Visibilité tridimensionnelle : étude analytique et apllications

Frédo Durand

Université Grenoble I – Joseph Fourier

Visibilité tridimensionnelle : étude analytique et applications

Frédo DURAND

Thèse présentée pour l'obtention du titre de Docteur de l'Université Joseph Fourier
Spécialité Informatique
Arrêté ministériel du 5 juillet 1984 et du 30 mars 1992
Préparée au sein du laboratoire *i*MAGIS-GRAVIR/IMAG-INRIA. UMR CNRS C5527.
Soutenue le 12 juillet 1999

Composition du jury :

Dinesh	MANOCHA	Rapporteur
Michel	POCCHIOLA	Rapporteur
Seth	TELLER	Rapporteur
Jacques	VOIRON	Examinateur
Jean-Claude	PAUL	Examinateur
Claude	PUECH	Directeur de thèse
Georges	DRETTAKIS	Co-directeur de thèse

Remerciements

Je dédie cette thèse à mes parents et à l'ensemble des professeurs qui m'ont subi pendant ma longue scolarité.

Claude a été plus qu'un directeur de thèse pour moi. Il m'a fait profiter de sa grande expérience, tant du domaine que de la recherche en général, et plus encore. J'ai particulièrement apprécié les voyages que nous avons fait ensemble, Barcelone et la Californie. L'ambiance qu'il sait faire régner au sein de l'équipe est une richesse rare. Il a vraiment été un "père spirituel" durant ces cinq années passées ensemble.

George a su m'arracher à mes rêveries théoriques quand il le fallait. Il n'a pas hésité à mettre les mains dans le code, je n'en aurai jamais fait autant sans lui. Il a effectué un travail considérable pour transformer ce que je rédigeais en phrases anglaises. Ils ont tous deux eu le mérite de supporter ma tête de mule et mon caractère de jeune coq...

Il m'étais impossible d'imaginer mon jury de thèse sans la présence de Seth Teller tant son travail fait référence en visibilité et tant sa thèse a été un modèle pour moi. La gentillesse, la disponibilité et l'enthousiasme dont il a fait preuve à chacune de nos rencontres ont été une motivation puissante à mes travaux.

J'ai été très honoré d'avoir Dinesh Manocha pour rapporteur. Il représente pour moi un exemple de ce que la bonne recherche en géométrie peut être : des bases théoriques sans faille au service de problèmes très concrets, avec des validations pratiques impressionnantes.

Les travaux de Michel Pocchiola ont constitué l'inspiration initiale de cette thèse ; sa présence dans mon jury me semblait donc indispensable. Je le remercie pour nos discussions toujours amicales et constructives.

Jean-Claude Paul est un mélange rare d'excellence et d'humilité. L'intérêt qu'il a montré envers mes travaux m'a réellement touché.

Travailler avec Jacques Voiron pour *La Lettre de l'Imag* avait été un vrai plaisir, je n'en ai été que plus honoré qu'il accepte de présider le jury de cette thèse.

Que le vénérable professeur Sillion veuille bien excuser mon humour parfois douteux. Au delà des discutions scientifiques fructueuse, j'ai été ravi de son amitié. Que les chaises portugaises nous pardonnent.

Leo Guibas m'a invité à l'université de Stanford où j'ai passé un été à la fois agréable et riche en inspiration. C'est grâce à lui et à Mark de Berg que sont nées les idées du chapitre 5. Leo m'a impressionné par l'élégance de sa réflexion géométrique et par son ouverture. L'esprit critique et l'intuition géométrique de Mark m'ont grandement aidé à affiner mes idées.

Les rencontres avec Dani Lischinski ont toujours été illuminées par sa bonne humeur. Je le remercie pour tous ses conseils et ses encouragements, et demande une fois de plus pardon à lui et à Daniel Cohen-Or pour avoir oublié la moitié du pic-nic lors d'une balade autour de Grenoble

La thèse de masters de Sylvain Petitjean m'a appris tout ce que je sais des singularités. Les échanges de mails et la rencontre que nous avons eus me font souhaiter travailler un jour avec lui sur la visibilité ou sur un autre sujet.

Je remercie Pierre Poulin pour ses critiques efficaces et constructives et James Stewart pour toutes les discussions avec le sourire. Outre leur aide scientifique indéniable, loués soient Yorgos Chrysanthou pour sa gentillesse ; Xavier Pueyo pour l'excellence de son français, son invitation à Girone et les conversations non informaticiennes ; Eugene Fiume pour son soutien et Nina Amenta pour son extraordinaire enthousiasme.

