link hierarchy below this level. This has the consequence of moving active links higher up in the hierarchy. Recall that the light transfers of links which are not stored are performed on-the-fly during the traversal of the link hierarchy.

Nonetheless, a complete representation of all light exchanges is maintained, albeit at a coarser level. Clearly, a more involved method could be used to tightly control the actual memory used by the links and appropriately modify the cut-off level of link creation.

An important consequence of moving links up the hierarchy is that large element hierarchy subtrees become candidates for removal. If the texture replacement only is used, very few surfaces are actually subdivided. Removing the clusters will further remove hierarchy subtrees, but in a view-dependent manner.



Figure 8: Image generated by the original HRC algorithm for the Medium Hall scene.



Figure 9: Image generated by the replacement by textures algorithm for the Complex Hall scene.

7 Implementation and Results

We have implemented the algorithm described previously as part of our hierarchical radiosity system. We present results of our implementation, and compare them to a “standard” BFA hierarchical radiosity with clustering (HRC) approach.

We have tested our approach of three scenes. This is a sequence of three “halls” containing 85,000 to 676,000 initial polygons each. These are called “Simple”, “Medium” and “Complex Hall” respectively and abbreviated *SH*, *MH* and *CH*. Examples of the hall scenes are shown in Figures (8, 9). We first show the statistics for the reference “standard” HRC solutions in Table 1.

Test Scene	<i>SH</i>	<i>MH</i>	<i>CH</i>
input polys	65K	169K	676K
mem_{init}	39MB	77MB	309MB
mem_{clust}	7MB	15MB	59MB
<i>links</i>	3.5M	7.5M	4.4M
$poly_{sub}$	137K	203K	2.5M
mem_{hier}	23MB	40MB	500MB
mem_{link}	94MB	202MB	118MB
<i>Time</i>	1h30mn	4h07mn	4h15mn
mem_{tot}	156MB	320MB	927MB

Table 1: Statistics for the reference “standard” radiosity solution. mem_{init} is the total initial memory *before* the solution, mem_{clust} is the part used by clusters, *links* is the number of links at the end of the solution, and $poly_{sub}$ is the number of hierarchical polygonal elements. mem_{hier} is the memory used by the hierarchy (excluding initial polygons), and mem_{link} that used by the links. *Time* is the total computation time; mem_{tot} is the total memory used, including the initial memory.

In Table 2 we show the results of our algorithm with texture replacement of subdivision only (no cluster reduction is performed), where the memory control target has been set to about 30% of initial memory mem_{init} . Clearly, our algorithm achieves significant overall savings in memory, the overall gains varying from 73 to 88% (excluding initial memory). It is important to note that

Test Scene	<i>SH</i>	<i>MH</i>	<i>CH</i>
mem_{link}	1.5MB	1.8MB	11.3MB
$mem_{hier+tex}$	13.3MB	29MB	155MB
<i>links</i>	55K	67K	426K
$poly_{sub}$	22K	3K	7.7K
$gain_{link}$	98%	99%	90%
$gain_{hier}$	42%	27%	70%
$gain_{mem}$	87%	88%	73%
<i>Time</i>	1h07mn	2h51mn	5h39mn
mem_{tot}	58.3MB	112MB	501MB

Table 2: Results of our algorithm. Notation is as in Table 1, except $mem_{hier+tex}$ which is the memory used by the hierarchy and the textures (excluding initial polygons), and $gain_{link}$, $gain_{hier}$, $gain_{mem}$ which are the percent memory gains for link, hierarchy and overall respectively (excluding initial memory).

scenes with very large memory consumption, can now be treated with much less memory (e.g., 501MB instead of 927MB; initial memory is 309MB). The computation time is comparable or even less than that of the standard solution. This is mainly due to the improved form-factor calculation, which reduces the values of many cluster self-link form-factors resulting in a slightly lower number of gather operations overall. The complex *CH* scene has fewer light sources in each room, resulting in a lower number of overall links, and a faster computation time.

Finally we present an example of the cluster reduction algorithm (Figure 10) showing very few artifacts. Here the memory gain was 80% mainly due to the links, since most of the objects are not subdivided in this scene.



Figure 10: Scene containing 43,000 polygons. (a) the original image and (b) the image generated with cluster reduction.

8 Conclusions and Future Work

We have presented a novel algorithm for controlling the memory consumption for hierarchical radiosity with clustering. We show how we can reduce the memory used by both the element hierarchy and the light-transport links. To do this, we introduce a new algorithm for the refine-gather-push/pull loop, which removes unnecessary links and permits the removal of a subtree of the element hierarchy during subdivision. The algorithm is based on the link or line-space [4] hierarchy, thus preserving the overall representation of global light transfers. Using this representation, our memory control algorithm can calculate global illumination solutions on a limited memory budget, by moving light transfers higher in the link hierarchy (in effect representing them in a more imprecise manner). The results of our implementation show that we can achieve significant savings in real memory consumption with little loss of visual quality or precision in the simulation of light.

In future work, we need to investigate more sophisticated memory control mechanisms, taking the memory consumption of the hierarchy directly into account. This would require storing parts of the hierarchy (clusters and surfaces) to disk during the computation, when the algorithm requires removal of the corresponding sub-tree. If, in addition, we can perform the first iteration of light transfer while reading in the scene, we could apply the simplification techniques described in this paper “on-the-fly” during the loading of the scene file. As a result, it should be possible to simulate environments of arbitrary size on any computer. This entire approach will probably require reorganizing the the order of lighting computations and sophisticated disk handling routines.

We believe that the ideas in this paper can be used with more appropriate representations for simplified clusters. These could take the form of image-based or volumetric primitives, or multi-resolution geometric simplifications. Finally, it will also be interesting to develop strategies to predict the utility of a link in future iterations reducing storage requirements of links.

Acknowledgements

Thanks to Frédo Durand for proof-reading and careful insights. This research was partly funded by Commission of the European Communities ESPRIT Reactive LTR project ARCADE (#24944).

9 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. In *Proc. of SIGGRAPH '91*, July 1991.
- [2] P. Bekaert, L. Neumann, A. Neumann, M. Sbert, and Y. Willems. Hierarchical monte carlo radiosity. In *9th EG Workshop on Rendering*, June 1998.
- [3] M. F. Cohen, S. E. Chen, J. R. Wallace, and D. P. Greenberg. A progressive refinement approach to fast radiosity image generation. In *Proc. of SIGGRAPH '88*, pages 75–84, August 1988.
- [4] G. Drettakis and F. X. Sillion. Interactive update of global illumination using a line-space hierarchy. In *Proc. of SIGGRAPH '97*, pages 57–64, August 1997.
- [5] T. A. Funkhouser. Coarse-grained parallelism for hierarchical radiosity using group iterative methods. In *Proc. of SIGGRAPH '96*, pages 343–352, August 1996.
- [6] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proc. of SIGGRAPH '97*, pages 209–216, August 1997.
- [7] S. Gibson and R. J. Hubbard. Efficient hierarchical refinement and clustering for radiosity in complex environments. *Comp. Graphics Forum*, 15(5):297–310, 1996.
- [8] P. Hanrahan, D. Salzman, and L. Aupperle. A rapid hierarchical radiosity algorithm. In *Proc. of SIGGRAPH '91*, pages 197–206, July 1991.
- [9] P. S. Heckbert. Adaptive radiosity textures for bidirectional ray tracing. In *Proc. of SIGGRAPH '90*, pages 145–154, August 1990.
- [10] K. Myszkowski and T. L. Kunii. Texture Mapping as an Alternative for Meshing During Walkthrough Animation. In *5th EG Workshop on Rendering*, pages 375–388, 1994.
- [11] F. Neyret. Modeling, Animating, and Rendering Complex Scenes Using Volumetric Textures. *IEEE Trans. on Visualization and Comp. Graphics*, 4(1):55–70, January 1998.
- [12] F. X. Sillion. A unified hierarchical algorithm for global illumination with scattering volumes and object clusters. *IEEE Transactions on Visualization and Comp. Graphics*, 1(3):240–254, September 1995.
- [13] F. X. Sillion, G. Drettakis, and B. Bodelet. Efficient impostor manipulation for real-time visualization of urban scenery. *Comp. Graphics Forum*, 16(3), September 1997.
- [14] B. Smits, J. Arvo, and D. Greenberg. A clustering algorithm for radiosity in complex environments. In *Proc. of SIGGRAPH '94*, pages 435–442, July 1994.
- [15] B. E. Smits, J. R. Arvo, and D. H. Salesin. An importance-driven radiosity algorithm. In *Proc. of SIGGRAPH '92*, pages 273–282, July 1992.
- [16] C. Soler and F. X. Sillion. Automatic calculation of soft shadow textures for fast, high-quality radiosity. In *9th EG Workshop on Rendering*, June 1998.
- [17] M. Stamminger, H. Schirmacher, P. Slusallek, and H-P. Seidel. Getting rid of links in hierarchical radiosity. *Comp. Graphics Forum (EUROGRAPHICS '98)*, 17(3), September 1998.
- [18] S. Teller, C. Fowler, T. Funkhouser, and P. Hanrahan. Partitioning and ordering large radiosity computations. In *Proc. of SIGGRAPH '94*, pages 443–450, July 1994.
- [19] A. J. Willmott and P. S. Heckbert. An empirical comparison of progressive and wavelet radiosity. In *8th EG Workshop on Rendering*, pages 175–186, June 1997.

3.5.6 A Practical Analysis of Clustering Strategies for Hierarchical Radiosity (EG'99)

Auteurs : J-M. Hasenfratz, C. Damez, F.X. Sillion et G. Drettakis

Actes Congrès Eurographis'99

Date : septembre 1999

A Practical Analysis of Clustering Strategies for Hierarchical Radiosity

Jean-Marc Hasenfratz, Cyrille Damez, François Sillion, George Drettakis

iMAGIS – GRAVIR/IMAG-INRIA, Grenoble, France

Abstract

The calculation of radiant energy balance in complex scenes has been made possible by hierarchical radiosity methods based on clustering mechanisms. Although clustering offers an elegant theoretical solution by reducing the asymptotic complexity of the algorithm, its practical use raises many difficulties, and may result in image artifacts or unexpected behavior. This paper proposes a detailed analysis of the expectations placed on clustering and compares the relative merits of existing, as well as newly introduced, clustering algorithms. This comparison starts from the precise definition of various clustering strategies based on a taxonomy of data structures and construction algorithms, and proceeds to an experimental study of the clustering behavior for real-world scenes. Interestingly, we observe that for some scenes light is difficult to simulate even with clustering. Our results lead to a series of observations characterizing the adequacy of clustering methods for meeting such diverse goals as progressive solution improvement, efficient ray casting acceleration, and faithful representation of object density for approximate visibility calculations.

1. Introduction and Motivation

In scenes with great geometric complexity containing hundreds of thousands or even millions of polygons global illumination algorithms require the grouping, or clustering, of the individual primitives. In this way light exchanges can be treated at the level of the clusters and thus the computational complexity of the radiosity solution becomes manageable^{14, 12}. Unfortunately, approximations made by hierarchical radiosity algorithms using clustering are very sensitive to the quality of the cluster hierarchy. Due to the complexity of the algorithms and data structures (by definition we are working with very complex models), no experimental analysis of the behavior of clustering algorithms has been undertaken today. Willmott and Heckbert¹⁵ provided an inspiring study for progressive refinement radiosity and hierarchical radiosity without clustering, but evidently this study was restricted in the type of scene considered.

In this practice-and-experience paper, we investigate clustering algorithms for hierarchical radiosity in a practical context. In particular, we have chosen an experimental approach, by comparing the performance of different clustering algorithms. We have concentrated our attention on models provided by real-world applications, in an attempt to uncover problems which real users of radiosity will encounter.

We propose a taxonomy of clustering algorithms, based both on the type of data structure used, and the type of construction algorithm. We then proceed to define requirements for a clustering algorithm. In particular, a clustering algorithm should provide the user with an intuitive time-quality tradeoff: the more time is spent on a solution, the better the quality of the solution. Several other desirable properties are also identified, such as limiting the overlap of clusters, optimizing the “tightness” of the fit of clusters around objects and appropriate size of the clusters with respect to the contained objects.

Once the requirements have been defined, we proceed with a series of experiments run on models used mainly in real-world applications. Various parameters are measured, including the image quality for varying simulation parameters, the cluster construction time, the quality of the hierarchy using different criteria and the speed of ray-tracing for each cluster hierarchy.

The results of these experiments have allowed us to observe a number of interesting properties of clustering, which are discussed in detail. This in-depth study of clustering leads to the understanding that there exists no universal, ideal clustering method, while explaining the relative merits of various approaches.

2. A Taxonomy of Clustering Algorithms

Clustering for hierarchical radiosity was introduced by Smits *et al.*¹⁴ and Sillion^{11, 12}. Clustering algorithms can be classified by considering two aspects important to their usage:

1. The choice of data structure used. Broadly speaking, two categories have been presented: regular, typically axis-aligned subdivisions of space and hierarchies of bounding volumes (HBV), which are more closely adapted to the object geometry. Examples of such structures include *k*-d trees and Octrees. Hybrids have also been proposed but have not been used to date in clustering for radiosity.
2. The type of construction algorithm. Again, two basic categories have been used: top-down and bottom-up construction.

2.1. Data Structure Choices

The data structures used for clustering have been mostly inherited from traditional spatial subdivision structures used in graphics.

2.1.1. Octrees and *k*-d Trees

Regular structures such as octrees or *k*-d trees have several advantages:

- They are fast to build (see Section 5.2), and easy to construct since the form of the clusters is (nearly always) predefined.
- They can provide fast ray-tracing since they use traditional ray-traversal mechanisms (see Section 5.2).

Their disadvantages are not specific to clustering for illumination, but due to the rather inflexible nature of their construction:

- Objects which intersect cell boundaries do not have a trivial placement in the tree. As a consequence a heuristic needs to be determined to place the object at an appropriate level.
- The tree can be very deep if the scene contains objects with large differences in scale.
- Since the shape of sub-clusters is prescribed by the subdivision mechanism, many empty clusters can be created. These clusters consume memory and resources since they are considered in all radiosity operations.
- Cluster boundaries do not tightly fit the set of contained objects, resulting in poor-quality estimates of the optical density, for the volumetric estimation of visibility¹².

The first clustering algorithm to use *k*-d trees for illumination was presented by Sillion¹¹. This approach uses a traditional, axis-aligned *k*-d tree into which objects are inserted. The strategy chosen for placement of objects intersecting cell boundaries is to put objects at the lowest level entirely containing them. For many models, this can have very negative consequences since a large number of objects can end

up at very high levels in the hierarchy, with adverse effects on computation speed. Since the refinement algorithm must, when refining a link to a cluster, create new links for each of its children, a high branching factor may result in long computational times.

2.1.2. Hierarchy of Bounding Volumes

Algorithms based on bounding volumes hierarchies have been used by several researchers^{14, 13, 7}. These algorithms use rectangular axis-aligned bounding boxes, and share the following qualities:

Advantages:

- If built correctly, the cluster hierarchy adapts well to the organization of the scene into individual objects (possibly each having its own sub-cluster hierarchy).
- An “intuitive” hierarchy can be produced for a scene with very different object sizes, without creating empty clusters.
- Clusters can be made to tightly fit their contents.

Disadvantages:

- Overlapping clusters are generally unavoidable.
- Bad clusters often result when the scene is considered as a set of individual polygons, without taking advantage of the object structure (*e.g.*, clusters mixing parts of different nearby objects).

Hybrid data structures have also been developed for clustering objects in different domains (notably for ray-tracing acceleration). Cazals *et al.*^{2, 3} construct hierarchies of uniform grids, and Klimazewski *et al.*¹⁰ present a similar approach. These methods are however designed to optimize ray-tracing by constructing regular grid structures, and are thus unsuitable “as is” for clustering. Some ideas however, in particular those concerning grouping of objects, by Cazals *et al.*^{2, 3}, could be applied in part to future clustering algorithms.

2.2. Construction Algorithms

In this section we re-visit the algorithms described above by construction algorithm type. The basic approaches are top-down and bottom-up construction. We will briefly discuss some issues of manual clustering, which is often used in industry and even in research.

2.2.1. Top-down Clustering

Typical top-down construction algorithms include the *k*-d tree (KDT) construction used by Sillion¹². An initial cell is created, and objects are subsequently added into the cell by appropriately subdividing the cell so that the object “fits” in a sub-cell. As mentioned above, objects crossing cell boundaries are placed high up in the hierarchy.

Christensen *et al.*⁴ build a hierarchy of bounding volumes starting with the bounding box of the entire scene. The bounding box is split into eight octants. For each surface contained in this bounding box, if the size of the object is smaller than that of an octant, the object is inserted into the octant containing its centroid. Otherwise, the object is attached as a direct child of the cluster at this level. A new bounding box of each octant is computed, and the algorithm continues recursively in the same manner. In this paper we refer to this algorithm and data structure as TF-OCT (“tight-fitting octree”).

2.2.2. Bottom-up Clustering

Bottom-up construction of clusters is inherently more complex, since it requires the examination of the existing objects and their mutual spatial relationships. Since it is in a certain sense an optimization process, the complexities of algorithms suited to such constructions rapidly become quadratic or higher in the number of objects to be processed.

Smits *et al.*¹⁴ mention they use a “modified Goldsmith-Salmon algorithm” without describing the specifics¹⁴. Silion and Drettakis employ a hierarchy of regular grids to filter the objects and clusters in order of increasing size, and produce candidate clusters based on spatial proximity¹³. The same algorithm is used by Gibson and Hubbard⁷. In the rest of this paper we refer to this method as PROXI, for “Proximity clustering”.

Note that as we choose to group objects based on a minimization function (*e.g.*, minimize the volume of the clusters created with respect to the objects being inserted)^{13,7}, an optimal solution requires an exhaustive test of all the combinations of groupings of the objects being considered. Clearly, this expense is extremely costly. A more appropriate grouping approach could be that of Cazals *et al.*³.

2.2.3. Manual Construction

The difficulty of clustering is such that automatic methods are not able to treat all scenes effectively. As a consequence user intervention is inevitable at some stage in the process. Examples of such intervention are the definition of “natural clusters” such as those defined by the group of polygons belonging to a single “object” (a chair for example) or a logical group such as the set of all objects on top of a desk. The special case of touching objects is also important, since it is a way of defining object hierarchies.

Some of these interventions can be handled at input (often the set of objects defining a chair object is defined as a group by the modeling program and then instanced). Such information should be incorporated by the clustering system and used to its advantage when available. Other cases are much harder (*e.g.*, the “objects on the desk” case, or touching objects).

2.3. Improved Approaches

An extended version of KDT, which we call OBT for “Overlapping Binary Tree”, was derived as an attempt to merge some of the benefits of Christensen’s octree construction and the binary trees obtained with KDT. The algorithm is very similar to the KDT construction, but here, objects crossing the cell boundaries are pushed down in the hierarchy only if the ratio of their size and the considered cell size, is higher than an overlap parameter which can be set by the user (when set to 0, a KDT hierarchy is obtained). The clusters produced by this algorithm can be larger than those of KDT, and therefore may overlap, but the parameter provided gives us control on the maximum potential overlap between adjacent clusters. Specifically, for a given value of the overlap parameter λ , the following relations are true:

- maximal size of the KDT cell to which an object of size S is assigned:

$$x = \frac{2S}{\lambda}$$

- maximal overlap between two adjacent OBT clusters of original size d :

$$o = \frac{8}{3}\lambda d$$

These relations ensure that large clusters will not be overwhelmed by large (and spatially sparse) collections of small objects.

As a consequence, the OBT algorithm produces a deeper hierarchy than KDT, and thus even more empty clusters. In order to get rid of empty clusters, we developed another algorithm that uses an auxiliary OBT hierarchy to build HBV clusters in a second pass. In this second step, we keep only the bounding box of the contents (objects and child clusters) of each non-empty OBT cluster. Therefore, each cluster in the obtained HBV hierarchy will have at most two child clusters. We call the resulting new structure and algorithm OKDT (Overlapping KDT).

Unfortunately, OKDT proved to be unable to separate objects formed of thin, long polygons, such as cylinders, because they were unlikely to be inserted deep in the hierarchy. As a consequence, several clusters contain too many surfaces. This had adverse consequences on ray-casting and refinement time (average branching factor being central for hierarchical algorithms). Therefore, in order to improve our algorithm we also tried applying a PROXI clusterisation process on each cluster containing more than a dozen polygons. We call this algorithm OKDT-P (Overlapping KDT with Proximity second pass).

3. Requirements for a Clustering Algorithm

The most important goal for a good clustering algorithm is to group the objects in a way which represents light transfer to or from the group in the most faithful manner. At the heart

of the hierarchical radiosity algorithm is the assumption that it is possible to replace a “complete” calculation by a simpler one, performed using simplified representations such as clusters. This idea can only be exploited if the resulting simplification only introduces modest perturbations in the calculation.

Unfortunately, a precise definition, or even a set of quantitative yardsticks allowing us to evaluate the quality of such a representation do not currently exist. Instead, a set of heuristics have been developed by various researchers in this domain (e.g. ^{4, 13, 7}). Considering the existing body of work, we can identify two sets of desirable properties for a clustering technique: those affecting the quality of the simulation, and those affecting the overall efficiency of the applications.

3.1. Requirements Regarding the Quality of the Results

We outline below some of the required properties in order to arrive at a satisfactory simulation of light transfer:

1. Monotonicity with respect to light transfer precision. If a light transfer previously represented at a certain level becomes represented at a finer level of the cluster hierarchy, the precision of the light transfer should be increased. If we consider the Time-error graph obtained by plotting the computation time as a function of the solution error, for different tolerance thresholds, we would like to obtain a *smooth and monotonic* function.
2. Overlapping clusters should be avoided as much as possible. Overlapping is problematic mainly because it implies the treatment of the transfer of light of a volume to itself, which is difficult to represent and to express in terms of the hierarchical radiosity formalism. This is especially true in the case where error bounds are estimated to drive the hierarchical refinement.
3. The nature of object group shapes should be preserved as much as possible. Clusters which contain large regions of empty space and scattered small objects should be avoided. Although this may seem evident, many automatic clustering algorithms have trouble respecting this requirement.
4. Objects should always belong to a cluster of “appropriate” size, with respect to their own dimensions. This requirement is difficult to quantify, but is especially important in the context of approximate visibility calculation, where the attenuation of light passing through clusters is estimated based on a volumetric analogy. This analogy relies on the calculation of an optical density for each cluster, which is only meaningful for clusters with well-distributed (i.e. “random”) collections of similarly-sized objects ¹¹.

3.2. Efficiency Considerations

Considering the major impact of cost considerations on the usability of clustering radiosity systems, particular attention must be paid to the two following aspects:

3.2.1. Building the Hierarchy

First, we are of course concerned about the efficiency of hierarchy construction. Even though the actual clustering phase is generally a preprocess, and can sometimes be stored with the model, it is still preferable to have efficient construction algorithms:

- Fast cluster construction is important in the modelling/design stage where the model changes significantly and thus long clustering times hinder lighting experimentation.
- For extremely large models it may be impractical to store an additional high-overhead data structure describing the cluster hierarchy.
- Finally, the application of hierarchical radiosity with clustering to dynamically changing environments ⁵ may require at least a partial rebuild of the cluster hierarchy at interactive rates.

3.2.2. Acceleration of Ray Casting

Another important problem is the potential use of the cluster hierarchy as a supporting data structure to accelerate ray casting queries. Such queries are used for image generation or more often, in the case of radiosity, for visibility calculations when computing form-factors. Indeed, ray casting is the method of choice for visibility estimation in hierarchical radiosity, because the hierarchical algorithm fragments the calculation into a large number of individual queries, each relative to a different emitter/receiver pair ⁹. Global approaches such as hemi-cube calculations are therefore inappropriate for hierarchical radiosity.

The requirements of ray-tracing acceleration are often contradictory with those of clustering for illumination. For example, structures which minimise the number of intersections on average in a statistical sense, such as the algorithm of Goldsmith and Salmon ⁸, result in clusters which contain elongated bounding boxes containing large empty spaces. These clusters are inappropriate for light transfer estimation or optical density estimation, as outlined above.

A possible solution is the creation of two separate structures, one for clustering and one for ray-tracing acceleration. This however would be undoubtedly far too expensive in memory for very large models. In practice, we have observed that the performance cost implied by the use of the clustering structure for ray acceleration is most often acceptable. Some comparative results concerning the performance of various cluster hierarchies as ray tracing accelerators are presented in Section 5.

4. Experimental methodology for evaluation clustering algorithms

4.1. Methodology

As stated earlier, clustering is necessary to make the illumination computation of industrial scenes tractable. Its most

important drawback is that it makes error control very difficult, where it would be necessary to allow fast solutions to be calculated with an acceptable precision. Given the previously described clustering algorithms, we need to determine the critical parameters for their behavior.

Until now, most of the scene models used in the literature were designed by researchers for research purpose. We feel that performing our study on such scenes would have in some way "hidden" most of the problems involved by clustering. In raw "industrial" scenes, poor quality initial meshing, dense distribution of objects or randomly ordered polygons can make clustering algorithms barely usable.

Thus, our approach has been to perform a sufficient number of experiments on a set of complex, "real life" scenes in order to understand and compare each algorithms behavior in terms of quality-versus-time tuning. We limited the range of our experiments to fairly coarse and fast calculations where the impact of clustering is significant.

4.2. Clustering Strategies Studied

The following table summarizes the set of clustering strategies considered in this paper.

- PROXI: Proximity cluster. Bottom-up construction after size filtering.
- OKDT: Overlapping k -d tree. Top-down allowing partial overlap.
- OKDT-P: Overlapping k -d tree with limited branching. OKDT modified to re-cluster cells with many children, using PROXI locally.
- TF-OCT: Tight-fitting octree. Top-down, layer-by-layer octree construction, with re-fitting of octree cell before subdivision.

We did not submit the KDT algorithm to our tests because our experience with it proved that its average branching factor was far too high for it to be usable with scenes as large as the one we used.

4.3. Test Scenes Chosen

We performed our tests on four different scenes, shown in Figure 1. Three of them are rather large industrial-type models while the fourth one is provided as a comparison to show that clustering usually behaves very well when applied to small scenes designed for research purposes. We have taken these scenes as representative scenes for the clustering problem, allowing us to identify the different problems of the algorithms which we will test.

AIRCRAFT (184,456 polygons)

Model of an aircraft cabin (courtesy of LightWork Design Ltd). All objects have been tessellated into (rather small) triangles to account for the rounded shapes.

VRLAB (30,449 polygons)

A virtual reality lab with two floors and mostly overhead lighting (courtesy of Fraunhofer Institut für Graphische Datenverarbeitung). This scene has a mixture of large polygons, likely to be subdivided, and very small patches (on chairs and desktop computers).

CAR (216,157 polygons)

A model of a car interior with very small details, lit by a single overhead console fixture (courtesy of BMW).

OFFICE (5,260 polygons)

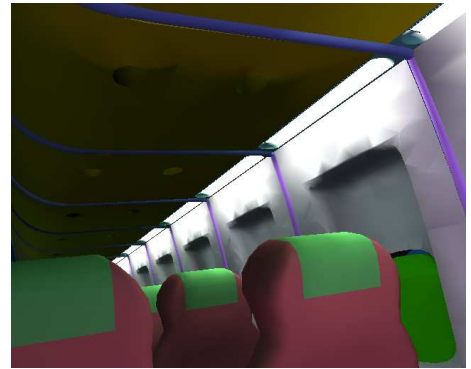
A model of a simple office scene. This model is much smaller than the other three and is provided to show that clustering performs well on this kind of small scene usually found in the literature.

4.4. Tests Performed

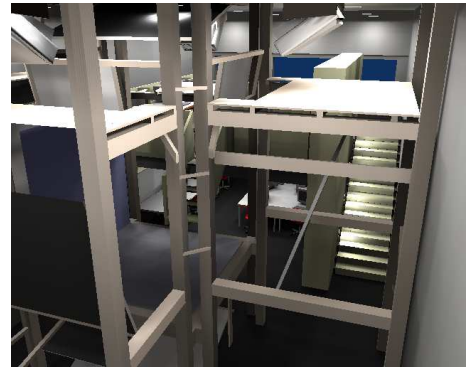
For each of these scenes, and for each clusterizer we decided to run the following experiments, designed to measure key aspects of clustering:

- **Data structure quality:** We listed the number of clusters in the hierarchy, as well as the average number of surfaces and clusters in each cluster node. These figures are needed to estimate both the memory cost of the hierarchy (compared to the cost of the actual geometry of the scene) and its efficiency for hierarchical radiosity. Recall that the computation speed of cluster based hierarchical radiosity depends on the average branching factor because the refinement algorithm must, when refining a link to a cluster, create new links for each child of this cluster.
- **Measure of time needed to build the cluster hierarchy.**
- **Solution quality:** To evaluate image quality we chose to use a "visual quality" error evaluation on images obtained for a given set of viewpoints rather than a view-independent error metric since we want to study the visual quality rather than the accuracy of the energy transfer quantities. Images were compared to reference images (resulting from a maximum precision calculation) using the following metric: we transform all pixels from RGB space into chromaticity space XYZ. The global image error is then the L_2 norm of the pixel-by-pixel difference image between the current and the reference image, evaluated in CIELUV space ⁶.
- **Algorithm usability:** To study the "quality versus time" behavior of each algorithm, we rendered each scene using each clustering algorithm, changing only the parameters controlling our refinement process (we used a BF-like refinement algorithm ⁹, and an error-bound based refinement, both of them showed similar behavior on clusters), measuring the time needed and evaluating the resulting image quality using the method previously described. Since we are interested in the effects of the cluster hierarchy, we do not perform tests on very precise solutions, which involve only surface-to-surface energy exchanges. Instead, we chose our refinement parameters in such a way that the proportion of energy gathered through links

AIRCRAFT



VRLAB



CAR (courtesy of BMW)



OFFICE

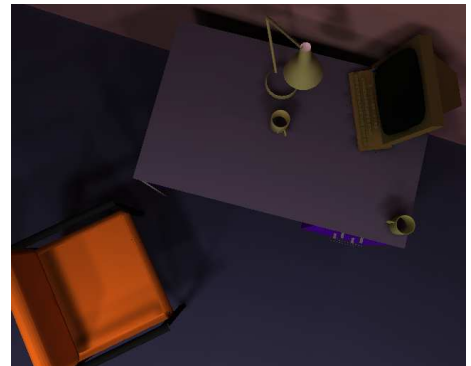


Figure 1: *The four test scenes.*

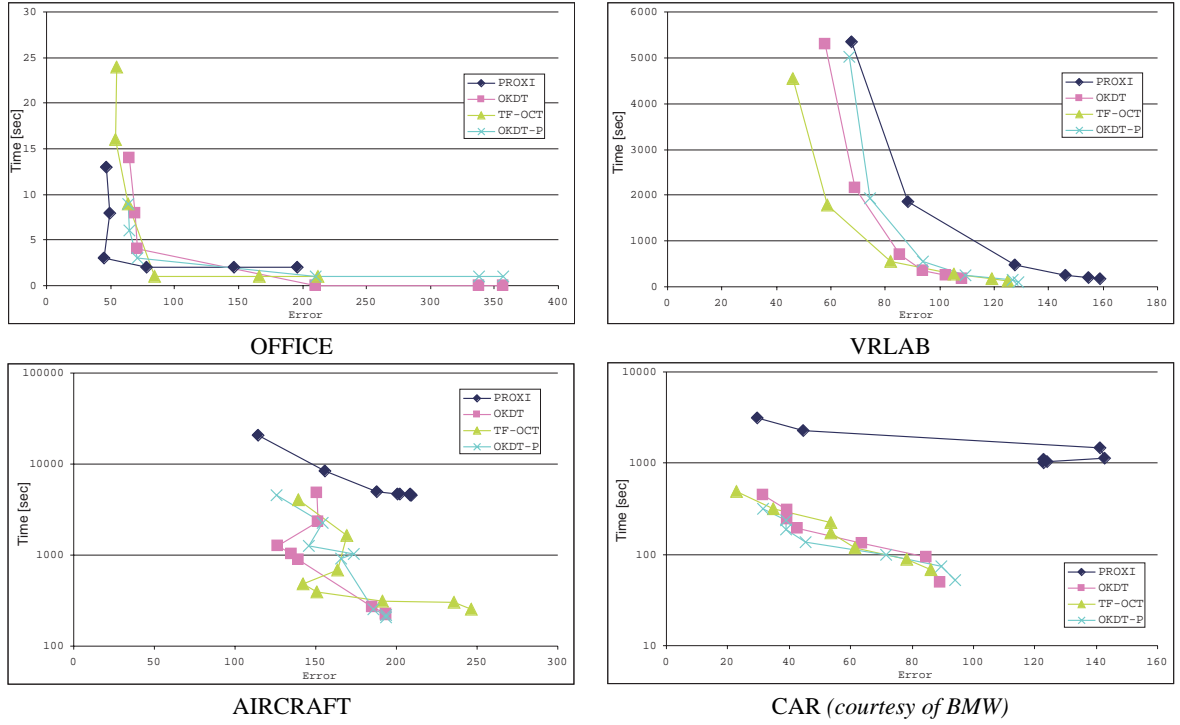


Figure 2: Time-error curves

involving clusters is significant (most of the time varying between 30% and 100%).

- **Ray casting acceleration:** As we said in Section 3.2.2, we have seen that it is desirable to use the cluster hierarchy as an acceleration structure to answer visibility requests (see Section 3.2). We decided to test the efficiency of each structure for ray-casting. Therefore, we chose 100,000 random pairs of surfaces in each scene, and measured the total time needed to search for a possible occlusion between each pair.

We also ran a separate set of experiments to evaluate the efficiency of the cluster hierarchy when using approximated volumetric visibility¹². This alternative to classical exact visibility computation accelerate rendering times when precise shadow area determination is not needed. Instead of the previously defined scenes, we used a model of several trees (Figure 5) representative of some applications where an average representation of light transfers may be sufficient. It is also a good test case for the volumetric visibility algorithm: the set of leaves being a good approximation of a turbid media. We calculated the images using each clustering algorithm and then compared the difference with a reference image obtained using exact surface visibility.

5. Results

In this section, we present a number of observations drawn from the tests explained previously. We begin with an analysis of the quality aspects of the simulations, looking at the evolution of our error measure with the user-defined error tolerance. We then consider in more detail the capacity of different clustering techniques to assist visibility calculations, with a particular emphasis on computational efficiency.

We ran all algorithms on a Silicon Graphics computer (MIPS R10000 at 250 MHZ) with 4GB of memory.

5.1. Evolution of Solution Error

Recall from Section 3 that a major demand on the clustering mechanism (together with the chosen refinement strategy) is that the evolution of computation time and solution quality should be regular and monotonic as the user changes the error tolerance (Figure 3). Ideally, this would result in a very regular and monotonic time/error curve. Such curves are presented in Figure 2 for the four test scenes.

Looking at the four Time-error curves, we see two very different types of behaviors: for the OFFICE and VRLAB scenes, all curves are regular and fairly monotonic, whereas for AIRCRAFT and CAR the variation of error is more erratic. Indeed, looking at a plot of error as a function of the

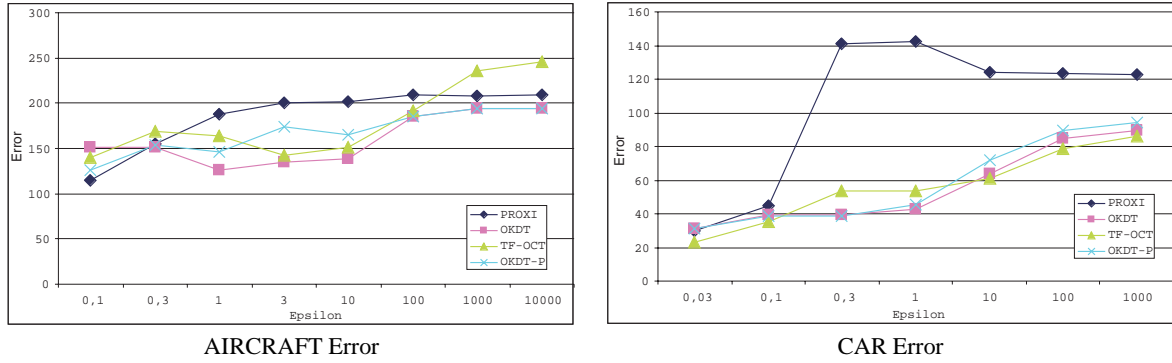


Figure 3: Error curves as function of user-supplied tolerance

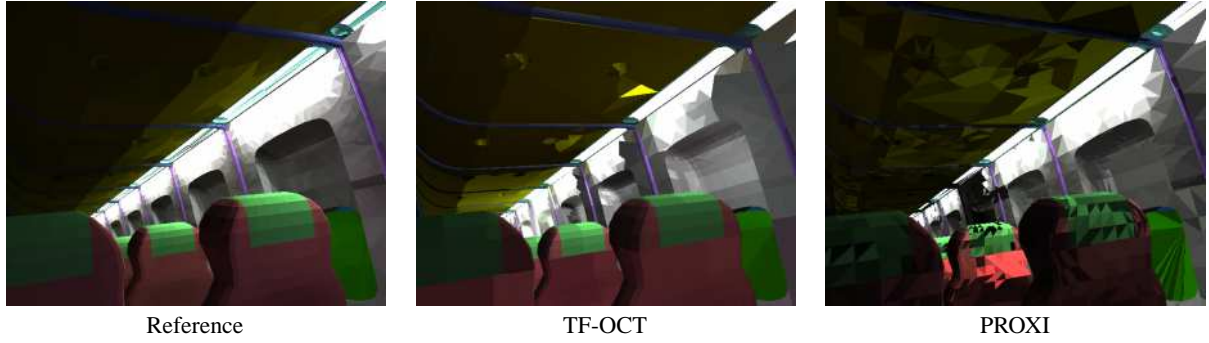


Figure 4: Example solutions exhibiting different (visual) forms of error (see also color plate).

user-supplied error tolerance, we find that the error in the solution does not always decrease as we reduce the tolerance. Because of limited space, we present only the error plots for AIRCRAFT and CAR (Figure 3); the corresponding curves for OFFICE and VRLAB are monotonically decreasing.

Why can the error increase when we decrease our tolerance? this unfriendly behavior occurs when links refined as a result of the error tolerance change produce a less accurate representation of radiosity exchanges. This is largely a question of refinement criteria, but is also influenced by the clustering strategy, as well as the distribution of objects in the scene. We observe that our scenes can be classified into two types: AIRCRAFT and CAR consist of many small polygons, because of a previous tessellation of the objects. VRLAB and OFFICE, on the other hand, contain objects of varying size, from large walls to small furniture components. Based on our experience and the results of the above experiments, we observe that clustering algorithms have more difficulty with the first type of scene (“polygon soup”). This is especially true of AIRCRAFT because 3D space is very densely populated, resulting in many interactions between clusters which are not separated by a significant distance. These interactions also present a particular challenge to the refinement criterion.

As an illustration of the typical errors created in an approximate solution for such scenes, consider the images in Figure 4. The two approximate solutions have a similar error under our measure, yet they appear quite different visually. The solution using TF-OCT clusters exhibits marked radiosity variations along axis-aligned boundaries, corresponding to the octree cells; on the other hand, the solution using PROXI shows a high variance of radiosity and a speckle pattern, due to the fact that nearby small objects can belong to many different and overlapping clusters, with markedly different radiosities.

5.2. Performance and Visibility Calculation with Clusters

We now consider the performance behavior of the clustering strategies in terms of construction time and as auxiliary structures for visibility calculations.

Construction Time

As explained in Section 2.2, bottom-up construction is a very expensive process since it amounts to an optimization procedure.

Observed computation times for the construction of the

cluster hierarchies support this prediction: the PROXI clustering strategy construction is always much slower than OKDT and TF-OCT, which operate top-down on simple recursive subdivision schemes. These two techniques always take less than 1% of the PROXI time. Interestingly, OKDT-P takes between 20% and 80% of the PROXI time, depending on the distribution of objects in the scene.

Approximate Visibility Calculations

The acceleration of visibility calculations using *equivalent extinction properties* of the clusters has been proposed by Sillion¹². In this approach, the transmittance factor between two points in the scene is evaluated by considering the entire segment between the points, and its intersections with all clusters, then combining the corresponding attenuation values, in an analogy with partially absorbing volumes. This method, also adopted by Christensen *et al.*⁴, is often faster than true ray casting using the surfaces, because of the smaller number of clusters and the ease of computation of ray-cluster intersections.

We computed approximate visibility using clusters in a scene dominated by direct lighting (from the sun), as shown in Figure 5. In this case, the shadow pattern on the floor is essentially an “X-ray image” of the cluster hierarchy, which greatly helps in the comprehension of the cluster distribution.

We observe a clear hierarchy in terms of shadow quality, in the order PROXI (best, notice high quality of trunk shadows), OKDT-P, OKDT, TF-OCT (poorest). Please see images in color section. This is consistent with the intuitive notion that PROXI starts from the objects and build clusters bottom-up, thereby building clusters that are very tight around the objects.

OKDT (and TF-OCT even more so) exhibits some incorrect shadows of large, blocky clusters, due to the constraints in the spatial subdivision. In this respect, OKDT-P effectively improves on OKDT, with a better fit around the objects and more precise shadows.

Ray Casting Acceleration

Interestingly, the computation times shown in Figure 5 increase with the quality of the shadows. This is consistent with the general observation that hierarchical structures with lower branching factors have more hierarchical levels and perform better for ray tracing acceleration.

This reasoning is supported by the analysis of cluster statistics on our test scenes. Figure 6 shows the variation of the following three quantities with the clustering technique, for each test scene:

- total number of clusters
- average number of child elements per cluster

- performance of ray tracing acceleration. This is measured by shooting a large number (100,000) of random rays through the scene and computing ray-surface intersections.

We first observe an obvious inverse correlation between the total number of clusters and the average number of children. In addition, TF-OCT has the largest branching factor for the cluster hierarchy because of its octal subdivision scheme. OKDT also has a fairly high number of children on average, because its construction mechanism offers no way to control this branching factor. Conversely, PROXI has a built-in mechanism limiting the number of children of any given cluster (this operates by grouping objects into overlapping sub-clusters). Therefore it exhibits the lowest branching factor. OKDT-P is intermediate, as expected, because by construction it avoids clusters with many children, handing them to the PROXI clusterizer. Still it avoids the overall large number of clusters of PROXI. A consistent best performer in terms of acceleration is therefore OKDT-P.

Finally, we note the conflicting nature of the two desires for (a) efficient ray tracing acceleration and (b) suitability for radiosity calculations (compare Figure 6 and Figure 2).

6. Conclusions and Future Work

We have presented an experimental analysis of clustering algorithms for hierarchical radiosity. A taxonomy of clustering algorithms was proposed, followed by a set of requirements for a good clustering algorithm. Guided by these requirements, we developed an experimental methodology based on an image-space quality measure. Extensive tests were run on scenes for the most part developed in real-world application contexts.

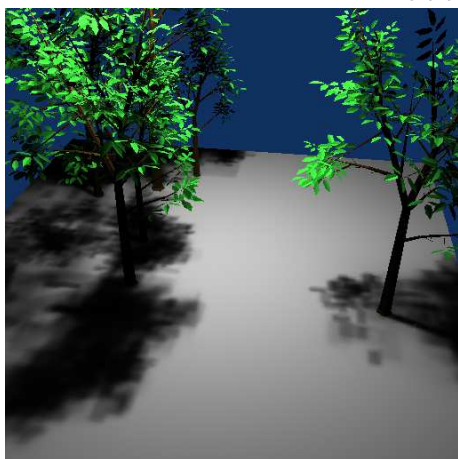
Drawing concrete conclusions from experimental tests such as those performed here is always a delicate task. Nonetheless, there are certain elements which we believe are clear enough to be singled out:

Clustering works well in many cases: in particular, for scenes containing objects of different sizes and a sufficient number of large initial surfaces (walls, floors etc.), all the clustering algorithms tested appear to perform well. The Time-error graphs are smooth and monotonic for these cases, presenting the user with an intuitive time-quality tradeoff.

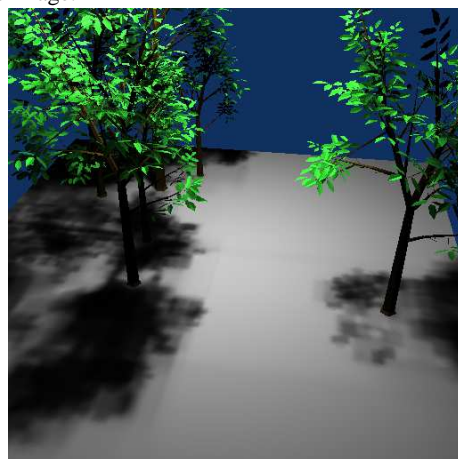
For scenes containing many small objects (“polygon soup”), existing clustering algorithms are less well-behaved. In particular, more time spent computing a solution does not always result in higher quality (see Section 5.1). This is even more troublesome since the scenes in question are typical of industrial “real-world” models, which are often the result of a fine tessellation of some unspecified and unrecoverable modeling format. It is clear that a new approach is required to treat such models, in order to build a hierarchy that follows the definition of objects. Reconstruction of individual



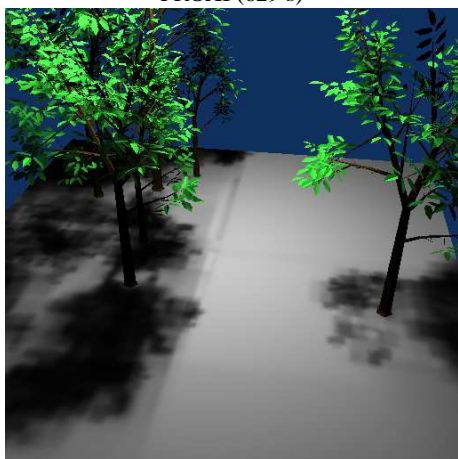
Reference image.



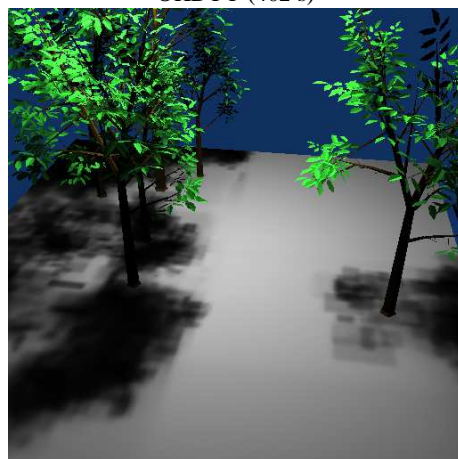
PROXI (629 s)



OKDT-P (402 s)



OKDT (281 s)



TF-OCT (166 s)

Figure 5: Influence of the clustering method on approximate visibility calculations (see also color plate).

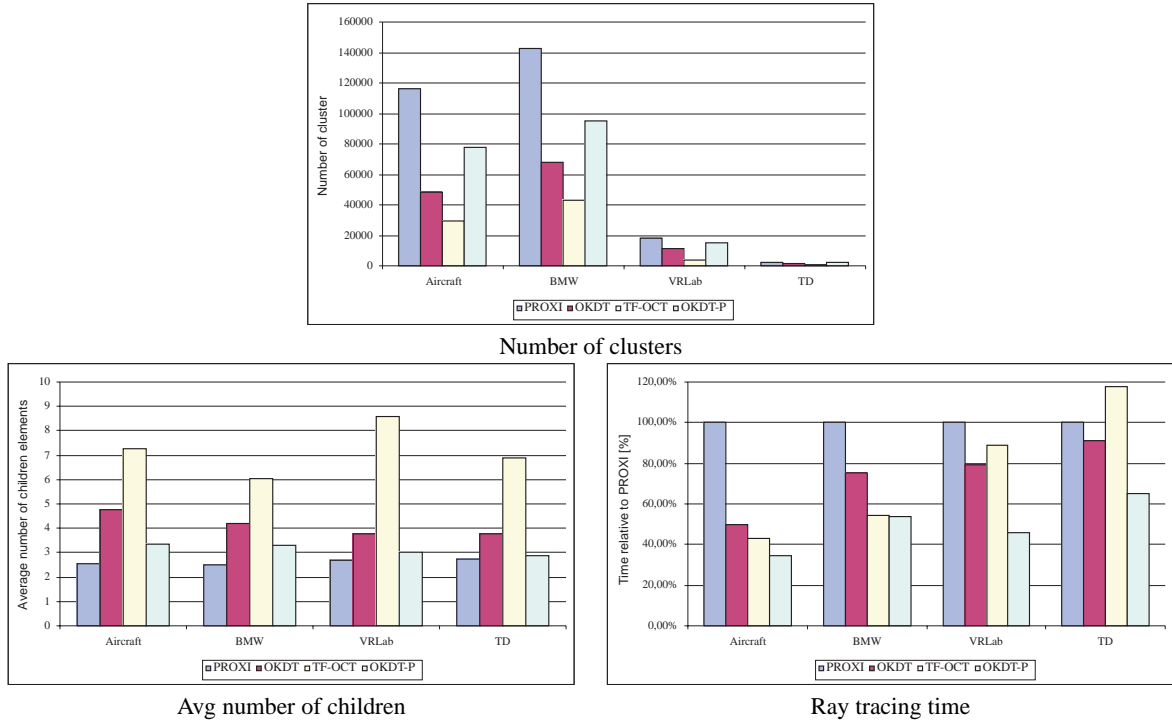


Figure 6: Statistics on the cluster hierarchies

objects is possible based on connectivity and surface properties, and multi-resolution object models could be developed to provide a hierarchy of representations.