Csol m'a fait profiter de sa profonde compréhension de la radiosité et des convolutions, de son humour discutable et des sourires amicaux de son Lampropeltis. Merci à Clairette d'avoir parfois réussi à nous empêcher de parler informatique à table.

Les discussions avec Nico me manquent, il a été un co-loc' et un collègue idéal. Vivement que j'aille squatter a New-York ! À Agata, je dois trop de choses pour en parler ici.

Jean-Dominique Gascuel est le gourou indispensable du labo, je n'aurais pu m'en sortir sans son aide technique. En plus son tarama est excellent.

Mathieu a apporté dans mon bureau sa curiosité, sa culture et son ardeur scientifique. Ses succès ne me surprennent pas du tout. Merci a Rara d'avoir été la maman de tous les thésards. Pardon pour les retouches d'image et pour le calimero. Stéphane et elle ont partagé mes premières publis dans un contexte pas toujours facile. Merci a Fred Cazals pour son dynamisme, le confit de canard de sa mère, les discussions boulot, les discussions pas boulot, etc. La thèse d'H^3 m'a énormément aidé a comprendre intimement la radiosité, sa cave et sa cuisine m'ont ravi. S'il existe des types bien sur terre, JC en fait partie, je l'admire pour beaucoup de chose, puisse-t-il m'apprendre l'humilité (mais ça parait pas gagné). Pardon Alexiiiis pour ta salle de bain, merci pour toutes les soirées, ainsi qu'à FF, deux incontournable des discussions politiques à iMAGIS. Je dédie à Ponpon les FFT qui ont servi à faire les convolutions du chapitre 5. Je te promets que je ne tournais pas au Boulard hors d'âge quand j'ai codé ça.

Bichont et Bichond sont toujours prêts à rendre service, une mention spéciale à Bichond que j'ai beaucoup sollicité pour boucler ce mémoire et à qui je pardonne ses goûts musicaux. Je demanderai sa canonisation. Merci à Jean-Marc et Joëlle pour la relecture et pour le reste. Bises à Gaspard. Jérémie tu viens quand tu veux me faire du Rougaï. Merci à Xadec pour l'aide sur Ville (l'ancien et le nouveau...).

Je n'oublie pas tous les autres imagiciens, l'ambiance dans ce labo est trop top cool géniale, comme on dit en français soutenu.

Je n'aurai jamais tenu physiquement et moralement sans les vins de *l'Echanson*, le fromage de la *Fromagerie des Alpages* et les kebabs du *Cesame*.

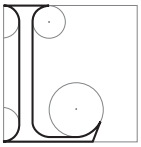
Je demande pardon à tous mes potes que j'ai trop délaissés ces quatre dernières années. Je dédie à Jojo tous les "trivial" de cette thèse et à Isa le mot "exotique" de la page 73 et les "mouais" que l'on peut parfois lire entre les lignes. Merci à Reno, Aude et Olivier pour m'avoir choyé quand je venais à Paris, pour les chipsters et Martini et pour bien d'autres chose. L'amitié et les coups de fil de Fouabulle, Sandrinette et Bruno m'ont aidé à ne pas péter trop les plombs. Merci à ma "grande sœur" Valérie, à Guitemie, Lola, Annick, Claire, Anne Roumier chérie, Élé, Pierrot, Anne-Claude, Chucky, Tom, Emmanuelle, Fred, Thierry et Pascal.

Enfin un immense pardon à Manue qui a dû souffrir de la concurrence de cette thèse, de mon manque de disponibilité et de mon stress.

SOMMAIRE

Introduction

> Des sciences démonstratives, la plus belle est celle à laquelle participe la science physique et la science géométrique, car de la science physique elle emprunte la perception, et de la science géométrique les démonstrations géométriques. Je n'ai rien trouvé où se réunissent ces deux arts de plus beau ni de plus parfait que la science des rayons...
>
> Ibn LUQA (mathématicien arabe du IXe siècle)



ES CALCULS de visibilité sont au cœur de bien des méthodes de synthèse d'images. Le calcul des objets visibles depuis un point de vue, les calculs d'ombres ou de pénombre comptent parmi les plus connus. De plus, des méthodes récentes de simulation de l'éclairage par exemple, réclament une information plus complète, puisqu'il faut déterminer pour toute paire de points s'ils sont mutuellement visibles. Dans bien des cas, la visibilité est le goulot d'étranglement, ce qui rend indispensable le développement de solutions efficaces.

La visibilité est malheureusement difficile à appréhender. Elle est par nature globale, un objet est visible de loin et des objets spatialement éloignés peuvent avoir des interactions très complexes. Les approches développées à ce jour ont cherché à répondre à des requêtes précises et limitées sans s'attacher à vraiment comprendre la cohérence et la globalité des propriétés de visibilité d'une scène tridimensionnelle. C'est tout d'abord ce manque d'un cadre permettant de décrire et de raisonner que cette thèse cherche à combler. Nous nous attacherons ensuite à exploiter cette meilleure compréhension pour développer des solutions nouvelles et efficaces.

Nous commençons cette introduction par un bref historique des problèmes de visibilité pour situer le contexte de cette thèse. Nous présentons les motivations qui ont guidé nos travaux, ainsi que les points qui expliquent la difficulté des questions abordées. Nous esquissons ensuite l'approche que nous avons suivie et les critères que nous nous sommes fixés, puis nous résumons nos contributions. Nous terminons cette introduction par une présentation de l'organisation du document.

Contexte

La synthèse d'images a connu une évolution fort différente de la peinture classique. En peinture, la projection perspective a longtemps été ignorée, alors que les questions de visibilité ne se sont en fait jamais posées : il est intuitif et immédiat pour un être humain de savoir quel objet est visible, quel objet en cache un autre, et

ce d'autant plus si la scène est devant ses yeux. L'aspect qualitatif (qu'est-ce qui est visible) est immédiat, alors que l'aspect quantitatif (en quel points se projettent les objets) n'a été résolu qu'à la Renaissance.

En synthèse d'images, les règles de la perspective ont dès le début été utilisées pour le calcul d'images d'objets tridimensionnels, tandis que la gestion des occultations s'est avérée bien plus problématique. L'intuition qualitative est difficile à traduire en algorithme. Les premières images étaient affichées en *fil de fer* : toutes les arêtes d'un polyèdre étaient tracées, qu'elles fussent cachées par d'autres objets ou non [1]. Les années 60 et 70 ont vu le développement d'algorithmes d'élimination des parties cachées, appelés alors algorithmes de *visibilité*. Cette question semble en pratique aujourd'hui résolue. Le *z-buffer* (tampon de profondeur), algorithme qui consiste à comparer la profondeur des objets pour chaque point de l'image s'est largement imposé. Des implémentations efficaces dans des cartes spécialisées permettent au moindre ordinateur d'afficher des milliers de polygones interactivement. Nous aurons néanmoins à revenir sur la question.

Les calculs d'ombres ont constitué le défi suivant. Le cas le plus simple est celui des ombres "dures" causées par une source ponctuelle. Il peut être vu comme le calcul des parties visibles depuis la source. La plupart des algorithmes ont été développés dans les années 70, mais aujourd'hui le problème du calcul des ombres dures ne semble pas pour autant définitivement résolu. Des problèmes de robustesse ou d'aliassage (effets de marches d'escalier) demeurent.

Les sources étendues ont commencé à être étudiées dans les années 80. Contrairement aux sources ponctuelles, un point de l'espace peut voir le tout, une partie, ou aucun point de la source lumineuse. Cela définit les zones éclairées, de pénombre et d'ombre. Le calcul de ces ombres douces a souvent été réduit à un échantillonnage par des sources ponctuelles ou à un filtrage d'ombres dures. Le calcul des limites d'ombre et de pénombre nécessite la prise en compte d'interactions complexes entre les bloqueurs. Aucune méthode n'a pour l'heure réussi à s'imposer et le calcul des ombres douces n'est pas encore véritablement sorti des laboratoires de recherche.

Les années 80 ont connu l'explosion de l'algorithme du lancer de rayon qui ramène tout problème de visibilité à une requête atomique : l'intersection entre un rayon lumineux et la scène. La méthode offre une souplesse inégalée grâce à sa simplicité, au prix d'un coût souvent élevé.