Of the clustering algorithms tested, the hierarchical bounding volumes (PROXI) approach seems to have the most predictable behavior in almost all cases. In particular, the Time-error curve is almost always monotonic and smooth. In addition, due to the nature of construction, it fits objects more tightly, which is a desirable property for clustering. However the overhead for PROXI is significant if not prohibitive in most cases: a much longer construction time (compared to all others tested), longer solution times, and in some cases a higher absolute error for very approximate simulations.

In terms of ray-casting cost, it appears that OKDT-P is the most rapid structure. Thus, if ray-casting cost is an issue (for example in interactive updates where efficiency is paramount), this may be the clustering algorithm of choice.

Finally, in terms of the quality of approximate visibility, a clear hierarchy was found with the following order PROXI (best), OKDT-P, OKDT, TF-OCT (poorest).

We hope these first conclusions will be useful to researchers and developers who wish to use clustering for hierarchical radiosity. Clearly, much remains to be done in this domain.

The error metric adopted for our tests is one of many possibilities. It is evident that different applications have different notions of error (for example in lighting design where an exact measure of energy prevails over image quality), and these different requirements will lead to different choices for clustering. These issues must be further investigated.

The initial, first-order, classification of scene “type” with respect to their behavior in the context of a clustering algorithm is an interesting avenue of research. Ideally, extensive experimentation would allow us to determine which algorithm is suitable for a given scene. This is however a very ambitious task, so even initial results would be worthy of further research.

The development of a novel clustering approach treating scenes containing many small unrelated polygons is also an interesting challenge.

To conclude, we believe that our analysis has shown the utility of clustering for many cases, identified some weaknesses of current algorithms and identified certain important properties of each algorithm with respect to their suitability for different tasks.

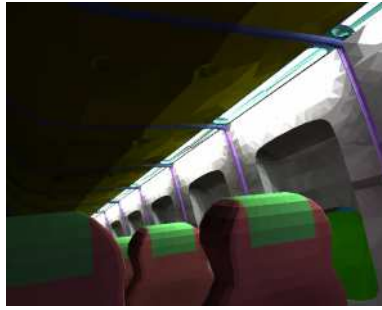
7. Acknowledgments

This work was supported by the European Union under the Esprit Long Term Research contract # 24944 “ARCADE:

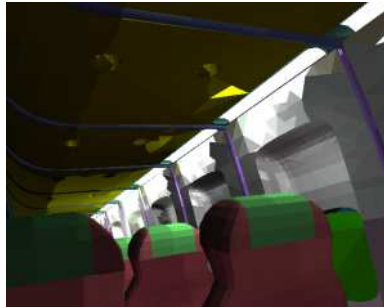
making radiosity usable”¹. Scenes were kindly provided by Lightwork Design Ltd, Fraunhofer Institute for Computer Graphics and BMW. iMAGIS is a joint research project of CNRS/INRIA/UJF/INPG.

References

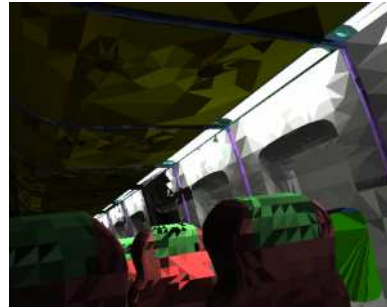
1. ARCADE: making radiosity usable. European Union. Esprit Long Term Research project #24944. <http://www-imagis.imag.fr/ARCADE>.
2. Frédéric Cazals, George Drettakis, and Claude Puech. Filtering, clustering and hierarchy construction: a new solution for ray tracing very complex environments. In F. Post and M. Göbel, editors, *Computer Graphics Forum (Proc. of Eurographics '95)*, volume 15, Maasticht, the Netherlands, September 1995.
3. Frédéric Cazals and Claude Puech. Bucket-like space partitioning data structures with applications to ray-tracing. In *Proceedings of the 13th International Annual Symposium on Computational Geometry (SCG-97)*, pages 11–20, New York, June4–6 1997. ACM Press.
4. Per Henrik Christensen, Dani Lischinski, Eric J. Stollnitz, and David H. Salesin. Clustering for Glossy Global Illumination. *ACM Transactions on Graphics*, 16(1):3–33, January 1997.
5. George Drettakis and François Sillion. Interactive update of global illumination using a line-space hierarchy. In *Computer Graphics Proceedings, Annual Conference Series: SIGGRAPH '97* (Los Angeles, CA), pages 57–64. ACM SIGGRAPH, New York, August 1997.
6. Mark D. Fairchild. *Color Appearance Models*, chapter 3, pages 90–93. Addison Wesley, 1998.
7. Simon Gibson and Roger J. Hubbard. Efficient hierarchical refinement and clustering for radiosity in complex environments. *Computer Graphics Forum*, 15(5):297–310, December 1996.
8. Jeffrey Goldsmith and John Salmon. Automatic creation of object hierarchies for ray tracing. *IEEE Computer Graphics and Applications*, 7(5):14–20, May 1987.
9. Pat Hanrahan, David Saltzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. *Computer Graphics*, 25(4):197–206, August 1991. Proceedings SIGGRAPH '91 in Las Vegas (USA).
10. Krzysztof S. Klimaszewski and Thomas W. Sederberg. Faster ray tracing using adaptive grids. *IEEE Computer Graphics and Applications*, 17(1):42–51, January 1997.
11. François Sillion. Clustering and volume scattering for hierarchical radiosity calculations. In G. Sakas, P. Shirley, and S. Müller, editors, *Photorealistic Rendering Techniques*. Springer Verlag, 1995. Proceedings of Fifth Eurographics Workshop on Rendering (Darmstadt, Germany, June 1994).
12. François Sillion. A unified hierarchical algorithm for global illumination with scattering volumes and object clusters. *IEEE Transactions on Visualization and Computer Graphics*, 1(3), September 1995. (a preliminary version appeared in the fifth Eurographics workshop on rendering, Darmstadt, Germany, June 1994).
13. François Sillion and George Drettakis. Feature-based control of visibility error: A multiresolution clustering algorithm for global illumination. In *Computer Graphics Proceedings, Annual Conference Series: SIGGRAPH '95* (Los Angeles, CA), pages 145–152. ACM SIGGRAPH, New York, August 1995.
14. Brian Smits, James Arvo, and Donald P. Greenberg. A clustering algorithm for radiosity in complex environments. In *Computer Graphics Proceedings, Annual Conference Series: SIGGRAPH '94* (Orlando, FL), pages 435–442. ACM SIGGRAPH, New York, July 1994.
15. Andrew J. Willmott and Paul S. Heckbert. An empirical comparison of progressive and wavelet radiosity. In Julie Dorsey and Philipp Slusallek, editors, *Eurographics Rendering Workshop 1997*, pages 175–186, New York City, NY, June 1997. Eurographics, Springer Wien. ISBN 3-211-83001-4.



Reference

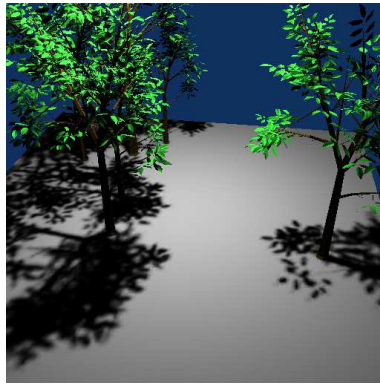


TF-OCT

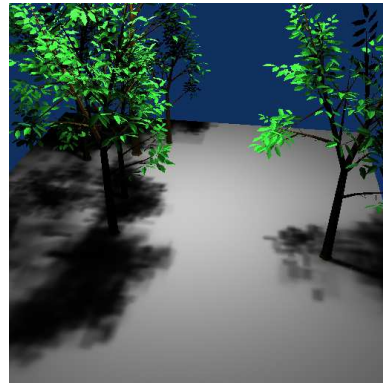


PROXI

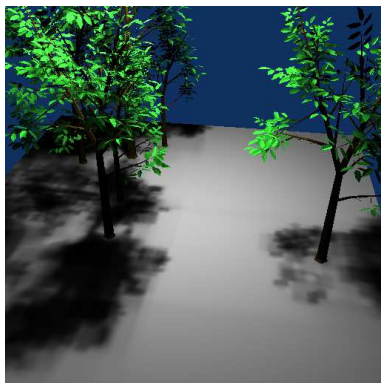
Figure 4: Example solutions exhibiting different (visual) forms of error.



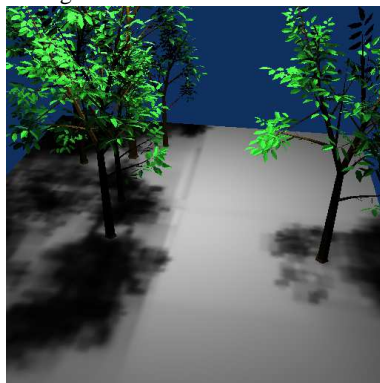
Reference image.



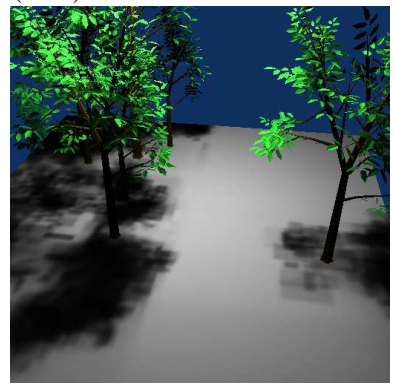
PROXI (629 s)



OKDT-P (402 s)



OKDT (281 s)



TF-OCT (166 s)

Figure 5: Influence of the clustering method on approximate visibility calculations.

3.5.7 A Clustering Algorithm for Radiance Calculation in General Environments (EGRW'95)

Auteurs : François Sillion, George Drettakis et Cyril Soler

Actes : 6th Eurographics Workshop on Rendering

Date : juin 1995

A Clustering Algorithm for Radiance Calculation In General Environments

François Sillion, George Drettakis*, Cyril Soler

iMAGIS **

Abstract: This paper introduces an efficient hierarchical algorithm capable of simulating light transfer for complex scenes containing non-diffuse surfaces. The algorithm stems from a new formulation of hierarchical energy exchanges between object clusters, based on the explicit representation of directional radiometric distributions. This approach permits the simplified evaluation of energy transfers and error bounds between clusters. Representation and storage issues are central to this type of algorithm: we discuss the different choices for representing directional distributions, and the choice between explicit storage or immediate propagation of directional information in the hierarchy. The framework presented is well suited to a multi-resolution representation, which may in turn significantly alleviate the storage problems. Results from an implementation are presented, indicating the feasibility of the approach and its capacity to treat complex scenes.

1 Introduction

The hierarchical radiosity algorithm permits the efficient computation of radiosity solution within well-understood error-bounds. Its main limitation is the “initial-linking” step, which for scenes of diffuse polygons adds a quadratic computational cost. As a consequence the algorithm is unusable for large environments. Recently presented clustering algorithms for hierarchical solutions [10, 6], avoid the quadratic cost by first clustering the environment and then refining the clusters.

Nonetheless, little work has been performed for non-diffuse environments. Two-pass algorithms [9, 11] and a general solution using directional representations [7] have treated more general environments in the context of progressive refinement radiosity. A hierarchical solution to general environments has also been proposed [1], but in the case of that algorithm the initial linking cost becomes $\mathcal{O}(n^3)$ in the number of initial polygons, making it unusable even for moderately complex scenes.

The processing of complex environments with general reflectors is a necessity, since almost all interesting scenes contain at least some percentage of non-diffuse materials. In this paper we present a framework which provides the necessary machinery for the treatment of non-diffuse environments in the context of a hierarchical clustering algorithm. This framework is a natural extension of previous clustering methods since, as noted before [6], clusters do not behave as isotropic scatterers, even if composed solely of diffuse surfaces. It is based on the representation of radiant intensity by directional distribution functions, and extends the spirit presented in [7] to hierarchical clustering. The result is the first efficient hierarchical algorithm permitting the efficient of complex, non-diffuse environments. In addition, this representation affords a smooth transition between the representation at the level of (non-diffuse and diffuse) surfaces to the

* The second author performed this research with an ERCIM fellowship (funded by the EU Commission), partially at UPC, Barcelona, Spain and GMD, St. Augustin, Germany.

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

level of clusters. Finally, the framework opens the way to an efficient multi-resolution representation of light properties for clusters.

In contrast with previous clustering approaches our new method is based on the storage of directional properties with the clusters. This approach requires the reconsideration of some of the quantities previously used since we are now dealing with directional energy exchanges between clusters. In Section 2 we characterise the directional properties of clusters which are used in our solution. In Section 3 we introduce the new algorithm which is based on the directional representation, in Section 4 we discuss the issues pertaining to possible approaches to storing directional distributions and in Section 5 we present some implementation issues and some first results. We conclude in Section 6 with a discussion of limitations and the directions for future research.

2 Characterization of directional energy transfer

As outlined above, we will be treating the light leaving and impinging on clusters as a function of direction. In particular we want to be able to store and manipulate directional functions to characterize the radiant behaviour of a cluster. In this section we discuss the physical quantities used, their representation and their relation to traditional radiosity variables.

For the most general discussion of directional light transfer, we consider light leaving the cluster, light impinging on the cluster, and light passing through the cluster. We also introduce a particular directional function useful for the expression of energy exchanges with distributions. In the remainder of this paper we will denote a direction in space by a unit vector, with the convention that \vec{u} represents an outgoing direction and \vec{v} an incident direction (See Fig. 1).

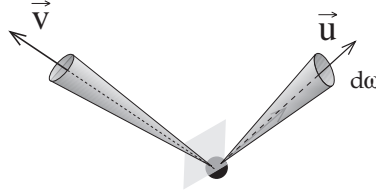


Fig. 1. Notations used for directional functions.

2.1 Outgoing Light

For the description of light leaving the cluster, we use *radiant intensity*, I , representing power per unit solid angle. At a point x on a surface, radiant intensity is related to radiance by the following formula:

$$dI(x, \vec{u}) = L(x, \vec{u}) dA (\vec{n} \cdot \vec{u}), \quad (1)$$

where \vec{n} is the surface normal and dA is the differential surface area around point x . In the case of a diffuse surface with radiosity B , radiant intensity is thus given by

$$dI(x, \vec{u}) = \frac{B}{\pi} dA (\vec{n} \cdot \vec{u}).$$

2.2 Incoming Light

For light arriving on a cluster, we use the standard (incoming) *radiance* quantity, defined as the amount of power received per unit area perpendicular to the direction of incidence and per unit solid angle.

With this definition, if the distribution of incident radiance at point x is $E(x, \vec{v})$, the incoming flux density per unit solid angle on a surface placed at x with normal direction \vec{n} is

$$E_s(x, \vec{v}) = E(x, \vec{v}) (\vec{v} \cdot \vec{n}) \quad (2)$$

2.3 The Tangent-sphere function

In Equations 1 and 2 above, the scalar products must be understood as being zero if the surface is not facing the right direction. For notational convenience we represent this extended scalar product as a function of \vec{u} . Let us define the *tangent-sphere* function $T_n(\vec{u})$ for a direction \vec{u} by

$$T_n(\vec{u}) = \begin{cases} \vec{u} \cdot \vec{n} & \text{if } \vec{u} \cdot \vec{n} \geq 0 \\ 0 & \text{Otherwise} \end{cases} \quad (3)$$

As shown in Fig. 2 the surface given in spherical coordinates by $r = T_n(\vec{u})$ has the shape of a sphere tangent to the plane orthogonal to \vec{n}

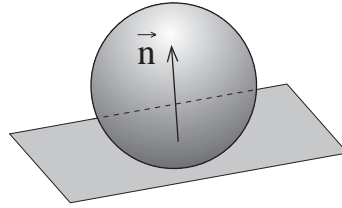


Fig. 2. Tangent-Sphere function.

Using this function, Equations 1 and 2 can be rewritten as

$$I(x, \vec{u}) = L(x, \vec{u}) \int dA T_n(\vec{u}) \quad (4)$$

and

$$E_s(x, \vec{v}) = E(x, \vec{v}) \int T_n(\vec{v}) \quad (5)$$

2.4 Extinction properties

The transmission properties of object clusters can be discussed using a fruitful analogy with semi-transparent volumes with optical extinction properties. Previous work along this line has proposed to compute equivalent isotropic extinction coefficients for object clusters based on the total area they contain [6] ($\kappa = A/4V$, where A is the total surface area of the objects in the cluster and V is its volume).

In the general approach presented here we lift the isotropic assumption and compute for each cluster a directional extinction coefficient, used to evaluate the attenuation of a light beam traversing the cluster in a given direction. The total projected area in a given direction can be precomputed and stored with each cluster. It is given by the following sum over the surfaces contained in the cluster:

$$\mathcal{A}(\vec{\nu}) = \sum_i A_i T_{-i}(\vec{\nu}) v \quad (6)$$

A directional extinction coefficient is then obtained with the following formula:

$$\kappa(\vec{\nu}) = \frac{\mathcal{A}(\vec{\nu}) v}{V} \quad (7)$$

$\kappa(\vec{\nu})$ is used as in [6] to compute approximate transmission through a cluster, as it represents the rate of attenuation per unit length in the direction of interest. Note that the factor of 4 from the isotropic formula is no longer present, since it accounted for the averaging over all directions. Plate 1 (see Appendix) shows results obtained with directional extinction.

2.5 Light Scattering

For now we only consider the transformation of incoming light into outgoing light to take place at surfaces. We assume that a surface oriented in direction \vec{n} is placed at the origin. The surface is small enough for all distributions to be safely assumed constant across its surface. One difficulty in expressing the general light scattering equation is that surface scattering is best described in a coordinate system that is local to the surface. Let us define a linear transformation M_{-n} such that $\vec{u} = M_{-n} \vec{u}'$ is the unit vector representing the direction of \vec{u} in a coordinate system attached to the surface. As shown in Fig. 3, both vectors are aligned, they simply have different coordinates because they are expressed in different frames of reference.

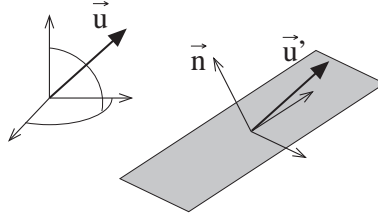


Fig. 3. Notations for the scattering equation.

Surface scattering In this paragraph we express all directions in the surface coordinate system. The radiance leaving the surface in a direction \vec{u} is given by

$$L(\vec{u}) = \int_{\vec{v} \in \Omega_+} E_s(\vec{v}) p_{bd}(\vec{u}, \vec{v}) d\omega_{-v} \quad (8)$$

where $E_s(\vec{v}) d\omega_{-v}$ is the incident flux density on the surface from the differential solid angle $d\omega_{-v}$ around direction \vec{v} . Ω_+ is the upper hemisphere (above the surface).

Expressing radiant intensity from incident radiance We now wish to express the scattering equation using the directional quantities defined above, and in a general (world) coordinate system, not tied to any particular surface. This simply requires a number of coordinate transformations using \mathbf{M}^{-1} . Combining Equations 4 and 8, we can express the radiant intensity leaving a surface in direction \vec{v} as

$$I(\vec{v}) = A T(\vec{n}) \mathbf{L}(\vec{v}) u \quad (9)$$

$$= A T(\vec{n}) \mathbf{L}(\mathbf{M}^{-1} \vec{v}) u \quad (10)$$

$$= A T(\vec{n}) u \int_{\vec{v}' \in \Omega_+} E(\vec{v}') p_{bd}(\mathbf{M}^{-1} \vec{v}') d\omega_{\vec{v}'} \quad (11)$$

Using Equation 5 and changing the integration variable to be a unit vector in the hemisphere above the oriented surface, $\vec{v} = \mathbf{M}^{-1} \vec{v}'$, we have

$$I(\vec{v}) = A T(\vec{n}) u \int_{\vec{v}' \in \mathbf{M}^{-1} \Omega_+} E(\vec{v}') \mathbf{F}(\vec{n}) p_{bd}(\mathbf{M}^{-1} \vec{v}') d\omega_{\vec{v}'} \quad (12)$$

Ideal diffuse case For ideal diffuse surfaces, the BRDF is a constant, and Equation 12 reduces to

$$I(\vec{v}) = A T(\vec{n}) \frac{\rho_d}{\pi} \int_{\vec{v}' \in \mathbf{M}^{-1} \Omega_+} E(\vec{v}') \mathbf{F}(\vec{n}) d\omega_{\vec{v}'} \quad (13)$$

The integral in Equation 13 represents the total incident flux density (irradiance) on the surface.

3 A Cluster-Based Illumination Algorithm for General Scenes

Existing radiosity clustering algorithms can be adapted to work with directional information, with little modification as described in this section. We assume here that the reader is familiar with hierarchical radiosity and clustering algorithms [3, 10, 6]. In these methods, a hierarchical subdivision structure of 3D space is used to collect surfaces into clusters. The main idea of the new general clustering algorithm is to associate to each cluster or surface a number of directional distributions representing its radiant properties. The scattering equation (12) must then be evaluated for each surface, using the appropriate incident radiance and radiant intensity distributions.

3.1 Form factor

Since we are using a radiant intensity distribution on the emitter, the estimation of energy transfer between a pair of objects is slightly different than with usual radiosity. Transfer estimates are needed in two stages of a hierarchical radiosity algorithm. First, a bound on the total energy transfer between two objects (or clusters) must be computed during the link refinement stage. Second, the actual energy transfer takes place in a *gathering* stage, where the incoming energy is computed across each link.

The notion of “form factor” used in our algorithm is redefined from purely algorithmic considerations: the form factor associated to each link is the scalar quantity by which the radiant intensity value of an emitter must be multiplied to obtain the incident irradiance (power per unit area perpendicular to the direction of propagation) on the receiver.

This quantity is simply derived from the expression of radiant intensity and irradiance, and is

$$F_{pq} = \int_p \int_q \frac{1}{r^2} d\mathbf{l} d\mathbf{l}' \quad (14)$$

3.2 Link refinement

For the purpose of making a refinement decision, a hierarchical subdivision criterion must be defined. Our preliminary implementation uses an estimate of the energy transferred between two objects q and p (objects can be surfaces or clusters [6]). To obtain this estimate we select two sample points in p and q , yielding a direction \mathbf{u} . Multiplying $I_q(\mathbf{u})$ with the “form-factor” F_{pq} we obtain an incident irradiance contribution on p from direction \mathbf{u} denoted by \mathcal{E}_{pq} . Note that, in a manner similar to Lischinski *et al.*'s work [5], an actual bound on this transfer can be computed, provided we store not only the average radiant intensity but also the maximum radiant intensity for each object. To obtain an energy value from incident irradiance requires a multiplication by the total projected area of the cluster's contents in direction \mathbf{u} , $\mathcal{A}(\mathbf{u})$, introduced in Section 2.4. Our estimate of the energy contribution of the link between q and p is thus

$$P = \mathcal{A}_p(\mathbf{u}) \mathcal{E}_{pq} \quad (15)$$

$$= \mathcal{A}_p(-\mathbf{u}) I_q(\mathbf{u}) F_{pq} \quad (16)$$

Note that the previous discussion ignores intra-cluster visibility issues. These are not treated in this paper, although recent work shows that it is possible to integrate their effect with reasonable cost [8]. It is interesting to note the benefit of storing the radiant intensity in the form of a directional distribution, since the transfer estimate does not require the interrogation of the cluster contents. This represents a potential gain over previous hierarchical clustering algorithms [6, 10].

3.3 Gather

Due to the change in quantities used to represent and store light, the traditional process of gathering across linked clusters or surfaces must be appropriately modified.

One of the most important choices to be made when representing directional properties in a hierarchy of clusters, is which properties to store explicitly at all levels in the hierarchy and which to store implicitly by pushing them down to the level at which additional storage cost is incurred. In particular the efficient treatment of incident energy contributions requires some attention. We consider here two alternatives, and discuss their relative merits.

Storing an incident radiance distribution The simplest directional clustering algorithm is probably one where incoming radiance is stored with each cluster, together with (outgoing) radiant intensity. The main advantage of this approach is that the amount of work performed for each link in the gathering phase is fixed, and does not depend on the clusters' complexity. This “constant-time” transfer computation, combined with the linear number of links with respect to the total number of surfaces [3, 10], results in a clustering algorithm with linear asymptotic complexity.

Unfortunately, storing incoming radiance is difficult and expensive. First, in the context of our framework we want to use a continuous, directional function representation. Incoming radiance is inherently discontinuous, as for instance the contribution of a

given source is non-zero only for directions reaching the source. This difficulty can be eliminated by estimating a continuous approximation to each source's contribution to the incident radiance.

Consider again the transfer from q to p . Since our refinement criterion has established the link at this level, it is reasonable to assume that the transfer is well represented by a point-to-point calculation. An estimation of the error incurred by this assumption must evidently be undertaken in the future. The incident *irradiance* on p is obtained as explained in Section 3.2. This irradiance can be spread across the solid angle subtended by q , using a simple parametric filter in the shape of a peak. We are investigating the use of rotated $\cos^n(\theta)$ distributions as convolution filters. Clearly however this operation involves a significant additional computational cost.

In addition, explicit storage implies the need for an expensive convolution operation when pushing the incoming radiance down the hierarchy of clusters. At the transition from clusters to surfaces the conversion from incident radiance to radiant intensity must be performed, as shown in Equations 12 for the general case and 13 for the diffuse case. Again this implies significant additional computation.

Immediate propagation of incoming contributions An alternative to storage of incoming radiance is to explicitly push incoming light down the hierarchy at each gather operation. To perform this we no longer consider radiance, but *irradiance*, computed as in Section 3.2. This quantity, accompanied by the incoming direction \vec{v} is pushed down the cluster hierarchy by simple addition. This irradiance is the term $E(\vec{v})d\omega_{\vec{v}}$ in Equation 12. At the surface level we need only evaluate Equation 12, replacing the integral by an “impulse” from direction \vec{v} with the surface irradiance value $\mathcal{E}_{pq}T_{\vec{v}}(\vec{v})$. This surface irradiance is used to scale the surface's BRDF, which reduces to a constant for diffuse surfaces.

3.4 Push/Pull

In our implementation we have chosen the option of immediate pushing of incoming radiance as opposed to storing the quantity as a directional function. Thus the traditional Push-Pull operation only needs to perform the “Pull” portion, since the “Push” occurs at the gathering stage. Since radiant intensity is a power quantity, the radiant intensity of a cluster is obtained from that of its sub-clusters by simple summation. The result is a combined directional function representing the total radiant intensity of the cluster.

4 Representation of Directional Distributions

Several storage schemes have been investigated in the context of simulating non-diffuse radiant exchanges. A major difficulty in selecting a representation is to achieve the best possible balance between the storage cost of each option and its suitability given a number of algorithmic requirements. Any finite representation of directional functions is based on the selection of a number of basis functions. The representation of a distribution then consists of its coordinate vector in the chosen basis.

Previous algorithms employ for example constant basis functions defined over the cells of a “global cube” [4], or spherical harmonics basis functions up to a prescribed order [2, 7]. The global cube approach has the advantage of simplicity, first because it is very easy to manipulate, but also because function products can be evaluated easily (since the basis functions have non-overlapping support). However it is inherently a discontinuous representation, prone to disturbing rendering artifacts.

Spherical harmonics, on the other hand, always produce continuous functions. But they are non-zero over the entire hemisphere, making the computation of function products much more expensive.

4.1 Spherical Harmonics

In our implementation we use spherical harmonics basis functions. These form an orthogonal basis of the set of distributions on the unit sphere. This infinite collection of basis functions is typically denoted by $Y_{l,m}(\theta, \phi)$ where $0 \leq l < \infty$ and $-l \leq m \leq l$. In direct analogy with a Fourier series in one dimension, any square-integrable function, $f(\theta, \phi)$, can be expressed in this basis, with a set of scalar coefficients $C_{l,m}$.

An approximate representation of a directional function is obtained by storing only the first few coefficients of this decomposition, up to a given maximum level. BRDFs can be encoded by such vectors of coefficients for use in a radiosity simulation [7].

Representation of diffuse surfaces using Tangent-Sphere functions In the diffuse case, all radiant intensity distributions are combinations of oriented Tangent-sphere functions (see Equation 13).

The decomposition of $T_{\vec{n}}(\vec{\omega})$ into spherical harmonics can be computed for a given direction \vec{n} . The simple shape of this function allows a very good approximation with only 9 coefficients ($l \leq 2$). The coefficients of this decomposition are thus functions of \vec{n} and they can themselves be decomposed using spherical harmonics of \vec{n} . This double decomposition was already used by Westin *et al.* to represent anisotropic BRDFs [12]. In our case it is stored in a data file, since the Tangent-sphere function is always the same.

The spherical harmonics representation of $T_{\vec{n}}(\vec{\omega})$ is obtained by evaluating the value of each coefficient for the direction \vec{n} . Since this only depends on the surface orientation, it is only performed once in the program, and is then stored with the polygon (and thus shared by all hierarchical elements on the surface).

Computation of the scattering integral If incident radiance is stored with the clusters, the integral in Equation 12 must be evaluated at each cluster-surface interface. The convolution of incident radiance and the BRDF is quite costly to compute, especially since function products are difficult to express with spherical harmonics coefficients. We are currently investigating an efficient algorithm to compute such convolutions, based on the use of recurrence relations, and the observation that the integral of a function is represented by its ($l = 0, m = 0$) coefficient.

5 Implementation and First Results

We have implemented the representation of radiant intensity and the equivalent push/pull operation in our testbed clustering system. As described above we have used spherical harmonics for the representation of directional functions. Our implementation is still preliminary in the sense that for now a limited number of orientations are allowed for non-diffuse surfaces. The color plates in the appendix demonstrate the versatility and high potential of the method.

5.1 Directional properties for clusters of diffuse surfaces

We first consider the anisotropic behaviour of clusters containing only diffuse surfaces. Plate 2 shows an example with over 6,000 surfaces. The ceiling receives no primary illumination, and is only illuminated by light reflected by the cluster. We see that the pattern of light on the ceiling is displaced with respect to the vertical direction.

As an indication to the reader of the relative cost of the storage of directional radiant intensity, comparisons are made to images generated using the algorithms presented in [6, 8], in which directional functions are not used. Since the refinement criteria are no

longer the same, we set our subdivision threshold so that the two executions result in similar number of links refined for two iterations. The following table gives the computation time (in seconds) and memory cost (in Mb) for directional (dir) and traditional (trad) clustering algorithms.

Name	Polygons	Time (dir)	Time (trad)	Mem. (dir)	Mem. (trad)
<i>Simple</i>	13	29.6	24.0	8.5	4.7
<i>Cubes</i>	6000	140.0	46.1	13.4	6.9

We see that the computation time for the directional approach is between 20% to 3 times higher. This can be explained by the additional expense in combining the tangent sphere functions and the directional representations of radiant intensity. The comparisons are given only as an indication; in the resulting images for the *Cubes* scene the directional algorithm obtains a much higher quality representation of the secondary illumination on the ceiling (see Plate 2).

The memory requirements for the directional representation are approximately twice that of the traditional clustering approach. These numbers are more meaningful since they are not affected as much by the different refinement criteria. If the growth factor is close to the indicated factor of two, this implies that memory utilization does not pose a major problem for our approach, since even very complex scenes will not require unmanageable amounts of memory.

5.2 Results for general reflectors

Plates 3 and 4 show simulations performed with a cluster of glossy surfaces. Both directional reflection and directional attenuation are demonstrated, by illuminating the scene from two different directions. Plate 5 illustrates the view-dependent character of radiant intensity distributions, with two different views of the same scene. Computation times for all these images range from 17 to 103 seconds.

6 Discussion and Conclusions

We have presented a general framework for the hierarchical representation of energy exchanges taking into account the non-uniform directional behavior of surfaces and object clusters. Although conceptually simple, this approach raises a number of practical issues, which we discuss below.

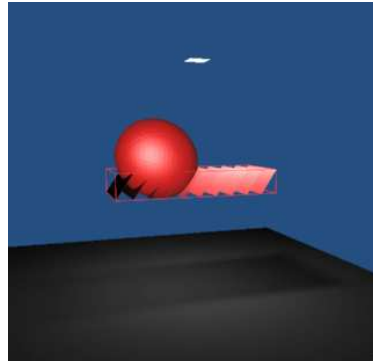
Benefits and limitations of the Approach The explicit representation of directional radiant functions for object clusters has several important benefits. First, it allows a smooth integration of non-diffuse reflectors in a clustering algorithm. Second, if incoming radiance is stored explicitly, it reduces the asymptotic complexity of the clustering algorithm. Third, the consideration of directional extinction properties greatly improves the applicability of the approximate transmission calculation based on the volume analogy. Finally, the method allows the simulation of non-isotropic scattering volumes with arbitrary phase functions. In practice we consider that the most useful feature is the ability to mix diffuse and non-diffuse reflectors in a scene at a moderate additional cost. In particular the overhead costs for diffuse reflectors remain reasonable, while allowing much more accurate transfers between clusters. We tend to prefer the option of implicit storage for incident radiance, since it appears very difficult to do away completely with any traversal of the hierarchy during the gathering stage. For instance, the consideration of intra-cluster visibility is much easier when each contribution is pushed down to the

surfaces [8]. The efficient representation of directional functions is a difficult issue. For general reflectors many spherical harmonics coefficients may be needed, resulting in high storage and computation costs.

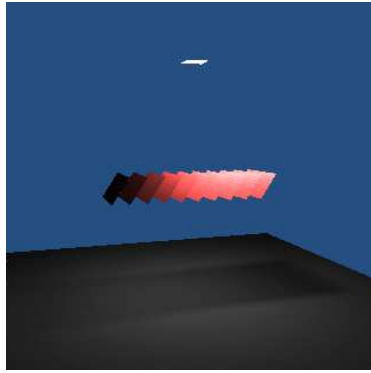
Future directions A major area of research for future work is the investigation of multi-resolution representations of directional functions. It may be possible to store different levels of detail at each cluster, instead of storing a complete distribution everywhere. This would dramatically lower the storage costs, while allowing true multi-resolution visibility computation through object clusters [8]. Another interesting direction is the computation (and storage) of complete scattering functions for all clusters. These will allow the direct transformation of incoming radiance to radiant intensity, similar to a volumic phase function. However the storage costs for such bidirectional phase functions may be prohibitive.

References

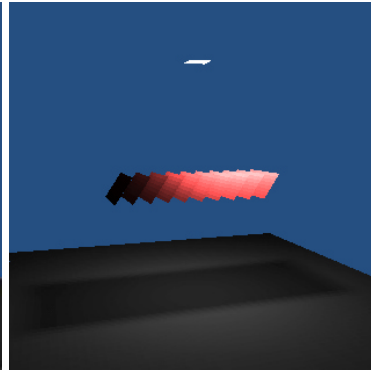
1. Larry Aupperle and Pat Hanrahan. A hierarchical illumination algorithm for surfaces with glossy reflection. In *Computer Graphics Proceedings, Annual Conference Series: SIGGRAPH '93* (Anaheim, CA, USA), pages 155–162. ACM SIGGRAPH, New York, August 1993.
2. Brian Cabral, Nelson L. Max, and Rebecca Springmayer. Bidirectional reflection functions from surface bump maps. *Computer Graphics*, 21(4):273–281, July 1987. Proceedings SIGGRAPH '87 in Anaheim (USA).
3. Pat Hanrahan, David Saltzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. *Computer Graphics*, 25(4):197–206, August 1991. Proceedings SIGGRAPH '91 in Las Vegas (USA).
4. David S. Immel, Michael F. Cohen, and Donald P. Greenberg. A radiosity method for non-diffuse environments. *Computer Graphics*, 20(4):133–142, August 1986. Proceedings SIGGRAPH '86 in Dallas (USA).
5. Dani Lischinski, Brian Smits, and Donald P. Greenberg. Bounds and error estimates for radiosity. In *Computer Graphics Proceedings, Annual Conference Series: SIGGRAPH '94* (Orlando, FL), pages 67–74. ACM SIGGRAPH, New York, July 1994.
6. François Sillion. A unified hierarchical algorithm for global illumination with scattering volumes and object clusters. *IEEE Transactions on Visualization and Computer Graphics*, 1(3), September 1995. (a preliminary version appeared in the fifth Eurographics workshop on rendering, Darmstadt, Germany, June 1994).
7. François Sillion, James Arvo, Stephen Westin, and Donald P. Greenberg. A global illumination solution for general reflectance distributions. *Computer Graphics*, 25(4):187–196, August 1991. Proceedings SIGGRAPH '91 in Las Vegas (USA).
8. François Sillion and George Drettakis. Feature-based control of visibility error: A multiresolution clustering algorithm for global illumination. In *Computer Graphics Proceedings, Annual Conference Series: SIGGRAPH '95* (Los Angeles, CA), pages 145–152. ACM SIGGRAPH, New York, August 1995.
9. François Sillion and Claude Puech. A general two-pass method integrating specular and diffuse reflection. *Computer Graphics*, 23(3):335–344, August 1989. Proceedings SIGGRAPH '89 in Boston (USA).
10. Brian Smits, James Arvo, and Donald P. Greenberg. A clustering algorithm for radiosity in complex environments. In *Computer Graphics Proceedings, Annual Conference Series: SIGGRAPH '94* (Orlando, FL), pages 435–442. ACM SIGGRAPH, New York, July 1994.
11. John R. Wallace, Kells A. Elmquist, and Eric A. Haines. A ray tracing algorithm for progressive radiosity. *Computer Graphics*, 23(3):315–324, July 1989. Proceedings SIGGRAPH '89 in Boston.
12. Stephen H. Westin, James R. Arvo, and Kenneth E. Torrance. Predicting reflectance functions from complex surfaces. *Computer Graphics*, 26(4):255–264, July 1992. Proceedings of SIGGRAPH '92 in Chicago (USA).



(a)



(b)



(c)

Plate 1. Using directional visibility information: (a) representation of the directional extinction coefficient for the cluster of slanted objects. (b) Simulation showing the varying attenuation in the shadow area. (c) Simulation using isotropic extinction: note the uniform attenuation in the shadow area.

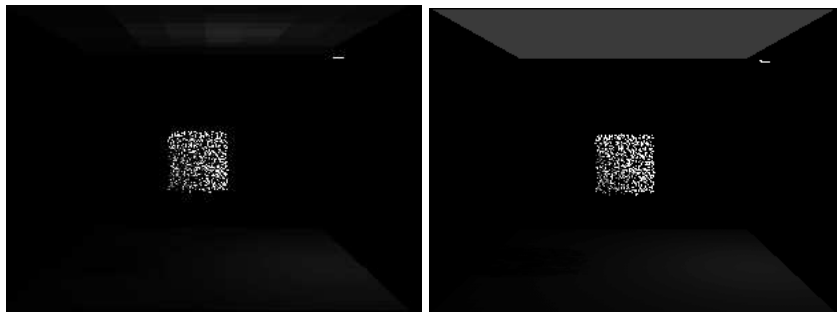


Plate 2. Solution for a scene with 6000 diffuse surfaces. (a) directional and (b) non-directional clustering.

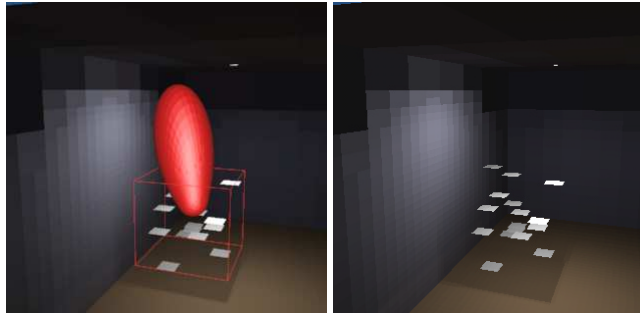


Plate 3. Simulation with a cluster of specular reflectors (overhead illumination): (a) distribution of radiant intensity for the selected cluster. (b) final image.

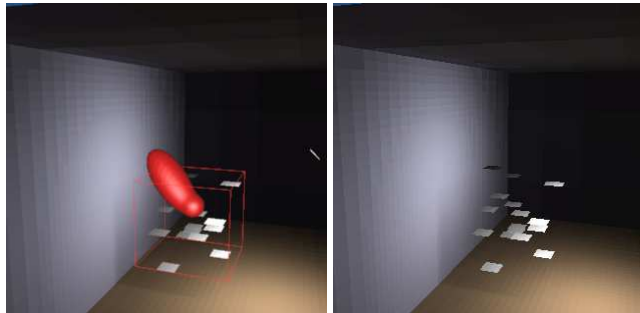


Plate 4. Simulation of the same scene (with illumination coming from the side). Comparing to Plate 3, note the change in secondary illumination and the change in the cluster's shadow.

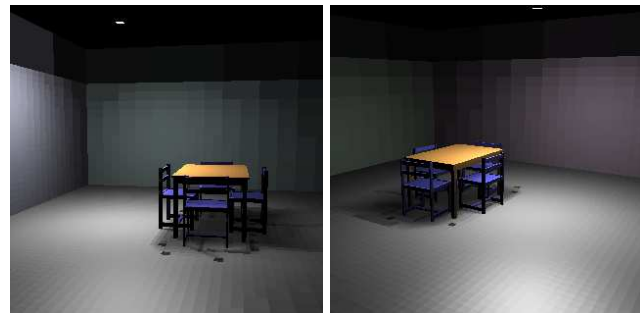


Plate 5. Two views of a scene with glossy surfaces (floor and table top). Note the differences in the appearance of the non-diffuse surfaces.

3.5.8 Efficient Glossy Global Illumination with Interactive Viewing (GI'99)

Auteurs : M. Stamminger, A. Scheel, X. Granier, F. Perez-Cazorla, G. Drettakis et François X. Sillion

Actes : Congrès Graphics Interface '99

Date : juin 1999

Efficient Glossy Global Illumination with Interactive Viewing

Marc Stamminger¹, Annette Scheel¹, Xavier Granier²,
Frederic Perez-Cazorla³, George Drettakis², François Sillion²

¹ Max-Planck-Institute for Computer Science, Saarbrücken, Germany

² iMAGIS-GRAVIR/IMAG-INRIA, Grenoble, France

³ GGG, University of Girona, Spain

Abstract

The ability to perform interactive walkthroughs of global illumination solutions including glossy effects is a challenging open problem. In this paper we overcome certain limitations of previous approaches. We first introduce a novel, memory- and compute-efficient representation of incoming illumination, in the context of a hierarchical radiance clustering algorithm. We then represent outgoing radiance with an adaptive hierarchical basis, in a manner suitable for interactive display. Using appropriate refinement and display strategies, we achieve walkthroughs of glossy solutions at interactive rates for non-trivial scenes. In addition, our implementation has been developed to be portable and easily adaptable as an extension to existing, diffuse-only, hierarchical radiosity systems. We present results of the implementation of glossy global illumination in two independent global illumination systems.

Key words: global illumination, glossy reflection, interactive viewing

1 Introduction

Real-world scenes contain materials with different reflective properties, varying from matte (diffuse) to shiny (glossy or specular). Global illumination research has made great advances for the treatment of diffuse environments in the recent years, in particular with the advent of the Hierarchical Radiosity (HR) algorithm [7] and the subsequent introduction of clustering [18, 15]. It is now possible to compute global illumination solutions of complex diffuse environments and perform interactive walkthroughs. Interactivity is achieved using the polygonal model which is appropriately subdivided into sub-polygons to capture shadows and lighting variations. Since the environments are diffuse, no updates are necessary at each frame, and the polygons are drawn as is. In contrast, scenes containing glossy surfaces cannot yet be treated in an interactive context. To generate images with glossy surfaces, ray-tracing based approaches are typically used, such as the RADIANCE system [27] or path-

tracing algorithms (e.g., [11, 23]). Some finite element approaches have been presented, but can only treat trivial scenes (e.g., [13, 1]) or require a second, ray-casting pass to generate an image [3]. Two approaches have been proposed which are capable of interactive viewing [16, 25], but they are limited in their capacity to treat non-trivial environments and reflective behaviours.

We present a novel solution which allows interactive viewing of globally illuminated glossy scenes. To reach this goal, we use a finite element representation of outgoing radiance at surfaces or clusters. This representation is used at each frame to evaluate the radiance leaving a glossy surface and reaching the eye, permitting interactive viewing.

A novel representation of incoming radiance in the form of a structure called *Illumination Samples* is presented, which is efficient both in memory and computation time. This structure replaces an explicit (and costly) finite-element representation of incoming radiance by sets of relevant point samples.

Furthermore, we demonstrate the importance and benefits of using an adaptive hierarchical representation of outgoing radiance improving on both computation time and memory consumption compared to previous approaches such as [16]. Our algorithm produces high quality glossy global illumination solutions which can be directly rendered for interactive walkthroughs, without the need for expensive second-pass final gather as in [3].

2 Previous Work

Most previous work in glossy illumination has been centered around ray-tracing. These start with distributed ray-tracing [28, 5] and the rendering equation [9] in the late eighties. A large body of research ensued focusing on Monte-Carlo stochastic algorithms. The goal of this research was to reduce the noise in the solutions, introduced by the stochastic nature of the Monte-Carlo methods (e.g. [22, 11, 14]). Monte-Carlo algorithms that proved to be useful in other research areas were successfully trans-

ferred to the global illumination problem [10, 24].

In parallel, several multi-pass methods have been developed which combine the advantages of ray-tracing and radiosity-style calculations [17]; others have integrated radiosity calculations a stochastic process [2]. The RADIANCE system [27], particle tracing [12] and photon-maps [8] are also interesting since they collect samples of illumination either on surfaces or in a separate structure, and use a ray-cast or trace to render the final image.

“Pure” finite element approaches for glossy illumination have appeared in two main flavours: three-point approaches [1, 13] and finite-element approaches using directional distributions [16, 3]. We concentrate on the last two methods in more detail, since they are closer to our new algorithm.

2.1 Wavelets and Final Gather

Christensen et al. [3] extended the wavelet methods which have been used for radiosity [29, 6, 4] to a radiance clustering algorithm. However, it still suffers from some computationally expensive steps which hinder interactive viewing. Patches store a radiance distribution which is represented using a four dimensional wavelet basis accounting for spatial and directional variations. Computing the transport coefficients involves evaluating a six-dimensional integral which is computationally expensive. Importance driven refinement reduces the number of interactions drastically; but the solution becomes view dependent and consequently cannot be used for interactive viewing.

Higher order wavelet bases were also investigated, which provide a sparser transport coefficient matrix and deliver smoother representations. However, the integrations are so complex that the authors resorted to the Haar basis with a smoothing final gather step, which is very time consuming and view dependent.

2.2 Radiance Clustering

The Radiance Clustering approach (RC) developed by Sillion et al. [16], used spherical harmonics to store exiting radiant intensity I on the hierarchical elements of a subdivision of the original scene. An “immediate-push” algorithm is used, which, during the gather operation of light across links, “pushes” the contribution all the way to the leaves. At the leaves, radiant intensity I is stored as a spherical harmonic function; the new contribution is reflected and added into this function.

The result can be visualised directly by sampling the spherical harmonic representations of I at each frame. Direct visualisation (i.e. with no acceleration) was performed for simple scenes; since for each frame radiance is evaluated at each vertex or leaf element, frame rates are not optimal. Furthermore, spherical harmonics are a

non-hierarchical representation, and the number of coefficients used is fixed in advance. As a result, there is no control over the level of detail required to represent the directionally dependent glossy illumination.

Our new algorithm provides solutions to the above problems and also reduces memory and time consumption. We start with an improved representation of incoming radiance, which avoids the memory overhead and multiple hierarchy passes of the “immediate-push” solution. In particular we introduce *Illumination Samples* which are an appropriate point sample set representation of incoming light. We then proceed with an adaptive hierarchical representation of outgoing radiance using Haar wavelets, which is well-suited to interactive viewing, and allows smooth control of the memory/quality tradeoff. This avoids the problems of non-adaptive representations which are either not sufficiently accurate or too memory-consuming. Appropriate directional refinement and simple heuristics for accelerated viewing are also introduced.

3 The Illumination Samples Algorithm

The goal of the new Illumination Samples algorithm is to extend an existing Hierarchical Clustering algorithm to also handle non-diffuse surfaces. Inter-surface light propagation is the same for diffuse and non-diffuse environments, with the difference that in a non-diffuse setup directional information about incident light must be maintained for a following glossy reflection step.

As in [16], patches and clusters are assumed to have no spatial extent. They store a hierarchical directional distribution for outgoing radiance which will be described separately in Section 4. In contrast to [3], our new algorithm does not differentiate between clusters and patches concerning the representation of exitant light.

3.1 Bounded Propagation

Our approach is based on the radiosity clustering method described in [20, 21], which can handle flat and curved surfaces as well as clusters in a uniform manner. Bounding boxes around the objects are used to bound the set of interacting directions. With this information, bounds on the form factor and exitant radiance at the sender are computed, delivering minimum and maximum values for the received radiosity. The difference is used to decide whether to refine a link.

This bounded radiosity approach can be applied to radiance computations easily, since the propagation of light, i.e. the transformation of exitant to incident light, is independent of material properties. The only difference lies in the evaluation of bounds on the radiance of the sender, which is even easier if we have a directional distribution for the sender’s exitant radiance. However, since this exitant radiance representation is only approxi-

mate, the resulting bounds are no longer conservative.

3.2 Incident Light

One way to integrate the directional information is to explicitly compute a finite element representation of it. In [3], each incident light contribution computed during propagation is projected separately onto a basis for incoming light (for clusters). This is rather costly (in memory and time) and results in significant blurring of incident light, which can exhibit very strong variations.