A partir du milieu des années 80 se sont développées les méthodes d'*éclairage global*. Elles visent à simuler toutes les interactions lumineuses dans une scène, en particulier les interreflexions. En effet, lorsque la lumière arrive sur un objet, elle est en général réfléchie et l'objet se comporte alors comme une source produisant de l'éclairage indirect. Calculer les effets d'ombrage revient alors à considérer tous les objets comme des sources de lumière. Cela nécessite de connaître depuis chaque point la visibilité de tous les autres points de la scène !

Les années 90 ont de plus connu une explosion de la taille des scènes à traiter. Les bases de données CAO sont énormes, le dernier Boeing, le B777, comporte par exemple des centaines de millions de polygones. On souhaite se promener dans des univers virtuels immenses comme des bâtiments entiers ou même des villes. Malgré la puissance croissante des ordinateurs et des cartes graphiques, l'affichage brutal de toute la scène est impossible en temps-réel (la dernière station de travail graphique haut de gamme Silicon Graphics, l'Onyx2 avec carte Infinite Reality, ne peut afficher à 30 images par seconde que 100 000 polygones).

Les calculs de visibilité n'apparaissent pas seulement en graphique. La vision par ordinateur a souvent été présentée comme le problème inverse de la synthèse d'images. Une façon de reconnaître un objet dans une image consiste à considérer toutes le vues possibles de cet objet et donc de résoudre un problème de visibilité pour chaque point de vue potentiel. Le choix du placement d'une caméra pour surveiller l'accomplissement d'une tâche doit éviter les occultations par les objets de l'environnement de travail. L'inspection de son environnement par un robot impose qu'au court de sont trajet toutes les régions soient visibles à un moment ou à un autre.

On le voit à la lecture de ce rapide tour d'horizon, les problèmes de visibilité ont évolué de questions de nature ponctuelle –vue depuis un point, ombres dures – à des problèmes de nature plus globales – visibilité d'une source étendue, visibilité mutuelle de toutes les paires de points de la scène. Cela s'est de plus accompagné d'un accroissement vertigineux de la taille des scènes à traiter.

[1] Cette représentation, avec l'inévitable affichage en vert sur fond noir, est restée dans bien des films le gage d'un modèle 3D et d'une activité informatique de pointe intense et sérieuse !

Motivations

Les motivations de cette thèse sont de deux natures complémentaires, pratiques et théoriques. La visibilité constitue un problème pratique au cœur de beaucoup de méthodes. Elle représente souvent le facteur limitant dont le coût restreint les performances. La visibilité présente parallèlement des problèmes géométriques fascinants dont la compréhension n'est qu'embryonnaire. Nous pensons néanmoins que la pléthore de contributions sur ce sujet rend nécessaire un tour d'horizon synthétique.

Des besoins pratiques

Afin de motiver nos travaux, nous nous sommes plus particulièrement intéressés à deux applications. Elle ont cependant été des moteurs et non un carcan. Nos travaux ont été validés sur ces applications, mais ils n'en restent pas moins une étude générale de la visibilité, qui a des implications dans d'autres domaines comme la vision ou la robotique.

La simulation de l'éclairage est un défi passionnant pour les calculs de visibilité. La radiosité [SP94, CW93b] est une méthode par éléments finis qui simule les interréflexions lumineuses dans une scène. La fonction d'éclairement est définis sur un maillage qui subdivise les polygones de la scène en sous-polygones.

Les interaction lumineuses entre toutes les paires de sous-polygones doivent être simulées, ce qui implique de calculer de manière quantitative leur visibilité mutuelle. La quantité de calculs de visibilité est de ce fait énorme et constitue le facteur limitant de la méthode [HSD94].

La précision de ces calculs est importante car ils sont utilisés pour déterminer les valeurs d'éclairement des points de la scène. Les erreurs entraînent des artefacts auxquels l'œil humain est très sensible. Les méthodes actuelles ont recours à des approximations par échantillonnage qui sont imprécises et fort coûteuses.

Les maillages réguliers habituels sont la cause d'effets d'aliassage (marche d'escalier) dans la pénombre. Pour obtenir des ombres de haute qualité il faut subdiviser les polygones le long des limites d'ombre et de pénombre. La tâche est déjà ardue lorsque l'on considère une source primaire puisque de multiples bloqueurs peuvent avoir des interactions complexes. De ce fait, les méthodes proposées à l'heure actuelle sont compliquées et sujettes à des problèmes de robustesse. Elles sont donc peu utilisées en pratique. L'éclairage indirect élève encore la complexité du problème puisque tout objet est une source secondaire et peut projeter des ombres. Cela explique qu'aucune méthode ne permet d'effectuer des simulations en traitant ces limites d'ombre secondaires.