An alternative is to reflect incident light contributions immediately after they have been computed [16], while their direction of incidence is still known. The reflection responses are then projected in finite element bases separately. This method circumvents the need to store the incident light, but the storage consumption is not reduced: two representations of exitant radiance are needed for the push/pull phase. In addition, this method is computationally expensive, because of the high number of BRDF evaluations, and the multiple hierarchy traversals involved in the immediate projection.

Our proposed solution is to combine the approaches of incident light representation and immediate reflection. We attach incident light to a receiving patch in the form of *Illumination Samples*. Light propagation is computed similarly to HR by refining links until each link represents what amounts to constant light power. Instead of simply summing the irradiance values at the receiver, an Illumination Sample with the direction to the sender and the transported irradiance is added to the receiver for each link. At the end of the propagation step, the illumination in the scene is represented as a set of point samples, distributed over the scene hierarchy.

3.3 Push/Pull and Reflection

A push step as in HR is needed to create a consistent representation of the incident light at the leaves, i.e. all light received by inner nodes is propagated to the children by passing its Illumination Samples downwards. Afterwards, each leaf has a large set of Illumination Samples describing its entire incident light field. Note that the number of hierarchy traversals is much smaller than in Radiance Clustering [16], where each sample is pushed down separately.

After the push step, the incident light has to be reflected according to the object's BRDF. Because Illumination Samples correspond to Dirac impulses, the reflection is an impulse response of the BRDF, i.e. it is the BRDF with a fixed incident light direction multiplied by the irradiance of the sample. The complete reflection is the sum of the impulse responses to each Illumination Sample. Therefore the BRDF must be evaluated once for each Illumination Sample to obtain the reflected radiance

in a particular direction.

Using an adaptive directional distribution described below, reflected light is projected onto an adaptive, hierarchical directional basis to obtain the new exitant light for each patch. These representations are then averaged bottom-up to obtain the distributions for inner nodes.

3.4 Shooting

With a gathering iteration scheme, the number of Illumination Samples and thus the time for push/pull increases from iteration to iteration. With a shooting scheme in the spirit of [19] this can be avoided. This requires an additional directional representation of *unshot radiance*, but it also avoids the storage and reflection of all Illumination Samples.

3.5 Error Analysis

Note that in our approach propagation and reflection are completely decoupled. Propagation computation does *not* consider the reflection properties of the receiver, which would allow the computation of incident light at a highly glossy patch more accurately than in the diffuse case. The spatial refinement of the patches is done during propagation, while the refinement level of the directional distributions is chosen during reflection. This distinction does not impose a problem on convergence, but it results in memory/computation savings.

Illumination Samples can be interpreted as Dirac-peaks from a particular direction describing incident light and are thus somewhat similar to the photons in the Photon Map approach of Jensen et al. [8]. However, Photon maps are not deterministic and their usage for lighting simulation is very different from ours.

With respect to a standard norm, with Dirac-peaks no convergent representation can be obtained. On the other hand, the Dirac-representation is only used to compute the reflection integral. From another point of view, this representation can be seen as intermediate data in a delayed numerical integration, where each Illumination Sample is a temporarily stored integration sample. So as long as the BRDF is numerically integrable, the computed reflection will converge.

4 Adaptive Representation of Outgoing Radiance for Interactive Display

To produce the finite-element solutions suitable for interactive display, we store outgoing light in the form of directional distributions attached to surfaces or clusters. As in Radiance Clustering [16], objects are assumed to have no spatial extent. Instead of the four dimensional radiance only the 2D *radiant intensity* distribution is stored with each object.

4.1 Directional Representations

For the representation of directional radiant intensities, we have implemented and examined two options: First, a uniform subdivision of the direction space, where each distribution is represented by a fixed number of coefficients (*non-adaptive basis*)¹. Second, we implemented an adaptive representation using Haar Wavelets.

The non-adaptive basis is more useful for smooth distributions, because all operations on the fixed subdivision basis are simple and fast. The adaptive Haar basis is better suited for strongly varying functions, because it can use more basis functions in the interesting regions and fewer in smooth regions. However, operations such as the evaluation of the distribution or addition of two distributions are more expensive.

“Non-adaptive” Representation

For a non-adaptive basis, we use a uniform subdivision of the direction space. To accomplish this task, a tetrahedron is subdivided. We thus obtain 4^{n+1} triangles if the level of subdivision is n . Since the number of vertices is lower than the number of triangles (this is $2 \cdot 4^n + 2$), we decided to store 3 floats for RGB only at the vertices.

Haar Representation

For the Haar representation, the domain of directions is parameterized by points on an octahedron. The vertices of the octahedron are selected to lie on the main axes, so each face corresponds to one octant of the directional domain. Simple sign considerations of a direction deliver the corresponding octahedron face.

A hierarchy of basis functions is built by assigning a first level basis function to each of the eight faces of the octahedron. These are then subdivided hierarchically in the usual manner.

In order to quickly compute an adaptive representation, a top-down approach was chosen. Assume that the function to be projected is f . For each of the first eight basis functions, f is sampled at the triangle corners and at its center. If f is almost constant over the triangle, the sample values will only vary slightly. For highly varying f , one can expect a wide range of function samples. Thus the difference between minimum and maximum sample is considered. If it is too large, the four finer basis functions partitioning the domain are considered recursively.

This top-down approach runs into problems if f has a sharp peak inbetween the samples. We alleviate this problem by enforcing a minimum subdivision level in the hope that the resulting sampling is dense enough to not miss any peaks.

¹This allows a comparison with [16]; to be complete we should have tested with spherical harmonics. Since no solution for general surface orientation currently exists, we used the “non-adaptive” basis.

There are several possibilities to decide whether the difference is too large. For the algorithm described in this paper, the difference is compared with the midpoint value, i.e. a maximum *percentage* deviation ϵ with respect to the center value is allowed. This turned out to be beneficial for our case (see Section 5.1); for other settings, different criteria can be used.

Comparison

To see the differences involved in using Haar or non-adaptive representations, consider the following example, which is an empty room (the geometry is taken from the RADIANCE test scene “Soda Shoppe” [26]).

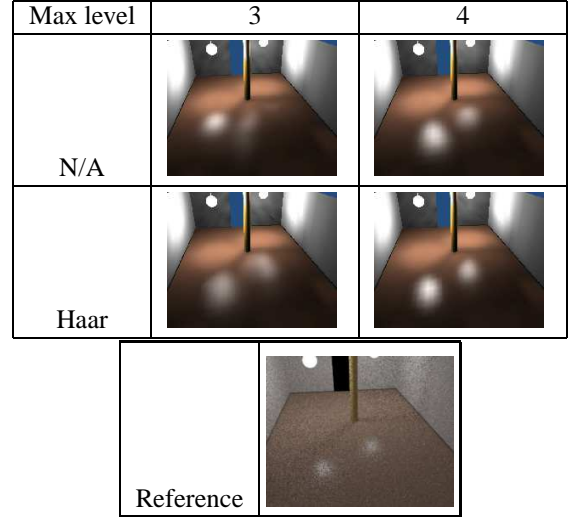


Figure 1: Comparison of the non-adaptive (N/A) vs. Haar.

Max Level	Distr	Triangles		Time	
		N/A	Haar	N/A	Haar
3	2878	782K	640K	510 s	722 s
4	2878	3127K	1829K	731 s	1125 s

Table 1: Comparison of Non-adaptive (N/A) and Haar basis for Radiance Clustering, showing the number of directional functions used and the computation time. Max Level is the maximum level of subdivision.

The reference image was computed using a path-tracer using next-event estimation [11]. The images in Figure 1 were generated using Radiance Clustering, with the non-adaptive and Haar basis (see Section 5 for more details on rendering). The “Max Level” parameter corresponds to the maximum permitted level of subdivision. Clearly, the non-adaptive basis fails to correctly represent the high-lights on the glossy floor for maximum subdivision level 3. For maximum level 4, the result is improved, but at the cost of 4 times more memory (see Table 1). In contrast,

the Haar basis uses less than three times as much memory for an “equivalent” improvement in quality. However, the Haar basis also takes more time. The reason is that arithmetic operations on the regular constant subdivision are of course simpler and faster.

This example demonstrates that for highly glossy scenes, small highlights can only be captured with the adaptive basis or a very fine non-adaptive basis representation, which in turn requires large amounts of memory. More importantly, the user, who has limited memory, can only change the quality in large “quanta”, and often will not be able to get a satisfactory result before running out of memory. Adaptive bases, such as Haar, alleviate this problem. However, the uniformity of the non-adaptive basis results in a smoother, more regular distribution, which becomes especially visible during interactive viewing.

5 Interactive Display

After the computation of a global illumination solution using Illumination Samples, we have a representation of outgoing radiant intensity, stored in the directional distribution function. At each frame during interactive display, we need to evaluate radiance for every glossy hierarchical leaf element in the direction of the viewpoint. This implies two requirements: (i) subdivision of the directional distributions appropriately so that a visually pleasing representation of glossy effects is produced and (ii) acceleration of the display process to avoid the cost of the evaluation of radiance at each element at every frame.

5.1 Refinement Issues for Display

Recall that we have decoupled directional subdivision, in the form of the Haar-based directional distribution functions, and the spatial subdivision, in the form of the “traditional” hierarchical radiosity element hierarchy. To display the solution, we interpolate radiance in the view direction by evaluating the directional distribution on each element. If subdivision in direction space is performed arbitrarily, the difference in subdivision of the directional function between neighbouring patches may be too abrupt.

This is the case for example if we compare absolute value differences between the center and the vertices of the triangles of the directional subdivision to decide whether to subdivide. The use of relative (percentage) differences avoids this problem since we approximate the form of the function, which varies more slowly across neighbours. The artifacts due to the absolute refinement can be seen in Figure 2.

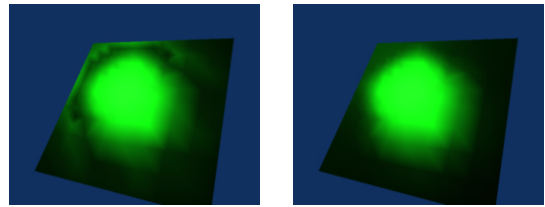


Figure 2: Artifacts when using the “absolute” refiner (left), which are absent when using the “relative” refiner (right).

5.2 Interactive Rendering

For efficient display we separate objects into two lists, so that diffuse objects can be rendered once and redisplayed in efficient, display-list mode. The other list, of glossy reflectors, is updated appropriately at each frame and displayed in immediate mode. The acceleration achieved obviously depends on the percentage of diffuse surfaces in the scene. For the scenes tested we achieve update rates varying from a few frames per second to a few seconds per frame for more complex scenes.

To achieve smooth shading for glossy surfaces, we add a field to the data structure associated with vertices in the hierarchy of elements. For planar surfaces, this field is updated during push-pull in a manner slightly different to that of radiosity; i.e. for a vertex belonging to a leaf element or to an edge, the radiant intensity is summed with the radiant intensity stored at the vertex. Since radiant intensity is in Watts/sr (see [16]), at display time we divide by the area of the surrounding elements.

The special case of indexed face-sets is treated separately. Indexed face-sets are common modelling primitives, and often result from the tessellation of curved objects such as spheres or cylinders. The advantage of such a primitive is that vertices are shared between adjacent elements. We can thus avoid the storage of the additional directional distribution at the vertices.

Each vertex stores the list of polygonal elements which share it. Its color is then the average radiant intensity of these polygons (i.e. I evaluated at the centers of the elements in the viewing direction). For more efficient display, we evaluate this color once per vertex for a given direction. Also, we recompute the color only if the direction changes “sufficiently” i.e. greater than a user-defined ϵ threshold. This allows the control of the quality/update rate tradeoff.

6 Implementation and Results

One major goal of our approach was the development of a solution which can be considered a simple “add-on” to an existing hierarchical radiosity system. We implemented the algorithm on two very different rendering architectures, namely BRIGHT (iMAGIS) and VISION (Uni-

versity of Erlangen).

We have tested our implementation on several test scenes, shown in Figures 3 and 6. The scenes in Figure 3 were used for the interactive viewing test in BRIGHT. The first scene shows three light sources colored red, green and blue, illuminating a very glossy, small reflector. This reflector in turn indirectly illuminates a diffuse wall. The second scene is a glossy sphere illuminated by a small source and a glossy floor. These in turn produce indirect glossy effects on the lower part of the sphere and the diffuse ceiling. Finally, the “Simple soda” scene is a simplified version of the “Soda Shoppe” scene. In BRIGHT, we require tessellation of all objects initially, which results in a high number of initial objects; in VISION, objects are not initially tessellated. This explains the low number of initial objects in the complete “Soda Shoppe” scene, used for Figure 6.

6.1 Radiance Clustering vs. Illumination Samples

In BRIGHT we have implemented both Radiance Clustering (RC) and the Illumination Samples (IS) approach. We have compared running time and memory usage for the RC and IS approaches. The threshold value has the same meaning for both approaches, since we are using a “relative” refiner. Visual inspection also shows that the resulting images are equivalent for the same parameter values. All timings are on a 195Mhz R10k SGI workstation.

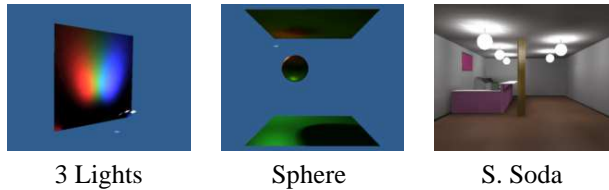


Figure 3: Test scenes using illumination samples

6.2 Memory Consumption

In Table 2 we show the memory statistics for the test scenes used. In particular we list the different scenes with the ϵ accuracy threshold (see Section 4.1), and the corresponding number of directional distribution basis functions used for the solution by the Radiance Clustering (RC) and Illumination Samples (IS) approach. The rightmost column shows the percent gain of the illumination samples approach.

Memory usage is clearly reduced using Illumination Samples compared to the Radiance Clustering approach for all scenes. The gain varies from 37 % to 41% in the best case. This is mainly due to the fact that Radiance Clustering requires the additional intermediate directional distribution functions to be able to correctly per-

	ϵ	m/M	IS	RC	
3 Lig.	0.5	1/3	8618	13866	38%
3 Lig.	0.1	1/3	8820	14068	37%
3 Lig.	0.5	1/4	27306	43510	37%
3 Lig.	0.5	1/5	79218	125944	37%
Sph.	0.5	1/3	2114324	3598720	41%
Soda	0.5	1/3	2097534	3339794	37%

Table 2: Gain in *memory* usage from the use of the Illumination Samples algorithm. ϵ is the accuracy threshold and m/M the min/max levels.

form the push-pull operation (see Section 2.2).

6.3 Computation Time

In Table 3 we show the computation time statistics for the test scenes used. In particular we list the different scenes with the ϵ threshold, and the corresponding computation time for the solution by the two approaches (RC and IS). The rightmost column shows the percent memory gain of the illumination samples approach.

	ϵ	m/M	IS	RC	
3 Lig.	0.5	1/3	44.6 s	90.1 s	50%
3 Lig.	0.1	1/3	44.2 s	90.0 s	51%
3 Lig.	0.5	1/4	49.3 s	95.5 s	48%
3 Lig.	0.5	1/5	58.6 s	105.3 s	44%
Sph.	0.5	1/3	4167.1 s	6492.9 s	34%
Soda	0.5	1/3	5207.6 s	7117.4 s	27%

Table 3: Gain in *computation time* from the use of the Illumination Samples algorithm. ϵ is the accuracy threshold and m/M are the min/max levels.

For all scenes the illumination samples approach provides a speedup of at least 27%. This is mainly due to the reduction in the number of hierarchy traversals, and also the reduction in the number of triangles used to represent the directional distributions as discussed above.

As expected and confirmed by the experimental results, the Illumination Samples approach reduces both memory and computation time with respect to Radiance Clustering. The images produced by both approaches are essentially indistinguishable.

6.4 Visual Quality/Comparisons

We qualitatively compare the visual quality of the images of Radiance Clustering and Illumination Samples with those from a path-tracer or the RADIANCE system. The path-tracer tests are rendered using an in-house implementation of the next-event estimation [11]. In Fig. 4 and 5 we show the reference path-tracer or RADIANCE images and the corresponding Illumination Samples images together with the computation times for the test scenes.

There are several interesting observations that we can infer from these examples:

- 1) In the case of the three light scene, RADIANCE runs

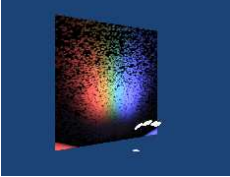
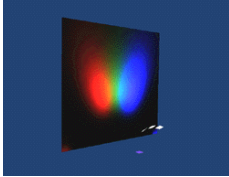
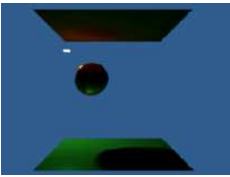
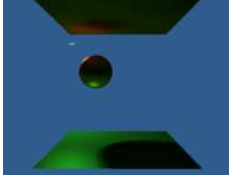
Reference	Illumination Samples
	
Radiance 6537s	59s
	
Radiance 1303s	4167s

Figure 4: Reference solutions (RADIANCE) compared to IS solutions for three lights and sphere scenes.



Reference	Illumination Samples
	
Path Tracer 51873s	5208s

Figure 5: Simple Soda reference solution (path-tracer at 455x364 resolution) compared to IS solution.

into sampling problems. Even after more than an hour and a half of computation it does not converge. In contrast, illumination samples achieves a solution in less than a minute, which is in addition viewable from any direction interactively².

2) The computation times of IS are either lower or in the same order of magnitude as those of the reference solutions. The important thing to remember is that the IS solutions can be viewed *interactively*, while the reference (path-tracer or Radiance) require the same amount of time (tens of minutes or even hours) for *every* image.

3) Path-tracing images are very noisy. The “smooth-shaded” solutions produced by IS do not suffer from this problem. Despite being approximate, the smooth-shaded images are therefore much better suited to interactive applications, where noise and flickering are very distracting.

We thus believe that our approach has great promise, since it can be used to generate low to moderate quality solutions for glossy environments, as well as produce solutions suitable for interactive viewing.

²A bi-directional path-tracer, or photon-map which would consider the reflection as a caustic may generate better results.

6.5 A More Complex Scene

As a last test we applied the Illumination Sample method to a more complex scene, the Soda Shoppe, one of the RADIANCE test scenes. Our version consists of 1,644 initial patches, several of which are non-planar, including the spherical light sources. About one third of the patches are non-diffuse. Since bounded form factor computation is used [20], no initial tessellation of these objects was necessary, which would have increased the initial complexity significantly. The scene is not yet really complex in the sense of an industrial-size model, but sufficiently non-trivial to impose severe problems on previous finite-element radiance methods.

The solution shown in Figure 6 was computed with the implementation of VISION, which incorporates the “shooting” solution described previously (Section 3.5). It was obtained in 8,488 seconds and contains 29,138 final patches. 91% of the computation time was spent on propagation, which in turn is dominated by visibility (97%), only 9% was used for push/pull including the reflection. 743,284 links were computed, resulting in 29,138 patches. Note that the used VISION implementation does not yet incorporate smooth reconstruction capabilities, so that the patch boundaries are clearly visible.



Figure 6: A solution to the glossy soda shoppe, computed in 8,488 seconds.

By far most of the computation is spent on visibility, as with diffuse radiosity computation. This indicates that we were able to reduce the overhead introduced by explicitly storing directional illumination information to reasonable levels. The more costly push/pull step was expected, but it is interesting to notice that it still requires only about 10% of the overall computation for this particular scene. For other scenes with more glossy objects, this percentage is somewhat larger, although always “reasonably small”. Note that this is only true for using shooting instead of gathering! For gathering the push/pull times can increase significantly.

7 Conclusions

We have presented a novel solution to global illumination simulation for glossy environments. Our new algorithm

is an important step towards interactive walkthroughs of globally illuminated glossy scenes: (i) We introduced the *Illumination Samples* algorithm which represents incoming light more accurately and efficiently, both in memory and computation time. (ii) We have used an adaptive hierarchical finite-element basis to store outgoing light, in a manner suitable for interactive viewing. This allows fine control of the memory/quality tradeoff, which was not possible in previous solutions. (iii) These algorithms can be implemented with marginal effort over an existing hierarchical radiosity system, by confining the modifications to a small number of phases and data structures. (iv) Interactive viewing of the glossy global illumination solutions is achieved by suitably refining the directional representation of outgoing light and accelerating the display process.

Our solution however is still quite memory-consuming requiring in the order of tens of Mbytes for the smallest scenes and hundreds of Mbytes for the more complex.

To remedy these problems we need to generalise the multi-resolution nature of our solution. Notably we will introduce a multi-resolution representation of outgoing light, which will allow significant savings in memory at the cost of lower visual quality. This approach will require a novel representation as well as a novel refinement algorithm. This representation can be used both for the actual lighting simulation as well as for display.

Acknowledgements

This research was funded by the ESPRIT Open LTR SIMULGEN (#25772) Thanks to Cyril Soler (iMAGIS) and Ignacio Martin (GGG) for coding help, and to Bruce Walter for proofreading. iMAGIS is a joint project of CNRS/INRIA/UJF/INPG.

8 References

- [1] L. Aupperle and P. Hanrahan. A hierarchical illumination algorithm for surfaces with glossy reflection. In *Computer Graphics (SIGGRAPH '93)*, pages 155–162, August 1993.
- [2] S. E. Chen, H. E. Rushmeier, G. Miller, and D. Turner. A progressive multi-pass method for global illumination. *Computer Graphics (SIGGRAPH '91)*, 25(4):165–174, July 1991.
- [3] P. H. Christensen, D. Lischinski, E. Stollnitz, and D. H. Salesin. Clustering for glossy global illumination. *ACM Trans. on Graphics*, 16(1):3–33, January 1997.
- [4] P. H. Christensen, E. J. Stollnitz, and D. H. Salesin. Global illumination of glossy environments using wavelets and importance. *ACM Trans. on Graphics*, 15(1):37–71, January 1996.
- [5] R. L. Cook, T. Porter, and L. Carpenter. Distributed ray tracing. *Computer Graphics (SIGGRAPH '84)*, 18(3):137–145, July 1984.
- [6] S. J. Gortler, P. Schröder, M. Cohen, and P. M. Hanrahan. Wavelet radiosity. *Computer Graphics (SIGGRAPH '93)*, 27:221–230, August 1993.
- [7] P. Hanrahan, D. Salzman, and L. Aupperle. A rapid hierarchical radiosity algorithm. *Computer Graphics (SIGGRAPH '91)*, 25(4):197–206, 1991.
- [8] H. Wann Jensen. Global illumination using photon maps. In Xavier Pueyo and Peter Schröder, editors, *Rendering Techniques '96 (7th EG Workshop on Rendering)*, pages 21–30. Springer, June 1996.
- [9] J. T. Kajiya. The rendering equation. *Computer Graphics (SIGGRAPH '86)*, 20(4):143–150, August 1986.
- [10] A. Keller. Quasi-monte carlo radiosity. In X. Pueyo and P. Schröder, editors, *Rendering Techniques '96 (7th EG Workshop on Rendering)*, pages 101–110, Porto, June 1996. Springer.
- [11] E. Lafortune. *Mathematical Models and Monte Carlo Algorithms for Physically Based Rendering*. PhD thesis, Katholieke Universiteit Leuven, 1996.
- [12] S. N. Pattanaik and S. P. Mudur. Computation of global illumination by Monte Carlo simulation of the particle model of light. *Third EG Workshop on Rendering*, pages 71–83, May 1992.
- [13] P. Schröder and P. Hanrahan. Wavelet methods for radiance computations. In *Photorealistic Rendering Techniques (5th EG Workshop on Rendering)*, pages 303–311, Darmstadt, June 1994. Springer.
- [14] P. Shirley, C. Wang, and K. Zimmerman. Monte carlo techniques for direct lighting calculations. *ACM Trans. on Graphics*, 15(1):1–36, 1996.
- [15] F. X. Sillion. A unified hierarchical algorithm for global illumination with scattering volumes and object clusters. *IEEE Trans. on Vis. and Comp. Graphics*, 1(3):240–254, September 1995.
- [16] F. X. Sillion, G. Drettakis, and C. Soler. A clustering algorithm for radiance calculation in general environments. In *Rendering Techniques '95 (6th EG Workshop on Rendering)*, pages 196–205. Springer, August 1995.
- [17] F. X. Sillion and C. Puech. A general two-pass method integrating specular and diffuse reflection. *Computer Graphics (SIGGRAPH '89)*, 23(3):335–344, July 1989.
- [18] B. Smits, J. Arvo, and D. Greenberg. A clustering algorithm for radiosity in complex environments. *Computer Graphics (SIGGRAPH '94)*, pages 435–442, July 1994.
- [19] M. Stamminger, H. Schirmacher, P. Slusallek, and H-P. Seidel. Getting rid of links in hierarchical radiosity. *Computer Graphics Forum (EUROGRAPHICS '98)*, 17(3):165–174, September 1998.
- [20] M. Stamminger, P. Slusallek, and H-P. Seidel. Bounded radiosity – illumination on general surfaces and clusters. *Computer Graphics Forum (EUROGRAPHICS '97)*, 16(3):309–318, September 1997.
- [21] M. Stamminger, P. Slusallek, and H-P. Seidel. Bounded radiosity – finding good bounds on clustered light transport. *Conf. Proc. of Pacific Graphics '98*, 1998. <http://www9.informatik.uni-erlangen.de/eng/research/pub1998/>.
- [22] E. Veach and L. Guibas. Bidirectional estimators for light transport. In *Photorealistic Rendering Techniques (5th EG Workshop on Rendering)*, pages 145–167, Darmstadt, June 1994. Springer.
- [23] E. Veach and L. J. Guibas. Optimally combining sampling techniques for Monte Carlo rendering. *Computer Graphics (SIGGRAPH '95)*, pages 419–428, aug 1995.
- [24] E. Veach and L. J. Guibas. Metropolis light transport. *Computer Graphics (SIGGRAPH '97)*, pages 65–76, aug 1997.
- [25] B. Walter, G. Alipay, E. P. F. Lafortune, S. Fernandez, and D. P. Greenberg. Fitting virtual lights for non-diffuse walkthroughs. In T. Whitted, editor, *SIGGRAPH '97*, pages 45–48, August 1997.
- [26] G. Ward. Radiance web site. <http://radiance.lbl.gov/radiance>.
- [27] G. J. Ward. The RADIANCE lighting simulation and rendering system. *Computer Graphics (SIGGRAPH '94)*, pages 459–472, July 1994.
- [28] T. Whitted. An improved illumination model for shaded display. *Communications of the ACM*, 23(6):343–349, June 1980.
- [29] H. R. Zatz. Galerkin radiosity: A higher order solution method for global illumination. *Computer Graphics (SIGGRAPH '93)*, pages 213–220, August 1993.

3.5.9 Hierarchical Lighting Simulation for Outdoor Scenes (EGRW'97)

Auteurs : K. Daubert and H. Shirmacher and F. Sillion and G. Drettakis

Actes : 8th Eurographics Workshop on Rendering

Date : juin 1997

Hierarchical Lighting Simulation for Outdoor Scenes

Katja Daubert, Hartmut Schirmacher*
François X. Sillion, George Drettakis**

iMAGIS ***
Laboratoire GRAVIR/IMAG-INRIA

Abstract: Lighting algorithms for outdoor scenes suffer from the sheer geometric and lighting complexity of such environments. In this paper we introduce an efficient, *hierarchical* solution to the problem of outdoor illumination. Data structures and sampling algorithms are presented, permitting the integration of complex and natural objects in a hierarchical radiosity simulation system. This new approach allows the hierarchical simulation of radiant energy exchanges in outdoor scenes for the first time, including terrain and botanical models as well as sunlight and skylight. This is accomplished by providing the necessary tools to treat terrain meshes as a hierarchy of light-exchanging objects, as well as an efficient hierarchical representation for the sky dome. In addition, refinement criteria are adapted to the particular characteristics of natural lighting. Results of our implementation are presented including naturally-lit images of terrain-maps, trees and buildings.

1 Introduction

Rendering technology has made tremendous progress in the last decade, and it has become possible to perform simulations of light exchanges for moderately complex environments, yielding images of impressive realism. These algorithms exploit hierarchical solution methods to correctly account for secondary illumination and result in view-independent solutions. A vast majority of this research has been concentrated on indoor scenes.

As realistic simulated images receive more widespread attention, new potential applications for lighting simulations constantly appear, and present new challenges for state-of-the-art technology. One such challenge is the accurate and efficient simulation of lighting effects for *outdoor* scenes. Such simulations are needed for instance in visual simulation applications for various driving or flight simulators, for visual site planning in the context of architectural projects or urban modeling, or for the simulation of radiant exchanges in vegetation with applications in remote sensing or in the simulation of botanical processes.

Outdoor scenes tend to be especially challenging to simulate, because they present a combination of several difficulties that are poorly handled by existing radiosity algorithms, and in particular hierarchical solution methods.

First, outdoor scenes exhibit great geometric complexity: on the one hand, the sheer scale of outdoor scenes is significantly larger than that of indoor scenes. Furthermore, outdoor scenes often contain natural objects (terrain or trees) that have complex shapes.

* {kadauber,htschirm}@immd9.informatik.uni-erlangen.de
currently at the University of Erlangen-Nuremberg, Computer Graphics Group.

** {Francois.Sillion,George.Drettakis}@imag.fr

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

A second major challenge with outdoor scenes is the complexity of the illumination itself. Natural illumination (i.e. from the sun and the sky) has specific properties such as the concentration of a very large energy flux in a small solid angle (from the sun), and the fact that the sky effectively “surrounds” all objects. These properties require particular care when sampling for illumination simulation. Moreover, because outdoor scenes are always complex, efficient algorithms must be developed, since brute force approaches are bound to be intractable.

In this paper we propose solutions to the problems of geometric complexity and natural lighting. Our approach is based on *hierarchical* lighting algorithms, which permit efficient and elegant solutions. In particular, we show that complex geometry can be handled using a generic hierarchical surface description mechanism in the context of a clustering algorithm. This mechanism is integrated into an object-oriented framework, and its generic nature opens the way for the definition of other hierarchical objects for lighting simulation (such as complex meshes representing curved surfaces). We introduce appropriate representation, sampling and refinement machinery for hierarchical treatment of natural lights (sun and skylight). The resulting system is, to our knowledge, the first capable of providing *hierarchical* lighting solutions of complex outdoor scenes.

2 Previous work

Previous work in lighting simulation for outdoor scenes has been restricted mainly to the modeling of natural light. Similarly, the treatment of the geometric complexity, specifically for the problems posed by certain outdoor structures has received limited attention. We briefly review the relevant literature.

2.1 Previous Work in Natural Lighting

Much work has been done on modeling daylight with different atmospheric effects and designing global illumination models including such light sources. Several models with different complexity were proposed to compute skylight intensities [1, 13, 8, 7, 3]. The public domain system *Radiance* [20] provides source code (“gensky” and “skybright”) to compute sky luminance using the CIE standard sky distribution model [1] for a given setup of hour, day, month, position on earth, and several weather-specific parameters. The sun position can also be specified directly by providing the solar angles.

To represent the skylight distribution, Nishita et al. [13] proposed a fixed set of band sources. In [3] subsampled band sources are inserted where the skylight distribution varies strongly, resulting in a representation with adaptive precision.

To speed up the time consuming integration of bands for global illumination, Tadamura et. al [19] use a regularly subdivided parallelepiped in a hemicube-like manner to determine the skylight contribution including visibility.

For indoor lighting, a set of “interface” surfaces can be used to represent the skylight coming through the windows of a room. In [9] the skylight contribution on window sample points is calculated in a preprocessing step and then window patches are used as area light sources for global illumination. This method is of course limited to indoor scenes.

The most closely related approach to our solution is that of Müller et al. [12]. In their algorithm the skylight distribution is directly stored in a set of radiosity patches which are placed far away from the rest of the scene to satisfy error bounds. This is

achieved by the use of a regular subdivision of the sky hemisphere. The sky patches are then used for shooting in the first iteration of the radiosity simulation. Their method is based on the progressive refinement radiosity algorithm. The initial shooting phase is expensive (computation time of three hours was reported for a scene of 47,500 patches), but permits rapid change of the sky lighting condition, by storing a list of form-factors.

2.2 Treating Complex Geometry

Many outdoor models, such as terrains or mesh structures resulting from scanned-data, or complex man-built structures (bridges, dams etc.) are typically represented with complex meshes. Level-of-detail approaches for the treatment of such complex geometries are well known in the modeling community (e.g., [14, 4]). Despite this wealth of algorithms, the hierarchical treatment of these mesh structures has not previously found its way into lighting simulation algorithms.

2.3 Limitations of the Previous Approaches

The vast majority of previously described natural lighting algorithms require time consuming preprocessing steps. The method proposed in [12] is the first to integrate the sky as a radiosity object, but however a fixed subdivision is used. This is a severe restriction, lacking the benefits of both hierarchical solution [6] and clustering [18] for skylight objects. This also results in expensive calculations and lack of an overall representation of *global* illumination. To achieve efficient integration of natural lighting in a radiosity system, it is important first to provide multiresolution representations of natural lights, and also to ensure that the entire sky is always taken into account. It is important to note that when progressive refinement radiosity is used, this condition forces us to shoot energy from *all* sky patches, at a significant expense, before any meaningful solution can be obtained.

Similarly, in the case of complex meshes representing outdoor structures, a true hierarchical representation is required. A generic clustering algorithm [18, 16] applied to the polygons constituting the mesh would however be unsuitable, since we would lose all neighbouring information amongst mesh elements, needed for the smooth reconstruction of the radiosity function.

3 Unified Lighting Framework

As was presented in [18], a natural complete framework for hierarchical radiosity with clusters can be based on the concept of an abstract *Hierarchical Element* or *H-element*. These elements can be clusters (groups of objects) or surfaces. In the context of an object-oriented system, this abstract class hierarchy permits the definition of generic operators (refine, gather, push-pull) on H-elements, thus entirely defining the hierarchical radiosity process.

We briefly overview the central concepts in what follows, which will allow the reader to understand the extensions required to seamlessly integrate the new features required for outdoor scenes.

3.1 Hierarchical Elements

At the heart of our hierarchical simulation framework is the identification of a scene with a *complete* hierarchy of *H-elements*, connected by *links* which represent light exchanges. The hierarchy is complete in the sense that its highest level encompasses the entire scene. H-elements can be of different nature, such as clusters (i.e., groups of objects), or portions of surfaces, but they all share a number of properties and characteristics, allowing the specification of many computational operations at an abstract level. H-elements all possess radiosity and reflectance information, as well as a set of links representing energy received at their level of the hierarchy. Specialised H-elements dealing with the specifics of clusters, or polygons, are obtained by subclassing the abstract level and overloading appropriate functionality.

3.2 Computational Operators

In the case of our clustering approach, input objects (e.g., surfaces, meshes etc.) are first grouped into clusters, and an initial self-link of the root node to itself is established. This link is a (very coarse) representation of all energy transfers in the scene. To achieve the desired quality of lighting simulation, the following sequence of operations needs to be performed [17, 2, 18].

1. Links between H-elements are first refined using a *refinement engine*, resulting in the subdivision of the corresponding H-elements where appropriate. This is described in more detail in the following paragraph.
2. Once the refinement process is complete, links exist between H-elements at various levels. Irradiance is now transported (*gathered*) across these links.
3. To maintain a consistent view of radiosity at every level of the hierarchy, irradiance is pushed (added) down the hierarchy, and radiosity is pulled (averaged) up.
4. After an appropriate number of iterations of the refine-gather-push/pull operations, the subdivided H-elements are displayed using the computed radiosity values.

Each one of these steps is defined at the highest possible, and thus most abstract, level of the class hierarchy. For operations such as gather and push-pull, few parts of the process require the definition of specialised methods.

3.3 Hierarchical Refinement Engine

The refinement engine (or "refiner") is the core of the hierarchical lighting simulation process. The refiner operates on links between two hierarchical elements.

The refiner first estimates how well the link represents the light transfer in question. This is typically performed by an error based criterion with respect to a user-defined tolerance. An example is the "BF" (radiosity times form-factor) criterion [6] Improved approaches exist [10, 5], using upper and lower bounds on the energy transfer across the link. If the link is judged of insufficient quality, it is refined, and thus one of the H-elements is typically subdivided, resulting in the creation of *sub-links*.

The estimation of the error criteria, and the subsequent gather operation on a link requires the calculation of an estimate of the form-factor between two H-elements or the irradiance received on an element due to the illumination from the other. This in turn requires an appropriate sampling of a kernel function involving the position, orientation, distance and relative visibility of the two elements. As shown in [18] the calculation

of form factors can be expressed at an abstract level using virtual functions with specialised behaviour for clusters or surfaces. A typical refiner therefore operates solely on H-elements and relies on virtual functions to perform the necessary subdivision (split).

In what follows, we show how the treatment of generic complex meshes and natural lighting can be integrated into this unified framework.

4 Geometry

4.1 Dealing with Geometric Complexity of Outdoor Scenes

For many outdoor scenes, we are presented with complex mesh representations (terrains, large man-made structures etc.). A simple, direct solution would be to pass these elements in an unstructured manner to a clustering algorithm [18]. As mentioned previously, this could result in inappropriate groupings of surfaces since the implicit mesh structure and connectivity information is lost. In addition, this would encumber the system with a large total number of polygons, even in the case where a vast majority are never lit or never participate in lighting.

We present a solution to this problem in the form of a general structure which encapsulates hierarchical mesh information. The power of this structure is its ability to hide the complexity of the geometric object (e.g., a terrain mesh or a tensor-product spline surface), by presenting a generic hierarchical interface. As hierarchical subdivision becomes necessary, the structure uses the hidden geometric complexity to create finer hierarchically subdivided levels. As an additional benefit, at the lowest level of subdivision the structure reverts to classical regular subdivision (e.g., quadtrees), with all the advantages this implies in terms of implicit representation and information sharing.

4.2 A Generic Hierarchical Mesh Structure

The definition of a generic hierarchical mesh class may include information permitting subdivision up to a “finest” level. For example, a terrain mesh will be given as an array of vertices describing its finest level of detail. Our generic interface hides this complex geometric information in what concerns the lighting simulation.

In our generic structure no mesh element uses all the geometric information (e.g., vertex array for terrains) constituting the mesh, since each element is approximated using only a small subset. Nevertheless each element has access to the geometric detail data. When subdivided, a mesh element at a given level constructs child elements which, by using more detailed geometric information, are a better local approximation of the mesh.

After each consecutive split, more geometric detail is used by each child element leading to a stage where finer geometric detail is no longer available in the original data. To allow for finer detail the splitting process does not construct additional mesh objects but regular hierarchical polygon elements (quadtrees) instead. This is possible in the context of our uniform framework because meshes and polygons are both derived from a single abstract hierarchical surface class.

Generic Surface Operations To successfully integrate generic mesh operations into our hierarchical radiosity system, we need to be able to maintain all connectivity information at every level of the hierarchy. In addition, the regular (e.g., quadtree or triangular) structures created below the finest level of mesh subdivision must be able to find

their neighbours across boundaries. An example is shown in the case of quadtrees in Fig. 1, although a similar problem exists for smooth surfaces etc.

To deal with this problem, we define an abstract class *generic edge*, allowing us to have a unified interface for neighbour finding, independent of the actual type of mesh in question. Generic edges are defined in an appropriately chosen *parameter space*, e.g. the mapping of the grid of terrain point data to the interval $[0, 1]^2$. Neighbourhood information is most notably needed to correctly perform per-vertex shading (Gouraud shading), and to correctly maintain a restricted subdivision when splitting elements.

A generic edge is defined by two points in parameter space which we call its *support* and parameters on this support defining the beginning and end of the generic edge. During mesh element subdivision, generic edges are defined between the newly constructed children.

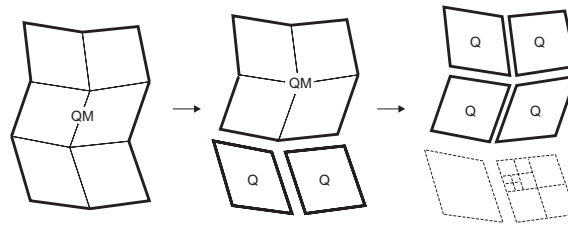


Fig. 1. A sequence of splitting operations, starting with one quad mesh (QM) and resulting in quadrilaterals (Q), which are subdivided into regular quadtrees. Connectivity is preserved across boundaries at *each* level of the hierarchy, due to the generic edge structure.

Neighbour Finding and T-vertex Detection Given the generic edge structure, neighbour finding can be correctly performed using the hierarchical mesh representation. Consider a patch with a generic edge e consisting of the support s and the parameter interval $[t_0 : t_1]$. To find the appropriate neighbouring patch sharing e , the following steps are performed:

1. Step up the mesh element hierarchy until the current element is using the edge e_0 , consisting of s and the parameters zero and one.
2. There must be a sibling of that patch sharing e_0 , which would not have been constructed otherwise. Find this sibling.
3. Step down the sibling's hierarchy to find the smallest element sharing the entire interval $[t_0 : t_1]$ of s .

Given this neighbourhood finding process, we can easily perform mesh restriction and T-vertex identification [15], thus allowing correct colour interpolation across vertices at different levels in the hierarchical mesh.

5 Natural Lighting

The incorporation of natural light sources is of course a key requirement for the creation of realistic simulations in outdoor scenes. The first problem to address is modeling these

sources. For outdoor scenes with moderate scale (such as a house or some part of a city) the sun can be modeled as a parallel source, and the sky is considered a hemispherical source at infinity.

In order to fully integrate sunlight and skylight in a hierarchical radiosity process with clustering, these light sources must become integral hierarchical elements which provide the same interface as the other elements in the simulation [18]. Thus we need to define corresponding H-elements (using specialised classes) which can “split” (be subdivided) to create child elements which represent the source in more detail and with less approximation error.

In addition, particular attention must be paid to the sampling mechanisms used for these sources, for the calculation of energy transfer and form factor estimates. For instance, form factor formulas based on distance calculations cannot be used directly with sources placed at infinity.

5.1 Hierarchical Representation of Skylight

The subdivision of the sky dome is done by a quadtree representation of the hemisphere’s parameter space, expressed by the altitude u and the azimuth v . u is the angle above the horizon, and v denotes the angle in the plane west from south (see Fig. 2). The initial sky patch is the whole hemisphere, $(u, v) \in [0, \pi/2] \times [0, 2\pi]$. Every hierarchical sky patch is therefore defined by a parameter range, and the constant radiance value associated with that range. Note that sky patches are characterised by *radiance* as opposed to radiosity, since they have no area and are positioned at an infinite distance.

The radiance is computed by sampling any given skylight distribution at a fine level of detail and creating a pyramidal representation of the averages of the sampled values. If an element is split, it retrieves the associated radiance value from the pyramid. If an element splits below the finest level of the pyramid, a new value is obtained by subsampling.

The skylight intensity values can also be modulated by a texture. The texture value is simply applied every time the skylight is sampled. When using monochromatic skylight models like the one provided by *Radiance* [20], the texture can be used to assign colours and draw clouds which will affect the simulation results.

Assuming full visibility, the contribution of a sky patch q with parameter range $[u_0, u_1] \times [v_0, v_1]$ to the irradiance on a given scene element p is obtained as follows:

$$\delta L_i^{(p)} = \mathcal{R}_{\sqrt{\cdot}\Pi} \mathcal{S}_{\sqrt{\cdot}} \Omega_{\Pi} \mathcal{J}_{\Pi} \quad (1)$$

where $\mathcal{R}_{\sqrt{\cdot}\Pi}$ is p ’s *receiver factor*, $\mathcal{S}_{\sqrt{\cdot}}$ is p ’s *scale factor*, Ω_q is the solid angle subtended by q when viewed from the any point of the scene, and J denotes radiance. \mathcal{R} and \mathcal{S} depend on p ’s element type: For surfaces, \mathcal{R} is the cosine between the surface normal and the line through the two elements’ samples and \mathcal{S} is 1. For volumes and clusters, \mathcal{R} is 1 and \mathcal{S} is $1/4$ [18]. In the case of partial visibility, a multiplicative correction factor is applied to model the fraction of mutual visibility [6].

The solid angle Ω_q is given by

$$\Omega_p = (v_1 - v_0) \cdot (\sin u_1 - \sin u_0). \quad (2)$$

5.2 Parallel Sources and Sunlight

Parallel sources are easy to implement because they never need to be split, and the irradiance cast on an element is just the source’s flux density modified by the receiver’s orientation and element type (receiver and scale factor). We thus obtain the irradiance of a parallel light q on a scene element p as follows (again assuming full visibility):

$$\delta L_i^{(p)} = \mathcal{R}_{\sqrt{\cdot}\Pi} S_{\sqrt{\cdot}\Pi} \phi_{\Pi} \quad (3)$$

where ϕ_q denotes the source’s flux density.

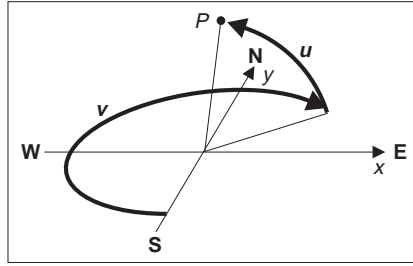


Fig. 2. Parameter space for a skylight object with u and v denoting altitude and azimuth

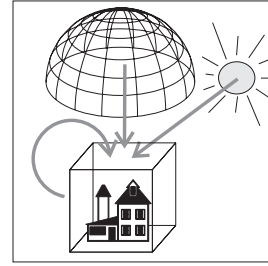


Fig. 3. Initial links for a scene with a skylight and a sunlight object

5.3 Adapting the Refinement Process

We refer to sunlight and skylight as “external” light sources because they are placed outside the scene geometry: therefore they are not included in the extent of the top-level cluster of the scene hierarchy, and their effect is not represented by the top-level “self-link” of the root cluster. To take into account the effect of external light sources, at the beginning of the simulation one link from each external light source to the scene’s root cluster is created (in addition to the top-level self-link). In the presence of n external lights, the simulation would thus start with $n + 1$ initial links. At this stage, the link from the skylight is established from the toplevel sky description. Figure 3 shows an initial link setup for a scene with sunlight and skylight.

Next, a refiner is called to recursively examine and subdivide these links. Because a refiner typically uses generic sampling routines to query the geometry of the elements considered, some modifications are needed to properly cope with external light sources. In particular, for special sources such as sunlight and skylight, the estimation and computation of irradiance must be left to the sender. In this way we can ensure that for ordinary surface-to-surface links the well-established form factor formulae are used to compute irradiance, and for external sources our new formulae are applied. Thus, the refinement engine does not need to know the type of the objects interacting with each other and uses a generic refinement procedure.

6 Results

The methods for hierarchical treatment of natural light sources and of terrain meshes have been implemented in BRIGHT, a hierarchical radiosity system with clustering, using a BF refinement engine. The skylight intensities and colours were computed using parts of the *Radiance* software system by Greg Ward [20] and modulating these monochromatic values by a texture. Any other skylight model could be used instead. All timings are on a Silicon Graphics Indy R5000 (150Mhz) machine.

6.1 Terrain

Figure 4 shows a terrain mesh illuminated by a sky dome and a parallel sunlight in the morning of a day in August. The radiosity solution for Figure 4 took about 16 minutes. The terrain scene is built from 30,373 polygons. The hierarchical subdivision resulted in 90,899 links and 33,967 leaf elements.

6.2 Natural Lighting

Figure 5 shows a house and several trees on a smaller terrain mesh. The total number of input polygons is 20,260. The scene is illuminated only by a hierarchical skylight object. The house casts (soft) shadows which are caused by the brighter parts of the sky.

If the same scene is also lit by a parallel sunlight source as in Figure 6, the shadows become sharper and the trees begin to cast shadows on the house walls and the ground. Fig. 7 shows the same scene from a different direction. Note the bright area in the sky which produced the subtle shadows of Fig. 5.

The hierarchical solution for Fig. 5–7 resulted in around 49,017 links and required approximately ten minutes. There are a total of 64,674 leaf elements. By selecting a higher error tolerance, our hierarchical algorithm can compute a coarser approximation of the natural illumination in the same scene in 148 seconds, using a total of only 1800 links (see Fig. 8) and resulting in less than 30,000 leaf elements, but maintaining comparatively good visual quality.

It is interesting to visualise the hierarchical structure of the solution by displaying the links to different levels of the hierarchy. Sky patches with low intensity are linked quite high up in the hierarchy (Fig. 9).

7 Conclusions

A new *hierarchical* solution to the problems of efficient and accurate outdoor lighting has been presented. A hierarchical abstraction was introduced, permitting the integration of moderately complex outdoor objects (such as terrain maps) and natural lighting (sky- and sun-light) into a unified hierarchical illumination framework.

In particular, an object-oriented approach was introduced to treat complex mesh structures using generic operators. The problem of hierarchically sampling the sky dome was then addressed and an appropriate refinement strategy described. These two contributions allow the treatment of complex, naturally-lit outdoor scenes, based on clustering and hierarchical radiosity, resulting in efficient and accurate lighting simulations.

In contrast to the only previous radiosity-based approach to outdoor lighting [12], our new method requires computation times for scenes of comparable size which are lower than those reported in [12]. More importantly, the hierarchical nature of our new algorithm ensures that global illumination effects are always accounted for at the chosen level of approximation, which is not possible in the progressive-refinement approaches previously used.

In future work, we will be investigating the use of the generic mesh framework for general complex mesh structures, which will require the correct definition of the parameterisation for non-regular structures. In terms of natural lighting, the sampling and refinement processes can be greatly improved, in the spirit of the error-based approaches of [11] or [5]. This will result in higher quality images with a smaller number of links and consequently lower computation times.

Acknowledgements. The first two authors are currently with the computer graphics group of the University of Erlangen-Nuremberg and performed their research at iMAGIS in Grenoble as part of the European student exchange program ERASMUS ICP-95-E-4060. Thanks to Frédo Durand for help with the sky texture.

References

1. CIE. Standardization of luminance distribution of clear skies, 1973. Publication TC 4.2.
2. Michael F. Cohen and John R. Wallace. *Radiosity and Realistic Image Synthesis*. Academic Press Professional, San Diego, CA, 1993.
3. Y. Dobashi, K. Kaneda, T. Nakashima, H. Yamashita, T. Nishita, and K. Tadamura. Sky-light for interior lighting design. In *Computer Graphics Forum*, volume 13, pages 85–96. Eurographics, Basil Blackwell Ltd, 1994. Eurographics '94 Conference issue.
4. Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Werner Stuetzle. Multiresolution analysis of arbitrary meshes. In R. Cook, editor, *SIGGRAPH 95*, Annual Conference Series, pages 173–182. ACM SIGGRAPH, Addison Wesley, 1995. held in Los Angeles, California, 6–11 August 1995.
5. S. Gibson and R. J. Hubbard. Efficient hierarchical refinement and clustering for radiosity in complex environments. *Computer Graphics Forum*, 15(5):297–310, December 1996.
6. Pat Hanrahan, David Saltzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. *Computer Graphics*, 25(4):197–206, August 1991. Proceedings SIGGRAPH '91 in Las Vegas (USA).
7. Kazufumi Kaneda, Takashi Okamoto, Eihachiro Nakame, and Tomoyuki Nishita. Photo-realistic image synthesis for outdoor scenery under various atmospheric conditions. *The Visual Computer*, 7:247–258, 1991.
8. R. Victor Klassen. Modeling the effect of the atmosphere on light. *ACM Transactions on Graphics*, 6(3):215–237, July 1987.
9. Eric Languénou and Pierre Tellier. Including physical light sources and daylight in a global illumination model. *3rd Eurographics Workshop on Rendering*, pages 217–225, May 1992.
10. Dani Lischinski. *Accurate and Reliable Algorithms for Global Illumination*. Ph.D. thesis, Cornell University, Ithaca, NY, 1994.
11. Dani Lischinski, Brian Smits, and Donald P. Greenberg. Bounds and error estimates for radiosity. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 67–74. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.
12. Stefan Müller, Wolfram Kresse, and Frank Schoeffel. A radiosity approach for the simulation of daylight. In *Eurographics Rendering Workshop 1995*. Eurographics, June 1995.
13. T. Nishita and E. Nakamae. Continuous tone representation of three-dimensional objects illuminated by skylight. *Computer Graphics*, 20(4):125–132, August 1986. Proceedings SIGGRAPH '86 in Dallas (USA).

14. J. Rossignac and P. Borrel. Multi-resolution 3D approximation for rendering complex scenes. In *Second Conference on Geometric Modelling in Computer Graphics*, pages 453–465, June 1993. Genova, Italy.
15. Hanan Samet. *Applications of Spatial Data Structures*. Addison-Wesley, Reading, Massachusetts, 1990.
16. François Sillion and George Drettakis. Feature-based control of visibility error: A multi-resolution clustering algorithm for global illumination. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 145–152. ACM SIGGRAPH, Addison Wesley, August 1995.
17. François Sillion and Claude Puech. *Radiosity and Global Illumination*. Morgan Kaufmann, San Francisco, 1994.
18. François X. 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, September 1995. ISSN 1077-2626.
19. Katsumi Tadamura, Eihachiro Nakamae, Kazufumi Kaneda, Masshi Baba, Hideo Yamashita, and Tomoyuki Nishita. Modeling of skylight and rendering of outdoor scenes. In R. J. Hubbard and R. Juan, editors, *Eurographics '93*, pages 189–200. Blackwell, 1993.
20. Gregory J. Ward. The RADIANCE lighting simulation and rendering system. In Andrew Glassner, editor, *Proc. SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 459–472. ACM SIGGRAPH.

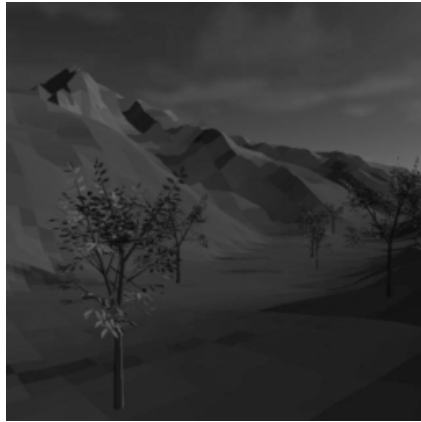


Fig. 4. Terrain, lit by sky-dome and sun.



Fig. 5. House, lit by sky-dome only.



Fig. 6. House lit by sky-dome and sun.



Fig. 7. House from left, view towards the sun.



Fig. 8. House, computed with less detail.

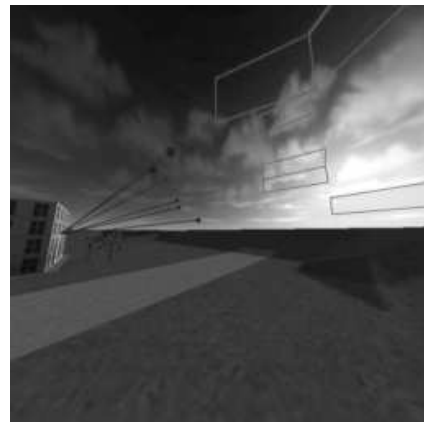


Fig. 9. Entire wall linked to large sky patches.

Algorithmes interactifs : rendu à base d'images, rendu haute qualité et environnements mixtes, réels-virtuels

Dans les chapitres précédents, nous nous sommes intéressés au départ aux algorithmes permettant le rendu de haute qualité, et ensuite au rendu par simulation de l'éclairage pour des scènes de grande complexité. Pour les solutions de radiosit , et en partie pour les solutions pour les environnements non-diffus pr sent es dans le chapitre pr c dent, nous pouvons nous d placer dans la sc ne, mais nous ne pouvons pas la modifier. Ceci est une forme d'interactivit  limit e.

Dans ce chapitre, nous pr sentons nos travaux plus r cents, qui ont tous un but commun, celui du rendu *interactif*. Nous commencons par le rendu   base d'images, qui permet un rendu simple (sans  clairage) mais pour des sc nes tr s complexe tel qu'une ville par exemple. Ensuite, nous pr sentons deux approches diff rentes pour un  clairage interactif de haute qualit . La premi re est une am lioration de la radiosit , et l'autre une m thode pour un rendu interactif en utilisant le lancer de rayons. Enfin nous concluons avec nos travaux dans un domaine nouveau et prometteur, qui permet le rendu interactif pour des sc nes *mixtes*, contenant   la fois des objets r els et des objets virtuels. Ce travail n cessite bien entendu des outils emprunt s   la vision par ordinateur pour la reconstruction g om trique. Nous montrons que la combinaison de ces approches avec la radiosit  et d'autres algorithmes tels que la g n ration de textures par exemple, nous permet d'interagir dans des environnements mixtes.

4.1 Rendu Interactif   Base d'Images

Pour arriver   faire un rendu interactif des sc nes contenant des centaines de milliers voire des millions d'objets. Une solution est de remplacer de la g om trie par des images. Ces techniques ont beaucoup emprunt  aux algorithmes d velopp s en vision par ordinateur.

Avec Fran ois Sillion et son stagiaire DEA B. Bodelet, nous avons d velopp  un algorithme pour l'affichage efficace de sc nes urbaines [SDB97]. Durant le d placement d'un observateur sur une rue, la partie du mod le de la ville qui n'est pas voisine des p t s de maisons environnantes est repr sent e par des images. Ceci est illustr  par la Figure 4.1. En haut nous voyons le mod le complet ;   gauche nous montrons l'image depuis le point de vue courant et   droite une vue d'ensemble, pour rendre compte de la complexit  totale de la sc ne. Au milieu nous voyons l'imposteur, qui repr sente la g om trie lointaine ; la vue d'ensemble montre qu'il ne s'agit que d'une petite partie de la sc ne totale. Enfin en bas, nous montrons la vue de l'utilisateur (toujours   gauche), qui montre que l'imposteur plus la g om trie locale donnent une image tr s proche de celle du d part, m me apr s des d placements du point de vue.

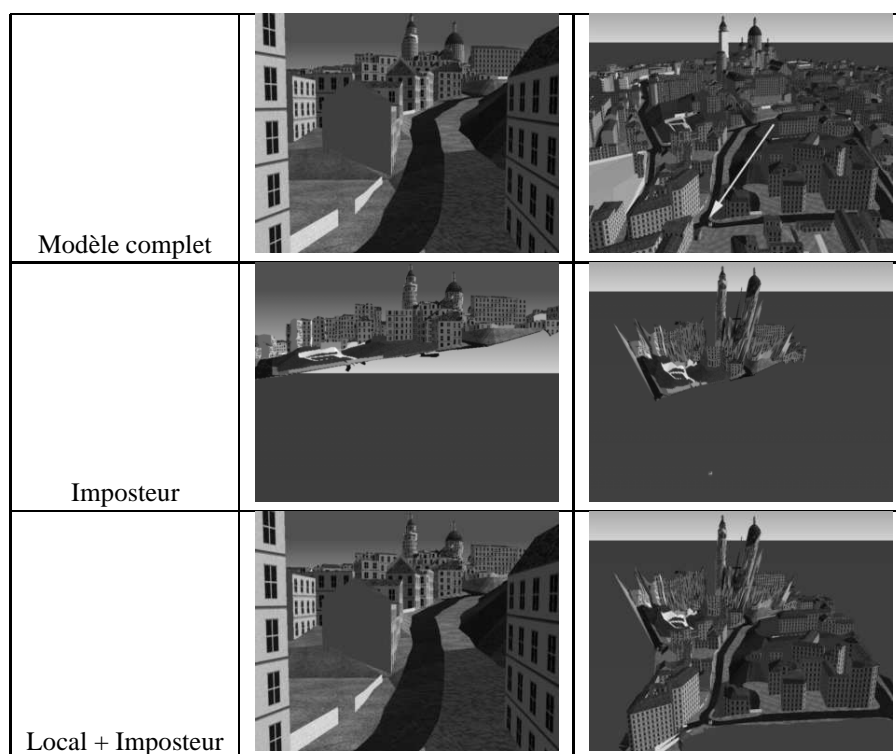


FIG. 4.1: Principes de la construction d'un imposteur. À gauche nous montrons la vue pour un utilisateur se promenant dans la rue. À droite en haut nous montrons cette position par une flèche. À droite nous voyons la vue d'ensemble.

En ajoutant de l'information de profondeur et de l'information de disparité, nous construisons un « imposteur », c'est-à-dire un maillage 3D représentant la partie éloignée. Cette approche est illustrée par la Figure 4.2, où nous voyons la texture de départ (a), l'image de profondeur (b), un contour extrait par des méthodes standard de Vision (c), ainsi que les lignes de disparité (d). À partir de ces lignes une triangulation est faite (e).

Contrairement aux méthodes précédentes de rendu à base d'images, ces imposteurs restent valables pour plusieurs positions de l'observateur, grâce au rendu 3D et à l'information de profondeur liée au maillage. Ceci est montré dans la Figure 4.2 (f) où l'image est rendue d'un point de vue autre que celui du départ. Nous voyons que la coupole est bien cachée, ce qui montre clairement les effets de parallaxe.

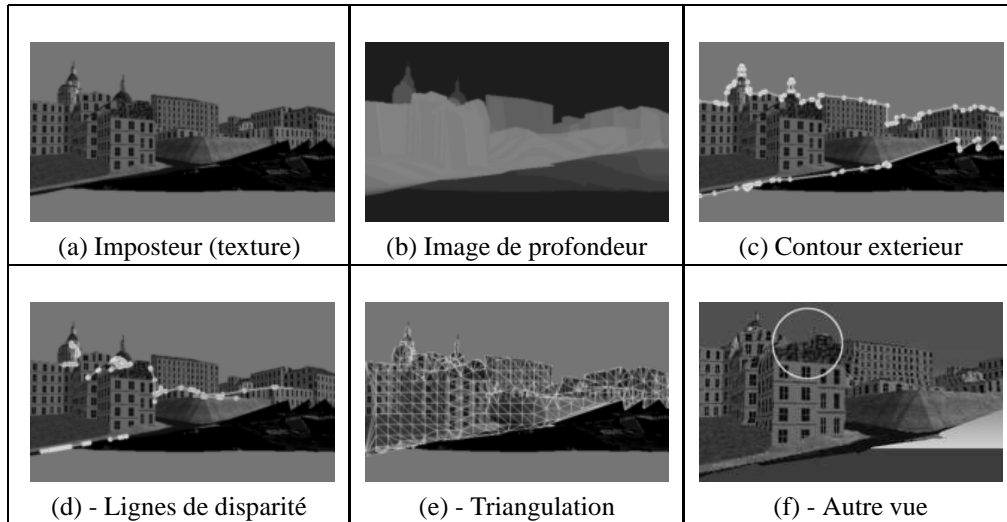


FIG. 4.2: Illustration des étapes de la construction de l'imposteur. La dernière image montre l'imposteur rendu d'un autre point de vue. Les effets de parallaxe sont visibles.

4.2 Rendu Interactif de Haute Qualité

Le rendu à base d'images présenté précédemment n'est pas directement adapté à un rendu de haute qualité, comprenant en particulier des effets d'éclairage global.

Dans la suite, nous développons deux approches très distinctes pour approcher ce but. La première se base sur une amélioration des algorithmes de radiosité hiérarchique, en utilisant toute la panoplie des algorithmes et des structures de données nécessaires à ces méthodes. La deuxième présente une approche très différente, s'appuyant exclusivement sur le rendu par lancer de rayons.

Une discussion des avantages et inconvénients de chaque méthode sera présentée au paragraphe 4.4.

4.2.1 Radiosité Interactive pour des Scènes Dynamiques

Une limitation majeure des algorithmes précédents de radiosité est le fait que le déplacement d'un objet entraîne un coût de calcul de l'éclairage très élevé, souvent égal à celui nécessaire pour créer la première image. Même avec les algorithmes de clustering ce coût est de l'ordre de dizaines des minutes, ce qui exclut l'utilisation interactive.

En partant de l'algorithme de clustering, nous avons ajouté une nouvelle représentation hiérarchique de l'espace de segments de droites contenus dans des liens entre deux objets. Ceci est illustré par la Figure 4.3, où nous montrons comment un lien est subdivisé. Les relations parent-enfants dans l'espace de droites ne sont pas codées séparément ; elles sont contenues dans la hiérarchie d'éléments, (Fig. 4.3 (c) et (d)).

Nous identifions l'espace modifié par le déplacement d'un objet en mouvement, par une traversée efficace de cette nouvelle hiérarchie. Les liens énergétiques affectés sont ainsi identifiés, permettant leur mise à jour rapide. Ceci est illustré par la Figure 4.4(b).

La solution de radiosité (représentée hiérarchiquement) est également mise à jour, en restreignant les modifications dans la partie de la hiérarchie d'objets et clusters véritablement affectée. Un exemple est montré dans la Figure 4.4(c). Pour y arriver, nous avons introduit une numérotation de la hiérarchie évitant ainsi de multiples parcours, qui ajoutaient un coût inacceptable.

Notre algorithme permet ainsi à l'utilisateur d'interagir avec une scène en déplaçant des objets, avec un temps de re-calcul de l'éclairage rapide. Ce re-calcul permet un affichage de quelques images par seconde

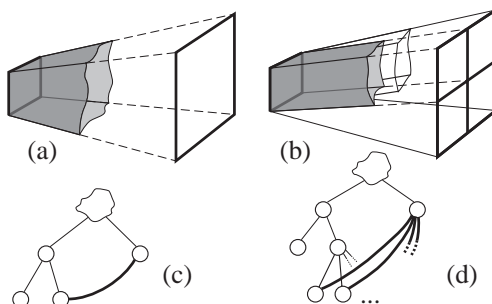


FIG. 4.3: Nous illustrons une hiérarchie de l'espace de segments : (a) le lien d'origine l^p (que l'on montre en faisceau) est subdivisé, créant ainsi quatre sous-liens (b), qui sont les enfants de l^p dans l'espace de droites. Les liens subdivisés (c) et les liens enfants (d) sont codés par la hiérarchie géométrique elle-même.

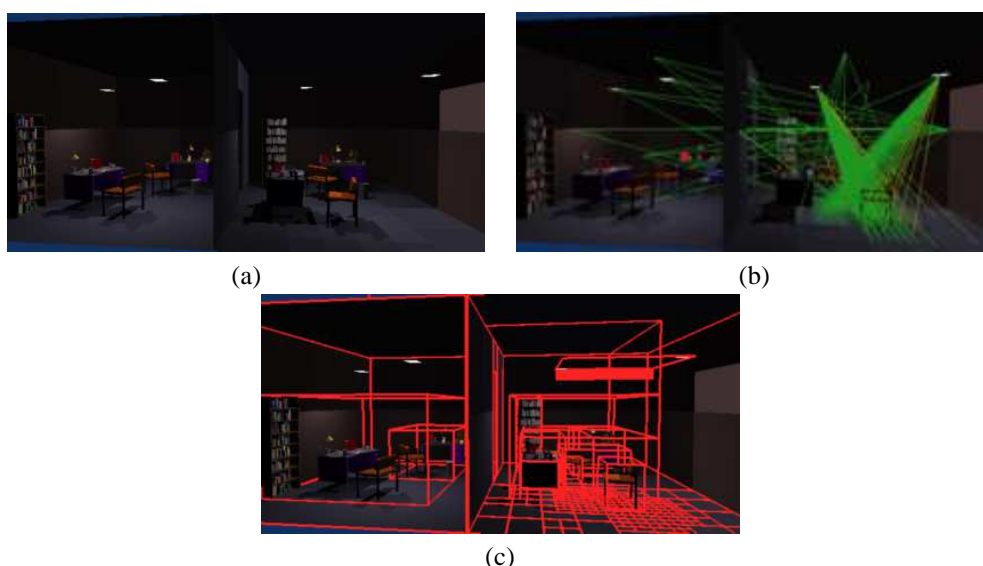


FIG. 4.4: Cet exemple contient une 14,572 polygones d'entrée. La chaise dans la pièce de droite se déplace vers la droite (a) - (b). La mise à jour nécessite autour de 0.3 secondes. Dans (b) nous montrons les liens identifiés comme modifiés, et en (c) la partie de la hiérarchie marquée comme « changé ». Notez que plusieurs milliers d'objets dans la pièce de gauche ne sont pas affectés.

pour des scènes de dizaines de milliers d'objets d'entrée [DS97], tout en traitant l'éclairage global. Nous avons enfin introduit une méthode simple pour contrôler le temps de mise à jour requis par l'algorithme à chaque pas de temps.

Grâce à notre nouvel algorithme, il est pour la première fois envisageable d'utiliser un rendu à base de solution de radiativité pour des applications interactives, comme le design, les maquettes virtuelles etc.

4.2.2 Rendu Interactif par Tracer de Rayon

Traditionnellement les systèmes interactifs de rendu sont limités par la vitesse des algorithmes produisant des images. La plupart du temps, comme c'est le cas pour tous les algorithmes décrits précédemment dans ce document, nous calculons toujours une image complète. Les accélérations portent sur les moyens d'y parvenir plus rapidement, souvent en utilisant des structures de données coûteuses et compliquées.

La nouvelle approche [WDP99] présentée dans ce paragraphe (développée dans le cadre du postdoc de Bruce Walter) se distingue de ce modèle en séparant la mise à jour de l'image de l'algorithme de rendu. Ainsi, une procédure séparée, appelé procédure d'affichage s'occupe de la production de l'image. Cette

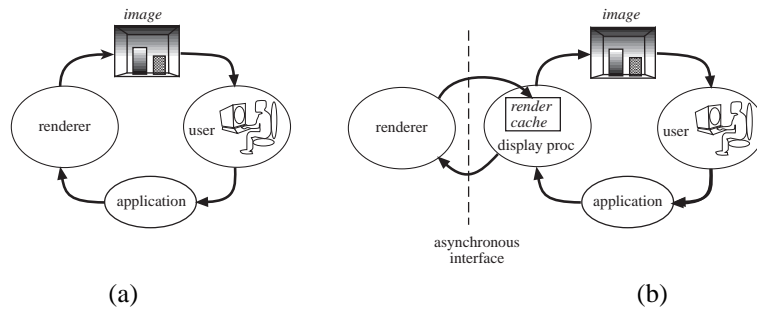


FIG. 4.5: (a) La boucle d'interaction classique, limitée par la vitesse de l'algorithme de rendu. (b) La boucle modifiée, en utilisant le *render cache* où on sépare le temps de mise à jour de la vitesse du rendu.

image est créée avec l'information partielle, fournie d'une façon asynchrone par le système de rendu. Ces deux modèles d'interaction graphique sont opposés dans les Figures 4.5(a) et (b).

Un autre aspect très différent de cette approche par rapport aux algorithmes présentés dans les chapitres et paragraphes précédents, est que cette méthode se base sur les algorithmes de rendu pixel-par-pixel, et notamment le tracer de rayons. L'avantage de cette approche est que le problème du rendu devient un problème d'échantillonnage ; nous pouvons donc appliquer des méthodes de reconstruction puissantes pour rendre une image approximative mais acceptable avec peu d'échantillons.

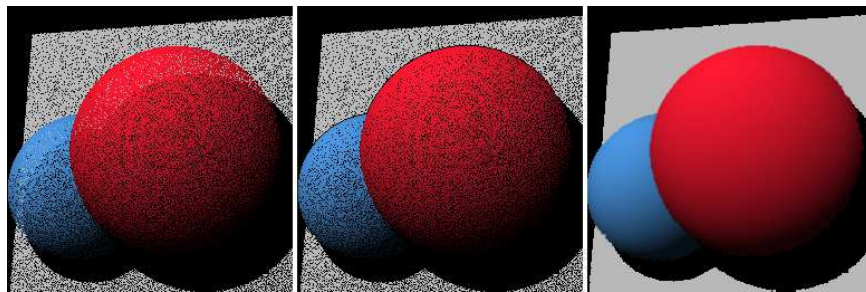


FIG. 4.6: Exemple de la reconstruction d'une image : Les points projetés (gauche) sont filtrés par la méthode d'exclusion par profondeur (depth-cull) (milieu) et par interpolation (droite). Le résultat final est l'impression d'opacité et d'occlusion acceptable.

Le processus est montré dans la Figure 4.6. Un tampon (*render cache*) de points est stocké pendant un mouvement du point de vue ou des objets. Pour générer une image, les points sont projetés sur l'écran, et il y a forcément certains pixels sur lesquels aucun point ne se projette, et certains pixels sur lesquels plusieurs points se projettent. De plus, certains points qui sont cachés par rapport au point de vue actuel restent visibles, car le point correspondant qui se trouve *devant* ne se trouve pas dans le tampon. Le résultat de cette première étape est à gauche dans la Figure 4.6.

Pour corriger les artefacts dus aux erreurs d'occultation, nous regardons les pixels voisins d'un point. Si la majorité de ses voisins n'ont pas la même profondeur, nous rejetons le point. Le résultat est une nette amélioration de la qualité de l'image (Figure 4.6 milieu).

Enfin, un lissage est appliqué aux pixels, dans un voisinage de 3x3 pixels, en n'utilisant que les pixels avec un point projeté. Le résultat final est dans la Figure 4.6 à droite.

Reste à trouver quels pixels sont à re-échantillonner pour l'image suivante. Étant donné qu'au départ nous supposons que l'algorithme de rendu ne peut fournir qu'un nombre faible d'échantillons par image pour un rythme de rendu de plusieurs images par seconde, il est très important de demander des points aux pixels où ils seront le plus utiles. Il est également important de bien étaler les échantillons sur l'ensemble de l'écran pour ne pas avoir une dégradation trop évidente dans une région de l'image.

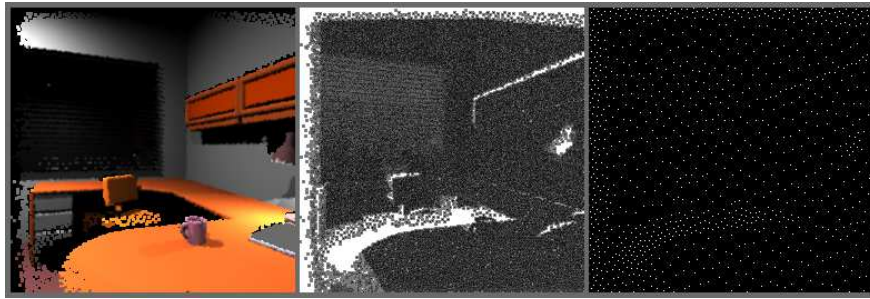


FIG. 4.7: L'image (gauche), l'image de priorité (milieu) et l'image après l'algorithme de *dithering*. L'utilisateur est en train de se déplacer en haut à gauche, et les régions de haute priorité sont celles précédemment cachées. L'algorithme de *dithering* utilisé arrive à la fois à bien étaler les échantillons demandés sur toute l'image, et les concentrer plus dans les régions où ils sont utiles.

Pour faire ceci, en parallèle avec la création de l'image, nous générons une « image de priorité » (voir Figure 4.7 milieu). Après un algorithme de « dithering » transforme cette image de niveau de gris en image binaire. Les pixels blancs sont les échantillons demandés au système de rendu.

Le résultat de cet algorithme est la possibilité d'atteindre près de 10-15 images par secondes, sur des images de résolution de 256x256 à 512x512 avec une qualité montrée dans la Figure 4.8. Plus de résultats et des séquences interactives enregistrées se trouvent à <http://www-imagis.imag.fr/Publications/walter>.



FIG. 4.8: Images d'une série d'utilisations interactives du *render cache*. Notez que toutes les images sont créées par lancer de rayons, à l'exception de l'image en bas à droite, qui est créée par tracer de chemins. Dans cette dernière image on distingue les effets des caustiques (sous les sphère en verre), très coûteux à produire habituellement.

4.3 Réalité Augmentée

Un domaine qui évolue énormément est celui de la réalité augmentée. Nous nous intéressons particulièrement à l'éclairage commun, c'est-à-dire les effets d'éclairage dûs aux interactions des objets réels



FIG. 4.9: La photo de la scène originale, utilisée pour l'extraction de la réflectance.

et synthétiques (ombre portée par un objet synthétique sur un objet réel par exemple).

Ce domaine a suscité un certain nombre de travaux récemment, notamment par l'équipe de l'université de Berkeley [Deb98, YDMH99]. Ces travaux sont orientés principalement vers la création « off-line » des scènes mélangées avec les effets de l'éclairage commun, pour la production des images fixes ou des animations. Dans nos travaux, nous avons insisté principalement sur l'aspect interactif : notre but est de donner la capacité à l'utilisateur d'interagir avec une scène mixte, réelle-virtuelle, tout en ayant les effets d'éclairage commun (ombre entre objets réels-virtuels, effets de lumière des sources réelles et virtuelles etc.).

Éclairage Commun pour la Réalité Augmentée

Les développements récents dans le domaine de la vision par ordinateur [FRL⁺97] permettent la modélisation facile à base d'images. En collaboration avec le projet ROBOTVIS à l'INRIA Sophia-Antipolis nous avons utilisé leur système pour construire un modèle approximatif d'une scène réelle. Pour la scène de la Figure 4.9, nous montrons la géométrie dans la Figure 4.10(a).

En se basant sur ce modèle et en utilisant les résultats de la radiosité dynamique présentée précédemment (paragraphe 4.2.1), nous avons développé une première méthode simple, permettant le déplacement interactif des objets virtuels dans une scène réelle avec des effets d'éclairage commun [DRB97].

En utilisant des méthodes développées par Fournier et al. [FGR93], nous utilisons une estimation très simple de la réflectance de la scène. Ensuite nous avons développé une méthode permettant la représentation de l'éclairage réel dans la scène (réelle) modélisée. Ceci est fait par la subdivision du modèle approximatif par le raffinement de la radiosité hiérarchique (voir Figure 4.10(b)).

Une fois cette représentation faite, nous sommes en position d'utiliser l'algorithme dynamique de [DS97] pour ajouter ou déplacer des objets synthétiques dans une représentation d'une scène réelle. L'ajout des objets synthétiques change forcément le maillage utilisé par la solution de radiosité comme le montre la Figure 4.10(c).

Notre implémentation montre que nous pouvons déplacer interactivement des objets synthétiques ainsi que les effets d'éclairage commun (par exemple une ombre sur une table réelle créée par l'objet virtuel), montré par la Figure 4.11.

Dans cette méthode, nous utilisons le rendu direct du résultat de la radiosité, c'est-à-dire les éléments de la solution (voir Figure 4.10(c) par exemple). Nous modulons leur couleur par un facteur de rapport entre la radiosité avant la modification de la scène et la valeur de la radiosité après la modification.

Les limitations de cette approche sont nombreuses : notamment, la seule modification possible est l'ajout et le déplacement d'un objet virtuel et nous ne pouvons pas modifier les intensités des sources lumineuses réelles, ni ajouter des sources virtuelles. De plus, le rendu direct des éléments de la radiosité ne donne pas toujours un résultat de très bonne qualité.

Le problème central est dans l'estimation de la réflectance. Dans la méthode qui vient d'être décrite, nous n'estimons pas une réflectance véritablement ; une valeur est simplement prise directement des pixels l'image. Comme le rendu se fait exclusivement par un rapport de valeurs, ceci donne un résultat convaincant. Mais si on souhaite changer la lumière réelle, il est nécessaire d'avoir une estimation de la réflectance.

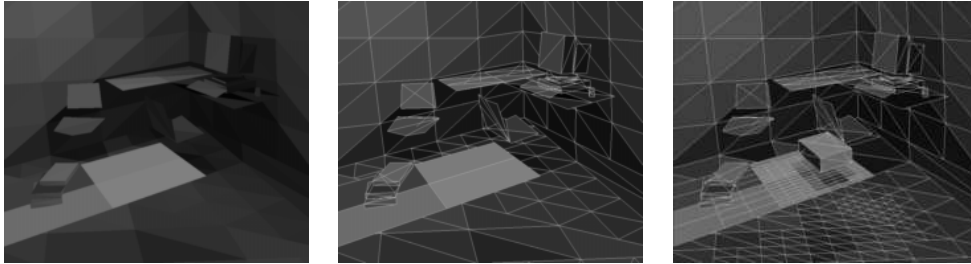


FIG. 4.10: (a) La radiosité, représentant l'éclairage réel de la scène. (b) Le maillage correspondant. (c) La radiosité modifiée après l'ajout de l'objet virtuel.



FIG. 4.11: (a) Rendu avec éclairage commun. Comparez avec Fig. 4.9 ; l'objet sur la table est virtuel, et donne une ombre sur la table réelle. (b) L'objet virtuel se déplace vers la gauche ; la mise à jour nécessite 2.5 secondes.

Ce problème est particulièrement difficile ; nous avons retenu une approche, dans le cadre de la thèse de Céline Loscos (que je co-dirige avec Claude Puech) qui se base sur de multiples images d'entrée [LFD⁺99]. Dans ce travail nous avons utilisé des techniques de l'université de Montréal pour la reconstruction géométrique avec la collaboration de M-C. Frasson et P. Poulin ; B. Walter et X. Granier d'iMAGIS ont également participé.

En particulier, nous prenons plusieurs images du même point de vue, mais en déplaçant une source de lumière connue. Comparez par exemple Figure 4.12, à gauche, la première et la deuxième ligne. Nous voyons que les ombres ont un placement différent, car la lumière a été déplacée.

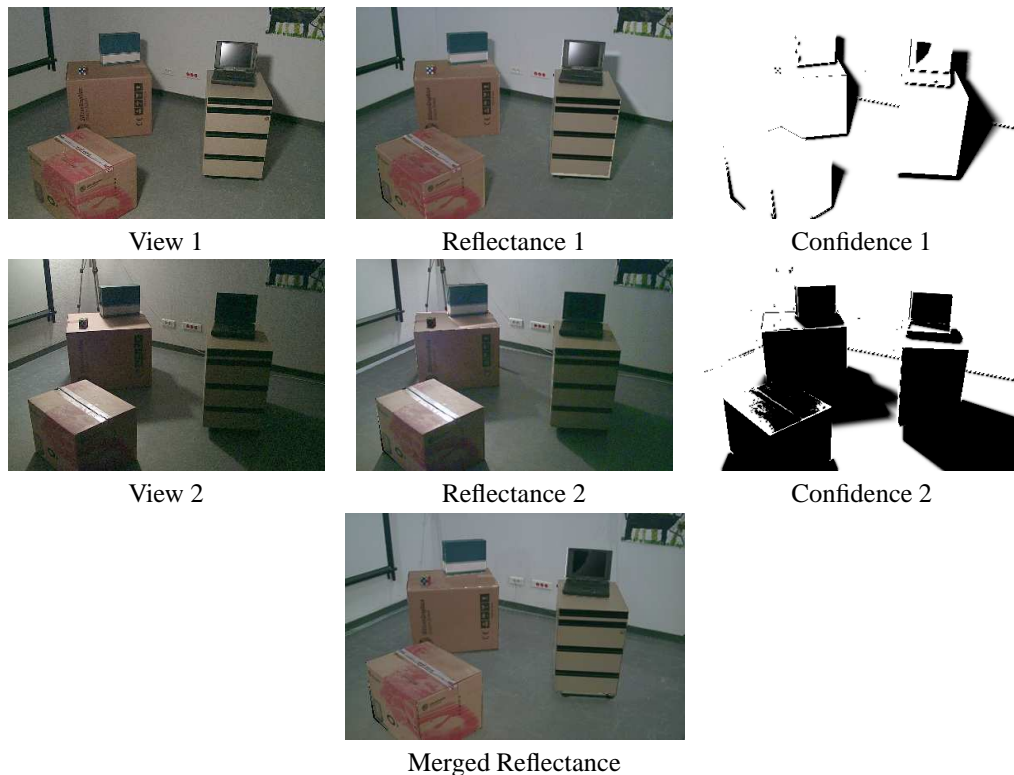


FIG. 4.12: Deux des sept images de radiance utilisées (gauche), les images de confiance (droite), et les images de réflectance (centre). Les valeurs sombres de confiance correspondent à une confiance basse. La réflectance moyenne est montrée en bas.

Deux ensembles d'images sont pris : celles avec le point de vue unique, mais en déplaçant la source, et celles utilisées pour la reconstruction géométrique. Une fois la géométrie reconstruite (en utilisant le système développé à l'université de Montréal [POF98]), nous estimons la réflectance pour chaque point de vue. Pour estimer la réflectance, nous tenons compte de la lumière directe depuis la source (pour chaque position) et de la lumière indirecte en utilisant un terme ambiant. Le résultat est une image de « réflectance », comme on peut le voir dans la Figure 4.12, colonne du milieu.

Une fois l'estimation de la réflectance de chaque point de vue faite, nous faisons une moyenne pondérée de ces estimations. Notamment, nous donnons une valeur de « confiance » à chaque pixel de réflectance par point de vue. Cette valeur est mise à la valeur de la visibilité au départ, donc élevée pour les régions éclairées, et basse pour les régions en ombre. Ensuite nous appliquons une série de filtres pour compenser les imprécisions de modélisation : nous étendons les régions d'ombre par un filtre min. Après nous appliquons un filtre de lissage pour éviter le passage rapide entre zone de haute confiance à des zones de basse confiance. Enfin nous appliquons un filtre de valeurs éloignées de la moyenne (outliers) pour compenser les effets spéculaires, que nous ne modélisons pas. Le résultat de ce processus peut être observé dans la colonne de droite de la Figure 4.12, où l'on voit les images de confiance après le filtrage. Nous faisons

ensuite une moyenne des réflectances individuelles, pondérées par la confiance, ce qui donne l'image de réflectance finale, montrée en bas de la Figure 4.12.

Une fois ce pre-traitement fait, nous initialisons un système de radiosité hiérarchique pour calculer l'éclairage indirect, d'une façon semblable à celle de la méthode précédente. Pour le rendu, nous utilisons le tracer de rayons pour le calcul de l'éclairage direct, et la radiosité pour obtenir la valeur de l'éclairage indirect. Pour des mises à jour rapides, nous identifions les régions de l'écran modifiées pour l'éclairage direct d'une façon efficace, en projetant la boîte englobante de l'objet déplacé et en utilisant la structure hiérarchique des shafts (comme pour la méthode précédente).

Des résultats de cette méthode sont montrés dans la Figure 4.13 Plus de résultats et des séquences interactives enregistrées se trouvent à <http://www-imagis.imag.fr/Celine.Loscos/relight.html>.

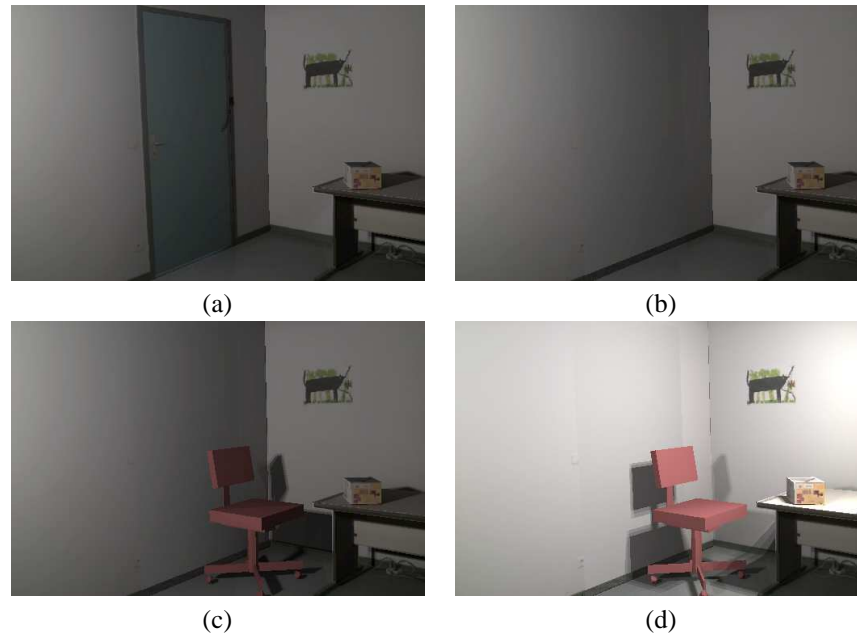


FIG. 4.13: Un exemple d'éclairage commun. (a) Image reprojetée sans modification, (b) enlèvement de la porte (qui nécessite 2.9 sec. résolution 512x340). (c) Ajout d'une chaise virtuelle en 3.4 sec. et (d) d'une source virtuelle (6.6 sec.).

4.4 Discussion

Dans ce chapitre nous avons présenté des travaux sur le rendu interactif. Notre travail se concentre surtout sur le rendu interactif de haute qualité, en utilisant deux approches, la radiosité et le lancer de rayons et également sur la réalité augmentée.

4.4.1 Rendu Haute Qualité

Les avantages de la méthode de radiosité sont liés d'abord sur la rapidité du calcul, surtout pour l'éclairage indirect ; ensuite sur le fait que les cartes d'accélération graphique peuvent être utilisées pour l'affichage, permettant un rendu interactif pour des scènes assez complexes.

Par contre plusieurs inconvénients peuvent être constatés : l'algorithme de radiosité est souvent limité à des scènes diffuses (les améliorations décrites dans le chapitre précédent ne permettant que certains cas limités de comportement non-diffus ; des structures de données lourdes en mémoire sont nécessaires, par la nature de l'algorithme d'éléments-finis ; enfin la gestion des problèmes de raffinement est souvent très complexe, et peut avoir des résultats très visibles sur la qualité de la solution.

Les méthodes de lancer de rayons, ont le grand avantage de ne pas être restreintes aux phénomènes d'éclairage qui peuvent être simulés ; de plus, aucune structure de données type maillage n'est nécessaire. Dans le cadre du « render cache » nous arrivons à conserver ces propriétés favorables, tout en ayant un temps de rendu interactif.

Les inconvénients par contre sont surtout le temps de calcul élevé, en particulier pour l'éclairage indirect. Dans le cadre du « render cache » nous pouvons également noter que malgré des débuts très prometteurs, des artefacts inacceptables subsistent encore, et que pour l'instant la résolution de l'image reste limitée.

4.4.2 Réalité Augmentée

La méthode récente [LFD⁺99], donne des résultats très prometteurs. Les limitations par contre sont d'abord liées à la qualité de la réflectance estimée et la restriction à un point de vue fixe. Les deux questions sont discutées dans les perspectives du chapitre suivant.

4.5 Articles

4.5.1 Efficient Impostor Manipulation for Real-Time Visualization of Urban Scenery (EG'97)

Auteurs : François X. Sillion, George Drettakis et Benoit Bodelet

Actes : Congrès Eurographics '97

Date : septembre 1997

Efficient Impostor Manipulation for Real-Time Visualization of Urban Scenery

François Sillion, George Drettakis, Benoit Bodelet

iMAGIS[†] – GRAVIR/IMAG - INRIA. B.P. 53, F-38041 Grenoble Cedex 9.

Abstract

Urban environments present unique challenges to interactive visualization systems, because of the huge complexity of the geometrical data and the widely varying visibility conditions. This paper introduces a new framework for real-time visualisation of such urban scenes. The central concept is that of a dynamic segmentation of the dataset, into a local three-dimensional model and a set of impostors used to represent distant scenery. A segmentation model is presented, based on inherent urban structure. A new impostor structure is introduced, derived from the level-of-detail approach. Impostors combine three-dimensional geometry to correctly model large depth discontinuities and parallax, and textures to rapidly display visual detail. We present the algorithms necessary for the creation of accurate and efficient three-dimensional impostors. The implementation of our algorithms allows interactive navigation in complex urban databases, as required by many applications.

Keywords: Visualization of large datasets, Image-based rendering, Image caching, Impostors, Urban scenes.

1. Introduction

Visualisation of urban environments is an exciting domain, with a growing number of important applications. Examples of such use include *city planning*, such as the visualisation of South Central Los Angeles, built and visualised by the UCLA School of Architecture for city planning¹, navigation aid systems for automobiles, driving and flight simulators for civilian and military use, climate and environmental studies, virtual tourism and education etc. The sheer size of the data required to represent a city presents particularly challenging problems for visualisation. In addition, the special structure of cities introduces additional difficulties: the volume of data visible from a given point can change drastically, from densely occluded (e.g., only a few neighbouring buildings visible), to panoramic views (e.g., from a hilltop) where millions of geometric primitives are visible.

Since all of the applications mentioned above require real-time feedback, rendering such large data sets using traditional techniques, such as frustum culling, is problematic. In this sense, the problem of urban visualisation can be seen

as a challenge of performing *appropriate* geometric simplification. Two contrasting approaches to simplification have recently become popular, either for visualisation or other applications. The first tendency targets the reduction of the geometric complexity of individual objects. These approaches often implicitly assume the availability of connectivity information, used to maintain a consistent topology². Such information is not readily available for most urban data. Simplification methods which do not preserve topology can make it difficult to control appearance properties such as color³. In addition, geometric simplification algorithms are rarely view-dependent, and thus inappropriate for walkthrough-type applications. The second approach assumes that no information is available other than a set of geometric primitives. As a consequence, complex geometry is substituted with an *image* “impostor”, or “cache”^{4,5,6}. The latter approaches have given impressive results, but are restricted in the type of scene they can treat, since they typically fail for densely occluded environments, and suffer from the fact that the number of frames for which these image impostors are valid is very limited due to the problems of parallax.

The new algorithm we present here introduces a solution which combines the strong points of both approaches, and exploits the specific nature of urban landscapes. On the one hand, we use the idea of image impostors, taking advantage

[†] iMAGIS is a joint research project of CNRS, INRIA, INPG and UJF. Contact E-mail: Francois.Sillion@imag.fr.

of the capabilities of texture for rapid display of detail information, and on the other hand we introduce the idea of impostors augmented with *three-dimensional* information, allowing longer cache life, and largely resolving the parallax problem. In addition, we provide a subdivision model of urban scenes, based on the inherent city structure, resulting in a segmentation scheme which is much better adapted than frustum culling. This segmentation provides a full three-dimensional model for the local neighbourhood (or “near field”), and the use of the augmented impostors for more distant landscape. The availability of the full 3D model for the near field can be very important, especially if operations such as collision detection (e.g., for games) or a faithful simulation of the illumination (interreflections for example) are required. The implementation of our new algorithm allows real-time visualisation of very large urban data-bases; the new three-dimensional impostor method results in longer cache life, and the capacity to treat parallax in many cases.

2. Previous work

The focus of most previous related work has been on the development of algorithms for visualisation of large scenes, notably based on database partitioning and culling, on level-of-detail approaches, and image-based rendering.

2.1. Database partitioning and visibility culling

The first algorithms for accelerating visualisation of large architectural models can be traced to the early stages of computer graphics research^{7,8}. Airey *et al.*, Teller and Séquin, Luebke and Georges^{9,10,11}, use spatial subdivision and attempt to precompute visibility relationships within a complex building scene. The central idea of these approaches is to predict the visible part of a scene for the next few frames, thus reducing the number of primitives which must be rendered. This is achieved using intelligent memory management and viewer motion prediction.

A different approach involved the hierarchical Z-buffer algorithm, which uses Z-buffer pyramids to rapidly eliminate occluded parts of the scene¹². Finally, the issue of constant-frame rate for visualisation of complex environments has been addressed by Funkhouser *et al.*¹³. A computational geometry approach was presented in¹⁴, in which large occluders are dynamically identified as the user moves and used to perform culling using an octree structure. This method successfully eliminates large portions of the model which are invisible but requires large occluders.

2.2. Image-based rendering

The idea of replacing full three-dimensional models by a rendered image can be traced back to the idea of environment maps¹⁵. Their direct application to rendering was used by Chen and Williams¹⁶, using range-images. This approach allowed limited motion around a viewpoint by using

pre-rendered images of the scene, which was then used to substitute three-dimensional rendering on low-end computers. More involved approaches include the use of panoramic images used in Quicktime-VR¹⁷ and plenoptic modelling¹⁸. These methods allow the choice of different viewpoints, and are noteworthy in the fact that they allow limited three-dimensional navigation of real scenes.

Maciel and Shirley introduced the idea of image “impostors”. In their work, a hierarchy of a three-dimensional complex model is created; on the faces of the bounding boxes images of the cluster contents are created. These images can be used to replace the contents when this representation is judged sufficient. In other approaches^{19,5,6} images of distant scenery are cached, and replace complex geometry based on an image discrepancy criterion: when an image replacing complex distant geometry is no longer accurate, the cache is invalidated. The Talisman²⁰ approach moves in the same direction.

Finally, the idea of representing three-dimensional scenes as a light-field was presented by^{21,22}. In such approaches the complexity of the geometry contained in a scene is relatively unimportant, and *slices* of this representation are extracted to generate images.

2.3. Shortcomings of previous approaches

The approaches briefly summarised above have resulted in the efficient visualisation of large data sets, in the case of partitioning and culling for environments which are always densely occluded (such as building interiors), or on the contrary for environments which are never densely occluded in the near-field for the image caching methods.

Inherently, urban environments require the treatment of dense occlusion in the near-field, and that of potentially large data-sets in the distant landscape. The urban data-space segmentation scheme, in conjunction with the new three-dimensionally augmented impostors provides a powerful solution to this challenge.

The dynamic octree culling approach using large occluders¹⁴ could be used as a first stage in our approach, but does not address the case where large amounts of 3D geometry are visible in the distance.

3. An efficient model for urban navigation

Urban models have several distinct properties, unlike other data sets. In general, they have very high geometric complexity, typically requiring very large datasets to represent even the simplest neighbourhood. In addition, even though at the street level there is a high degree of occlusion, it is often the case that the amount of data visible can change abruptly and unpredictably. This occurs for example when turning from a narrow street into a large open square with a

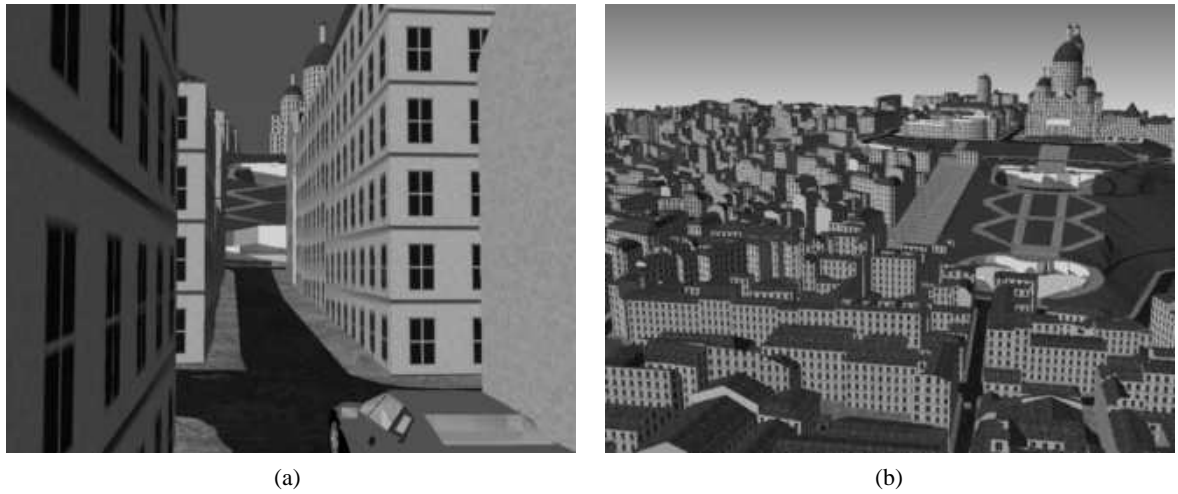


Figure 1: Example view of an urban scene. (a) from the street level, (b) from higher up, looking in the same overall direction (See also the color section at the end of this volume).

view to a hill. Furthermore, urban models are typically quite large, since they extend over several square kilometers.