Enfin, les avancées récentes [HSA91] permettent, grâce à une approche hiérarchique, de concentrer les calculs sur les interaction énergétiques "importantes". Le critère qui guide le raffinement des calculs est fondamental pour l'efficacité de la méthode, et il est naturel de penser que la visibilité doit être prise en compte. Malheureusement, le réglage fastidieux et imprévisible de ces seuils explique que les approches hiérarchiques n'aient pas réussi à s'imposer dans l'industrie.

Pour résumer, la simulation de l'éclairage nécessite des calculs intensifs et précis de visibilité entre toutes les paires d'objet d'une scène, ainsi que la détermination des limites d'ombre et de pénombre, potentiellement par rapport à tous les objets. Les problèmes de visibilité y sont globaux et très complexes.

Nous nous sommes également intéressés à l'affichage de scènes très complexes. Dans bien des cas, seule une faible partie est effectivement visible. Prenons l'exemple d'une balade dans une ville. Les façades proches cachent le plus souvent la majorité de la scène, qu'il est inutile d'essayer d'afficher. Les techniques d'*occlusion culling* (élimination d'occultation) consistent à rapidement éliminer des groupes d'objets invisibles avant de les envoyer à la carte graphique qui utilise ensuite un z-buffer pour calculer l'image finale.

Les calculs y sont prudents : on ne déclare pas invisible un objet visible, mais l'inverse peut se produire sans poser de problème puisqu'un z-buffer est ensuite utilisé pour calculer l'image finale. Un excès de prudence ralentit l'affichage (car trop d'objets sont envoyés à la carte graphique) mais l'image calculée est juste. De nombreuses méthodes ont été proposées pour effectuer ce calcul pour chaque image et des implémentations dans les cartes graphiques commencent à apparaître.

Cependant, dans le cadre d'une transmission *via* réseau ou si la scène est trop grande pour être entièrement chargée en mémoire, on souhaite se limiter aux objets visibles. La lenteur des transmissions ou des accès-disque rend toutefois impossible d'estimer et de charger les objets visibles à chaque image. Une information

plus globale sur le voisinage du point de vue est nécessaire pour permettre une approche plus prédictive qui anticipe le chargement. C'est pourquoi des calculs d'occultation depuis un volume de l'espace sont nécessaires.

Les travaux antérieurs sur ce sujet sont malheureusement restreints à des cas particuliers comme les intérieurs de bâtiment ou ne traitent que les occultations dues à un seul objet convexe à la fois. Les occultations dues à la conjonction de plusieurs objets sont pourtant cruciales pour une efficacité accrue. La taille des scènes à traiter interdit les calculs géométriques complexes pour chaque objet. Les contraintes sont différentes de celles de l'éclairage global : l'efficacité et la capacité de traiter des scènes gigantesques priment sur l'exactitude des calculs.

Un besoin de compréhension

La plupart des auteurs s'attachent à *résoudre* des problèmes de visibilité. Rares sont ceux qui proposent un regard analytique ou descriptif, sur la nature des problèmes et des propriétés sous-jacentes. La recherche de solutions immédiates à des problèmes concrets et précis a conduit à se ramener à des problèmes connus et mieux maîtrisés. Cela a donné lieu à des solutions dont l'efficacité n'est pas discutable, mais la spécificité des questions de visibilité a souvent été occultée. Nous pensons qu'il est utile de mieux comprendre la visibilité pour ensuite la traiter plus efficacement.

La visibilité dans le plan a suscité un grand intérêt dans la communauté théorique de géométrie algorithmique . Des travaux tel que le complexe de visibilité [PV96b] permettent d'appréhender les propriétés de visibilité du plan dans un cadre élégant et puissant. Les problèmes de visibilité s'y expriment naturellement et la cohérence de visibilité y est intimement explicitée.

Malheureusement, la littérature théorique est plus pauvre dès lors que la troisième dimension est abordée. Les auteurs s'intéressent le plus souvent à la complexité théorique de problèmes comme le calcul d'une vue ou le lancer de rayon. Les résultats disponibles sont des réductions d'algorithmes qui évitent de traiter directement les questions de visibilité et qui n'apportent que peu de compréhension sur les phénomènes impliqués. Même en théorie l'approche est avant tout constructive et non descriptive ou analytique.