As a consequence of these properties, traditional methods such as frustum culling are not very well adapted to the needs of urban visualisation. Consider for example the scene depicted in Figure 1: the model is constructed from polyhedral data representing a $2\text{km} \times 3\text{km}$ area of Paris around Montmartre (courtesy of IGN) and consists of 140,800 polygons (note that this rather large number of primitives still only allows a crude geometrical description of the buildings).

The image on the left shows a view taken from street level; clearly only few of the objects are actually visible, but simple culling strategies cannot decide what is relevant for this view and must consider the entire depth of the scene within the frustum. In this example frustum culling selects 45,200 polygons, which still far exceeds the real-time capabilities of current mid-range computers. The full depth complexity of the model can be estimated on the right-hand image taken from above.

Note that culling away objects based on the distance to the viewpoint would not be a good idea since it would eliminate landmarks such as the church in the background. Therefore a difficult character of urban scenes becomes apparent with this example: while much of the complexity is typically hidden (for ground views) due to important occlusion, some directions let the user view objects at far distances.

In the new approach presented here we use the idea of a *local neighbourhood*, which has a natural interpretation in the context of urban landscapes. Cities are organised by streets and blocks, and can thus be directly segmented into a local

region (for example the blocks adjacent to the street we are moving on) and a distant landscape which contains the rest of the city geometry.

The central principle of our approach is to maintain full three-dimensional geometric information of the local neighborhood, while maintaining an approximate view of the distant landscape, which we call an *impostor*. We will be using inherent urban structure to guide this segmentation and the partitioning, as well as to aid in building and maintaining the accuracy of the impostors.

A straightforward approach implementing these ideas would be the use of textures, as impostors, potentially with the aid of some hierarchical structure⁴. However, such an approach suffers from the constant need to update the image caches which have become invalid due to parallax.

A more promising solution consists in using additional three dimensional information to create augmented impostors, containing for example depth information, contours of the sky-scape etc. These 3D impostors have longer “cache life”, compared to simple texture-based approaches and are more accurate since the additional information can be used to control their validity. Furthermore, the segmentation based on the urban structure potentially allows us to select the number and location of impostors in an optimal fashion.

This approach gives us the additional benefit of having more control over the error committed with respect to depth, while maintaining the advantage of textures which capture fine, but distant, detail very compactly. The principles introduced above are illustrated in Figure 2, where we show the local model associated with a selected viewer configuration. An impostor is constructed to complement the local model

for nearby viewpoints. The images on the right clearly show the three-dimensional nature of the impostor.

3.1. Segmentation of the city model

As mentioned above, we propose the segmentation of the model into a nearby local neighbourhood, and more distant landscape, represented by impostors. The segmentation will typically be based on natural urban subdivision (“blocks” divided by streets for example).

We note that the directions along which distant objects can be visible usually correspond to streets or non-built areas, which can be extracted from an analysis of the urban database. Impostors can therefore be associated to these “important” directions, so that they reproduce exact views at well-chosen vantage points in the city, for a given resolution. This segmentation will also permit the combination of distant views in an efficient manner, based on the topological relationships in the arrangement of impostors, further enhancing the “cache-lifetime” of the impostors.

We consider an interactive application in which the user navigates in the urban scene. At any time the viewer location, and direction of sight, are known. Segmenting the model actually means that we partition the space of viewer position and direction of sight into a number of cells, each possessing all the necessary data to quickly render images for viewers in the corresponding neighborhood. Note that the word “neighborhood” here should conceptually refer to a portion of a five-dimensional space (position and direction). However in practice the segmentation operates in spaces of fewer dimensions thanks to the very high structure of urban data.

In the remainder of this paper, we will use the following terminology. For each cell, we define:

- The *local model* as a fraction of the scene 3D model, extracted from the complete model using an appropriate segmentation technique.
- The *distant model* as the remainder of the 3D scene.
- An *Impostor* as a textured three-dimensional object used to render distant geometry.

Each cell can possess one or more impostors as needed. The essence of the rendering algorithm is then to always draw the combination of the impostors of the current cell and its local model, to obtain a view of the entire model.

The desirable characteristics for a segmentation of the model are therefore primarily linked to its ability to represent distant landscape for cells that are as large as possible (to minimize their number), using the simplest possible impostors (to minimize their cost) and with the smallest possible image error (to avoid annoying artifacts). In particular, the use of a local 3D model and a set of impostors raises the issue of the boundary region between the models. Misalignment of the two boundaries results in “cracking” problems which should be minimized. It is very important to fol-

low the structure of the urban landscape to perform the segmentation: $k-d$ trees, quadtrees or other regular structures are not well suited to this task since they will not produce meaningful chunks of data.

3.2. Definition of three-dimensional impostors

A three-dimensional impostor is initially based on the generation of an image which captures small details of the distant urban landscape in the form of texture information, and exactly complements the view of the current cell’s local model for a specific vantage point.

Nonetheless, additional three-dimensional information should be stored with the image from the outset. This is necessary to perform appropriate deformations of the image when the user moves around the initial vantage point, and therefore allow cells to reach a certain size. Instead of deforming, or morphing, the texture image, we can also build from it a collection of three-dimensional elements, which can then be visualised efficiently just like other 3D data.

The main issue for the generation of optimal impostors is the determination of what should be represented as 3D information, and what can be safely left as texture information. 3D information is needed to correctly reproduce parallax effects as the user moves and portions of the distant model occlude other portions. However, small geometric details on facades, for instance, need not be integrated in the 3D information. Thus the addition of three-dimensional information to the impostors can be thought of as a constrained form of geometric simplification, whereby an initial dense mesh (the initial impostor image) is simplified to reduce polygon count while approximating the data within a given tolerance.

Considering that a “simplified mesh” has been created on the impostor image, this mesh can be reprojected in 3D by means of the depth information from a z-buffer, yielding a 3D textured polygonal mesh.

Important desirable properties for impostors therefore center on the ability to reconstruct an accurate view of the distant landscape in a neighborhood around the initial vantage point where the impostor was created. For this we need rich texture information and the relevant 3D structure to recreate important parallax effects, but the 3D complexity should be minimized to avoid excessive costs. The trade-off between geometric and texture detail has been discussed elsewhere²³, in the context of a single object. Simple and accurate validity criteria are also needed to keep reconstruction error in the final image under control. In particular, the “cracking” problem mentioned above should be carefully monitored. In this respect, we see that using simple planar impostors, or image caches^{5,6}, is not a viable solution because we always render large portions of the model on an impostor.

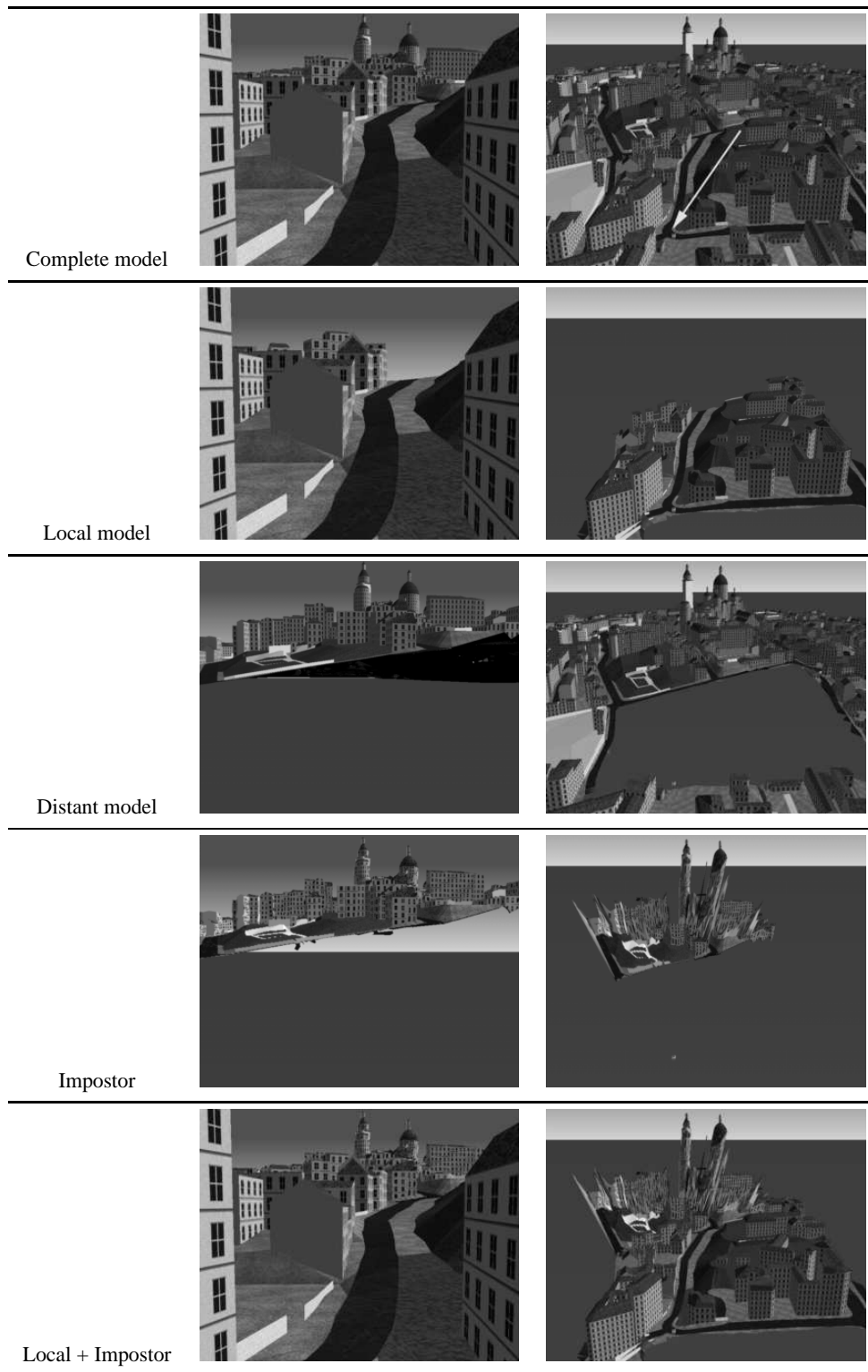


Figure 2: Principles of model segmentation and use of an impostor. The left column shows views created for a user walking in a street (the viewer location is indicated by the arrow in the top right image). The right-hand column shows a bird's eye view of the scene (See also color section).

4. Practical algorithms

The principal ideas outlined above have been used to guide the development of several practical algorithms in our urban visualisation system. We present below the algorithm used to segment an urban model based on street connectivity data and a construction method for three-dimensional impostors.

These approaches are not necessarily the most efficient, but provide a first solution to the specific requirements of the visualisation of complex urban environments. We have implemented all the algorithms presented next in our prototype system, with satisfactory results, which will be presented in the following section.

In our current implementation, we have decided to focus on urban navigation for pedestrians or land vehicles. This particularity is important in that it means that there is a lot of occlusion by nearby buildings along the streets, therefore the directions in which distant landscape is visible are well identified. Flying over a city (for games or other simulation applications) would require different segmentations or different sets of impostors. Note that the difficulty would not arise from the fact that distant objects are visible, but rather from the inability to easily predict in which direction this happens.

4.1. Segmentation of the model

A user walking or driving on the ground is typically constrained to a network of streets, therefore this defines a subset of the model where the viewpoint can be. We can safely restrict our algorithms to the creation of impostors that let us recreate views for such possible points.

Visibility in a street is typically blocked sideways by adjacent buildings, but can extend quite far away in the directions of the street endpoints, or above the adjacent buildings/ground (“local” skyline).

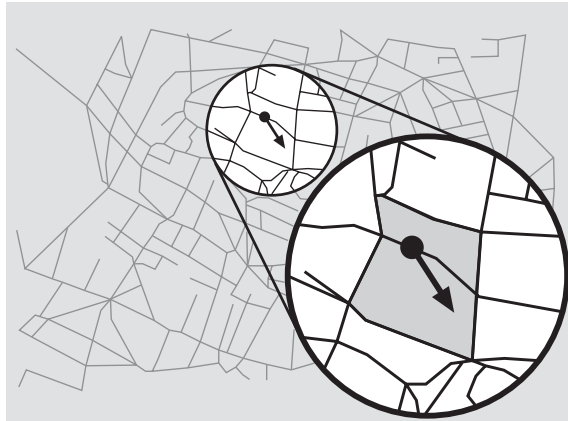


Figure 3: Blocks adjacent to the current street are selected to compose the local model.

A graph of the network of streets is first constructed. In our prototype implementation, this graph is created from auxiliary data depicting the geometry of the streets in Paris. While this data allows us to quickly build a topological structure, an equivalent graph can easily be constructed by hand for a given urban dataset. It is indeed probably desirable to build the graph from the geometrical data, since the use of an independent geometry file actually revealed several misalignment problems, with streets running through houses etc.

The graph is represented internally using a classical winged-edge data structure, allowing fast navigation in the streets, as well as easy access to “neighboring” geometrical information. The area of interest is therefore treated as a 2D polyhedron, whose faces are city blocks (and point to appropriate buildings and landmarks) and whose edges are portions of streets.

We define a cell of the segmented model as an oriented edge of the street graph. Topological information is then used to extract the local 3D model around a point (a street) as the set of blocks “near” that street. The simplest definition of “near” is that blocks must touch the current street segment (Figure 3). We use this definition in our implementation. Other definitions can be used, in particular to ensure smooth transitions between cells as the viewer moves through the database. In particular, a pointwise notion of connectivity, counting as adjacent all blocks sharing a graph vertex, seems important around the endpoints of a street segment. We are currently investigating better segmentations using this notion. Obviously the chosen selection mechanism should depend on the particular morphology of the city model at hand: short and curved street segments place particular requirements in terms of impostor life span, compared to long and straight streets.

Because visibility of distant objects is in practice mostly limited to the directions of the street ends, we associate two impostors with each street (one with each cell, i.e. a directed edge in the graph). This implicitly assumes a densely built environment, where visibility is blocked by buildings along each side of the street.

4.2. Impostor creation

Recall that the goal of the three-dimensional impostor construction is a representation which maintains the advantages of a texture-based image cache, while at the same time containing richer information related to depth and contouring. This additional data permits the reuse of the impostors for many more frames than in traditional image-caching. In our implementation, impostors are created either off-line as a pre-process, or on demand when the user enters a new area of the model. Since impostors are associated to edges, there are twice as many impostors as streets in the model. The main stages of our construction algorithm are listed below, and illustrated by the images in Figure 4. Please refer to these images while reading this section.



(a) - Impostor texture



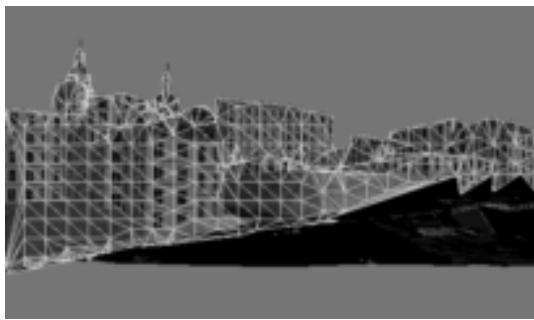
(b) - Depth Image



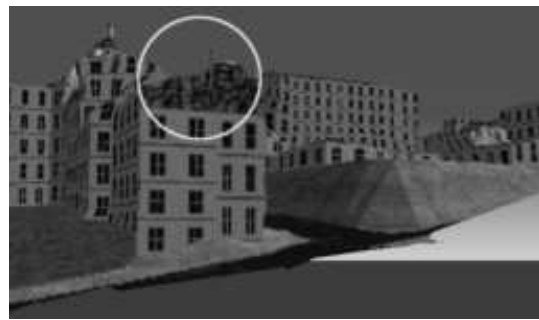
(c) - External contour



(d) - Depth disparity lines



(e) - Impostor triangulation



(f) - Other view

Figure 4: Illustration of the steps taken to create an impostor. The last image shows the impostor rendered from a different viewpoint and demonstrates the appearance changes due to parallax (See also color section).

The successive steps of the impostor creation algorithm are:

1. Create an image of the distant scenery (to be used as the impostor texture).
2. Save the corresponding depth image (contents of the z-buffer).
3. Extract the external contour of the image.
4. Identify the significant depth disparity contours.
5. Perform a constrained triangulation of the impostor.
6. Store the list of 3D triangles along with the texture image.

Creation of the impostor image

We begin by generating an image of the distant landscape (that is, the entire model of which the “local” model has been removed) from a given point of view (Fig. 4-a). In our case we create the view from the edge origin, looking towards the other end. We also temporarily extract a depth image from the z-buffer (Fig. 4-b).

Identification of important features

A standard image-processing contour extraction based on thresholding is then applied to the image to separate actual relevant data from the background ²⁴. While the separation from an empty background (such as for the skyline) is trivial, care must be taken that some distant portions of the model can be visible *from below*, because we are only considering the distant model, and therefore the ground description of the local model is missing from the rendered data. We use a special color to identify the bottom face of ground elements, and use this information in our contour extraction algorithm.

The resulting contour captures all the details of typical city sky-scapes, including spires, rooftops etc (Fig. 4-c), to the resolution of the computed image, and is referred to as the *external* contour.

The external contour can be considered to be a two-dimensional polygon in the image-plane used for rendering (although there may of course be multiple contours, the structure of urban landscapes, with the presence of the ground, is such that in the vast majority of cases there is a single contour in the image. Therefore we will always speak of “the” contour, but our algorithms apply to all available contours). We want to subdivide this polygon into simple polygons such as triangles in the plane, which can then be reprojected in 3D. Because our goal is to allow the recreation of parallax in the impostor, we next identify lines of maximal depth discrepancy (Fig. 4-d).

This is accomplished by creating a depth discrepancy image from the depth image. Depth discrepancy is defined here as the maximum difference between the depth at a pixel and that of its neighbors. We then extract “interesting” contours (after a thresholding operation to select pixels with an important discrepancy) and fit a set of line segments to the resulting contours.

In practice, we apply the following simple algorithm iteratively, until no more pixels with a depth discrepancy above a given threshold are available:

1. Find pixel with largest discrepancy.
2. Follow chain of pixels until we are blocked by the external contour or the discrepancy falls below the threshold.
3. Compute a polygonal approximation of the chain.

Triangulation and 3D reprojection

The external contour and the discrepancy lines provide the important information needed to recover parallax effects within the impostor. To avoid precision issues as well as extreme triangle aspect ratios, we augment the impostor with a set of points selected on a regular grid within the external contour to impose a minimal sampling density. Finally we perform a constrained Delaunay triangulation of all these points, with the constraint that all external contour segments and all discrepancy line segments be included in the triangulation ²⁵ (Fig. 4-e).

All vertices of the triangulation are then reprojected in 3D using the information of the depth image, and the resulting set of 3D triangles, together with the corresponding texture image, constitutes the impostor. Texture coordinates for each triangle vertex are simply the image coordinates of the corresponding pixel location.

A view of the impostor from a viewpoint other than the point of creation is shown in Fig. 4-e. Note the parallax effects, when the most distant buildings become obscured by nearby ones. This effect is obtained exclusively from the perspective rendering of the 3D triangles composing the impostor.

5. Results

We present in Figure 5 comparative images for different viewer positions in the same area. Notice first the overall quality of the images in the central column, produced by the combination of the local 3D model and the impostor (compare to the right-hand column rendered using the complete 3D model). The images in this central column are produced at a sustained rate of between 11 and 19 frames per second, while a complete 3D rendering only achieves between 1 and 2.5 frames per second (using frustum culling and the *Performer* high-performance graphics library).

Second, the three-dimensional impostors succeed in providing an accurate view of distant landscape, taking into account some parallax changes. Note in particular how the church towers in the background correctly recede behind the closer buildings.

Finally a small amount of cracking is visible when the user comes very close to the boundary between the local model and the impostor. This suggests that more work is needed to keep all errors under complete control.

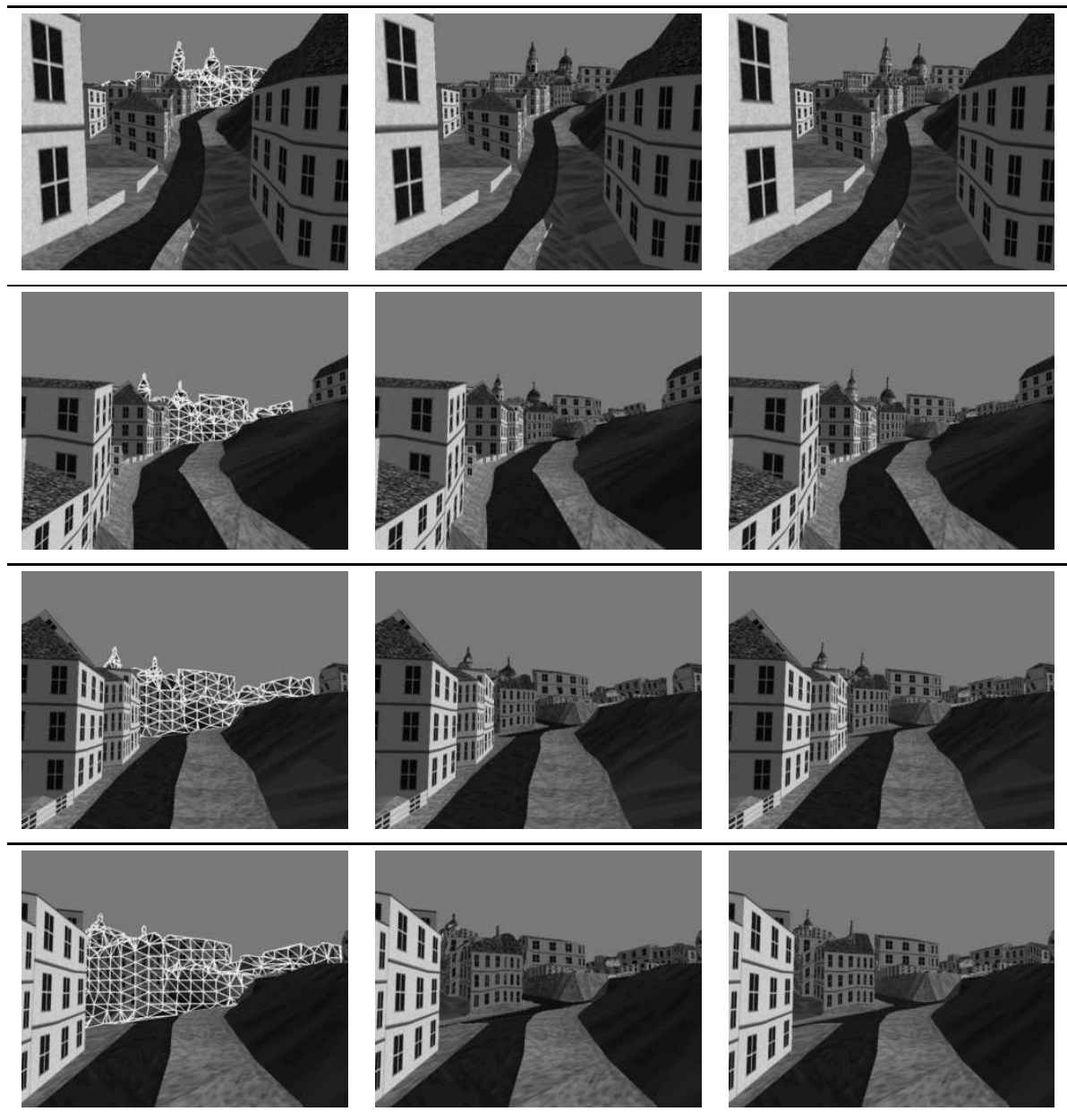


Figure 5: Images extracted from an interactive tour of the Montmartre area. A sustained frame rate of about 15 fps was obtained. In the left column the impostor was colored to make it visible. The center column shows the view displayed during interactive navigation. The right-hand column is a reference view rendered using the entire model (1 to 2 fps). Note the self-masking effects within the impostor, due to the evolution of parallax: in particular the church in the background disappears behind the closest buildings as the viewer moves forward (See also color section).

Note that since the life time of impostors corresponds to the time the user spends in a given street, impostors remain active in areas whose size exceeds several tens of meters.

5.1. Performance

The results of our implementation are quite satisfactory. The algorithm achieves over 11 frames per second for the Montmartre model (recall that this is a model of around 140,000 polygons). All timings were computed on an SGI Indigo² 200 MHz R4400 computer. We observe that the speedup value is only an indication, very much dependent on the complexity of the data. A more complex model containing more detail information on the facades in the form of geometry, for instance, would not affect the complexity of the impostors, and may produce even greater acceleration factors.

In terms of space, a typical impostor, starting with a texture image of 512x512, results in about 1,000 polygons. This is much less than the typical *visible* geometric complexity of the distant model. The average size of the local model is approximately 4,000 polygons, therefore the total amount of 3D geometry being drawn at any given time is reduced to about 5k polygons.

The creation of an impostor takes about 12 seconds on the above computer, of which the initial offscreen rendering takes approximately 10. These figures could therefore be greatly reduced using hardware rendering (for instance with the pixel buffer mechanism available in recent SGI software). All of these operations can be seen as preprocessing, and can be stored with the model.

The above model has 1168 edges. Precomputing and storing all the impostor images clearly represents a significant expense. While the calculation time can be largely amortized by later usage, memory requirements are potentially large. However we observe that very little of this memory is needed at any given time, since very few impostors are active. A concurrent process can therefore be used to pre-fetch impostors in a neighborhood of the viewer.

6. Conclusions and Future Directions

This paper introduces a new framework for the efficient segmentation and visualisation of urban landscapes. The central idea is the efficient segmentation of the urban model based on inherent city information (such as streets and blocks), resulting in two distinct sub-sets at each frame: the local neighbourhood and distant scenery. We use the full three-dimensional model to render the local neighbourhood, thus providing detailed geometric information to the viewer. For distant scenery, we introduce “impostors” of the complex data set, augmented with appropriate three-dimensional information. These impostors are initially based on an image-view of the distant scenery; the contour(s) of the building

sky- and ground-line are extracted, as well as lines of depth discrepancy. The result is polygonalised and reprojected into three-dimensional space using the depth-buffer information. As a consequence, the impostor can be used for a much larger number of frames than previous image-caching techniques. In addition, parallax distortions can be captured to a certain extent.

We have implemented our new approach, achieving near real-time visualisation of an real complex urban model. We have shown how important sky-line features (rooftops, spires etc.) are preserved using our 3D impostors, and that parallax effects such as mutual occlusion of two buildings contained in the impostor, are successfully rendered.

Much research remains to be done. One important issue is the improvement of transitions between cells of segmented models. In the current approach, the impostors are completely correct at the view points for which they were generated. One approach would consist in morphing between impostors. We are currently investigating solutions to this problem, based on storing different views at edge endpoints, and developing appropriate combination algorithms.

Segmentation models should take advantage of the characteristics of urban structure. Different cities, built in different parts of the world in different cultures, can have distinct properties. For example, the sky-scraper sky-lines of North-American cities differ from the large boulevards and monuments of Paris. Such characteristics have an important influence on occlusion and the regularity of visibility changes.

An important issue is the error measure used to determine the validity range for impostors. The addition of three-dimensional information adds different types of distortion, but due to its richer nature can provide improved discrepancy measure compared to simple pixel differences of image caches.

Although the problem of cracking is greatly diminished using 3D impostors, it has not been completely eliminated. Different possibilities exist, such as sharing boundaries or vertices between the model and the impostor. Better heuristics can be used to extract relevant data in impostors, e.g., using decimation techniques.

Finally, we hope that the insight gained by studying urban visualization, and designing algorithms tailored to this application, will prove useful in developing more general solutions to the issue of impostor usage for the visualization of very large databases.

7. Acknowledgements

The authors wish to thank the French “Institut Géographique National” for the TRAPU urban data set, as well as Mathieu Desbrun and Frédo Durand for their help in preparing the images. Computer vision contouring code was courteously provided by Luc Robert and by Radu Horaud. Dani

Lischinski provided crucial help with the Delaunay triangulation code.

References

1. W. Jepson, R. Liggett, and S. Friedman, "An environment for real-time urban simulation", in *1995 Symposium on Interactive 3D Graphics* (P. Hanrahan and J. Winget, eds.), pp. 165–166, ACM SIGGRAPH, (Apr. 1995).
2. M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbury, and W. Stuetzle, "Multiresolution analysis of arbitrary meshes", in *SIGGRAPH 95* (R. Cook, ed.), Annual Conference Series, pp. 173–182, ACM SIGGRAPH, Addison Wesley, (1995). held in Los Angeles, California, 6–11 August 1995.
3. J. Rossignac and P. Borrel, "Multi-resolution 3D approximation for rendering complex scenes", in *Second Conference on Geometric Modelling in Computer Graphics*, pp. 453–465, (June 1993). Genova, Italy.
4. P. W. C. Maciel and P. Shirley, "Visual navigation of large environments using textured clusters", in *1995 Symposium on Interactive 3D Graphics* (P. Hanrahan and J. Winget, eds.), pp. 95–102, ACM SIGGRAPH, (Apr. 1995).
5. J. Shade, D. Lischinski, D. Salesin, T. DeRose, and J. Snyder, "Hierarchical image caching for accelerated walkthroughs of complex environments", in *SIGGRAPH 96 Conference Proceedings* (H. Rushmeier, ed.), Annual Conference Series, pp. 75–82, ACM SIGGRAPH, Addison Wesley, (Aug. 1996).
6. G. Schaufler and W. Stürzlinger, "A three dimensional image cache for virtual reality", in *Computer Graphics Forum, 15(3). Proceedings Eurographics '96* (J. Rossignac and F. Sillion, eds.), pp. 227–236, Blackwell, (Sept. 1996).
7. J. H. Clark, "Hierarchical geometric models for visible surface algorithms", *Communications of the ACM*, **19**(10), pp. 547–554 (1976).
8. C. B. Jones, "A new approach to the 'hidden line' problem", *Computer Journal*, **14**(3), pp. 232–237 (1971).
9. J. M. Airey, J. H. Rohlf, and F. P. Brooks, Jr., "Towards image realism with interactive update rates in complex virtual building environments", in *Computer Graphics (1990 Symposium on Interactive 3D Graphics)* (R. Riesenfeld and C. Séquin, eds.), vol. 24, pp. 41–50, (Mar. 1990).
10. S. J. Teller and C. H. Séquin, "Visibility preprocessing for interactive walkthroughs", in *Computer Graphics (SIGGRAPH '91 Proceedings)* (T. W. Sederberg, ed.), vol. 25, pp. 61–69, (July 1991).
11. D. Luebke and C. Georges, "Portals and mirrors: Simple, fast evaluation of potentially visible sets", in *1995 Symposium on Interactive 3D Graphics* (P. Hanrahan and J. Winget, eds.), pp. 105–106, ACM SIGGRAPH, (Apr. 1995).
12. N. Greene and M. Kass, "Hierarchical Z-buffer visibility", in *Computer Graphics Proceedings, Annual Conference Series, 1993*, pp. 231–240, (1993).
13. T. A. Funkhouser and C. H. Séquin, "Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments", in *Computer Graphics (SIGGRAPH '93 Proceedings)* (J. T. Kajiya, ed.), vol. 27, pp. 247–254, (Aug. 1993).
14. S. Coorg and S. Teller, "Temporally coherent conservative visibility", in *Proc. 12th Annual ACM Symposium on Computational Geometry*, (1996).
15. J. F. Blinn and M. E. Newell, "Texture and reflection in computer generated images", *Communications of the ACM*, **19**, pp. 542–546 (1976).
16. S. E. Chen and L. Williams, "View interpolation for image synthesis", in *Computer Graphics (SIGGRAPH '93 Proceedings)* (J. T. Kajiya, ed.), vol. 27, pp. 279–288, (Aug. 1993).
17. S. E. Chen, "Quicktime VR - an image-based approach to virtual environment navigation", in *SIGGRAPH 95 Conference Proceedings* (R. Cook, ed.), Annual Conference Series, pp. 29–38, ACM SIGGRAPH, Addison Wesley, (Aug. 1995). held in Los Angeles, California, 06–11 August 1995.
18. L. McMillan and G. Bishop, "Plenoptic modeling: An image-based rendering system", in *SIGGRAPH 95 Conference Proceedings* (R. Cook, ed.), Annual Conference Series, pp. 39–46, ACM SIGGRAPH, Addison Wesley, (Aug. 1995). held in Los Angeles, California, 06–11 August 1995.
19. M. Regan and R. Pose, "Priority rendering with a virtual reality address recalculation pipeline", in *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)* (A. Glassner, ed.), Computer Graphics Proceedings, Annual Conference Series, pp. 155–162, ACM SIGGRAPH, ACM Press, (July 1994).
20. J. Torborg and J. Kajiya, "Talisman: Commodity Real-time 3D graphics for the PC", in *SIGGRAPH 96 Conference Proceedings* (H. Rushmeier, ed.), Annual Conference Series, pp. 353–364, ACM SIGGRAPH, Addison Wesley, (Aug. 1996). held in New Orleans, Louisiana, 04–09 August 1996.
21. M. Levoy and P. Hanrahan, "Light field rendering", in *SIGGRAPH 96 Conference Proceedings* (H. Rushmeier, ed.), Annual Conference Series, pp. 31–42, ACM SIGGRAPH, Addison Wesley, (Aug. 1996).

22. S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, "The lumigraph", in *SIGGRAPH 96 Conference Proceedings* (H. Rushmeier, ed.), Annual Conference Series, pp. 43–54, ACM SIGGRAPH, Addison Wesley, (Aug. 1996).
23. A. Certain, J. Popović, T. DeRose, T. Duchamp, D. Salesin, and W. Stuetzle, "Interactive multiresolution surface viewing", in *SIGGRAPH 96 Conference Proceedings* (H. Rushmeier, ed.), Annual Conference Series, pp. 91–98, ACM SIGGRAPH, Addison Wesley, (Aug. 1996). held in New Orleans, Louisiana, 04-09 August 1996.
24. B. Jähne, *Digital Image Processing*. Springer Verlag, (1995). Third edition.
25. D. Lischinski, "Incremental delaunay triangulation", in *Graphics Gems IV* (P. S. Heckbert, ed.), pp. 47–59, San Diego, CA: Academic Press Professional, (1994).

4.5.2 Interactive Update of Global Illumination Using A Line Space Hierarchy (SIGGRAPH'97)

Auteurs : George Drettakis et François X. Sillion

Actes : Congrès SIGGRAPH '97

Date : août 1997

Interactive Update Of Global Illumination Using A Line-Space Hierarchy

George Drettakis and François X. Sillion

iMAGIS[†]- GRAVIR/IMAG - INRIA

Abstract

Interactively manipulating the geometry of complex, globally illuminated scenes has to date proven an elusive goal. Previous attempts have failed to provide interactive updates of global illumination and have not been able to offer well-adapted algorithms controlling the frame rate. The need for such interactive updates of global illumination is becoming increasingly important as the field of application of radiosity algorithms widens. To address this need, we present a novel algorithm which provides interactive update rates of global illumination for complex scenes with moving objects. In the context of clustering for hierarchical radiosity, we introduce the idea of an implicit *line-space hierarchy*. This hierarchy is realized by augmenting the links between hierarchical elements (clusters or surfaces) with *shafts*, representing the set of lines passing through the two linked elements. We show how line-space traversal allows rapid identification of modified links, and simultaneous cleanup of subdivision no longer required after a geometry move. The traversal of line-space also limits the amount of work required to update and solve the new hierarchical system after a move, by identifying the modified paths in the scene hierarchy. The implementation of our new algorithm allows interactive updates of illumination after object motion for scenes containing several thousand polygons, including global illumination effects. Finally, the line-space hierarchy traversal provides a natural control mechanism allowing the regulation of the tradeoff between image quality and frame rate.

Keywords: Global illumination, Dynamic environments, Hierarchical radiosity, Form-factors, Interactivity, Frame-rate control.

1 Introduction

The use of realistic global illumination is becoming more and more widespread. As a consequence, users demand more flexibility, and better interaction with lighting systems. Ideally, a user would like to be able to interact with a scene and interactively perceive at least some degree of global illumination effects. A major limitation of current systems is the inability to move or change geometry in a scene, with simultaneous update of global illumination effects. Applications such as virtual studios, tele-conferencing in virtual environments, driving simulators etc., all require interactive manipulation

of the scene geometry, without loss of important illumination information.

Although significant advances have been made towards accelerating radiosity calculations for changing geometry [7, 4, 11], all previous approaches fail to provide interactive global illumination updates even for moderately complex scenes, and do not provide a way to control the quality/speed tradeoff for interactive display. The new solution we present uses the subdivision of line-space implied by the link structure of hierarchical radiosity to achieve efficient interactive radiosity updates for scenes of moderate complexity.

The goal of our approach is to provide a unified framework which will allow a user of a hierarchical radiosity system to move objects in a lighting simulation, and interactively perceive global illumination changes. We also provide a mechanism permitting the user to sacrifice quality for speed, but still maintain at least some of the important visual cues due to global illumination.

In any hierarchical radiosity system, and in particular a clustering-based approach, the elements of the scenes are linked together following the potential interactions of light between such pairs. These links induce a subdivision of the line-space of the scene, or more accurately the space of line-segments, following the flow of light between scene elements. We augment links with an explicit representation of all lines passing between two elements via a *shaft* [9]. In particular, when an object moves, we can efficiently identify the parts of the system which are modified, by hierarchically descending in this line-space. This traversal permits efficient cleanup of the mesh where subdivision is no longer needed because of geometry changes, and allows us to mark the paths in the hierarchy which are modified. As a consequence, fast resolution of the modified part of the system of equations is achieved. Finally, the line-space hierarchy provides a natural way to control the expense incurred at every frame. This is achieved by limiting the descent into line-space by the time available at each frame.

We present an implementation of these ideas which shows that, using the line-space hierarchy we can achieve interactive updates of illumination (2-3 frames per second) for scenes of moderate complexity (up to about 14,500 input polygons). This includes the treatment of scenes almost exclusively lit by secondary illumination.

2 Context and Previous Work

The fact that the movement of an object often causes limited changes to a global illumination solution became evident early on in graphics research, in particular for “radiosity” algorithms. Several algorithms have been proposed which deal specifically with changes to geometry, and their evolution follows closely the progress of radiosity solutions. In what follows we present a brief overview of these methods.

2.1 Progressive radiosity solutions

The initial “full-matrix” radiosity solution [8], led to the development of an algorithm which took advantage of coherence properties of the hemi-cube, used to calculate form-factors [3]. This solution suffered from all the limitations of full-matrix radiosity (including quadratic storage and lack of adaptive subdivision), and most notably was limited to predetermined trajectories. Despite these draw-

[†]iMAGIS is a joint research project of CNRS/INRIA/UJF/INPG. Address: iMAGIS/GRAVIR, BP 53, F-38041 Grenoble Cedex 09 France. Email: {George.Drettakis|Francois.Sillion}@imag.fr

backs, this algorithm is noteworthy in the insight that form-factor changes can be limited spatially, by means of a swept volume restricting the part of space affected by the move.

The advent of progressive refinement solutions led to the development of two similar approaches which took advantage of the “shooting” process of radiosity propagation [4, 7]. By treating all object movements as deletions and re-insertions in the scene, shadows were removed by re-shooting energy, and new shadows were correctly inserted where appropriate by shooting “negative energy”. This approach achieved impressive update times for direct illumination, even though global updates remained very expensive. An improvement to these methods was developed by Müller et al. [11], who added an intelligent data structure maintaining shadow-lists accelerating potentially modified interactions between surfaces. The main drawback of all these approaches is due to the fact that they are based on progressive refinement, for which the global energy balance is hard to control. In addition these methods cannot achieve interactive update rates (several frames per second) for scenes of moderate size or larger.

2.2 Hierarchical radiosity solutions

Some of the limitations of progressive refinement solutions can be addressed in the context of hierarchical radiosity. An overall “snapshot” of the global energy balance is maintained in the link structure, and the corresponding hierarchy. When an object moves, a limited number of links, and thus a limited part of the hierarchy, are modified. This can be seen by examining the block form-factor matrices resulting from hierarchical subdivision. In Fig. 1(a) the chair has just moved to the right. The block form-factor matrix (collapsed up to a certain hierarchical level for display) is shown in Fig. 1(b), “warmer” colors representing larger values of form-factors. The matrix in Fig. 1(c) represents the difference in the matrices produced by the move. As we can see, few regions of the matrix change.

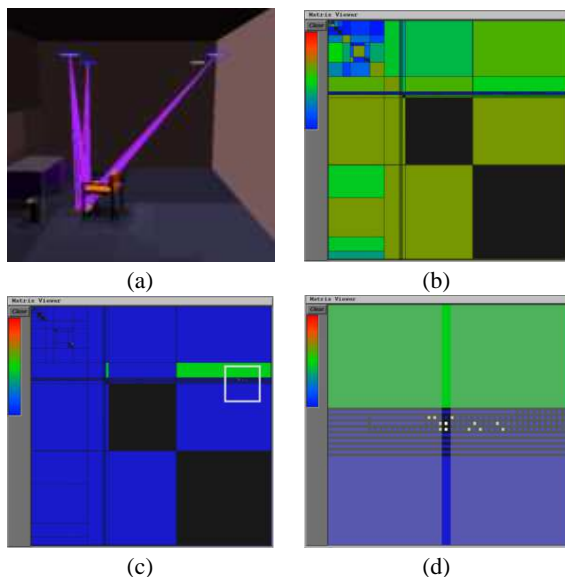


Figure 1: The chair in (a) has moved to the right. The block form-factor matrix is shown in (b), color coded (see text), before motion. In (c) we show the difference (dark blue = no change); (d) zoom of the modified matrix area, yellow blocks are the links shown in (a).

By zooming into a modified region (Fig. 1(d)), we realize that the changes are due to the moving shadow of the chair. In particular the region of the matrix selected in Fig. 1(d) corresponds to the element

shown in red in Fig. 1(a), and the yellow blocks are a collapsed representation of the links arriving at this element.

Two first solutions have been developed exploiting hierarchical radiosity for dynamic environments. Forsyth et al. [6] present the idea of “promoting” and “demoting” links based on a refinement criterion which moves links up and down in the hierarchy, depending on the position of the objects at each frame. This is similar to the idea of “ghost” links presented by Shaw [12]. A first approach to the problems incurred by changes in visibility was presented in Shaw’s work, by introducing special “shadow” links connected to the source and containing blocker information. A final optimisation was presented by which a “motion volume” is built with the bounding planes of the two positions of the dynamic object; All links are then tested against this volume to determine if they may have changed or not.

2.3 Shortcomings of previous methods

The algorithms for hierarchical radiosity in dynamic environments described above achieve significant improvement over the progressive refinement approaches developed earlier. Nonetheless, interactive updates rates are not achievable using these methods, especially for complex scenes, and since they were developed using traditional hierarchical radiosity, the quadratic cost of initial linking presents an important obstacle to their practical use.

More importantly, all previous approaches lack a unified mechanism which can rapidly identify the part of the system modified and at the same time control the simulation quality/time tradeoff in a coherent manner. In what follows we show that the *line-space hierarchy*, exploited by accessing the links attached to the scene hierarchy, provides this functionality. We will show how the hierarchical traversal of line-space allows rapid identification of the parts of the system which change, provides an efficient mechanism to update the hierarchy, and finally allows fine control of the quality/speed tradeoff for dynamic environments.

2.4 Objectives: user interaction with the scene

We assume that any object in the scene can be chosen to be dynamic. The only restriction is that each potential dynamic object must be included in a cluster of its own. When the user selects a dynamic object, the corresponding cluster is attached to the root of the hierarchy.

Changing the dynamic object during the simulation is not too complicated, since we can identify the links which were affected by the previous and the new dynamic object. This can be performed efficiently by traversing line space as described later, updating the links and the corresponding visibility information. Since the dynamic object clusters are attached to the root, a change in the object chosen can be accompanied by a re-insertion into the cluster hierarchy.

Once the object is chosen, the user can freely interact with the object to change its position (for example by translation or rotation). At each frame, all that is required is the previous and current position of the dynamic object bounding box. What is required next is to identify the links of the system whose shaft cuts either of these bounding boxes.

3 Hierarchical Line-Space Traversal

To rapidly identify the necessary modifications of the global illumination system of equations, we need a data structure which will isolate the parts of space which are affected by the motion of an object. The *line-space hierarchy* we introduce is such a representation encoded using the traditional link structure of hierarchical radiosity.

3.1 Introduction to the Line-Space Hierarchy

In the original hierarchical radiosity algorithm [10], the scene, as well as subsequent subdivision, is represented as a hierarchy. In addition, when the clustering algorithm is used, the polygons of the scene are grouped together to form a complete hierarchical representation of the environment. We will refer to such an element as an H-element h_e (Hierarchical element [13]), be it a cluster or a surface element. This scene hierarchy is augmented by *link* information, which is used to represent radiant exchanges between two H-elements of this hierarchy. A typical hierarchical solution process proceeds by *refining* the links: when the link is considered to insufficiently or incorrectly represent the light transfer, the element is subdivided and sub-links are created [10, 15]. The decision to subdivide is based on a “subdivision criterion” which may be based on error-estimation or magnitude of energy transfer across a link. We will refer to a “refiner”, which is the module responsible for the refinement operation.

We represent the set of light paths in a hierarchical manner by augmenting the link structure. The shaft shown for example in Fig. 2(a) represents the entire set of lines which pass between the two elements. More precisely, this is the set of *line segments* rather than infinite lines. For simplicity however, we will use the term *line-space hierarchy* throughout this paper. In this respect, we build a coarse approximation to structures which encode visibility information of maximal free segments such as the visibility complex [5].

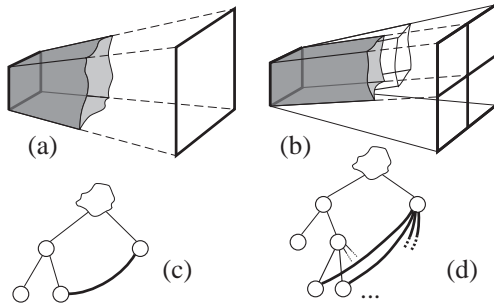


Figure 2: We show here a line space hierarchy: (a) the original link l^p (shown as a shaft) is subdivided, resulting in four sub-links (b), which are the line-space children of l^p . P-links (c) and the subsequent children links (d) are embedded in the H-element hierarchy.

When a link is subdivided, the four resulting links can be considered as children of the original link, Fig. 2(b). To store this hierarchy, subdivided links are not discarded, and are stored as *passive links* or *p-links* with the H-elements. Thus our new link hierarchy is actually *embedded* in the H-element hierarchy itself. For example, the thick black line in Fig. 2(c) corresponds to a (subdivided) p-link, and the four thick lines in Fig. 2(d) to the four resulting links. These ideas are related to the approach developed by Teller and Hanrahan [17], where a similar link hierarchy was used to incrementally maintain blocker lists, as well as the “ghost-link” idea [12] or link “demotion/promotion” approach [6].

3.2 Data structures

To explicitly work in line-space, we need a representation of the set of lines between surfaces. One possibility would be the approach using Plücker coordinates as was presented by Teller [16], or the intercepts of lines on two parallel planes [1]. Another approach would be that of ray-classification [2].

We have preferred to use the *shaft* structure [9] for its simplicity, and because it is well defined and easy to manipulate for the case when one endpoint of a link is a bounding box. In addition

the shaft structure permits efficient intersection tests with bounding boxes, which is an operation central to the algorithms presented below. Since our algorithm operates in the context of clustering radiosity, this allows the use of the same structure for links between any combination of cluster and surface element. Another interesting aspect of the shaft representation is that shafts are truncated, and thus operate on *line segments*, as opposed to infinite lines.

The line-space hierarchy is built during the traditional refinement process of clustering hierarchical radiosity. The list of links arriving at a H-element is stored on the element itself, as well as all the p-links (see Figs. 2 and 3). Both active links and p-links are augmented with the shaft corresponding to the part of line-space they respectively cover.

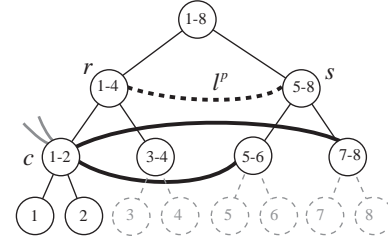


Figure 3: Numbering of the H-element hierarchy into index ranges.

One final component is required to allow efficient traversal the line-space hierarchy. Consider a given passive link $l^p_{s \rightarrow r}$, stored at H-element r and emanating from H-element s . Since this p-link has been subdivided, it is possible that the element s has been subdivided also. In addition, what originally was the p-link $s \rightarrow r$ is now a set of links at possibly different levels of the hierarchy, emanating from children of s , also at different levels. Links originating from other sources may also co-exist on the element r and its children. To be able to quickly identify the links which are children of $l^p_{s \rightarrow r}$, we need to be able to tell if a given link originates from an H-element which is a child of s . To avoid repeated traversals of the hierarchy which become overwhelmingly expensive, we apply a numbering scheme of all hierarchical elements. In particular, we traverse the entire hierarchy once at the beginning of the lighting simulation process, and assign an interval to each node, corresponding to the largest possible number of children which can be created. This number can be calculated based on the value of the *area threshold* A_e [10], which limits the size of the smallest surface. To see this, consider the partially subdivided hierarchy shown in Fig. 3, where subdivided nodes are represented with solid lines. The index ranges assigned to each node correspond to the maximum subdivision which can possibly be incurred, shown as the complete tree, where currently unsubdivided nodes are shown with dashed lines.

3.3 Traversing Line-Space

Consider that we are situated on a given H-element h_e of the hierarchy and we wish to traverse the part of line space related to a p-link arriving at h_e . To effect the line-space descent, we need to visit exclusively the children links of the passive link considered. Each passive link l^p is connected to a source node s , which has a corresponding index range following the numbering scheme elaborated above. To visit the line-space children of l^p , we descend to each H-element child c of the current node h_e , and examine only those passive links originating from s or s 's children, arriving at c . These are the links attached to a node with an index within the interval of the index of s .

For example, in Fig. 3, we are at receiver node r . P-link l^p is the thick dashed line. When descending to its line-space children, we visit for example H-element c . The only links we consider however

are those emanating from H-element in the range $[5 - 8]$, i.e., children of the original H-element s , linked by l^p . This procedure is summarized in Fig. 4.

```

traverseLineSpace(H-element  $h_e$ , IndexRange  $ind$ )
{
  for each p-link  $l^p$  of  $h_e$ 
    // find the "source" node  $q$ 
    H-element  $q = l^p \rightarrow \text{LinkedNode}()$ 
    if(  $ind$  contains  $q \rightarrow \text{IndexRange}()$  )
      for each child  $c$  of  $h_e$ 
        // limit traversal to the index of  $q$ 
        traverseLineSpace(  $c, q \rightarrow \text{Index}()$  )
}

```

Figure 4: Hierarchical Line Space Traversal

3.4 Efficient illumination update

To perform an update we first need to find the links affected by object motion. We consider as potentially changed the links whose *shafts* are touched by the bounding box of the *dynamic object* (i.e., the object which moves) before or after the move. After motion, the subdivision of parts of the hierarchy (for example due to shadow motion) may no longer be needed. We need to identify these parts of the hierarchy and perform the cleanup.

For the links which have been thus identified, new form-factor values need to be computed, since visibility may have changed with respect to the dynamic object. We call this operation *link update*.

To perform form-factor update efficiently, we maintain two-part occlusion information with each link: occlusion with respect to the dynamic object, and occlusion with respect to the rest of the scene. Thus at each frame, visibility is checked only with respect to the dynamic object. This information must of course be updated when we change dynamic objects; a line traversal to find the sub-spaces affected by the old and new object suffices to achieve this. Also, when a new link is created, its visibility with the rest of the scene is computed (but not for reinstated p-links).

After link update, it may be the case that certain links have to be refined, in particular when a visibility change occurs. The new values of the radiosity need to be computed and gathered on the links. The hierarchical system must then be updated to reflect the new position of the moving object. Finally the new system must be solved. The line-space hierarchy provides the necessary mechanism to efficiently perform all of the above steps. We thus summarize our new approach as follows:

1. Find the links potentially changed by traversing line-space and remove subdivision unnecessary due to geometry change;
2. Update the modified links and refine where necessary;
3. Solve the hierarchical system efficiently.

Step (1) occurs during the line-space traversal, resulting in the information necessary to perform steps (2) and (3).

4 Rapid Identification of Modified Links and Subdivision Cleanup

We next show how line-space traversal can be used to rapidly identify the links which have potentially changed. This traversal allows simultaneous removal of subdivision which is no longer required due to the new position of the moving object.

4.1 Finding the modified links efficiently

To find the modified links, the line-space traversal can be thought of as “zooming-in” to the region of space which has changed. The traversal algorithm starts at the root of the hierarchical elements (a cluster whose extent is the bounding volume of the scene), and descends recursively. At a given hierarchical node h_e , we visit all its passive links. For each such p-link l^p , we determine whether its corresponding line-space (represented by the shaft), was affected by the object motion. This is performed by testing the link shaft against the previous and current positions of the dynamic object. If the link was affected, we will potentially descend into its corresponding line-space. Given the new position of the dynamic object, we first test whether passive link l^p would no longer satisfy the subdivision criteria; this could occur for example if the p-link was partially occluded by the previous position of the dynamic object, but is completely unoccluded in the new position. If this is the case, the link can be reinstated as active, and the descent into the line-space hierarchy is stopped. If, on the other hand, the passive link l^p is maintained, the line-space children of l^p are visited, and the same process is applied recursively.

Finally, when the traversal of the passive links of h_e is complete, we examine all of the active links corresponding to the current part of line space (links originating from s or s ’s children) to determine if they are affected by the object motion. If this is the case, i.e., the active link shaft cuts the dynamic object before or after the move, it is added to a list of candidates for update and potential refinement. An example of a candidate list is shown in Fig. 5(b); notice that few of the numerous links describing energy exchanges in the left room or with the left wall of the right-hand office appear in this list.

4.2 Cleanup of unnecessary subdivision

The traversal described above also provides the benefit of cleaning up unneeded refinement in the same step as the determination of the links which need to be updated. In particular when we have decided that a p-link l^p (linked to H-element s) on H-element h_e does not merit refinement due to the new position of the dynamic object, we can remove the entire set of links and p-links which are (line-space) children of l^p . This is performed by descending to the children of H-element h_e and removing all passive and active links linked to s and its children. After this removal, if there are no links remaining on any of the children of h_e (or its intermediate nodes), the subdivision is cleaned up. A similar approach to subdivision cleanup presented in [12] required a special pass, and multiple hierarchy passes to mark attached sources and their children.

The line space traversal is summarized in Fig. 6. In this algorithm, notice that when a potential change is found (either for a p-link re-installation or the required update of an link), the corresponding H-element is marked changed. For example, the H-elements marked “changed” are shown in red in Fig. 5(c). This will be used in what follows to limit the system solution at each frame. In addition, this approach is particularly efficient in the context of a clustering algorithm, since the number of links is limited.

5 Fast System Solution By Hierarchy Pruning

The line-space traversal algorithm described above results in fast identification of the links which need to be refined and at the same time allows the un-refinement of unnecessary subdivision. Once this step has been performed, we need to update the links which are changed, and potentially refine certain links. This additional refinement will be required for example around new shadow boundaries. Once these steps are complete, we have a complete hierarchical system which is ready to be resolved. In hierarchical radiosity [10, 15], system resolution is performed by performing complete sweeps of

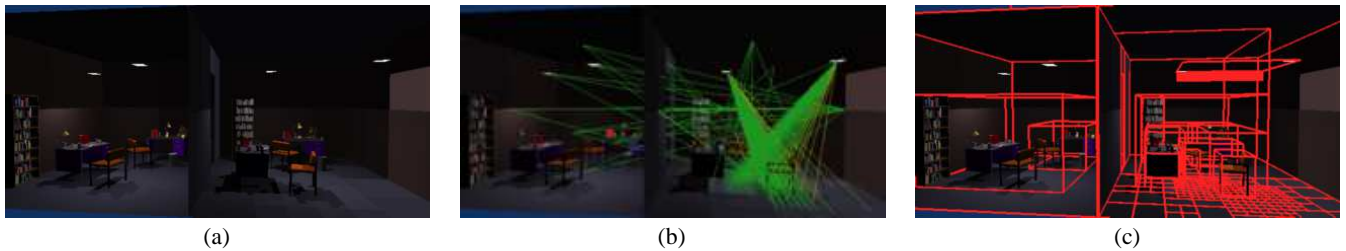


Figure 5: This example contains 14,572 input polygons. The chair in the right-hand room moves to the right (a) - (b). The average update time is approximately .3 s. In (b) we show the links modified and in (c) the parts of the hierarchy affected by the move and marked “changed” by the line-space traversal. Note how the several thousand surfaces of the left-hand room are unaffected.

```

findModifiedLinks(H-element he, IndexRange ind)
{
  for each p-link pl of he {
    H-element q = pl→LinkedNode() // “source” node
    if( ind contains q→IndexRange()
       and affectedByMotion(pl→Shaft()) {
      if( wouldBeRefined( pl ) ) {
        for each child c of he
          // limit traversal to the index of q
          findModifiedLinks( c, q→IndexRange() )
          if a child changed
            he→setChanged()
      }
    }
    else
      reInstateLink( pl )
      he→setChanged()
  }
  // if any links of he are modified they are added
  // to the list of links to refine, and he is marked changed
  checkAddLinks( he )
}

```

Figure 6: Efficient Modified Link Identification

the hierarchy: *Gather* and *Push-Pull*. In the context of clustering [13], irradiance is gathered through the links onto the H-elements; this irradiance is subsequently pushed down the hierarchy. Finally, the radiosity values are computed at the leaves, and pulled up the hierarchy to provide the new solution.

Such an approach was used for a dynamic solution in [12] for example. For large scenes containing thousands of input polygons, these multiple sweeps of the hierarchy result in an unacceptable overhead in an interactive context. Furthermore, such complete hierarchy traversal is unnecessary: only a small number of elements has actually been changed.

In what follows, we show that we can perform a *Gather* exclusively for links which have been affected by object motion, during link update. The line-traversal algorithm returns a list of links potentially changed. The form-factors for these links are recomputed to reflect the new position of the dynamic object. This update operation may result in the need for certain links to be refined.

Once the refinement is complete, the global solution can be efficiently performed. In particular the marking of the changed paths performed during line-space traversal allows us to limit the *Push-Pull* to the subsections of the hierarchy which have been modified, for one bounce of illumination. Subsequent bounces may potentially be required but can also be treated efficiently.

There is one essential assumption for the following discussion: we assume that before any motion is performed, the system has run to convergence. We define a solution as converged when no more links can be refined for the given error threshold, and the energy balance has been computed using all links.

5.1 Link update and in-place *Gather*

Once a link has been identified by the line-space traversal algorithm as potentially changed, we need to update its form-factor value. There are two possibilities: either the link has changed little and as a consequence it will not be refined or the link has changed in a way which requires further subdivision.

In the first case we need to modify the irradiance of the receiving patch by the difference of the previous irradiance and the current irradiance. For source s and receiver r , B_s the radiosity of H-element s , F^k the form-factor before the move and F^{k+1} the form-factor after the move, this difference is simply:

$$I_{diff} = B_s (F^{k+1} - F^k) \quad (1)$$

For clusters, we also need to distribute the irradiance down to the cluster contents.

When refinement occurs, we need to remove the irradiance which arrived at the receiving node from the previous transfer on the link. After subdivision, the new irradiance will be transferred when the sub-links are established. For element r , the irradiance I_r becomes:

$$I_r^{k+1} = I_r^k - F^k B_s \quad (2)$$

After subdivision, the new links will be established at a different level in the hierarchy, and we then simply add in the new irradiance. We thus avoid the *Gather* sweep of the hierarchy.

5.2 Non-recursive refinement for modified links

At each frame, certain links will be identified as requiring refinement. This typically occurs when the dynamic object touches a link shaft for the first time. Previous hierarchical radiosity (e.g., [10]) performed recursive refinement of links. In such an approach, when a link is refined, we immediately attempt to refine its children and so on recursively until subdivision is no longer possible given the current subdivision criteria.

For the requirements of controlling the solution, we need to achieve two goals: (a) refine the most important links first and (b) potentially truncate the refinement process if we wish to limit the amount of time spent for a frame. The latter point becomes essential for the time/quality control algorithm presented in Section 6.1.

To achieve this, when refining a link, the resulting refined sub-links are added to a heap. The refiner then extracts the link with the highest potential power transfer for refinement, thus achieving goal (a). The refiner keeps track of the number of links it still needs to refine n_l , and the number n_r of links already refined. When the sum of n_l and n_r exceed the limit fixed by the subdivision process, the refinement is terminated. The n_l form-factors of the remaining links are then updated to correspond to the new position of the dynamic object, and established at the given level without being refined.

5.3 Global solution

Once the line-space traversal and the refinement are complete, we have a new hierarchy for which the irradiance at each node corresponds to the new position of the object. In addition, we have marked as “changed” all the elements in the hierarchy which have been modified, as well as all the paths in the hierarchy leading to these elements. The only remaining step is the *Push-Pull* process of the hierarchical solution which will result in the correct hierarchical representation of radiosity at every level of the hierarchy.

As mentioned earlier, we exploit the information of the paths and elements marked “changed”, to accelerate the *Push-Pull* since it will only be performed on a small part of the hierarchy.

5.3.1 Single Bounce

We modify the *Push-Pull* procedure to visit only the parts of the hierarchy which are modified. At a given node, we check if it is marked “changed”: if it is we proceed as normal, descending to the children, and if not we use the radiosity already calculated at this level instead of continuing the hierarchy descent. This procedure is summarized in the pseudo-code of Fig. 7.

```

Spectrum partialPushPull(Helement* he, Spectrum IrradDown)
{
    Spectrum RadUp
    if he→changed() { // normal PushPull
        if ( he→isLeaf() )
            RadUp = ComputeLeafRad
        else foreach child c of he
            RadUp += partialPushPull(c, IrradDown + he→Irrad())
    }
    else // use previous values
        RadUp = he→Radiosity()
    he→Radiosity = RadUp
    return RadUp
}

```

Figure 7: Partial Push-Pull

This update algorithm will result in a system which is updated to reflect the new position of the dynamic object. This new state includes changes to all links, for direct *and* indirect illumination. In most cases, this is largely sufficient, since it represents a system which in many cases has converged. This solution always provides updated shadows due to primary illumination, as well as for some shadows due to indirect illumination.

5.3.2 Subsequent bounces of illumination

It may be the case however that the system has changed sufficiently so that subsequent iterations are needed to achieve convergence. Consider the example of a dark room with a door (initially closed) opening to a bright corridor (see Fig. 10). When opening the door, some direct light will flow into the room due to links which will be refined. If the previous algorithm were to be used as described above, certain light transfers would not occur (e.g., from the floor to the ceiling), since the radiosity values in the hierarchy would not be modified until after the call to *partialPushPull*.

To determine the modified links, we perform a partial traversal of the hierarchy, visiting only the elements marked “changed” by the line-space traversal. At each H-element h_e , the H-elements it is linked to are stored, that is the H-elements for which h_e acts as a “source.” In addition, we store the old value of radiosity, corresponding to the previous dynamic object position. Thus at every H-element changed by the first bounce, we can calculate the difference in irradiance, and perform an in-place gather to the receiving node as described above. Some links may need to be refined, and as such are inserted into the heap. Subsequent refinement is then

performed, followed by a new partial push pull. Care must be taken to re-initialize the “changed” markings of every node, as well as the old value of radiosity after each iteration. In particular, H-elements updated by the difference in irradiance are marked as “changed” as well as their parents and children. The difference in global effects is generally limited, and thus the extents of these updates is not very large, even for cases where global illumination effects are very important (e.g., Fig. 10).

6 Controlling Simulation Time and Rendering Quality

The solution presented above allows rapid updates for scene of moderate complexity. Nonetheless, the update rate can be too slow for certain applications where 1 or 2 frames per second is simply not acceptable. In addition, for very complex scenes, we need to be able to provide the user with the choice of “give me N frames per second, with the best possible quality.”

The limitation of the algorithm presented above is that there is no control on the number of links that need to be updated. Thus, if the dynamic object moves into a part of space which contains a large number of links, the line-space traversal will create a large candidate list. A large amount of time will thus be required to update the form-factors of these links, and to potentially refine some of them.

To avoid this problem we present an algorithm that updates as much of the modified link hierarchy as possible, in the sense of a user-defined time limit, and unrefines the rest. This can be performed naturally with the aid of the line-space hierarchy, and is very much in the spirit of the original hierarchical radiosity approach.

6.1 Controlled update algorithm

To achieve a controlled update time the user first selects a target frame rate. The system then calculates the number of link updates that it can perform in a given frame. Thus we consider as our “time” or cost unit, the time required for a link update.

To achieve the control of the number of link updates, and adapt the hierarchy to the desired frame rate, we first calculate the number of child links and p-links for each p-link in the hierarchy. This is performed while traversing line-space to find modified links, as presented above in Fig. 4, and “pulling” the number of links resulting from the subdivision of each p-link.

To perform the link update, we traverse line-space for a second time. During this sweep, we only traverse the parts of the hierarchy actually modified.

The update control algorithm is very similar to the line space traversal. The important difference is that we always update the links of the current node *first*. We are thus assured that the state of the hierarchy above the current node h_e is correctly up to date. This is an essential requirement, since we can not otherwise truncate the update without incurring inconsistencies. The time spent updating links of this node is subtracted from the remaining time.

We then compute the minimum amount of work required if we are to descend in line-space. This is equal to the total of all line-space children active and passive links arriving at the children of h_e . If this number is larger than the remaining “time”, we cannot guarantee that we will be able to update the remaining sub-links in the allotted time. In this case, we reinstate the appropriate p-links of h_e and cleanup underlying links (and remove subdivision if necessary). We then stop the descent.

If we can still descend, we continue the traversal of line-space. We assign to each child the fraction of time calculated as follows. We sum the total number n_{cl} of children links stored with each p-link of h_e , in the current index range. The fraction assigned to each child c of h_e is the number of active sub links of p_l arriving at c plus the number of active sub links below c , also stored during the

push-pull of line-space. This can be seen as a local and adaptive form of progressive multi-gridding [14].

```

controlledUpdate(Helement  $he$ , IndexRange  $ind$ , int  $remainingCost$ )
{
    updateLinks( $h_e$ )
     $remainingCost = remainingCost - \text{links updated}$ 
     $pLinkCost = \text{number of children p-links and links in } ind \text{ range}$ 
    if  $pLinkCost \leq remainingCost$  {
        for each p-link  $l^p$  of  $he$ 
            H-element  $q = l^p \rightarrow \text{LinkedNode}()$  // "source" node
            if  $ind$  contains  $q \rightarrow \text{IndexRange}()$ 
                for each child  $c$  of  $he$  with  $c \rightarrow \text{changed}()$ 
                     $cost = \text{fraction of child links of } l^p \text{ in } c$ 
                    controlledUpdate( $c, q \rightarrow \text{IndexRange}(), cost$ )
            }
        else
            update and reinstate all p-links of  $h_e$  in  $ind$  range
    }
}

```

Figure 8: Controlled Update Algorithm

A very important feature of this controlled update strategy is that no matter how much time is allocated for the update, a consistent solution is computed. This is due to the fact that the set of links always completely covers the entire line space. The availability of a *complete* hierarchy is a benefit of using hierarchical clusters. In other words, the algorithm ensures that all possible energy transfers are accounted for, although they may be included in links very high up the hierarchy (probably as a significant approximation, inducing some error). This consistency comes in contrast to all previous approaches to incremental updates of radiosity solutions, and is an important asset for many practical applications.

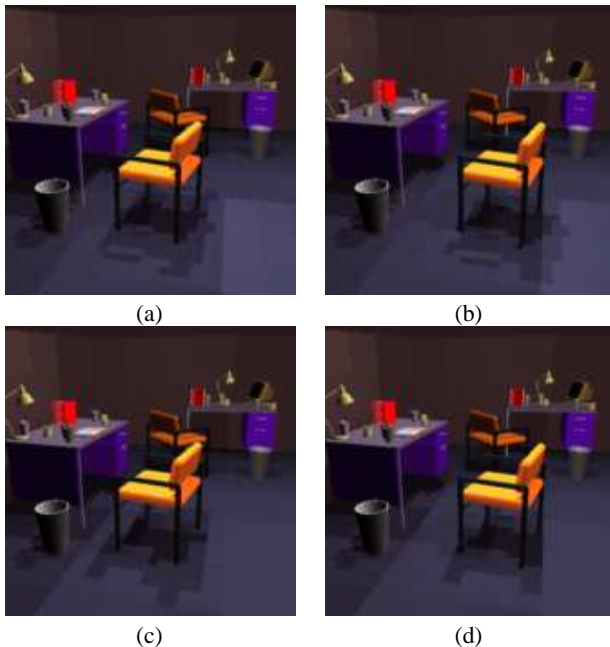


Figure 9: The chair is the dynamic object, moving to the right, with no additional bounces. Update rates (a)-(b) average .5s. In (c) and (d) the same motion is shown with time/quality tradeoff, limited to 330 link updates per frame. The average update takes .3s.

6.2 Maintaining consistent quality

The controlled line-space descent presented above performs well for many configurations. Nonetheless, due to the quadtree nature

of the subdivision on surfaces, it is often the case that new refinement does not occur as desired. In particular we may notice that subdivision is not propagated across quadtree edges.

To counter this problem, we influence link refinement. Consider a p-link inserted into the heap for refinement because the dynamic object cuts its shaft, and that there was no interaction in the position at the previous frame. In this case, we increase the key used to sort elements in the heap, making refinement of this link more probable.

7 Implementation and Results

We have implemented the new algorithms on a clustering hierarchical radiosity algorithm following [13]. The modules for line-space traversal and all modified refinement routines required around 5000 additional lines of C++. All test results are reported on an Indigo 2 Impact, with an R4400 processor at 200MHz.

7.1 Basic algorithm

The first example is a single scene containing 5,405 polygons shown in Fig. 9, with four light sources. For this scene, we show one chair moving to the right, which is an object containing 870 polygons. The update rates are close to 0.5 seconds per frame, which can be satisfactory in many cases. The breakdown of the computation time for the first move for example is: 0.08 s. for line-space traversal, 0.40 s. for link update/refinement and 0.05 s. for partial push-pull.

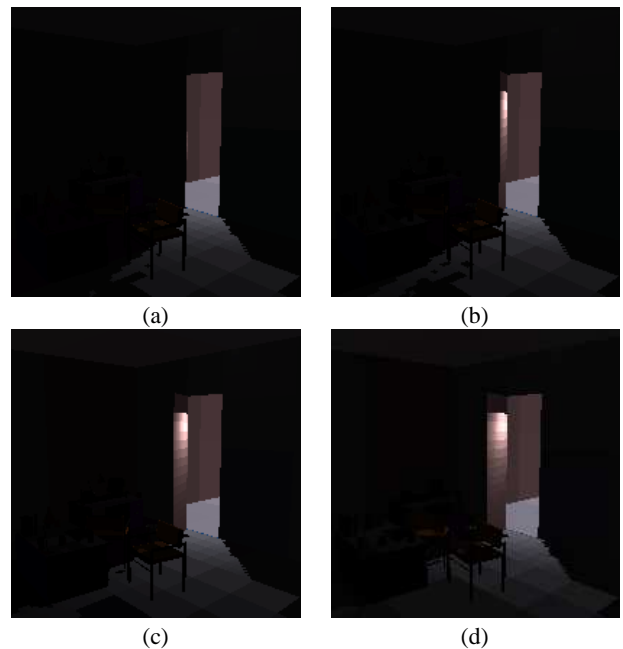


Figure 10: In this example, a door opens onto a dark room (5,295 polygons). Additional bounces are performed, and global illumination is updated. Updates takes (a)-(b): 3.20 s.; (b)-(c): 3.47 s.; (c)-(d): 2.52 s.

The second example is shown in Fig. 10, where the additional iterations are required. Initially the room is completely black. While the door is gradually opened, global illumination is rapidly updated. We see that a reasonable representation of global illumination is provided with update rates of around 3 seconds per frame. For the second update for example, the single bounce took a total of 2.15 seconds and the subsequent two bounces 0.55 and 0.5 seconds each.

The third example shows that we can handle complex geometry without significant degradation in speed. In Fig. 5, we have a scene with 14,572 input polygons, with the chair in the right-hand

room moving to the right. The update takes around 0.6s. per frame, but sometimes improves to 0.2s per frame (5 fps) when the chair traverses “sparse” regions of line-space (i.e. with few links).

7.2 Time/quality tradeoff

The controlled update algorithm was tested on the scene used previously to demonstrate the basic approach. In Fig. 9(c) and (d) we present the output of the algorithm for a selected value of 330 link updates per frame, which results in an average update rate of approximately .3 seconds/frame. The shadows are of lower quality compared to the image shown in Fig. 9, but are still definitely usable.

7.3 Higher quality movement

Our approach is also well adapted if the user requires higher quality, and is prepared to wait longer (several seconds). An example is shown in Fig. 11, where an update takes about 4 s., but we see that the quality of shadows is improved compared to Fig. 9. To achieve this, we simply decrease the ϵ threshold for BF refinement [10].

8 Conclusions and Future Work

We have presented a new framework for the efficient calculation of incremental changes of global illumination using hierarchical radiosity. This set of algorithms allows interactive updates of the lighting solutions in scenes with moving geometry. The heart of our approach is the introduction of a *line-space* hierarchy associated with the links between scene elements. This hierarchy is distinct from, but related to, the hierarchy of surfaces and clusters in the scene, and can be traversed efficiently using the implicit correspondence between the two hierarchies. In terms of data structures, the main addition with respect to hierarchical radiosity is the introduction of shaft structures representing a portion of line segment space associated to each link.

The line-space traversal algorithm is beneficial in many respects. First, it allows the fast identification of the links that should be modified in response to object motion. These links are collected to accelerate later processing. In addition, it permits simultaneous cleanup of object subdivision that has become unnecessary due to visibility changes induced by object motion. The line-space traversal marks the modified parts of the hierarchy, resulting in an accelerated solution of the modified system.



Figure 11: Higher-quality solution, 4 s. per frame on average.

The set of links marked during line-space traversal embodies all of the form factor changes in the scene, although at a hierarchical level which may not be sufficiently detailed for the desired accuracy. Hierarchical refinement of these links is carried out in a non-recursive manner, using a set of refinement criteria to resolve fine radiosity variations and shadow details. The marking mechanism results in a very fast solution of the updated system of equations.

Finally, the line-space hierarchy also provides a natural mechanism to control the time/quality tradeoff for incremental updates, by constantly monitoring the expected cost of refinement operations

and ensuring that the refinement budget is not exceeded. The nature of the refinement algorithm ensures that the solution is always consistent in that it takes into account all possible energy transfers in the scene, although possibly at high hierarchical levels.

Future work includes the design of improved quality control mechanisms. The very notion of solution quality is difficult to define. For instance, it is well accepted that the presence of shadows for moving objects is an important visual cue for understanding object location and motion, but these shadows need not necessarily be very precise. Recent work on accelerated approximate shadow calculations using the hierarchy of clusters and feature-based error metrics may be applicable to dynamic updates [14].

Finally, more involved visibility structures, which provide detailed line-space information, such as the Visibility Complex [5], will probably be very useful for dynamic updates of illumination.

Acknowledgements

Thanks to Frédo Durand, Mathieu Desbrun and Rachel Orti for crucial help.

References

- [1] N. Amenta. Finding a line transversal of axial objects in three dimensions. In *Proc. 3rd ACM-SIAM Sympos. Discrete Algorithms*, pages 66–71, 1992.
- [2] James Arvo and David Kirk. Fast ray tracing by classification. *Computer Graphics*, 21(4):55–64, July 1987. Proceedings SIGGRAPH '87 in Anaheim.
- [3] Daniel R. Baum, John R. Wallace, Michael F. Cohen, and Donald P. Greenberg. The back-buffer algorithm : an extension of the radiosity method to dynamic environments. *The Visual Computer*, 2:298–306, 1986.
- [4] Shenchang Eric Chen. Incremental radiosity: An extension of progressive radiosity to an interactive image synthesis system. *Computer Graphics*, 24(4):135–144, August 1990. Proceedings SIGGRAPH '90 in Dallas.
- [5] Frédo Durand, George Drettakis, and Claude Puech. The 3d visibility complex, a new approach to the problems of accurate visibility. In X. Pueyo and P. Schröder, editors, *Rendering Techniques '96*, pages 245–257. Springer Verlag, June 1996. Proc. 7th EG Workshop on Rendering in Porto.
- [6] David Forsyth, Chien Yang, and Kim Teo. Efficient radiosity in dynamic environments. In Sakas et al., editor, *Photorealistic Rendering Techniques*, Darmstadt, D, June 1994. Springer Verlag. Proc. 5th EG Workshop on Rendering.
- [7] David W. George, François Sillion, and Donald P. Greenberg. Radiosity redistribution for dynamic environments. *IEEE Computer Graphics and Applications*, 10(4), July 1990.
- [8] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Bat-tail. Modeling the interaction of light between diffuse surfaces. *Computer Graphics*, 18(3):213–222, July 1984. Proc. SIGGRAPH '84 in Minneapolis.
- [9] Eric A. Haines. Shaft culling for efficient ray-traced radiosity. In Brunet and Jansen, editors, *Photorealistic Rendering in Comp. Graphics*, pages 122–138. Springer Verlag, 1993. Proc. 2nd EG Workshop on Rendering (Barcelona, 1991).
- [10] Pat Hanrahan, David Saltzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. *Computer Graphics*, 25(4):197–206, August 1991. SIGGRAPH '91 Las Vegas.
- [11] Stefan Müller and Frank Schöffel. Fast radiosity repropagation for interactive virtual environments using a shadow-form-factor-list. In Sakas et al., editor, *Photorealistic Rendering Techniques*, Darmstadt, D, June 1994. Springer Verlag. Proc. 5th EG Workshop on Rendering.
- [12] Erin Shaw. Hierarchical radiosity for dynamic environments. Master's thesis, Cornell University, Ithaca, NY, August 1994.
- [13] François Sillion. A unified hierarchical algorithm for global illumination with scattering volumes and object clusters. *IEEE Trans. on Vis. and Comp. Graphics*, 1(3), September 1995.
- [14] François Sillion and George Drettakis. Feature-Based Control of Visibility Error: A Multiresolution Clustering Algorithm for Global Illumination. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 145–152. ACM SIGGRAPH, August 1995.
- [15] Brian E. Smits, James R. Arvo, and David H. Salesin. An importance-driven radiosity algorithm. *Computer Graphics*, 26(4):273–282, July 1992. Proc. SIGGRAPH '92 in Chicago.
- [16] Seth J. Teller. Computing the antipenumra of an area light source. *Computer Graphics*, 26(4):139–148, July 1992. Proc. SIGGRAPH '92 in Chicago.
- [17] Seth J. Teller and Patrick M. Hanrahan. Global visibility algorithms for illumination computations. In J. Kajiya, editor, *SIGGRAPH 93 Conf. Proc. (Anaheim)*, Annual Conf. Series, pages 239–246. ACM SIGGRAPH, August 1993.

4.5.3 Interactive Rendering using the Render Cache (EGRW'99)

Auteurs : Bruce Walter, George Drettakis et Steven Parker

Actes : 10th Eurographics Workshop on Rendering

Date : juin 1999

Interactive Rendering using the Render Cache

Bruce Walter[†], George Drettakis[†], Steven Parker[‡]

[†]iMAGIS¹-GRAVIR/IMAG-INRIA
B.P. 53, F-38041 Grenoble, Cedex 9, France
[‡]University of Utah

Abstract.

Interactive rendering requires rapid visual feedback. The *render cache* is a new method for achieving this when using high-quality pixel-oriented renderers such as ray tracing that are usually considered too slow for interactive use. The render cache provides visual feedback at a rate faster than the renderer can generate complete frames, at the cost of producing approximate images during camera and object motion. The method works both by caching previous results and reprojecting them to estimate the current image and by directing the renderer's sampling to more rapidly improve subsequent images.

Our implementation demonstrates an interactive application working with both ray tracing and path tracing renderers in situations where they would normally be considered too expensive. Moreover we accomplish this using a software only implementation without the use of 3D graphics hardware.

1 Introduction

In rendering, interactivity and high quality are often seen as competing and even mutually exclusive goals. Algorithms such as ray tracing [29] and path tracing [14] are widely used to produce high-quality, visually compelling images that include complex effects such as reflections, refraction, and global illumination. However, they have generally been considered too computationally expensive for interactive use.

Interactive use has typically been limited to lower-quality, often hardware accelerated rendering algorithms such as wireframe or scan-conversion. While these are perfectly adequate for many applications, often it would be preferable to achieve interactivity while preserving, as much as possible, the quality of a more expensive renderer. For example, it is desirable to use the same renderer when editing a scene as will be used for the final images or animation.

The goal of this work is to show how high quality ray-based rendering algorithms can be combined with the high framerates needed for interactivity, using only a level of computational power that is widely and cheaply available today. Once achieved, this creates a compelling visual interface that users quickly find addictive and are reluctant to relinquish. In the typical visual feedback loop, as illustrated in Figure 1(a), the expense of the renderer often strictly limits the achievable framerate².

The *render cache* is a new technique to overcome this limitation and allow interactive rendering in many cases where this was previously infeasible. The renderer is shifted out of the synchronous part of the visual feedback loop and a new display process introduced to handle image generation as illustrated in Figure 1(b). This greatly

¹iMAGIS is joint project of CNRS, INPG, INRIA and Université Joseph Fourier.

²In this paper we will use framerate to mean the rate at which the renderer or display process can produce updated images. This is usually slower than the framerate of the display device (e.g., a monitor or CRT).

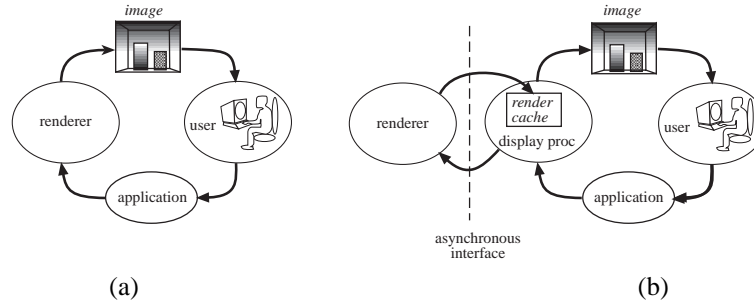


Fig. 1. (a) The traditional interactive visual feedback loop where the framerate is limited by the speed of the renderer. (b) The modified loop using the *render-cache* which decouples display framerate from the speed of the renderer.

reduces the framerate’s dependence on the speed of the renderer. The display process, however, does not replace the renderer and depends on it for all shading computations.

The display process caches recent results from the renderer as shaded 3D points, reprojects these to quickly estimate the current image, and directs the renderer’s future sampling. Reprojection alone would result in numerous visual artifacts, however, many of these can be handled by some simple filters. For the rest we introduce several strategies to detect and prioritize regions with remaining artifacts. New rendering samples are concentrated accordingly to rapidly improve subsequent images. Sampling patterns are generated using an error diffusion dither to ensure good spatial distributions and to mediate between our different sampling strategies.

The render cache is designed to make few assumptions about the underlying rendering algorithm so that it can be used with different renderers. We have demonstrated it with both ray tracing [29] and path tracing [14] rendering algorithms and shown that it allows them to be used interactively using far less computational power than was previously required. Even when the renderer is only able to produce a low number of new samples or pixels per frame (e.g., 1/64th of image resolution), we are able to achieve satisfactory image quality and good interactivity. Several frames taken from an interactive session are shown in Figure 2.

1.1 Previous Work

The render cache utilizes many different techniques and ideas to achieve its goal of interactivity including progressive refinement for faster feedback, exploiting spatio-temporal image coherence, using reprojection to reuse previously results, image space sparse sampling heuristics, decoupling rendering and display framerates, and parallel processing. The contribution of the render cache is to show how these ideas can be adapted and used simultaneously in a context where *interactivity* is considered of paramount importance, in a way that is both novel and effective.

One way to provide faster visual feedback and enhanced interactivity is to provide the user with approximate intermediate results rather than waiting for exact results to be available. This is often known as progressive refinement and has been used by many researchers (e.g., the “golden thread” of [3] or progressive radiosity [9]).

Many researchers have explored ways to exploit spatio-temporal image plane coherence to reduce the computational costs in ray tracing sequences of images. With a fixed

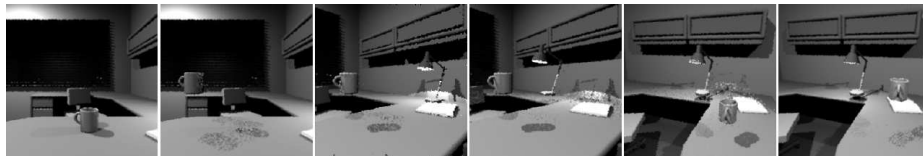


Fig. 2. Some frames from an interactive editing session using the render cache. The user is given immediate feedback when changing the viewpoint or moving objects such as the mug and desk lamp in this ray traced scene. While there are some visual artifacts, object positions are rapidly updated while other features such as shadows update a little slower. The user can continue to edit and work without waiting for complete updates. On a single processor, this session ran at ~ 5 fps and lasted about one minute. No graphics hardware acceleration was used. See Appendix for larger color images.

camera, changing materials or moving objects can be accelerated by storing partial or complete ray trees (e.g., [25, 13, 6]).

Sequences with camera motions can be handled by storing the rendered points and reprojecting onto the new camera plane. There are several inherent problems with reprojection including that the mapping is not a bijection (some pixels have many points map to them and some have none), occlusion errors (if the reprojected point is actually behind an occluding surface in the new view), and non-diffuse shading (a point's color may change when viewed from a different angle). Many different strategies for mitigating these problems have been proposed in the image-based literature (e.g., [8, 18, 17, 16, 26]), which relies heavily on reprojection.

Reprojection has also been used in ray tracing to accelerate the generation of animation sequences (e.g., [2, 1]). These methods save considerable computation by reprojecting data from the previous frame and only recomputing pixels which are potentially still incorrect. At a high level their operations are similar to those of the render cache but their goal (i.e. computing exact frames) is different. Our goal of interactivity requires the use of fast reconstruction heuristics that work reasonably well even in the presence of inexact previous frames and a prioritized sparse sampling scheme to best choose the limited number of pixels that can be recomputed per frame.

Sparse or adaptive image space sampling strategies (e.g., [21, 19, 11, 5]) can greatly reduce ray tracing costs. While most work has concentrated on generating single images, some researchers have also considered animations (e.g., using a uniform random sampling to detect changed regions [2] or uniform deterministic sampling in an interactive context [4]). The render cache introduces a new sampling strategy that combines several different pixel update priority schemes and uses an error diffusion dither [10] to mediate between our conflicting goals of a uniform distribution for smooth image refinement and concentrating samples in important regions for faster image convergence.

Parallel processing is another way to accelerate ray tracing and global illumination rendering (e.g., see [24] for one survey). Massive parallel processing can be used to achieve interactive ray tracing (e.g., [20, 22]), but this is an expensive approach. A better alternative is to combine parallel processing with intelligent display algorithms. For example, Parker et. al. [22] who used frameless rendering [4] to increase their framerate, could benefit from the render cache which produces better images and requires significantly fewer rays per frame.

The Post-Rendering 3D Warp [16] is an alternative intelligent display process. It displays images at a higher framerate than that of the underlying renderer by using

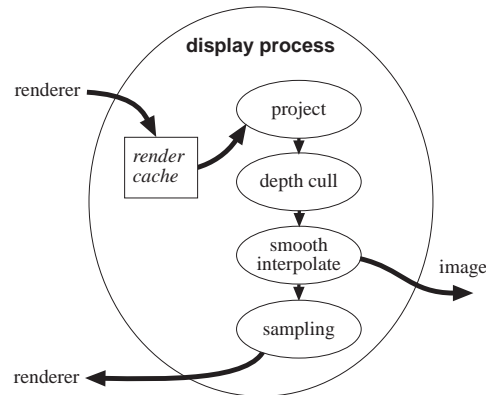


Fig. 3. The display process: The render cache receives and caches sample points from the renderer, which are then projected onto the current image plane. The results are filtered by the depth culling and interpolation steps to produce the displayed image. A priority image is also generated which is used to choose which new samples will be requested from the renderer.

image warping to interpolate from neighboring (past and future) rendered frames. One drawback is that the system must predict far enough into the future to render frames before they are needed for interpolation. This is trivial for a pre-scripted animation, but extremely difficult in an interactive context.

Another example is the “holodeck” [15] which was designed as a more interactive front end for Radiance [28]. It combines precomputation, reprojection, and online uniform sampling to greatly increase interactivity as compared to the Radiance system alone. Unlike the render cache though, it is not designed to handle dynamic scenes or long continuous camera motions and uses a less sophisticated sampling strategy.

2 Algorithm Overview

Our display process is designed to be compatible with many different renderers. The main requirement is that the renderer must be able to efficiently compute individual rays or pixels. Thus, for example, ray tracing like renderers are excellent candidates while scan conversion renderers are not. The display process provides interactive feedback to the user even when the renderer itself is too slow or expensive to produce complete images at an interactive rate (though the number of visual artifacts will increase if the renderer would take more than a few seconds to produce a full image).

There are several essential requirements for our display process. It must rapidly generate approximations to the current correct image based on the current viewpoint and the data in the render cache. It must control which rays or samples are rendered next to rapidly improve future images. It also must manage the render cache by integrating newly rendered results, and discarding old data when appropriate or necessary.

Image generation consists of projection, depth culling, and interpolation/smoothing steps. Rendered points from the cache are first projected onto the current view plane. We try to enforce correct occlusion both within a pixel by using a z-buffered projection, and among neighboring pixels using the depth culling step. The interpolation step fills in small gaps in the often sparse point data and produces the displayed image.

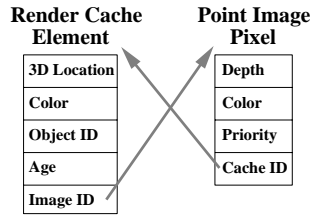


Fig. 4. The fields in the render cache and point image. Each point or element in the render cache contains a 3D location, a color, and an object id all provided by the renderer. They also have an age which is incremented each frame and an image id field which tells which pixel (if any) this point currently maps to. Each pixel in the point image contains a depth, a color, a priority, and the cache id of the point (if any) that is currently mapped to this pixel.

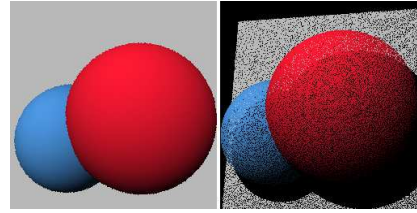


Fig. 5. Results of z-buffered point projection for a simple scene containing a white plane behind two diffuse spheres. The points generated for one viewpoint (left) are projected on the image plane for a new viewpoint (right). Notice that there are many gaps where no point projected onto a particular image (shown as black) and some points of the lighter plane are showing through gaps in the darker sphere points which should be occluding them.

Simultaneously, the display process also builds a *priority image* to guide future sampling. Because we expect that only a small subset of pixels can be rendered per frame, it important to direct the rendered sampling to maximize their benefit. Each pixel is given a priority value based on the relative value of rendering a new sample at that pixel. We then use an error diffusion dither [10] to choose the new samples to request. Using a dither both concentrates more samples in important regions and ensures that the samples are well spaced and distributed over the entire image. A diagram of the display process steps is shown in Figure 3.

3 Image Generation

Images are generated in our display process by projecting rendered points from the render cache onto an augmented image plane called *point image* (see Figure 4 for corresponding data fields). The projection step consists of a transform based on the current camera parameters as specified by the application program and z-buffering to handle the cases when more than one point maps to the same pixel. Whenever a point is mapped to a pixel, their corresponding data fields (see Figure 4) are updated appropriately including writing the point's color, depth, and a priority based on its age to the pixel.

The raw results of such a projection will contain numerous artifacts as illustrated in Figure 5. We handle some of the simpler kinds of artifacts using some filters while relying our sampling algorithm and newly rendered samples to resolve the more difficult artifacts in future frames. We have deliberately chosen to use only fast, simple techniques and heuristics in our system to keep its computational requirements both as light and consistent as possible.

3.1 Depth Culling and Smoothing/Interpolation

Some of the points, though formerly visible, may currently lie behind an occluding surface (e.g., due to object or camera motion). Visual artifacts, such as surfaces incorrectly “showing through” other surfaces, occur if such points are not removed (see Figure 6).

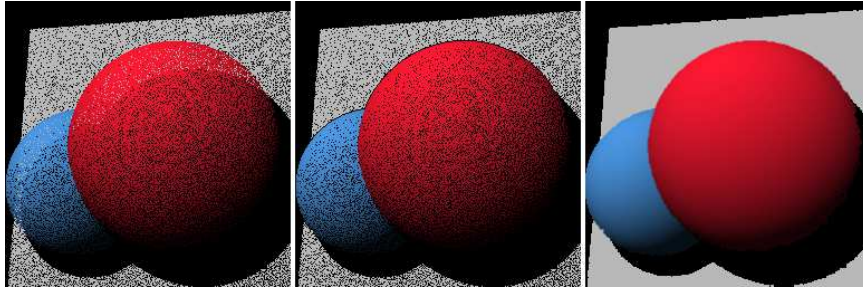


Fig. 6. Image reconstruction example: The raw projected points image (left) from Figure 5 is filtered by our depth-cull (middle) and interpolation (right) to produce an image that gives the correct impression that the surfaces are opaque and continuous.

Projection only removes occluded points if a point from the occluding surface maps to the same pixel. We remove more occluded points using a depth culling heuristic that searches for points whose depth is inconsistent with their neighbors.

Each pixel's 3x3 neighborhood is examined and an average depth computed, ignoring neighboring pixels without points. If the point's depth is significantly different from this average, then it is likely that we have points from different surfaces and that the nearer surface should now be occluding the farther one. Based on this assumption, we remove the point (i.e. we treat it as if no point had mapped to this pixel) if its depth is more than some threshold beyond the average depth (currently we use 10%). This heuristic both correctly removes many points which should have been occluded and falsely removes some genuinely visible points near depth discontinuity edges. Fortunately the incorrect removal artifacts are largely hidden by the interpolation step.

Next we use an interpolation and smoothing filter to fill in small gaps in the point image (see Figure 6). For each pixel, we again examine its 3x3 neighborhood and perform a weighted average³ of the corresponding colors. The weights are 4, 2, and 1 for center, immediate neighbor, and diagonal neighbors respectively, and pixels without points receive zero weight. This average becomes the pixel's displayed color except when there are no points in the neighborhood making the average invalid. Such pixels either retain the color they had in the previous frame or are displayed as black depending on the user's preference.

The quality of the resulting image depends on how relevant the cached points are to the current view. Actions such as rapid turning or moving through a wall can temporarily degrade image quality significantly. Fortunately the sparse sampling and interpolation tend to quickly restore image quality. Typically the image quality becomes usable again by the time that just one tenth of the cache has been filled with relevant points.

4 Sampling

Choosing which samples the renderer should compute next is another essential function of the display process. Since we expect that the number of new samples computed per frame to be much smaller than the number of pixels in the displayed image (typically by a factor between 8 and 128), it is important to optimize the placement of these sparse

³As described the system performs some slight smoothing even in fully populated regions. If this is considered objectionable, smoothing could easily be disabled at those pixels that had a point map to them.



Fig. 7. A image produced by the display process (left) along with its corresponding priority image (middle) and the dithered binary image specifying which sample locations will be requested next from the renderer. In this case the user is moving toward the upper left and the high priority regions are due to previously occluded regions becoming visible. Note that the dithering algorithm causes new samples to be concentrated in these high priority regions while staying well spaced and distributed over the entire image region.

samples. Samples are chosen by first constructing a grayscale sampling *priority image* and then applying an error diffusion dither algorithm. We use several heuristics to give high priority to pixels that we suspect are likely to contain visual artifacts.

The priority image is generated simultaneously with image reconstruction. Each point in the render cache has an age which starts at zero and is incremented each frame. When a point in the render cache maps to a pixel, that pixel's priority is set based on the point's age. This reflects the intuition that it is more valuable to recompute older samples since they are more likely to have changed. The priority for other pixels is set during the interpolation step based on how many of their neighbors had points map to them. Pixels with no valid neighbors receive the maximum possible priority while pixels with many valid neighbors receive only a medium priority. The intuition here is that it is more important to sample regions with lower local point densities first.

Choosing sampling locations from the priority image is equivalent to turning a grayscale image into a binary image. We want our samples to have a good spatial distribution so that the image visually refines in a smooth manner by avoiding the clumping of samples and ensuring that they are distributed over the whole image. We also want to concentrate more samples in high priority regions so that the image converges more quickly. A uniform distribution would not properly prioritize pixels, and a priority queue would not ensure a good spatial distribution. Instead we utilize a simple error diffusion dithering algorithm [10] to create the binary sample image (see Figure 7). The dithering approach nicely mediates between our competing sampling goals at the cost of occasionally requesting a low priority pixel.

In our implementation, scanlines are scanned in alternating directions and the priority at each pixel compared to a threshold value (total priority / # samples to request). If above threshold, this pixel is requested as a sample and the threshold subtracted from its priority. Any remaining priority is then propagated, half to the next pixel and half to the corresponding pixel on the next scanline.

4.1 Premature Aging

By default, points age at a constant rate, but it is often useful to prematurely age points that are especially likely to be outdated or obsolete. Premature aging encourages the

system to more quickly recompute or discard these points for better performance.

A good example is our color change heuristic. Often a new sample is requested for a pixel which already contains a point. We consider this a *resample*, record the old point's index in the sample request, and ensure that the requested ray passes exactly through the 3D location⁴ of the old point. We can then compare the old and new colors of resampled pixels to detect changes (e.g., due to changes in occlusion or lighting). If there is a significant change, then it is likely that nearby pixels have also changed. Therefore, we prematurely age any points that map to nearby pixels. In this way we are able to automatically detect regions of change in the image and concentrate new samples there. Another example is that we prematurely age points which are not visible in the current frame since it is likely that they are no longer useful.

4.2 Renderer and Application Supplied Hints

While we want our display process to work automatically, we also want to provide ways for the renderer and application to optionally provide hints to increase the display process' effectiveness. For example, the renderer can flag some points as being more likely to change than others and thus should be resampled sooner. The display process then ages these points at a faster rate. Some possible candidates are points on a moving objects, points in their shadows, or points which are part of a specular highlight. Together with the resample color change optimization, this can greatly improve the display process's ability to track the changes in such features.

The noise inherent in Monte Carlo renderers can cause the display process to falsely think that a sample's color has changed significantly during resampling. Falsely triggering the color change heuristic can prematurely age still valid points and wastefully concentrate samples in this region. To avoid this, the renderer can provide a hint that specifies the expected amount of noise in a result. This helps the display process to distinguish between significant color changes and variation simply due to noise.

We have also added a further optimization to help with moving objects. The application can provide rigid body transforms (e.g., rotation or translation) for objects. The display process then updates the 3D positions of points in the render cache with the specified object identifiers. This significantly improves the tracking of moving objects as compared to resampling alone though resampling is still necessary.

4.3 Cache Management Strategy

We use a fixed size render cache that is slightly larger than the number of pixels to be displayed. Thus each new point or sample must overwrite a previous one in the cache. The fixed size cache helps keep the computational cost low and constant. In dynamic environments, this also ensures that any stale data will eventually be discarded.

New points or samples that are resamples of an old point (see above) simply overwrite that point in the cache. Since the old point is highly likely to be either redundant or outdated, this simple strategy works well.

For other new points, we would like to find some no longer useful point to overwrite. One strategy is to replace the oldest point in the cache as it is more likely to be obsolete. However, we decided that doing this exactly would be unnecessarily expensive. Instead we examine a subset of the cache (e.g., groups of 8 points in a round robin order) and replace the oldest point found.

⁴Otherwise requested rays are generated randomly within the pixel to reduce aliasing and Moiré patterns.

5 Implementation and Results

We have implemented our display process and a simple test application that allows us to change viewpoints and move objects. The display process communicates with renderers using a simple abstract broker interface. This abstract interface allows us to both easily work with different renderers (two ray tracers and two path tracers so far) and to utilize parallel processing by running multiple instances of the renderers simultaneously. The broker collects the sample requests from the display process, distributes them to the renderers when they need more work and gathers rendered results to be returned to the display process. It is currently written for shared memory parallel processing using threads, though a message passing version for distributed parallel processing is also feasible.

The render mismatch ratio is a useful measure of the effectiveness of the render cache and our display process. We define this ratio as the number of pixels in a frame divided by the number of new samples or pixels produced by the renderer per frame. It is render cache's ability to handle higher mismatch ratios that allows us to achieve interactivity while using more expensive renderers and/or less computational power.

Working with a mismatch ratio of one is trivial; render and display each frame. Mismatch ratios of two to four can easily be handled using existing techniques such as frameless rendering [4]. The real advantage and contribution of the render cache is its ability to effectively handle higher mismatch ratios. In our experience, the render cache works well for mismatch ratios up to 64 and can be usable at even higher ratios. In many cases this allows us to achieve much greater interactivity with virtually no modification to the renderer. Performance in particular cases will of course depend on many factors including the absolute framerate, scene, renderer, and user task.

5.1 Results

Our current implementation runs on Silicon Graphics workstations using software only (i.e. we do not use any 3D graphics hardware). Our experience shows that the render cache can achieve interactive ray tracing even on single processor systems whose processing power is equivalent to that of today's PC computers. Specialized or expensive hardware is not required, though we can also exploit the additional rendering power of parallel processing when available.

Timings for the display process running on a 195Mhz R10000 processor in an SGI Origin 2000 are shown in Table 1. The display process can generate a 256x256 frame in 0.07 seconds for a potential framerate of around 14 frames per second. In a uniprocessor system, the actual framerate will be lower because part of the processors time must also be devoted to the renderer. In this case, we typically split the processors time evenly between the display process and the renderer for a framerate of around 7 fps. Even on a multiple processor machine it may be desirable to devote less than a full processor to the display process in order to increase the number of rendered samples produced.

Using larger images is trivial though it reduces the framerate. The time to produce each frame scales roughly linearly with the number of pixels to be displayed since all the data structures sizes and major operations are linear in the number of pixels.

We have tested the render cache in various interactive sessions using both ray tracing [29] and path tracing [14] renderers and on machines ranging from one to sixty processors. Some images from example sessions are shown in Figures 2 and 8 (see color plates in Appendix) and videos are available on our web page⁵.

⁵<http://www-imagis.imag.fr/Publications/walter>

Initialize buffers	0.0046 secs
Point projection	0.0328 secs
Depth cull	0.0085 secs
Interpolation	0.0139 secs
Display image	0.0027 secs
Request new samples	0.0053 secs
Update render cache	0.0027 secs
Total time	0.0705 secs

Table 1. Timings for the display process’ generation of a 256x256 image produced on a single 195Mhz R10000 processor. The display process is capable of producing about 14 frames per second in this case, though the actual framerate may be slower if part of the processors time is also devoted to rendering.

In all cases tested, the render cache provides a much more interactive experience than any other method using the same renderers that we are aware of (e.g., [22, 4]). The reprojection correctly tracks motion and efficiently reuses relevant previously rendered samples. While there are visual artifacts in individual frames, the prioritized sparse sampling smoothly refines the images and allows us to quickly recover from actions that make the previous samples irrelevant (e.g., walking through a wall). We still rely on the renderer for all shading calculations and need it to produce an adequate number of new samples per frame. Compared to previous methods though, we require far fewer new samples per frame to maintain good image quality.

All the sessions shown in Figure 8 used 320x320 resolution and ran at around 8 fps. The first three sessions used ray tracing and between two and four R10000 processors. An image from a sequence where the user walks through a door in Greg Larson’s cabin model is shown in the upper left. In the upper right, an ice cream glass has just been moved in his soda shoppe model, and its shadows are in the process of being updated. In the lower left, the camera is turning to the right in a scene with many ray traced effects including extensive reflection and refraction.

The lower right of Figure 8 shows a path tracing of Kajiya’s original scene. Path tracing simulates full global illumination and is much more expensive. The four processor version (shown) is no longer really adequate as too few new samples are rendered per frame resulting in more visual artifacts. Nevertheless, interactivity is still much better than it would be without the render cache. We have demonstrated good interactivity even in this case when using a sixty processor machine.

6 Conclusions

The render cache’s modular nature and generic interfaces allows it to be used with a variety of different renderers. It uses simple and fast algorithms to guarantee a fast consistent framerate and is designed for interactivity even when rendered samples are expensive and scarce. Reprojection and filtering intelligently reuse previous results to generate new images and a new directed sampling scheme tries to maximize the benefits of future rendered results.

Our prototype implementation has shown that we can achieve interactive framerates using software only for low but reasonable resolutions. We have also shown that it can enable satisfactory image quality and interactivity even when the renderer is only able to produce a small fraction of new pixels per frame (e.g., between 1/8 and 1/64 of

the pixels in a frame). We have also demonstrated it working with both ray tracing and path tracing and efficiently using parallel processors ranging from two to sixty processors. Moreover, we have shown the render cache can handle dynamic scenes including moving objects and lights.

We believe that the render cache has the potential to significantly expand the use of ray tracing and related renderers in interactive applications and provide interactive users with a much wider selection of renderers and lighting models to choose from.

6.1 Future Work

There are many ways in which the render cache can be further improved. Higher frame-rates and bigger images are clearly desirable and will require more processing power. With its fixed-size regular data structures and operations, the render cache could benefit from the small-scale SIMD instructions that are becoming common (e.g., AltiVec for PowerPC and SSE for Pentium III). It is also a good target for graphics hardware acceleration as its basic operations are very similar to those already performed by current graphics hardware (e.g., 3D point projection, z-buffering, and image filtering).

The lack of good anti-aliasing is one clear drawback of the render cache as presented here. Unfortunately since anti-aliasing is highly view dependent, we probably do not want to include anti-aliasing or area sampling within individual elements in the render cache [8]. This leaves supersampling as the most obvious solution though this will considerably increase the computational expense of the display process.

Although the render cache works well for renderer mismatch ratios up to 64, more work is needed to improve its performance at higher ratios. Some of the things that will be needed are interpolation over larger spatial scales, better very sparse sampling, and methods to prematurely evict obsolete points from the render cache.

Because the render cache works largely in the image plane, it is an excellent place to introduce perceptually based optimizations and improvements. Some examples include introducing dynamic tone mapping models (e.g., [27, 23]) or using perceptual based sampling strategy (e.g., [5]).

We also like to see our display process used with a wider variety of renderers such as Radiance [28], bidirectional path tracing, photon maps [12], and the ray-based gather passes of multipass radiosity methods [7].

Acknowledgements

We would like to thank people at the Cornell Program of Computer Graphics and especially Eric Lafortune for helpful early discussions on reprojection. We are indebted to Peter Shirley for many helpful comments and contributions. Also thanks to Greg Larson for making his mgl library and models available and special thanks to Al Barr for resurrecting from dusty tapes his original “green glass balls” model as used in Jim Kajiya’s original paper.

References

1. S. J. Adelson and L. F. Hodges. Generating exact ray-traced animation frames by reprojection. *IEEE Computer Graphics and Applications*, 15(3):43–52, May 1995.
2. S. Badt. Two algorithms taking advantage of temporal coherence in ray tracing. *The Visual Computer*, 4(3):123–132, Sept. 1988.
3. L. D. Bergman, H. Fuchs, E. Grant, and S. Spach. Image rendering by adaptive refinement. In *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20, pages 29–37, Aug. 1986.
4. G. Bishop, H. Fuchs, L. McMillan, and E. J. Scher Zagier. Frameless rendering: Double buffering considered harmful. In *Computer Graphics (SIGGRAPH '94 Proceedings)*, pages 175–176, July 1994.

5. M. R. Bolin and G. W. Meyer. A perceptually based adaptive sampling algorithm. In M. Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, pages 299–310, July 1998.
6. N. Brière and P. Poulin. Hierarchical view-dependent structures for interactive scene manipulation. In *SIGGRAPH 96 Conference Proceedings*, pages 83–90, Aug. 1996.
7. S. E. Chen, H. Rushmeier, G. Miller, and D. Turner. A progressive multi-pass method for global illumination. In *SIGGRAPH 91 Conference Proceedings*, pages 165–174, July 1991.
8. S. E. Chen and L. Williams. View interpolation for image synthesis. In J. T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 279–288, Aug. 1993.
9. M. F. Cohen, S. E. Chen, J. R. Wallace, and D. P. Greenberg. A progressive refinement approach to fast radiosity image generation. *Computer Graphics*, 22(4):75–84, August 1988. ACM Siggraph '88 Conference Proceedings.
10. R. W. Floyd and L. Steinberg. An adaptive algorithm for spatial greyscale. In *Proceedings of the Society for Information Display*, volume 17(2), pages 75–77, 1976.
11. B. Guo. Progressive radiance evaluation using directional coherence maps. In M. Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, pages 255–266, July 1998.
12. H. W. Jensen. Global illumination using photon maps. In *Rendering Techniques '96*, pages 21–30. Springer-Verlag/Wien, 1996.
13. D. A. Jevans. Object space temporal coherence for ray tracing. In *Proceedings of Graphics Interface '92*, pages 176–183, May 1992.
14. J. T. Kajiya. The rendering equation. In D. C. Evans and R. J. Athay, editors, *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20, pages 143–150, Aug. 1986.
15. G. W. Larson. The holodeck: A parallel ray-caching rendering system. In *Second Eurographics Workshop on Parallel Graphics and Visualisation*, Rennes, France, Sept. 1998.
16. W. R. Mark, L. McMillan, and G. Bishop. Post-rendering 3D warping. In *1997 Symposium on Interactive 3D Graphics*, pages 7–16. ACM SIGGRAPH, Apr. 1997.
17. N. Max and K. Ohsaki. Rendering trees from precomputed Z-buffer views. In *Eurographics Rendering Workshop 1995*. Eurographics, June 1995.
18. L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. In R. Cook, editor, *SIGGRAPH 95 Conference Proceedings*, pages 39–46, Aug. 1995.
19. D. P. Mitchell. Generating antialiased images at low sampling densities. In M. C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, pages 65–72, July 1987.
20. M. J. Muuss. Towards real-time ray-tracing of combinatorial solid geometric models. In *Proceedings of BRL-CAD Symposium*, 1995. <http://ftp.arl.mil/mike/papers/>.
21. J. Painter and K. Sloan. Antialiased ray tracing by adaptive progressive refinement. In *Computer Graphics (SIGGRAPH '89 Proceedings)*, pages 281–288, July 1989.
22. S. Parker, W. Martin, P. Sloan, P. Shirley, B. Smits, and C. Hansen. Interactive ray tracing. In *Symposium on Interactive 3D Computer Graphics*, April 1999.
23. S. N. Pattanaik, J. A. Ferwerda, M. D. Fairchild, and D. P. Greenberg. A multiscale model of adaptation and spatial vision for realistic image display. In *Computer Graphics*, July 1998. ACM Siggraph '98 Conference Proceedings.
24. E. Reinhard, A. Chalmers, and F. W. Jansen. Overview of parallel photo-realistic graphics. In *Eurographics '98 State of the Art Reports*. Eurographics Association, Aug. 1998.
25. C. H. Séquin and E. K. Smyrl. Parameterized ray tracing. In J. Lane, editor, *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23, pages 307–314, July 1989.
26. J. W. Shade, S. J. Gortler, L. He, and R. Szeliski. Layered depth images. In M. Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, pages 231–242, July 1998.
27. G. Ward. A contrast-based scalefactor for luminance display. In P. Heckbert, editor, *Graphics Gems IV*, pages 415–421. Academic Press, Boston, 1994.
28. G. J. Ward. The RADIANCE lighting simulation and rendering system. *Computer Graphics*, 28(2):459–472, July 1994. ACM Siggraph '94 Conference Proceedings.
29. T. Whitted. An improved illumination model for shaded display. *Communications of the ACM*, 23(6):343–349, June 1980.

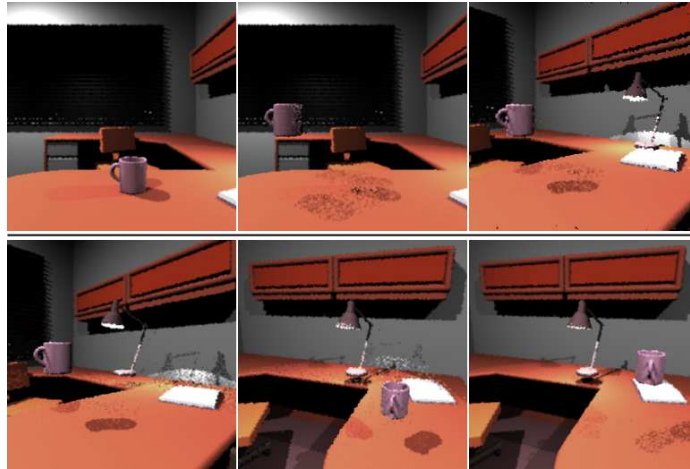


Fig. 2. Some frames from a render cache session. See main text for more detail.



Fig. 8. Some example images captured from interactive sessions. Some approximation artifacts are visible but the overall image quality is good. All scenes are ray traced except the lower right which is path traced. In the upper right image we have just moved the ice cream glass and you can see its shadow in the process of being updated on the table top.

4.5.4 Interactive Common Illumination for Computer Augmented Reality (EGRW'97)

Auteurs : George Drettakis and Luc Robert and Sylvain Bugnoux

Actes : 8th Eurographics Workshop on Rendering

Date : juin 1997

Interactive Common Illumination for Computer Augmented Reality

George Drettakis *, Luc Robert **, Sylvain Bougnoux **

* iMAGIS/GRAVIR-INRIA, ** ROBOTVIS

Abstract: The advent of computer augmented reality (CAR), in which computer generated objects mix with real video images, has resulted in many interesting new application domains. Providing *common illumination* between the real and synthetic objects can be very beneficial, since the additional visual cues (shadows, interreflections etc.) are critical to seamless real-synthetic world integration. Building on recent advances in computer graphics and computer vision, we present a new framework to resolving this problem. We address three specific aspects of the common illumination problem for CAR: (a) simplification of camera calibration and modeling of the real scene; (b) efficient update of illumination for moving CG objects and (c) efficient rendering of the merged world. A first working system is presented for a limited sub-problem: a static real scene and camera with moving CG objects. Novel advances in computer vision are used for camera calibration and user-friendly modeling of the real scene, a recent interactive radiosity update algorithm is adapted to provide fast illumination update and finally textured polygons are used for display. This approach allows interactive update rates on mid-range graphics workstations. Our new framework will hopefully lead to CAR systems with interactive common illumination without restrictions on the movement of real or synthetic objects, lights and cameras.

1 Introduction

Computer augmented reality (CAR) is a booming domain of computer graphics research. The combination of virtual or synthetic environments with real video images (RVI) has lead to many new and exciting applications. The core research in this area concentrates on the problems related to registration and calibration for real-time systems (see for example [3, 4]). Since many of these problems are still largely unresolved, little attention has been given to the problems of the interaction of *illumination* between the real and synthetic scenes.

Pioneering work in this domain has been performed by Fournier et al. [15]. This work (see Section 2.3 for a brief review), has shown how the computation of common illumination between the real and synthetic scene results in a greatly improved graphical environment with which the user can interact. The use of real video images eliminates the need to model complex environments in great detail, and, by nature, provides a realistic image to the user. In what concerns common illumination, the introduction of virtual objects in a real scene becomes much more natural and convincing when light exchanges between real and synthetic objects (such as shadows and interreflections) are present in the composite images presented to the user.

In this work we present a new common illumination framework, by addressing the following three stages: (a) camera calibration and modeling, (b) common illumination

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

** INRIA, BP93 06902 Sophia-Antipolis, Cedex, France, E-mail: Luc.Robert@inria.fr

updates and (c) rendering. The goal is to build a system which can compute common illumination at interactive update rates. The work reported here is in preliminary form; as such we have restricted the configuration we will be treating to the case of moving computer generated objects in a static real scene viewed by a static camera.

By using advanced vision techniques, we have replaced the tedious and inaccurate manual modeling process with a flexible and precise vision-based approach. This method allows us to model the real scene to the level of detail required, and to extract camera parameters simply and automatically. We use fast hierarchical [17, 25, 26] and incremental update [9] techniques for radiosity, permitting interaction with virtual objects in the CAR environment. Interactive update rates (a few seconds per frame) of the mixed real/synthetic environment, including common illumination is achieved by using a texture-based rendering approach on suitable hardware. We believe that the combination of advances in vision, illumination and graphics provides a framework which will lead to general interactive common illumination for CAR.

2 Previous and Related Work

2.1 Reconstruction of 3D models From Images

A number of techniques have been proposed for producing 3D models from images in photogrammetry and computer vision. The photogrammetry approach mostly focuses on accuracy problems, and the derived techniques produce three-dimensional models of high quality [2]. However, they generally require significant human interaction. Some commercial products, such as *Photomodeler*, already integrate these techniques. In computer vision, a number of automatic techniques exist for computing structure from stereo or motion (e.g., [8, 21, 11]). With these techniques, the three-dimensional models are produced much more easily, but they are less accurate, potentially containing a small fraction of gross errors.

Alternate representations have been proposed for realistic rendering from images. With image interpolation techniques [12, 22, 24], the scene is represented as a depth field, or equivalently, as a set of feature correspondences across two reference images. Although these implicit 3D representations are suited to rendering, they are not adapted to our framework since we need *complete* 3D data to perform radiosity computation.

Some recent approaches have been proposed to reduce the effort in the production of explicit 3D models of high quality, either by imposing constraints on the modeled scene [7], or by combining automatic computer vision processes with human interaction [13]. We follow this last approach in this paper.

2.2 Computer Augmented Reality

Much work has recently been performed in the domain of computer augmented reality. The main body of this research concentrates on the requirements of real-time systems [3]. In terms of illumination, these systems provide little, if any, common lighting information. Examples of work including some form of shadowing between real and synthetic objects are presented in [27] and [20].

Common illumination requires full 3D information, and thus should use explicit modeling of the real world objects. Similar requirements exist for the resolution of occlusion between real and virtual objects (e.g., [4]).

The wealth of excellent research in this domain will undoubtedly be central in the future work in common illumination (see Section 6.1). For now however, we concentrate on the issues directly related to illumination. The reader interested in an in-depth survey should refer to [3].

2.3 Radiosity and Common Illumination for CAR

In what follows, we consider the following configuration: we have an image I , which we call the “target image”, and, using techniques developed below, a set of geometric elements approximating the scene. All quantities related to the image will be noted “ $\hat{\cdot}$ ”. The most closely related previous research in common illumination is that of Fournier et al. [15]. We will be adopting many of the conventions and approximations used in that approach. In [15] many basic quantities are defined in a rather ad-hoc manner using information taken from image I . The average reflectivity of the scene $\hat{\rho}$ is selected arbitrarily. This can also be set as the average pixel value.

Once a value for $\hat{\rho}$ is set, the *overall reflectivity factor* R is defined as:

$$R = \frac{1}{1 - \hat{\rho}} . \quad (1)$$

The concept of “ambient radiosity” [6], \hat{B}_A is then used, permitting a first estimation of the exitance values E_i of the sources:

$$\hat{B}_A = \frac{R \sum_{all\ i} E_i A_i}{\sum_{all\ i} A_i}, \quad (2)$$

where E_i is the exitance of each object i and A_i its area. Another approximation of \hat{B}_A is given by:

$$\hat{B}_A = \frac{\sum_{all\ xy} p_{xy}}{N \hat{\rho}}, \quad (3)$$

where p_{xy} is the intensity of the pixel xy of the target image I , and N the total number of pixels of I . Equations (2) and (3) allow us to approximate the values of E_i if we know the number and area of the real sources.

Fournier et al. also proposed a first approximation of the radiosity on each geometric element i , which we call \hat{B}_i , which is the average value of the pixel intensities covered by element i .

In our approach, we improve the ease of modeling, as well as the lighting update and final display speeds compared to [15]. Nonetheless, to achieve these improvements, we sacrifice certain advantages of Fournier et al.’s system: we currently can only handle a static camera and real scene, and the quality of rendering may be slightly degraded compared to that obtained by ray-traced correction to a real image. Such degradation is mainly due to slight texture/polygon misalignment. However, since this paper is an attempt at defining a new approach to common illumination, we consider the above mentioned shortcomings as challenges for future research (Section 6.1).

3 Semi-Automatic Image-Driven Modelling Using Computer Vision

In this section we describe the creation of the three-dimensional model using vision-assisted techniques. We first compute the intrinsic parameters of the camera (focal length, aspect ratio) by using an image of a calibration pattern. Twelve images are then used to automatically build a set of panoramic images. The relative positions/orientations of the cameras are then computed, based on point correspondences. We thus construct a geometric model of the room by computer-vision assisted, image-based interaction. Finally, textures are extracted and de-warped automatically. The whole process took approximately 4 hours for the scene shown in Figure 1.

3.1 Camera Calibration Using a Target

The intrinsic parameters of the camera (see [10] for more details about the imaging geometry of cameras) are computed using the calibration technique described in [23]. We need to take one image of a non-planar calibration pattern, i.e., a real object with visible features of known geometry. With minimal interaction (the user only needs to click the approximate position in the image of 6 reference points), an estimate of the camera parameters is computed. This estimate is then refined by maximising, over the camera parameters, the sum of the magnitudes of the image gradient at the projections of a number of model points.

The output of the process is a 3×4 matrix, which is decomposed as the product of a matrix of intrinsic parameters and a 4×4 displacement (rotation, translation) matrix (computation described in [10]).

3.2 Image Acquisition and Mosaicing

Though the minimum number of viewpoints for stereo reconstruction is two, we acquired images from four distinct viewpoints for better accuracy of the reconstructed 3D geometry. The viewpoints lie approximately at the vertices of a 1-meter-wide vertical square in one corner of the room.

To enlarge the field-of-view, we built panoramic images using mosaicing [28, 19]. At each viewpoint, we took three left-to-right images with an overlap of approximately 50% between two consecutive images. During this process, we were very careful at each viewpoint not to translate the camera but restrict motion to rotation. This guarantees that there exist linear projective transformations which warp the left and right images onto the center one.

For each triple of images, we computed these transformations automatically [30]. The two warped images and the center images were then “pasted” on the same plane. An example of mosaic is shown in Figure 1.

3.3 Computation of the Relative Geometry of the Cameras

In the next stage, we estimate the relative geometry of all the cameras, i.e., the rotations \mathbf{R}_{1i} and translations \mathbf{t}_{1i} of all cameras with respect to, say, the first one. For this, we identify corresponding points across the images. This is done in a semi-manual manner. Using the system `totalcalib` developed at ROBOTVIS (Figure 2), the user first clicks on a reference point in one image. The system then searches for matches in the other images, using window-based cross-correlation. This is shown in Figure 2 (a), with the annotated white points. The matches proposed correspond to the regions which are

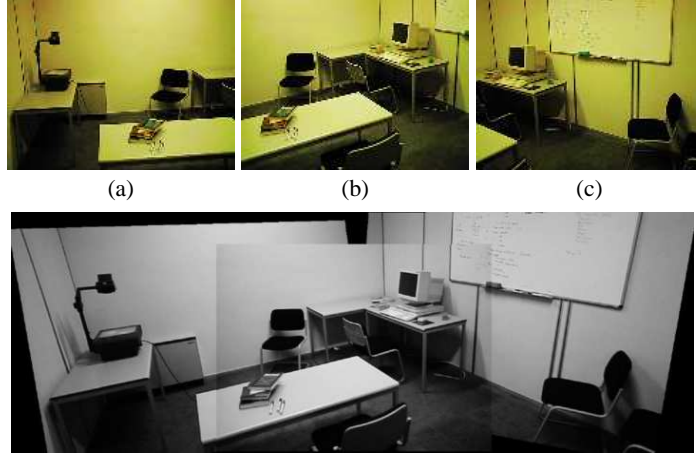


Fig. 1. Three original images, and the resulting mosaic (see text and also Colour Section).

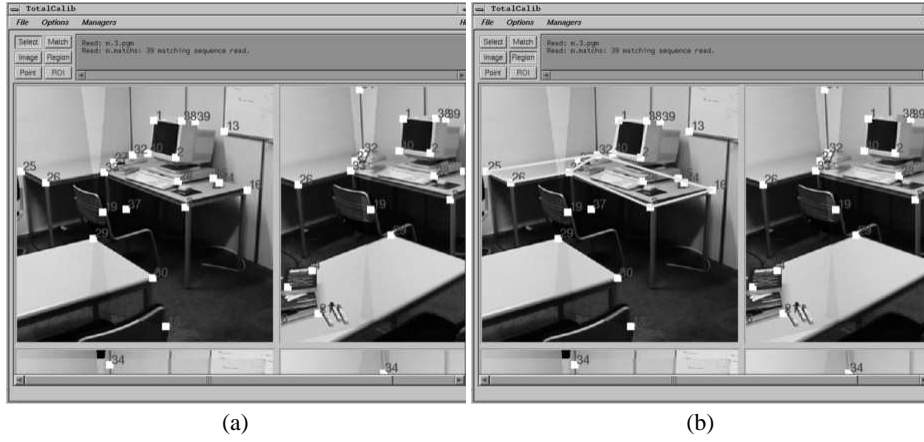


Fig. 2. (a) A totalcalib session; matched points are shown in white and are annotated. (b) Selection of the regions to reconstruct (e.g., the white polygon on the table-top).

most similar to the image around the reference point. In most cases these points indeed represent the same object as the reference point. If not, the user can manually correct the errors.

Based on the point correspondences, we compute the fundamental matrices F_{1i} (see appendix 7) using the non-linear method described in [29]. The minimum number of correspondences is 8 in theory, but for better accuracy we used about 30 points spread over the whole scene (see Figure 2).

From F_{1i} and the intrinsic parameters, we then derive, using the technique described in [18], the rotation \mathbf{R}_{1i} and translation \mathbf{t}_{1i} . In fact, each translation is known only up to a scale factor, which corresponds to choosing an arbitrary unit for distances in space. Translation \mathbf{t}_{1i} ($i > 2$) is rescaled with respect to \mathbf{t}_{12} by using point correspondences

visible in images 1,2 and i and comparing space distances computed with image pairs $(1, 2)$ and $(1, i)$.

From this initial estimate, we then run a non-linear minimisation process known as *bundle adjustment* in photogrammetry [2], which refines the estimate of the rotations and translations. We end up with an estimate of rotations and translations of all cameras with respect to the first camera.

3.4 Building the 3D Model and Extracting/De-warping the Textures

To build the polygons of the three-dimensional model, we first define the geometry of their vertices using the same semi-automatic technique. Their 3D coordinates are then obtained by inverting the projection equations. This process is known as *reconstruction* in computer vision or *intersection* in photogrammetry. We then manually define the topology of the polygons by selecting and connecting vertices in the images (see Figure 2(b)). The resulting model is stored in a standard 3D format.

For each polygon, we finally compute a texture image by de-warping the original image and bringing it back to the plane of the polygon. In this process, the resolution of the texture image can be chosen arbitrarily, as well as the directions of the axes of texture coordinates. The x -axis is chosen parallel to the longest edge of the polygon, which in most cases maximises the fraction of the texture image which lies inside the polygon and will be actually rendered. The choice of the texture resolution is based on the following criterion: when projecting one pixel of the texture image onto the reference image, one should obtain a small quadrilateral whose dimensions are all smaller than one pixel. This guarantees that the final synthesized images have approximately the same level of detail as the initial ones.

4 A Fast Hierarchical Method for Common Illumination

Recent advances in global illumination technology allow us to calculate the lighting efficiently, using hierarchical radiosity [17], clustering [25, 26] and incremental update methods [9]. To initialise the system, the calculation of certain basic parameters is required. We adopt many of the conventions used by Fournier et al. [15], adapting them appropriately to the application and the requirements at hand.

Two main stages are required: (a) initialisation of basic parameters such as exitance values for the real sources, radiosity and reflectance for the real video image (RVI) objects, and (b) the creation of a full hierarchical radiosity system, including the cluster hierarchy and “line-space” hierarchy of links and shafts required for the incremental solution.

4.1 Initialising the Basic Parameters

As discussed in Section 2.3 the basic approximations proposed in [15] can be used to estimate the set of parameters required to create a hierarchical representation of the (real) light transfer in the CAR scene. In the same spirit as this approach, we define the reflectance of each patch i to be:³

³ This is easier to calculate than the neighbourhood in [15].

$$\hat{\rho}_i = \frac{\hat{B}_i}{\hat{B}_A} \times \hat{\rho} \quad (4)$$

Note that during subdivision, \hat{B}_i is updated to reflect the average intensity of the pixels covered by the newly subdivided sub-element. The calculation of \hat{B}_i is performed by rendering the polygon textured with the corresponding part of the target (real) image I into an offscreen buffer and averaging the resulting pixel values. Once the new \hat{B}_i is computed for the child element, the value $\hat{\rho}_i$ is updated.

It is important to note that this approach is a coarse approximation, since we cannot distinguish between shadows and obstacles in the image. Since we are simply computing an overall correction to illumination, we accept this approximation for now, but resolving this issue is definitely part of required future work.

Since we have an initial geometric model of the real sources, we can easily estimate their exitance. If (as is the case in the examples presented in Section 5), we have sources of equal power and area, the relations of equations (2) and (3) suffice to approximate E_i . If on the other hand we have a larger number of different sources, we need to estimate their value. This can be done easily by creating a link hierarchy using the \hat{B}_i 's and simply pulling \hat{B} up the hierarchy. If we have m sources, by selecting m elements we have m equations giving us a good approximation of the E_i 's.

4.2 Creating a Hierarchical Radiosity System

Once the values of E_i and ρ_i are estimated (we set $\rho_i = \hat{\rho}_i$ for each surface element), we have everything we need to perform a normal hierarchical radiosity iteration. Consider for example Figure 3(a), which shows the radiosity calculation for the real scene previously presented in Figure 1(b). Note that we only use *one* image of the mosaic (Figure 1(b) in our case) from which to extract textures.

The refinement stage of hierarchical radiosity proceeds as usual, and is left to run to “convergence”, i.e. when the radiosity values no longer change much. Once completed, we have what we call an *original* value for the radiosities of all real objects. We store this value, \tilde{B}_i , on each hierarchical element (cluster, surface or sub-patch). The value \tilde{B}_i is a (relative) representation of the illumination due to real sources.

The next step is the addition of computer generated objects. This is performed by adapting the methods described in [9]. We thus group the synthetic objects into “natural” clusters (i.e. a chair or a desk lamp) and we add them into the scene. To update the existing hierarchical radiosity system, we use the line-space traversal approach to efficiently identify the links affected by the CG object being inserted, and we incrementally perform the appropriate modification to illumination. As a result all patches now have a (possibly modified) radiosity value B_i .

4.3 Display

To achieve interactive update rates, we need to display the result of the combination of real and synthetic environments at interactive rates. This requirement precludes the use of the ray-casting approach of [15]. Our solution is to exploit the real-time texture capacities currently available on mid- and high-range graphics workstations.

To display the effects of a change due to the interference of a CG object with an RVI object, we simply modulate the texture by the ratio: B_i / \tilde{B}_i . This operation requires the capacity to modulate “positively” and “negatively”, so we must coherently re-scale this ratio to always lie between zero and one.

4.4 Interactive Common Illumination of CG Objects in a Real Scene

Using an implicit hierarchical description of the line segment space contained between hierarchical elements, we can rapidly identify the links modified [9]. This is achieved by keeping a hierarchy of shaft structures [16] associated with the links and inactive (or *passive*) refined links.

In the case of the CAR application, special treatment is required to ensure that lighting effects created by CG objects are well represented. The refinement process is thus adapted to reflect this, by imposing finer subdivision for shadows or additional illumination due to the interaction of CG objects with the real scene (see Figure 3(c)).

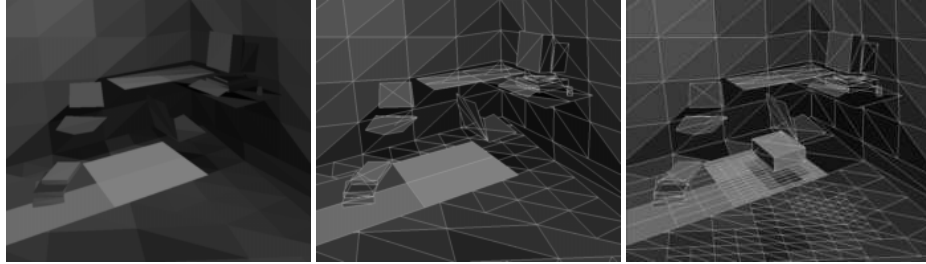


Fig. 3. (a) The radiosity \tilde{B} computed by the initialisation phase of the algorithm. (b) The corresponding mesh. (c) The radiosity B and the mesh after the addition of the CG object.



Fig. 4. (a) The complete CAR rendering using RVI texture polygons for display, including the CG object. (b) The CG object moves to the left: update takes 2.5 seconds. (See Colour Section).

5 Results

The RVI scene we have used was modeled with 98 input polygons. This is a coarse representation, but sufficient for the example we wish to show here. We have a total of 4 512x512 textures (for the walls and floor), and 2, 2 and 6 textures of resolution 256x256, 128x128 and 64x64 respectively, for the detail objects of the scene.

In Figure 3(a) we show the result of the initialisation step where the geometry is displayed using the original radiosity \tilde{B}_i . Notice the low level of subdivision. The cor-

responding mesh is shown in Figure 3(b). After subdivision, the number of leaf elements is 512.

In Figure 4(a) we show the complete CAR image, including the CG object, and the corresponding shadow on the table top in the foreground. Notice how the mesh (Figure 4(b)) is much finer in the regions affected by the computer graphics object, with a total of 905 leaf elements.

The addition of the CG object took 2.8 seconds. When moving the dynamic object (see Figure 4), the update to illumination requires on average 2.5 seconds, on an Indigo 2, R4400 200Mhz High-Impact.

6 Future Work and Conclusions

The methodology we presented here was intended, as mentioned above, as a first step in a new direction for the treatment of common illumination for CAR. We thus consider it important to indicate why we believe that our framework is a suitable starting point for the treatment of more general configurations.

The ultimate goal is to have seamless, real-time mixing of real and synthetic scenes, with shared realistic illumination. There is a lot of work to be done before this goal can be achieved, much of which is related to hardware, vision, registration and sensing (see [3] for more detail). We concentrate here on the issues directly or indirectly related to common illumination and display.

6.1 Future Work

The first restriction to lift is that of a static camera. As a first step, we will be using pre-recorded real video sequences and attempt to mix real and synthetic scenes. Several problems result from this, notably camera calibration and correct rendering.

To deal with the problem of camera calibration, a first approach could be to use a set of “keyframes” for which the process described in this paper is applied, and to use point-tracking techniques to update the projective matrices as we move from one point to the other.

For rendering, the problem is posed by the fact that different de-warped textures will be associated with the same geometry at different viewpoints. Simple interpolation schemes will not work, since the occlusion configuration will have changed. Thus an obstacle elimination scheme will have to be developed, using known vision techniques enhanced with the available 3D and lighting information.

A different restriction to overcome is to permit motion of CG light sources. This requires an adaptation of the incremental update method [9], most notably in what concerns the representation of direct lighting shadows and corresponding refinement. The motion of real sources will result in similar problems.

Moving real objects is also an important challenge. In the context of pre-recorded sequences, much of the difficulty will be overcome by the explicit 3D modeling of the object. The removal of real objects is also an interesting challenge, and will require the use of some of the techniques developed for the treatment of the texture de-warping problem, in particular to effect a “removal” of a real object. Image-based rendering approaches [5] may prove useful here as well.

The move to real-time video acquisition and common illumination will be the greatest challenge of all. We believe that the knowledge and experience acquired in resolving the problem for pre-recorded sequences will prove extremely fruitful for the development of a solution working in real time.

6.2 Shortcomings of the Current Approach

Despite the encouraging first results, there are several shortcomings in the approach presented here.

In the system presented we have not shown the addition of virtual lights. This is not too hard to achieve, but requires some modification to the incremental update approach, since the addition of a light source typically affects a large part of the environment. In addition, special attention must be taken in the re-scaling of the image before display since the addition of a source can add an order (or orders) of magnitude to the radiosity values of the scene.

In the lighting simulation phase we should use the variation of the RVI texture to aid refinement, and distinguish between variation due to occluding objects and shadows. The use of the obstacle removal techniques for textures can aid in this (see Section 6.1).

The refinement process is central: the modulation of the RVI textures by the changes in visibility is unforgiving. If a partially visible link is too far up the hierarchy, the resulting “spread shadow” is very visible, and spoils the effect of seamless real/synthetic merging. Since in synthetic-only environments these effects can usually be ignored, little has been previously done to address these issues.

The estimation of the initial parameters is also a major problem. The current estimations are very much ad-hoc. Nonetheless, what is important is not the precision of the approximation (since we are adding fictional objects, there is no “correct” solution), but the effect of more accurate choices which could result in more convincing results.

6.3 Conclusions

We have introduced a new framework for dealing with the problem of common illumination between real and synthetic objects and light sources in the context of computer augmented reality. We first use state-of-the-art vision techniques to calibrate cameras and estimate projection matrices, as well as recent image-based modeling approaches to create a model of the real environment. We then use rapid incremental hierarchical radiosity techniques to insert computer generated objects and manipulate them interactively. To achieve interactive display we use radiosity-modulated textures.

We have developed a working system for the restricted case of a static camera and static real environment. The prototype system we present shows that it is possible to create convincing CAR environments in which CG objects can be manipulated interactively. Compared to previous work in common illumination (notably [15]), our framework allows easier modeling and calibration, faster illumination updates and rapid display of CAR scenes.

Nonetheless, much more remains to be done. We have briefly discussed some possible future research paths, by removing the restrictions one by one, to achieve interactive common illumination for first a moving camera, then moving lights and finally moving real objects. We will initially be investigating these issues for pre-recorded video sequences, before taking the plunge into real-time acquisition.

In conclusion, we believe that the use of advanced, user-friendly image-based vision approaches to modeling and camera calibration, in conjunction with rapid incremental lighting and texture-based rendering, are a promising avenue leading to interactive common illumination for CAR. It will probably be a long time before we can interact naturally with virtual objects or creatures in our living room, but any solution to such a goal necessarily requires real-time common illumination.

Acknowledgements Thanks to Céline Loscos and François Sillion for carefully re-reading the paper and for many fruitful discussions. Thanks to Frédo Durand for help with the video as well Rachel Orti, Mathieu Desbrun and Agata Opalach for final production.

References

1. Boston, MA, June 1995. IEEE Computer Society Press.
2. K.B. Atkinson, editor. *Close Range Photogrammetry and Machine Vision*. Whittles Publishing, 1996.
3. Ronald T. Azuma. A survey of augmented reality. In *Presence: Teleoperators and Virtual Environments (to appear) (earlier version in Course Notes #9: Developing Advanced Virtual Reality Applications, ACM SIGGRAPH (LA, 1995), 20-1 to 20-38, 1997.* http://www.cs.unc.edu/azuma/azuma_AR.html.
4. D. E. Breen, R. T. Whitaker, E. Rose, and M. Tuceryan. Interactive occlusion and automatic object placement for augmented reality. *Computer Graphics Forum*, 15(3):C11–C22, September 1996.
5. Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 279–288, August 1993.
6. Michael F. Cohen, Shenchang Eric Chen, John R. Wallace, and Donald P. Greenberg. A progressive refinement approach to fast radiosity image generation. *Computer Graphics*, 22(4):75–84, August 1988. Proceedings SIGGRAPH '88 in Atlanta, USA.
7. P.E. Debevec, C.J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In Holly Rushmeier, editor, *SIGGRAPH*, pages 11–20, New Orleans, August 1996.
8. Umesh R. Dhond and J.K. Aggarwal. Structure from stereo - a review. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(6):1489–1510, 1989.
9. George Drettakis and François Sillion. Interactive update of global illumination using a line-space hierarchy. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings (Los Angeles, CA)*, Annual Conference Series. ACM SIGGRAPH, August 1997.
10. Olivier Faugeras. On the evolution of simple curves of the real projective plane. Technical Report 1998, INRIA, 1993.
11. Olivier Faugeras. *Three-Dimensional Computer Vision: a Geometric Viewpoint*. MIT Press, 1993.
12. Olivier Faugeras and Stéphane Laveau. Representing three-dimensional data as a collection of images and fundamental matrices for image synthesis. In *Proceedings of the International Conference on Pattern Recognition*, pages 689–691, Jerusalem, Israel, October 1994. Computer Society Press.
13. Olivier Faugeras, Stéphane Laveau, Luc Robert, Gabriella Csurka, Cyril Zeller, Cyrille Gauchlin, and Imed Zoghalmi. 3-d reconstruction of urban scenes from image sequences. *CVGIP: Image Understanding*, 1997. To appear.
14. Olivier Faugeras, Tuan Luong, and Steven Maybank. Camera self-calibration: theory and experiments. In G. Sandini, editor, *Proc 2nd ECCV*, volume 588 of *Lecture Notes in Computer Science*, pages 321–334, Santa Margherita Ligure, Italy, May 1992. Springer-Verlag.
15. Alain Fournier, Atjeng S. Gunawan, and Chris Romanzin. Common illumination between real and computer generated scenes. In *Proceedings Graphics Interface '93*, pages 254–263. Morgan Kaufmann publishers, 1993.
16. Eric A. Haines. Shaft culling for efficient ray-traced radiosity. In Brunet and Jansen, editors, *Photorealistic Rendering in Comp. Graphics*, pages 122–138. Springer Verlag, 1993. Proc. 2nd EG Workshop on Rendering (Barcelona, 1991).
17. Pat Hanrahan, David Saltzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. *Computer Graphics*, 25(4):197–206, August 1991. SIGGRAPH '91 Las Vegas.
18. R.I. Hartley. In defence of the 8-point algorithm. In *Proceedings of the 5th International Conference on Computer Vision [1]*, pages 1064–1070.

19. M. Irani, P. Anandan, and S. Hsu. Mosaic based representations of video sequences and their applications. In *Proceedings of the 5th International Conference on Computer Vision* [1], pages 605–611.
20. P. Jancène, F. Neyret, X. Provot, J-P. Tarel, J-M. Vézien, C. Meilhac, and A. Véroust. Res: computing the interactions between real and virtual objects in video sequences. In *2nd IEEE Workshop on networked Realities*, pages 27–40, Boston, Ma (USA), October 1995. <http://www-rocq.inria.fr/syntim/textes/nr95-eng.html>.
21. C.P. Jerian and R. Jain. Structure from motion. a critical analysis of methods. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(3):572–587, 1991.
22. L. McMillan. Acquiring immersive visual environments with an uncalibrated camera. Technical Report TR95-006, University of North Carolina, 1995.
23. L. Robert. Camera calibration without feature extraction. *Computer Vision, Graphics, and Image Processing*, 63(2):314–325, March 1995. also INRIA Technical Report 2204.
24. S.M. Seitz and C.R. Dyer. Physically-valid view synthesis by image interpolation. In *Proc. IEEE Workshop on Representation of Visual Scenes*, pages 18–25, Cambridge, Massachusetts, USA, June 1995.
25. François Sillion. A unified hierarchical algorithm for global illumination with scattering volumes and object clusters. *IEEE Trans. on Vis. and Comp. Graphics*, 1(3), September 1995.
26. Brian Smits, James Arvo, and Donald P. Greenberg. A clustering algorithm for radiosity in complex environments. In Andrew S. Glassner, editor, *SIGGRAPH 94 Conference Proceedings (Orlando, FL)*, Annual Conference Series, pages 435–442. ACM SIGGRAPH, July 1994.
27. Andrei State, Gentaro Hirota, David T. Chen, Bill Garrett, and Mark Livingston. Superior augmented reality registration by integrating landmark tracking and magnetic tracking. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings (New Orleans, LO)*, Annual Conference Series, pages 429–438. ACM SIGGRAPH, Addison Wesley, August 1996.
28. Richard Szeliski. Video mosaics for virtual environments. *IEEE Computer Graphics and Applications*, 16(2):22–30, March 1996.
29. Zhengyou Zhang, Rachid Deriche, Olivier Faugeras, and Quang-Tuan Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial Intelligence Journal*, 78(1-2):87–119, 1994.
30. I. Zoghiani, O. Faugeras, and R. Deriche. Using geometric corners to build a 2d mosaic from a set of images. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, Puerto Rico, June 1997. IEEE.

7 Appendix: Epipolar geometry

In the general case of one or two cameras observing a non-planar scene from two different viewpoints, the three-dimensional geometry of the scene and of the cameras can be characterised by the *epipolar geometry* of the cameras, a purely projective geometric property which depends only on the configuration of the cameras. It tells us that given one point in one image, we can draw a line in the second image on which the corresponding point (i.e., the point representing the same physical point in space) necessarily lies. The epipolar geometry is captured by a 3×3 singular matrix called the *fundamental matrix* [14]: Two image points $\mathbf{m}_1, \mathbf{m}_2$ represent the same point in space if and only if

$$\mathbf{m}_2^T \mathbf{F}_{12} \mathbf{m}_1 = 0 \quad (5)$$

The fundamental matrix is related to the intrinsic and extrinsic parameters of the two cameras:

$$\mathbf{A}_2^T \mathbf{F}_{12} \mathbf{A}_1 = [\mathbf{t}]_{\times} \mathbf{R} \quad (6)$$

where $\mathbf{R} = \mathbf{R}_1 \mathbf{R}_2^T$, $\mathbf{t} = -\mathbf{R}_1 \mathbf{R}_2^T \mathbf{t}_2 + \mathbf{t}_1$ represent the inter-camera motion and $[\mathbf{t}]_{\times}$ is the antisymmetric matrix such that $\forall \mathbf{x}, [\mathbf{t}]_{\times} \mathbf{x} = \mathbf{t} \times \mathbf{x}$.

The fundamental matrix can be computed from point correspondences in the images, without knowing anything about the intrinsic parameters of the cameras (focal length, aspect ratio, etc.). Robust programs which automatically perform this computation [29] are now publicly available⁴.

⁴ <ftp://krakatoa.inria.fr/pub/robotvis/BINARIES>



Fig. 5. The mosaic resulting from 3 original images. Textures are extracted using one image only.



Fig. 6. (a) The complete CAR rendering using the RVI texture polygons for display, including the CG object (b) The CG object has moved to the left: the update takes 2.5 seconds.

4.5.5 Interactive Virtual Relighting and Remodeling of Real Scenes (EGRW'99)

Auteurs : C. Loscos and M.-C. Frasson and G. Drettakis and B. Walter and X. Granier and P. Poulin

Actes : 10th Eurographics Workshop on Rendering

Date : juin 1999

Interactive Virtual Relighting and Remodeling of Real Scenes

Céline Loscos[†], Marie-Claude Frasson^{†‡}, George Drettakis[†],
Bruce Walter[†], Xavier Granier[†], Pierre Poulin[‡]

[†]iMAGIS¹-GRAVIR/IMAG-INRIA

B.P. 53, F-38041 Grenoble, Cedex 9, France

[‡]Département d'informatique et de recherche opérationnelle, Université de Montréal

Abstract. Lighting design is often tedious due to the required physical manipulation of real light sources and objects. As an alternative, we present an interactive system to *virtually* modify the lighting and geometry of scenes with both real and synthetic objects, including mixed real/virtual lighting and shadows.

In our method, real scene geometry is first approximately reconstructed from photographs. Additional images are taken from a single viewpoint with a real light in different positions to estimate reflectance. A filtering process is used to compensate for inaccuracies, and per image reflectances are averaged to generate an approximate reflectance image for the given viewpoint, removing shadows in the process. This estimate is used to initialise a global illumination hierarchical radiosity system, representing real-world secondary illumination; the system is optimized for interactive updates. Direct illumination from lights is calculated separately using ray-casting and a table for efficient reuse of data where appropriate.

Our system allows interactive modification of light emission and object positions, all with mixed real/virtual illumination effects. Real objects can also be virtually removed using texture-filling algorithms for reflectance estimation.

1 Introduction

Designing the illumination of real environments has always been a difficult task. Lighting design for home interiors for example, is a complex undertaking, requiring much time and effort with the manipulation of physical light sources, shades, reflectors, etc. to create the right ambiance. In addition other physical objects may need to be moved or otherwise changed. The problem is even more complex on movie sets or exterior lighting design. The fundamental trial-and-error nature of the relighting process makes it painful and often frustrating; more importantly, the requirements of constructing and moving real objects and light sources make testing many different potential designs often impossible.

Ideally, we would like to perform such processes entirely synthetically. The lighting designer would simply photograph the environment to be relit and/or remodeled, and then create the different conditions by computer simulation so that they can be evaluated appropriately.

Evidently, such a goal is very hard to accomplish. In this paper we provide first solutions to a subset of this goal, inspired by techniques developed for computer augmented reality, and common illumination between the real and the synthetic scenes [2, 10].

Our method starts with a preprocess, in which real geometry is reconstructed from

¹iMAGIS is joint project of CNRS, INPG, INRIA and Université Joseph Fourier.

a series of photos [20], taken from several different viewpoints. A second set of images (which we call *radiance images*) are taken from a *fixed* viewpoint with a real light source in different positions. The geometry and radiance images are used to extract an approximate reflectance at each pixel for the given point of view. Because reflectance is harder to estimate in shadowed regions, we try to have each visible surface point unshadowed in at least one image. We compensate for geometric and photometric imprecision by filtering and combining results from the individual radiance images. The result of this new approach is an acceptable estimate of reflectance, called a *reflectance image*; in the process, shadows are removed in a satisfactory manner.

Our main goal is to provide *interactive* manipulation of mixed real and virtual environments with common illumination. To achieve this we have separated the calculation of direct and indirect illumination. The reflectance image is used to initialise a hierarchical radiosity system with clustering [23], optimized for dynamic updates [6]. This structure is used for rapid updates of *indirect* light, while *direct* light is computed on a pixel-by-pixel basis. For direct light many components can be pre-computed and cached in a table for rapid use, and in other cases the changes are limited to small regions of screen space, permitting interactive updates. Working on a pixel-by-pixel basis results in high quality direct shadows and also facilitates the removal of real objects, since we can simply manipulate the reflectance image using texture generation methods.

It is important to note outright that we do not attempt to extract *accurate* reflectance values. The goal is to achieve *convincing* relighting at interactive rates. To this end we can ignore inaccuracies and small artifacts, if the overall effect is believable.

2 Previous work

A large body of literature exists in computer vision on reconstructing 3D scenes from photos [8]. However the quality of the extracted 3D models has only recently become satisfactory for computer graphics applications with the presentation of interactive systems such as *Photomodeler* [19], *REALISE* [9, 15], *Façade* [4], and others [20]. While they all include some form of texture extraction and mapping, none treat the extraction of surface properties and re-illumination. Sato *et al.* [21] present a system to extract 3D geometry, texture, and surface reflectance, but it is limited to controlled environments.

With the development of an ever increasing number of computer augmented reality applications, it becomes important to handle the common illumination between real and synthetic scenes. While some previous papers [10, 5] present preliminary solutions, they all require significant user intervention and are limited in different ways in the lighting or geometric conditions they can treat. Recent developments to *Façade* [2] include surfaces property extraction, but rendering times of the *Radiance* [24] system used for image generation are far from interactive.

Nakamae *et al.* [18] developed a solution for merging virtual objects into background photographs, and estimated the sun location to simulate common illumination effects in outdoor environments. More recently Yu and Malik [27] proposed a solution to virtually modify the illumination with different virtual positions of the sun in outdoor scenes.

Loscos and Drettakis [16, 17] have developed an approach to remove shadows, thus enabling synthetic relighting. This technique attempts to remove shadows by computing the best possible approximation using a single image. Despite successful results for certain cases, certain visual artifacts remain in the shadow regions.

In our method, as mentioned in the introduction, we separate direct lighting, which can be easily computed for each pixel, from indirect, or global lighting. Since we will



Fig. 1. The 7 radiance images used for the example presented in this paper.

be interactively modifying the scene, we need to be able to update the global illumination rapidly. To do this, we have used some of the ideas developed by Shaw [22] and Drettakis and Sillion [6].

Removal of real objects from a reconstructed scene requires some form of hole-filling in the real images/textures containing the real objects being removed. Heeger and Bergen [13] have developed a method to synthesize texture images given a texture sample. They use a series of linear filters to analyse the sample and create a texture that matches the sample appearance. Their method is successful on “stochastic” textures (e.g., stucco) but fails on “deterministic” textures (e.g., bricks). El-Maraghi [7] has provided a public domain implementation of their algorithm.

Igehy and Pereira [14] integrate a composition step into the Heeger and Bergen algorithm in order to “erase” flaws (e.g., stains or undesired features) from images. They manually create a mask which indicates which part of the image is to be covered by the synthesized texture and which part keeps its original texture.

3 Overview of the Method

Our goal is to allow interactive synthetic relighting and remodeling of real environments including both removing real lights or objects, and adding virtual ones. To accomplish this, we need to build approximate geometric and reflectance models of the environment and quickly estimate the illumination in modified configurations. We also want our method to be tolerant of measurement and modeling errors in order to work on a broad class of environments. Our process consists of several preprocessing steps followed by an interactive relighting session.

We begin by taking two sets of photographs of the target environment. The first is taken from multiple viewpoints under normal lighting conditions and is used to build an approximate geometric model provided by our photomodeling system [20]. The second set is taken from the fixed viewpoint that will be used during the interactive editing session. These photos use controlled lighting that consists of a single known light source that is moved between photos. We typically use between 5 and 7 such photos (e.g., Fig. 1). This second set, which we will refer to as the *radiance images*, is used to estimate the reflectance on all the visible surfaces.

To recreate sharp shadows, the direct lighting is estimated on a per pixel basis from the fixed viewpoint using ray casting. For each pixel we store its corresponding 3D point and surface, its estimated local reflectance, and its visibility and form-factors to each light. This structure is illustrated in Fig. 2.

Each radiance image is used to estimate the reflectances at each pixel, but may be

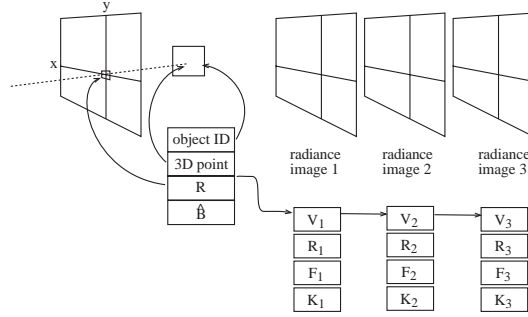


Fig. 2. A per pixel data structure is stored for the interactive view as well as for each radiance image. The visibility to each light V_i , the form-factor to each light F_i , the estimated reflectance at this pixel R_i , and the confidence level K_i of the pixel are stored for each radiance image i . The interactive view stores the merged reflectance R , the ambient term \hat{B} , the object's surface ID and the 3D point corresponding to each pixel.

unreliable in some regions such as shadows. We generate a more robust reflectance estimator by assigning a confidence for each estimate and combining them from the multiple images accordingly. If we remove real objects, we also estimate the reflectance in regions of the image that become visible. This is accomplished by adapting a texture-filling algorithm.

Once the geometric and reflectance models are extracted, they are used to initialise an hierarchical radiosity system that enables dynamic simulation of the indirect lighting in the environment.

After completing these preprocessing steps, we are ready to interactively model and relight our scene. When we modify the lighting or the geometry of the scene (either real, virtual or both), we efficiently update direct and indirect light. The regions of the image for which direct illumination must be recomputed are efficiently identified in screen space using polygon ID maps and the shaft data structures used for dynamic global illumination. These same structures also allow efficient recomputation of indirect light.

4 Preprocessing

The main goal of the preprocessing steps is to initialise the data structures that will be used during the interactive session. First surface reflectance at each pixel is estimated, and a pixel-based data structure for precomputed direct lighting quantities is initialised. Finally the hierarchical radiosity system is set up for rapid indirect lighting updates.

The process begins by building a geometric model of the environment using our photomodeling system [20]. The user specifies a set of corresponding points in the set of photographs taken from multiple viewpoints. The system uses these to solve for the camera parameters and 3D positions of the points. The user connects these points together into polygons to form a geometric model of the scene and can specify additional constraints to improve the model. All further processing uses the radiance images, with the light source positions measured by the user.

4.1 Pixel Data Structure

The radiance images are all taken from the fixed viewpoint that we will use in our interactive remodeling session. The physical light source we used is a simple garden light covered by white semi-transparent paper to achieve a more diffuse effect. Using a fixed viewpoint simplifies the capture of the real scene (since we need a small number of images); in addition working in image space allows more efficient data structures to be used for display, and generally simplifies the algorithms developed.

Much of the computation is done on a per pixel basis for this viewpoint using an augmented pixel data structure. At each pixel we store (see Fig. 2):

- The 3D point P which projects to the center of this pixel
- The polygon ID of the visible surface containing this point
- The form-factor F_i to each light source from this point
- The visibility V_i to each light source from this point
- The estimated surface reflectance R at this point

We create one such data structure for each radiance image plus an additional one for interactive use which also stores the indirect radiance \hat{B} estimated by the radiosity system for this point. The radiance images additionally store a confidence K_i (≤ 1) at each pixel which indicates how reliable we think its reflectance estimate is.

The polygon ID and 3D point P are obtained by using an item buffer [25] and z-buffer depth values. The form-factor F_i is computed using a standard point-to-polygon technique [1]. The visibility V_i is the fraction of the light source which is visible from point P and is estimated by ray casting from the point to the light source. The number of rays is varied adaptively from 4 to 64, with the higher number being used in regions of penumbra. Initially, confidence K_i is set equal to V_i , since we have less confidence in regions in shadow.

4.2 Reflectance Recovery Algorithm

If we assume that our surfaces are diffuse then there is a simple relation between the radiance L seen by a pixel in the camera, the reflectance R at point P , and the incident light on point P given by:

$$L = R \left(\sum_i F_i V_i E_i + \hat{B} \right) \quad (1)$$

where E_i is the emittance of light i , $F_i V_i E_i$ is the direct illumination due to light i and \hat{B} accounts for all indirect light. The emittance value is currently set arbitrarily, and an appropriate scaling factor applied to compensate during display.

If all the quantities in question were available and exact, we could solve exactly for the reflectance at each pixel using a radiance image i with its single light source via:

$$R_i = \frac{T^{-1}(C_i)}{F_i V_i E_i + \hat{B}} \quad (2)$$

where C_i is the pixel color recorded by the camera and $T(\cdot)$ is the response function of the camera. This function was unavailable for our camera² so we have used a simple scaling factor, though it could be accurately estimated using the method of Debevec and Malik [3].

²A Kodak DC260 digital camera.

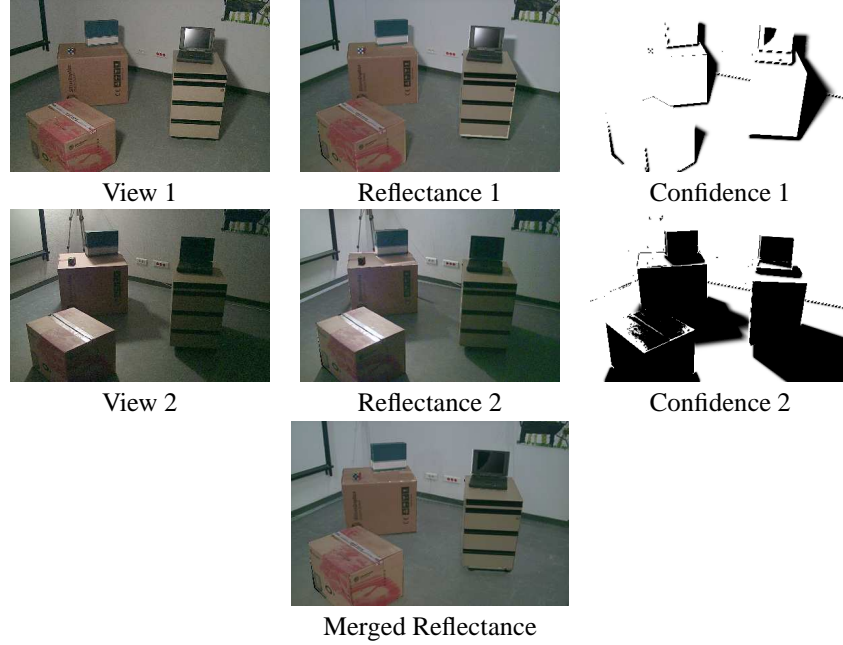


Fig. 3. Two of the seven radiance image views (left), the confidence images (right), and the resulting reflectance (center), extracted using Eq.(2). Dark values are for lower confidences. The merged reflectance is shown at the bottom.

As a first approximation to indirect lighting \hat{B} , we have used an ambient term equal to the average image color times a user specified average reflectance [10]. The resulting reflectance gives satisfactory results for our test cases, although more involved indirect lighting calculations may be necessary in other contexts when more accurate reflectance is needed. Some experiments were performed with an iterative approach to reflectance estimation using our radiosity solution, without much improvement in the reflectance estimate. Nonetheless, this is clearly a topic of future work.

Because of the many approximations in our system including the geometry, indirect light, and diffuse assumption, we know that our reflectance estimates will sometimes be quite inaccurate (e.g., in shadow regions where the indirect term dominates). We compensate for this by combining the reflectance estimates from multiple radiance images to form a much more robust reflectance estimator.

For each radiance image i , we also estimate our confidence K_i for each pixel reflectance estimate. The computation of K_i values is explained in next section. The merged pixel reflectance is formed by a weighted average of individual reflectance estimates:

$$R = \frac{\sum_{i=0}^n K_i \times R_i}{\sum K_i} \quad (3)$$

4.3 Filtering Confidence Values

As mentioned above, we initially set the confidence equal to the visibility V with respect to the light source, to reflect the fact that our reflectances are often inaccurate in shadow regions where indirect light dominates. However there are also other condi-

tions that can cause inaccurate reflectance estimates including geometric error, specular highlights, saturation in the camera, and even the movable light source being visible in some images. We use a series of filters to try to identify and reduce the confidence in such problem regions.

Near shadow boundaries visibility V depends heavily on the exact geometry configuration and thus may be unreliable due to inaccuracies in our reconstructed model. To reflect this, we first expand low confidence regions using a 5×5 minimum filter where the pixels confidence is replaced by the minimum confidence in its neighborhood. Abrupt changes in the confidence can also cause objectionable artifacts in the combined results, therefore we next apply a 5×5 smoothing filter.

Lastly, to detect other problem regions, we apply an outlier filter. For each pixel, we compute the median of its high confidence reflectance estimates (e.g., those with $K_i > 0.75$) from the individual radiance images. Estimates which differ by more than a user supplied threshold from this median are assumed to be outliers and have their confidence set to zero. This allows to automatically detect and discount problem regions such as specular highlights and the light source tripod which is visible in some radiance images. Afterwards another smoothing filter (3×3) is applied. Examples of resulting confidence images are shown in Fig. 3 for two views.

Once the confidences have been computed, we combine the reflectance estimates using Eq. (3). The result is more robust and contains fewer artifacts than any of the individual reflectance estimates from the radiance images as shown in Fig. 3.

4.4 Texture Filling for Real Object Removal

Removing a real object from the scene leaves a gap, or previously invisible region, for which we need reflectance estimates. We fill in this missing information using texture synthesis in a technique similar to Igehy and Pereira [14]. We use El-Maraghi's [7] implementation of Heeger and Bergen's [13] texture synthesis in our system.

To synthesize the textures needed, we extract a texture sample from the *reflectance image* from every polygon that now covers the region to fill. The extraction of the sample is currently done manually, but we are experimenting with automatic extraction procedures. This sample is fed to the synthesis algorithm which generates a texture of the same size as the region to fill. The generated texture is applied to the reflectance using a masking process, described in Section 5.3. The generated textures are stored for objects marked as "removable" accelerating the interactive remodeling operations.

It should be noted that texture generation is performed on the reflectance image and is thus not hindered by shadows or lighting variations during the object removal. The reprojection of the shadows with the new scene will generate a correct image of the scene without the real object.

4.5 Initialising the Hierarchical Radiosity System

To bootstrap the hierarchical radiosity system, the reflectance values recovered by Eq. (3) are reprojected onto the corresponding polygons, initialising the reflectance values. For the polygons invisible in the image used for the interactive session, we take a sample of the texture during the photomodeling session and get an average value using Eq. (2). For parts of polygons invisible from the fixed viewpoint, we use an average reflectance value computed from the visible parts.

With this approximation, a first radiosity solution is computed by our system, using an implementation of hierarchical radiosity with clustering [23]. The subdivision is set



Fig. 4. (a) The original view of the scene and (b) the corresponding radiosity mesh used to simulate indirect light and dynamic updates; note the coarse subdivision.

to a relatively coarse level since such a level is sufficient for computing indirect light, which varies slowly. An example mesh is shown in Fig. 4(b).

Recall that direct effects, including direct shadows, are treated separately for display. Direct light is however computed by the radiosity system, but simply ignored for display. The subdivision is fixed at the beginning of the process to a minimum area threshold. Nonetheless, we maintain the hierarchical nature of the radiosity algorithm, since links are established at different levels of the hierarchy, using a “standard” BF refiner [12]. Thus we will only need to update links and the radiosity values when performing interactive modifications.

5 Interactive Modification of Scene Properties

Once the reflectance has been computed for each pixel and the radiosity system set up, we can perform interactive modification of scene properties. The modifications that our system permits are related to lighting and geometry. The former includes changing a real light or adding virtual lights; the latter includes adding and moving virtual objects and removing real objects.

The web page <http://www-imagis.imag.fr/Membres/Celine.Loscos/relight.html>, contains high-resolution images and online movie sequences of interactive sessions. All timing results reported below have been taken on a SGI R10000 195Mhz processor.

5.1 Modifying Illumination

When we modify a light source emittance, two operations need to be performed:

- For indirect illumination, we need to compute a new radiosity solution. Given that the subdivision and the link structure are fixed after the initial solution, updating indirect illumination simply requires a few successive sweeps of the hierarchy to “gather” and “push-pull” [12] radiosity and is very fast (less than .05 seconds in our test scenes, since their polygon count is low).
- For display, the direct lighting component is recomputed at each pixel. Indirect illumination is displayed using hardware smooth-shading of the elements of the hierarchical radiosity subdivision, which are then blended into the final image. This results in the addition of the indirect irradiance \hat{B} at each pixel.

In the pixel structure, we have stored the visibility and form-factor with respect to each light source. Thus the computation of the direct component is very rapid.

When displaying an image, we compute the following color at each pixel:

$$C = R \left(\sum_{s=0..n_s} F_s V_s E_s + \hat{B} \right) \quad (4)$$

for the n_s (real or virtual) light sources in the scene. Before inserting any virtual light source, the scene is lit only with its original light ($n_s = 0$). Shadows are *reprojected* due to the visibility term V_s , since they have been removed from the reflectance.

An example is shown in Fig. 5. The original photo is shown in (a), reprojected initial lighting conditions in (b), and we show the addition of a virtual light source in (c). The entire update for adding the virtual light takes 3.1 seconds broken down as follows: visibility 2.5 sec., shaft/radiosity operations 0.4 sec., indirect light blending and other 0.2 sec. Recall that in the case of the light source insertion, we are required to update *all* the pixels of the image. During dynamic updates, we cast a small number of rays to the light sources, resulting in aliased shadows. An additional “shadow clean-up” could be performed when the user stops modifying the scene, with a higher shadow sampling rate.

5.2 Modifying Scene Geometry

To allow interactivity when adding, removing or moving objects and lights, we maintain a shaft data structure [11], inspired from the work of Drettakis and Sillion [6]. Updating the entire table requires in the order of a few minutes for visibility values, especially when using many rays per light source; using the method described below reduces this time to fractions of a second.

A hierarchical shaft [11] data structure is constructed from the first radiosity solution, and corresponds to each light transfer link. When we add an object it is first attached to the root cluster of the scene; links are established to the light sources as appropriate, based on the refinement criterion, and visibility information is computed.

The hierarchy of shafts is used for two purposes: (a) to identify the pixels for which *direct* illumination has changed (i.e., the shadow regions of the new object); and (b) to identify the links for which visibility needs to be updated (i.e., all links whose shaft is cut by the new object), for both direct and indirect light transfers.

To achieve the above, we descend the hierarchy of shafts, finding those intersected by the new object. The hierarchical elements attached to the end of a shaft originating at a light source are marked as “changed”. While descending, the visibility of the modified links is updated. With all necessary links updated, we recompute a radiosity solution with only gather and push-pull steps.

The pixel data structure is then updated and displayed. The bounding box of the initial and final position of the moving object are first projected onto the image-plane, limiting the region of the screen directly affected by the motion. For this region a new item buffer is performed, and the pixels under the previous object position are found as well as those under the new position, since the polygon IDs will have changed. For these pixels, reflectances are kept to the original values for the “uncovered” pixels and updated to that of the virtual object for the newly covered pixels. New form-factors and visibility values are then computed for all the pixels changed in the modified region.

For the pixels associated with patches tagged as “changed”, visibility with respect to the sources is recomputed. These are not as localized as the directly affected pixels, but their number is often small.

The entire pixel table is then traversed to update the indirect illumination value at

each pixel, based on the new global illumination calculation; again, this is performed with hardware rendering of the hierarchical radiosity patches.

When inserting a new light source, the form-factor and visibility with respect to the source need to be computed for every pixel.

When removing an object, we perform a similar process. We delete every link and all corresponding shaft structures of the removed object.

When moving an object, the process is equivalent, but we do not have to delete the links. We just have to update the information (form-factors and visibilities). Shafts due to the moving object are deleted and reconstructed with its new position.

In Fig. 5(d) we show the insertion of a virtual object in a scene lit with the original light and an added virtual light source. The insertion requires 1 sec., of which visibility accounts for .5 sec., shafts .1 sec. and the rest .4 sec. When moving the virtual object, we achieve update rates of about 1 sec. per frame, with a similar breakdown to that of the object insertion (Fig. 5(e)).

5.3 Removing Real Objects

When the user chooses to remove an object, she indicates the object to the system. Similarly to virtual objects, we know *exactly* which region of the screen will have to be filled, since the correspondences between polygons and pixels are known through the polygon IDs stored in the pixel data structures. We automatically create two masks corresponding to this region: a weight mask and a texture mask [14]. At first, each contains “1” over the region to fill and “0” elsewhere. We extend the weight mask a few pixels to compensate for inaccuracies in the removed object geometry (to avoid leaving any color from the removed object in the image).

The object is then removed from the scene and a new item buffer is performed to update the polygon IDs. The polygon IDs present in the region to be filled indicate from which polygons we have to extract textures. The texture mask is filled with these new IDs and the weight mask is blurred around its “0/1” borders. This allows the composition of the synthesized texture with the texture from the image: when the mask is 0, the color of the pixel will be the color in the reflectance image, when the mask is 1 the color will be taken from the synthesized texture and a fractional weight will allow a smooth transition from the synthesized texture to the original image (e.g., the original colors present in the image).

The reflectance is then updated for the pixels affected, as well as the visibility and form-factors, as in the case of virtual object motion/removal. Results of object removal are shown in Fig. 6.

A second example of real object removal is shown in Fig. 7. In the context of an interior redesign, we may want to remove doors for example, which is hard to do in the real world. This is shown Fig. 7(b). Note that due to approximate reflectance estimation, the texture generation results in slightly visible discontinuities. A virtual object has been added in (c) and a different lighting configuration created in (d).

6 Conclusion

We have presented a new approach to synthetic relighting and remodeling of real environments. Our approach is based on a preprocessing step to recover approximate reflectance properties from a sequence of radiance images. Radiance images are taken from a fixed viewpoint with varying illumination (i.e., different positions of the same light source), using a simplified reconstructed model of the scene. Using the information in the images and the 3D reconstructed model, we create reflectance images for

each light position by estimating direct illumination and light source visibility as well as indirect light. The reflectance images are merged by a weighted average based on the confidence level we have in the reflectance at each pixel in each radiance image. In our case, this is based on visibility (points in shadow have low confidence); a filtering step is applied to compensate for errors in geometric reconstruction and illumination computation.

After the reconstruction has been performed we can interactively modify scene properties. This is achieved by efficiently identifying regions of the screen which need updating, and performing a pixel-by-pixel update for direct light. Indirect lighting is treated separately with an efficient hierarchical radiosity structure, optimized for dynamic updates.

In our implementation we can virtually modify real light intensity, insert and move virtual objects, and even remove real objects *interactively*. Despite inevitable artifacts, the quality of the images is sufficient for the purposes of interactive lighting design and limited remodeling.

Independently to our work, Yu *et al.*[26] have recently developed more robust techniques for reflectance estimation, including specular effects in particular. These are based on capturing images of the entire scene, and computing radiosity to estimate the reflectance using clever iterative methods and high-dynamic range images. We believe that our approach can benefit from such improved reflectance estimation (for example to remove the artifacts in texture generation in Fig. 7) as well as for the reflectance of objects which are not visible in the radiance image. On the other hand, we believe that both our interactive approach, especially for global illumination, as well as our confidence maps could be useful for such approaches.

In future work, using the high dynamic range radiance images of Debevec and Malik [3] will allow us to achieve more accurate reflectance extraction. Once we have more confidence in the original radiance most of the errors in the reflectance estimation will be due to indirect light. The hierarchical radiosity framework has the added advantage that it can be used to bound indirect illumination errors and thus should allow us to achieve better results.

We also need to investigate ways to allow motion of the viewpoint, which is currently an important limitation of our approach. Also, the texture generation approaches we have used are limited to stochastic textures. With some user intervention, it may be possible to achieve satisfactory results with deterministic textures also.

From a more practical point of view, we can add the synthetic motion of real objects simply into our system. A view-independent texture of the real object is required, which can be provided by our photomodeling system, as well as a modified rendering routine. As was discussed in the results, the largest expense in the updates is the calculation of visibility for direct lighting. These calculations can be easily parallelized, and we hope to achieve good speedups in a parallel version, enhancing interactivity.

References

1. D. R. Baum, H. E. Rushmeier, and J. M. Winget. Improving radiosity solutions through the use of analytically determined form-factors. In *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23, pages 325–334, July 1989.
2. P.E. Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *SIGGRAPH '98 Conference Proceedings*, Annual Conference Series, pages 189–198, July 1998.
3. P.E. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. In *SIGGRAPH '97 Conference Proceedings*, Annual Conference Series, pages 369–378, Au-

- gust 1997.
4. P.E. Debevec, C.J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *SIGGRAPH '96 Conference Proceedings*, Annual Conference Series, pages 11–20, July 1996.
5. G. Drettakis, L. Robert, and S. Bougnoux. Interactive common illumination for computer augmented reality. In *Rendering Techniques '97 (8th Eurographics Workshop on Rendering)*, pages 45–56. Springer-Verlag, June 1997.
6. G. Drettakis and F. Sillion. Interactive update of global illumination using a line-space hierarchy. In *SIGGRAPH '97 Conference Proceedings*, Annual Conference Series, pages 57–64, August 1997.
7. T. El-Maraghi. An implementation of Heeger and Bergen's texture analysis/synthesis algorithm with source code. <http://www.cs.toronto.edu/~tem/2522/texture.html>.
8. O. Faugeras. *Three-Dimensional Computer Vision — A Geometric Viewpoint*. MIT Press, 1993.
9. O. Faugeras, S. Laveau, L. Robert, G. Csurka, and C. Zeller. 3D reconstruction of urban scenes from sequences of images. Tech. report 2572, INRIA Sophia-Antipolis, May 1995.
10. A. Fournier, A.S. Gunawan, and C. Romanzin. Common illumination between real and computer generated scenes. In *Proc. of Graphics Interface '93*, pages 254–262, May 1993.
11. E. A. Haines and J. R. Wallace. Shaft Culling for Efficient Ray-Traced Radiosity. In *Photo-realistic Rendering in Computer Graphics (Proceedings of the 2nd Eurographics Workshop on Rendering)*, New York, NY, 1994. Springer-Verlag.
12. P. Hanrahan, D. Salzman, and L. Aupperle. A rapid hierarchical radiosity algorithm. In *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 197–206, July 1991.
13. D.J. Heeger and J.R. Bergen. Pyramid-Based texture analysis/synthesis. In *SIGGRAPH '95 Conference Proceedings*, Annual Conference Series, pages 229–238, August 1995.
14. H. Igehy and L. Pereira. Image replacement through texture synthesis. In *Proceedings of the 1997 IEEE International Conference on Image Processing*, 1997.
15. F. Leymarie, A. de la Fortelle, J. Koenderink, A. Kappers, M. Stavridi, B. van Ginneken, S. Muller, S. Krake, O. Faugeras, L. Robert, C. Gauclin, S. Laveau, and C. Zeller. Realise: Reconstruction of reality from image sequences. In *International Conference on Image Processing*, volume 3, pages 651–654, Lausanne (Switzerland), 1996. IEEE Signal Proc. Soc.
16. C. Loscos and G. Drettakis. Interactive relighting of real scenes. Tech. report 0225, INRIA Rhône-Alpes, November 1998.
17. C. Loscos, G. Drettakis, and L. Robert. Interactive modification of real and virtual lights for augmented reality. In *SIGGRAPH '98 Technical Sketch (Visual Proceedings)*, July 1998.
18. E. Nakamae, K. Harada, T. Ishizaki, and T. Nishita. A montage method: The overlaying of the computer generated images onto a background photograph. In *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20, pages 207–214, August 1986.
19. Photomodeler. <http://www.photomodeler.com>.
20. P. Poulin, M. Ouimet, and M.-C. Frasson. Interactively modeling with photogrammetry. In *Rendering Techniques '98 (9th Eurographics Workshop on Rendering)*, pages 93–104. Springer-Verlag, June 1998.
21. Y. Sato, M.D. Wheeler, and K. Ikeuchi. Object shape and reflectance modeling from observation. In *SIGGRAPH '97 Conference Proceedings*, Annual Conference Series, pages 379–387, August 1997.
22. E. Shaw. Hierarchical radiosity for dynamic environments. *Computer Graphics Forum*, 16(2):107–118, 1997.
23. F. X. 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, September 1995.
24. G.J. Ward. The RADIANCE lighting simulation and rendering system. In *Proceedings of SIGGRAPH '94*, Annual Conference Series, pages 459–472, July 1994.
25. H. Weghorst, G. Hooper, and D. P. Greenberg. Improved computational methods for ray tracing. *ACM Transactions on Graphics*, 3(1):52–69, January 1984.

26. Y. Yu, P.E. Debevec, J. Malik, and T. Hawkins. Inverse global illumination: Recovering reflectance models of real scenes from photographs. In *SIGGRAPH '99 (to appear)*, 1999.
27. Y. Yu and J. Malik. Recovering photometric properties of architectural scenes from photographs. In *Computer Graphics (ACM SIGGRAPH '98 Proceedings)*, July 1998.

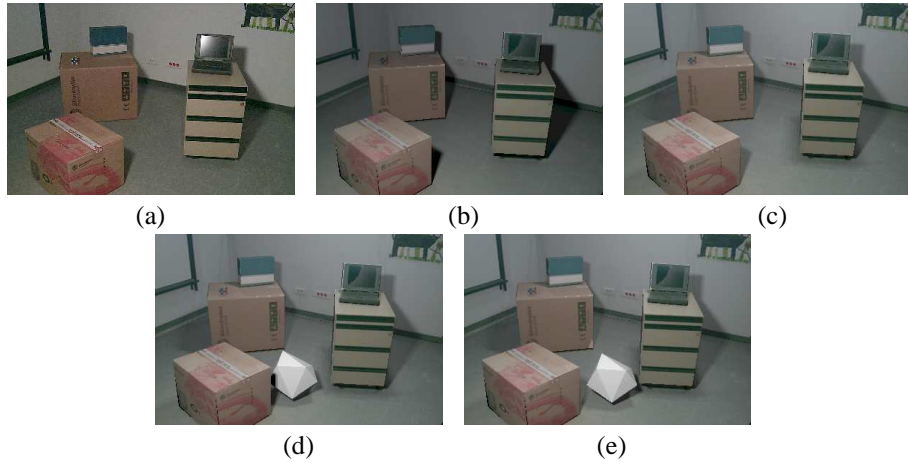


Fig. 5. (a) The original radiance image (photo). (b) Original reprojected lighting conditions, displayed using the recomputed direct and indirect components, (c) a virtual light has been inserted into the scene adding the light took 3.1 seconds (for 400x300 resolution). (d) A virtual object has been inserted into the scene with both lights on; adding the object required 1 sec. (e) Moving the virtual object requires 1.1 sec.



Fig. 6. Texture filling examples for real object removal. (a) Initial reflectance image (b) The laptop is removed. The laptop was removed entirely *synthetically* since no additional image was captured. (c) The original relit image. (d) The relit image after removal. Removal of the laptop took 0.7 sec., since generated textures are pre-computed for “removable” objects.

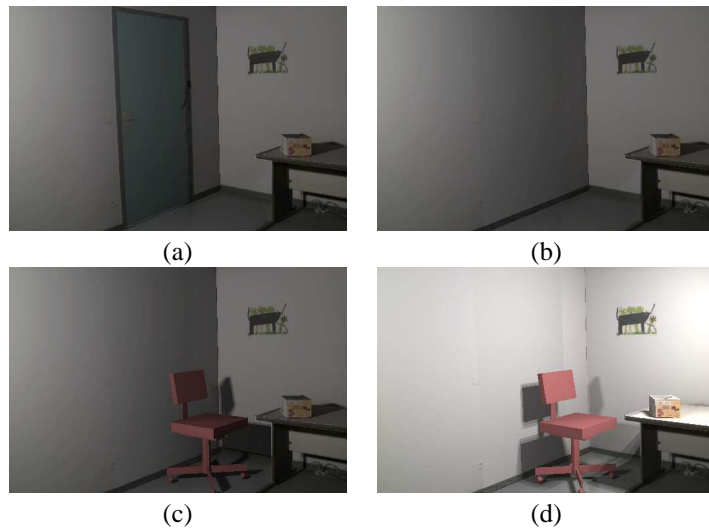


Fig. 7. A second real object removal example. (a) The original relit image, (b) the relit image after removal of the door, which took 2.9 sec., for a resolution of 512x341. (c) A virtual chair has been added to the scene, requiring 3.4 sec., and (d) a virtual light added (needing 6.6 sec.).

Conclusion et Perspectives

Nous avons développé trois thèmes principaux dans ce mémoire : les calculs haute précision passant par la définition et l'utilisation des structures de données de visibilité ; les calculs de l'éclairage pour des scènes de grande complexité (géométrie et photométrie) et enfin sur des algorithmes de rendu interactifs, pour le rendu à base d'image, de haute qualité et pour des scènes mixtes, réelles-virtuelles.

Nous allons tenter de résumer notre contribution dans chaque thème :

- En ce qui concerne les calculs de haute qualité pour la visibilité, dans un premier temps nous avons étendu les travaux, développés pendant ma thèse sur les maillages de discontinuité. Nous avons appliqué ces maillages dans le cadre plus général d'un algorithme *d'échantillonnage structuré*, qui essaie de concentrer les échantillons utilisés pour la représentation de la lumière là où ils sont utiles ; les maillages de discontinuité ont été aussi utilisés pour l'éclairage global et pour des mises à jour interactives pour l'éclairage direct.

Dans un deuxième temps, dans le cadre de la thèse de Frédo Durand, nous avons effectué une étude approfondie des propriétés de la visibilité globale, c'est-à-dire des relations de visibilité de chaque objet par rapport à chaque autre. Cette étude a donné lieu à des avancées importantes, notamment dans la définition d'une structure générale le *Complexe de Visibilité* ainsi que sa simplification pratique le *Squelette de Visibilité* qui a été ensuite utilisé pour une application d'éclairage global.

- Sur les algorithmes d'éclairage pour des scènes très complexes nous avons essayé d'éliminer le coût de la simulation, en nous attaquant au cœur du problème, c'est-à-dire le raffinement et la visibilité. Nous avons également proposé une solution pour réduire le coût mémoire de cette méthode. Pour les scènes non-diffuses, nous avons développé des solutions basées sur le stockage de la radiance sortante par des fonctions directionnelles discrètes et hiérarchiques.
- Pour le rendu interactif, nous avons développé un algorithme de rendu à base d'images ; deux algorithmes de rendu interactif haute qualité ont été également introduits, en utilisant la radiosité et le lancer de rayons. Nous avons enfin présenté deux nouvelles méthodes pour la réalité augmentée en tenant compte de l'éclairage commun entre objets réels et virtuels.

Ces travaux ont soulevé beaucoup de questions sur les méthodes choisies pour résoudre les problèmes de visibilité, de l'éclairage, du rendu interactif et de la réalité augmentée.

Pour la visibilité analytique il est clair que les limitations principales des méthodes développées sont la robustesse des calculs et la consommation mémoire qui est trop élevée. Les deux facteurs sont critiques pour le passage à l'échelle : nous ne pouvons pas traiter de « vraies » scènes d'une taille habituelle (centaines de milliers ou millions d'objets) sans que ces deux questions soient résolues.

Pour les méthodes de radiosité hiérarchique, il est clair que la réduction de mémoire et le traitement de la visibilité sont des questions très importantes ; les problèmes qui demeurent sont surtout liés aux critères

de raffinement. La gestion de scènes non-diffuses est également très importante, mais les contraintes de mémoire des méthodes développées ici semblent empêcher pour cette voie le passage à l'échelle.

Les travaux sur les méthodes de rendu interactif et la réalité augmentée sont de premiers pas dans des domaines qui sont en train de démarrer. Le problème de la lenteur des algorithmes d'éclairage global par tracer de rayons demeure, ainsi que les problèmes d'images haute résolution pour la solution interactive du « render cache ».

5.1 Perspectives

Le résumé présenté ci-dessus montre les voies que nous comptons explorer dans l'avenir.

Nous nous intéressons au développement des solutions robustes et hiérarchique pour la visibilité analytique. Nous espérons ainsi pouvoir calculer d'une façon hiérarchique et paresseuse le squelette de visibilité pour des scènes de taille réaliste. Une voie prometteuse sera sans doute une approche hiérarchique qui consistera à calculer un squelette localement pour des groupes ou clusters d'objets et ensuite calculer un squelette entre les groupes. Les problèmes qui sont posés sont nombreux, car il s'agit de définir la visibilité approximative. Nous pensons que ca sera sans doute une quantité fortement dépendante de l'application.

Pour l'éclairage nous nous orientons vers les solutions générales, qui pourront traiter des scènes non-diffuses. Nous sommes en train d'examiner deux voies possibles : la première utilisera la radiosit  hi rarchique avec clustering pour la partie diffuse et le tracer de particules pour le sp culaire, et la deuxi me sera une nouvelle approche stochastique, qui peut  tre vue comme une suite de l'algorithme de « Metropolis ».

Nous espérons que la premi re m thode permettra  galement le contr le de la qualit , d'une fa on similaire   la m thode de la radiosit , c'est- -dire par des bornes d'erreur qui peuvent  tre sp cifi es par l'utilisateur. La deuxi me m thode sera sans doute  troitement li e   l'approche du « render cache » dans le but d'obtenir un syst me d' clairage global g n ral interactif.

La r alit  augment e est un domaine en pleine expansion, et les travaux d' clairage commun ne commencent gu re    tre utilis s. Dans un premier temps nous espérons surmonter les restrictions   un seul point de vue et   une r flectance diffuse,  ventuellement en utilisant des m thodes semblables   celles de l' quipe de Berkeley [YDMH99]. Nous espérons  galement appliquer ces m thodes aux sc nes de l'ext rieur,  ventuellement dans le contexte d'une application arch ologique, en collaboration avec la Fondation du Monde Hell nique (www.fhw.gr).

Bibliographie

- [CCWG88] Michael F. Cohen, Shenchang Eric Chen, John R. Wallace, and Donald P. Greenberg. A progressive refinement approach to fast radiosity image generation. In John Dill, editor, *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22, pages 75–84, August 1988.
- [CDP95] Frédéric Cazals, George Drettakis, and Claude Puech. Filtering, clustering and hierarchy construction : a new solution for ray tracing very complex environments. *Soumis à Eurographics '95*, 1995.
- [CGIB86] Michael F. Cohen, Donald P. Greenberg, David S. Immel, and Philip J. Brock. An efficient radiosity approach for realistic image synthesis. *IEEE Computer Graphics and Applications*, 6(3) :25–35, March 1986.
- [CLSS97] 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, January 1997.
- [DDP96] Frédo Durand, George Drettakis, and Claude Puech. The 3d visibility complex, a new approach to the problems of accurate visibility. In X. Pueyo and P. Schröder, editors, *Rendering Techniques '96*, pages 245–257. Springer Verlag, June 1996. Proc. 7th EG Workshop on Rendering in Porto.
- [DDP97a] Frédo Durand, George Drettakis, and Claude Puech. The 3d visibility complex : a unified data-structure for global visibility of scenes of polygons and smooth objects. In *9th Canadian Conference on Computational Geometry*, Kingston, Canada, August 1997.
- [DDP97b] Frédo Durand, George Drettakis, and Claude Puech. The Visibility Skeleton : A Powerful and Efficient Multi-Purpose Global Visibility Tool. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings (Los Angeles, CA)*, Annual Conference Series. ACM SIGGRAPH, August 1997.
- [DDP99] F. Durand, G. Drettakis, and C. Puech. Fast and accurate hierarchical radiosity using global visibility. *ACM Transactions on Graphics*, march 1999.
- [Deb98] P.E. Debevec. Rendering synthetic objects into real scenes : Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *SIGGRAPH '98 Conference Proceedings*, Annual Conference Series, pages 189–198, July 1998.
- [DF93] George Drettakis and Eugene L. Fiume. Accurate and consistent reconstruction of illumination functions using structured sampling. *Computer Graphics Forum (Eurographics '93)*, 13(3) :273–284, September 1993.
- [DF94] George Drettakis and Eugene Fiume. A fast shadow algorithm for area light sources using backprojection. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 223–230. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.
- [DF96] George Drettakis and Eugene Fiume. Structured penumbral irradiance computation. *IEEE Transactions on Visualization and Computer Graphics*, 2(4) :299–313, December 1996.

- [DRB97] G. Drettakis, L. Robert, and S. Bugnoux. Interactive common illumination for computer augmented reality. In Julie Dorsey and Philipp Slussalek, editors, *Rendering Techniques '97*, pages 45–56, 8th Eurographics Workshop on Rendering, St. Etienne, June 1997.
- [Dre94a] George Drettakis. Simplifying the representation of radiance from multiple emitters. In Sakas et al., editor, *Photorealistic Rendering Techniques*. Springer Verlag, Darmstadt, June 1994. Proc. 5th EG Workshop on Rendering (Darmstadt, D, June 1994).
- [Dre94b] George Drettakis. *Structured Sampling and Reconstruction of Illumination for Image Synthesis*. Ph.d. thesis, Department of Computer Science, University of Toronto, 1994.
- [DS96] George Drettakis and Francois Sillion. Accurate visibility and meshing calculations for hierarchical radiosity. In X. Pueyo and P. Schröder, editors, *Rendering Techniques '96*, pages 269–279. Springer Verlag, June 1996. Proc. 7th EG Workshop on Rendering in Porto.
- [DS97] George Drettakis and Francois Sillion. Interactive update of global illumination using a line space hierarchy. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings (Los Angeles, CA)*, Annual Conference Series. ACM SIGGRAPH, August 1997.
- [DSSD97] K. Daubert, H. Shirmacher, F. Sillion, and G. Drettakis. Hierarchical lighting simulation for outdoor scenes. In Julie Dorsey and Philipp Slussalek, editors, *Rendering Techniques '97*, pages 229–238, 8th Eurographics Workshop on Rendering, St. Etienne, June 1997.
- [FGR93] A. Fournier, A.S. Gunawan, and C. Romanzin. Common illumination between real and computer generated scenes. In *Proc. of Graphics Interface '93*, pages 254–262, May 1993.
- [FRL⁺97] Olivier Faugeras, Luc Robert, Stphane Laveau, Gabriella Csurka, Cyril Zeller, Cyrille Gaucelin, and Imed Zoghlami. 3-d reconstruction of urban scenes from image sequences. *Computer vision graphics and Image processing*, 1997. To appear.
- [GD99] X. Granier and G. Drettakis. Controlling memory consumption of hierarchical radiosity with clustering. In James Stewart and Scott Mackenzie, editors, *Graphics Interface '99*, jun 1999.
- [GTGB84] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modelling the interaction of light between diffuse surfaces. In *Computer Graphics (SIGGRAPH '84 Proceedings)*, volume 18, pages 212–22, July 1984.
- [HDS99] Jean-Marc Hasenfratz, Cyrille Damez, Franois Sillion, and George Drettakis. A practical analysis of clustering strategies for hierarchical radiosity. In P. Brunet and R. Scopigno, editors, *Computer Graphics Forum (Eurographics '99 Conference Proceedings)*, volume 18 :3. Eurographics, September 1999.
- [Hec92] Paul Heckbert. Discontinuity meshing for radiosity. *Third Eurographics Workshop on Rendering*, pages 203–226, May 1992.
- [HSA91] Pat Hanrahan, David Salzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 197–206, July 1991.
- [HSD94] Nicolas Holzschuch, François Sillion, and George Drettakis. An efficient progressive refinement strategy for hierarchical radiosity. In *Fifth Eurographics Workshop on Rendering*, Darmstadt, D, June 1994.
- [LD97] Céline Loscos and George Drettakis. Interactive high-quality soft shadows in scenes with moving objects. In D. Fellner and S. Kalos, editors, *Computer Graphics Forum (Proc. of Eurographics '96)*, volume 17, Budapest, Hungary, September 1997.
- [LFD⁺99] C. Loscos, M.-C. Frasson, G. Drettakis, B. Walter, X. Granier, and P. Poulin. Interactive virtual relighting and remodeling of real scenes. In G. Ward-Larson and D. Lischinski, editors, *Rendering Techniques '99 (10th Eurographics Workshop on Rendering)*. Springer-Verlag, June 1999.
- [POF98] P. Poulin, M. Ouimet, and M.-C. Frasson. Interactively modeling with photogrammetry. In G. Drettakis and N. Max, editors, *Rendering Techniques '98 (9th Eurographics Workshop on Rendering)*, pages 93–104. Springer-Verlag, June 1998.

- [PPD98] Eric Paquette, Pierre Poulin, and George Drettakis. A light hierarchy for fast rendering of scenes with many lights. In N. Göbel and F. Nunes Ferreira (guest editor), editors, *Computer Graphics Forum (Eurographics '98 Conference Proceedings)*, pages 63–74. Eurographics, September 1998. held in Lisbon, Portugal, 02-04 September 1998.
- [SD95] François Sillion and George Drettakis. Feature-based control of visibility error : A multi-resolution clustering algorithm for global illumination. *soumis pour publication a SIGGRAPH 95*, 1995.
- [SDB97] Francois Sillion, George Drettakis, and Benoit Bodelet. Efficient impostor manipulation for real-time visualization of urban scenery. In D. Fellner and S. Kalos, editors, *Computer Graphics Forum (Proc. of Eurographics '96)*, volume 17, Budapest, Hungary, September 1997.
- [SDS95] François Sillion, George Drettakis, and Cyril Soler. A clustering algorithm for radiance calculation in general environments. In P.M. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95*, Wien, August 1995. Springer Verlag.
- [SDS96] Eric J. Stollnitz, Tony D. DeRose, and David H. Salesin. *Wavelets for Computer Graphics : Theory and Applications*. Morgann Kaufmann, San Francisco, CA, 1996.
- [Sil95] François Sillion. A unified hierarchical algorithm for global illumination with scattering volumes and object clusters. *IEEE Trans. on Vis. and Comp. Graphics*, 1(3), September 1995.
- [SSG⁺99] M. Stamminger, A. Scheel, X. Granier, F. Perez-Cazorla, G. Drettakis, and François X. Sillion. Efficient glossy global illumination with interactive viewing. In James Stewart and Scott Mackenzie, editors, *Graphics Interface '99*, jun 1999.
- [SSSS98] M. Stamminger, H. Schirmacher, P. Slusallek, and H-P. Seidel. Getting rid of links in hierarchical radiosity. *Comp. Graphics Forum (EUROGRAPHICS '98)*, 17(3), September 1998.
- [War94] Greg Ward. A contrast-based scalefactor for luminance display. In Paul Heckbert, editor, *Graphics Gems IV*, pages 415–421. Academic Press, Boston, 1994.
- [WDP99] Bruce Walter, George Drettakis, and Steve Parker. Interactive rendering using the render cache. In G. Ward-Larson and D. Lischinski, editors, *Rendering Techniques '99 (10th Eurographics Workshop on Rendering)*. Springer-Verlag, June 1999.
- [YDMH99] Y. Yu, P.E. Debevec, J. Malik, and T. Hawkins. Inverse global illumination : Recovering reflectance models of real scenes from photographs. In *SIGGRAPH '99 (to appear)*, 1999.