Un besoin de synthèse

Nos recherches nous ont amené à lire de nombreux travaux traitant de visibilité en synthèse d'images, mais aussi en vision par ordinateur, en robotique ou en géométrie algorithmique. De nombreuses similarités existent, que ce soient dans les problématiques ou dans les approches adoptées. Nous nous sommes cependant rendu compte d'une certaine imperméabilité de ces domaines et de la méconnaissance – la nôtre en premier lieu – des travaux voisins. Cela nous amène parfois malheureusement à re-développer des techniques, ou pire, à ignorer des solutions qui pourraient faire la lumière sur nos problèmes.

La quantité de travaux publiés et leur dissémination rend malheureusement difficile l'abord de la question sans faire l'impasse sur des pans entiers, ou sans se noyer dans un nombre de références inextricable.

Il nous a paru fructueux de réaliser un tour d'horizon qui inclût tous ces domaines et permît d'avoir une vision globale sur la visibilité. En particulier nous jugeons utile de faire le lien entre différentes approches et de proposer des interprétations parfois absentes des articles originaux. Une telle synthèse peut être utile à ceux qui débutent sur le sujet tout comme à ceux qui veulent s'informer sur les problèmes et techniques étudiés dans d'autres communautés.

Difficulté des problèmes de visibilité

La plupart des approches antérieures ont considéré la visibilité comme à une boîte noire qui à des requêtes fournit des réponses. Les travaux à ce jour ont plus cherché à optimiser le traitement des requêtes qu'à comprendre les aspects structurels de la visibilité.

Empruntons à Koenderink et van Doorn [Kv76] la métaphore de l'étude d'une fonction ou d'une équation. La plupart des méthodes de visibilité permettent de calculer la valeur de la fonction : quel est le point vu selon ce rayon ? Ces deux points sont-ils mutuellement visibles ? Ces requêtes ponctuelles ne nous procurent cependant pas d'information plus globale ou structurelle qui permettrait de mieux décrire la fonction : est-elle monotone, continue, quelles-sont ses inflexions, etc.

Dans cette section, nous résumons les obstacles conceptuels qui font de l'étude de la visibilité tridimensionnelle un sujet difficile. Nous commençons par expliquer l'échec logique d'une approche tridimensionnelle directe. Nous essayons ensuite de dégager des intuitions qui plaident pour l'existence d'une cohérence de visibilité. Nous décrivons le travers de certains travaux qui décrivent la visibilité comme si les objets étaient transparents. Nous finissons en expliquant pourquoi les *graphes d'aspects*, s'ils présentent une contribution majeure à l'étude de la visibilité présentent un cadre qui ne nous satisfait pas entièrement.

Nature de la visibilité

La visibilité ne présente aucune localité spatiale. D'un point de l'espace on peut voir des objets très éloignés, alors qu'un objet très proche peut être complètement occulté. C'est l'un des écueils qui expliquent la difficulté de l'étude de la visibilité et le fait que des approches spatiales ne répondent pas véritablement au fond du problème. On a cependant cherché à rapporter les problèmes de visibilité à des problèmes purement tridimensionnels, le plus souvent à de la localisation ou à de la détection de collision.

Intéressons-nous à l'exemple du lancer de rayons. Les approches classiques accélèrent cette méthode en optimisant l'intersection d'un rayon avec la scène. Cela passe le plus souvent par l'utilisation d'une structure spatiale comme une grille régulière ou des boîtes englobantes qui permettent de savoir quels sont les objets spatialement proches du rayon. On recherche la "collision" entre le rayon et la scène. De telles méthodes ont fait leurs preuves, elles accélèrent notablement les calculs. Cependant aucune compréhension n'a été apportée. Le problème est contourné, on se ramène juste à un problème conceptuellement plus simple, savoir si deux entités tridimensionnelles ont une intersection.

Quelle est la nature des propriétés de visibilité ? Quel est le cadre qui permet de les étudier de manière à en faire ressortir la structure ? La visibilité s'exprime naturellement grâce aux rayons lumineux. L'exemple du lancer de rayons est paradigmatique. Une image est constituée par l'ensemble des rayons lumineux qui arrivent à l'œil, les parties éclairées d'une scène sont celles accessibles par des rayons issus d'une source de lumière.

Les rayons permettent d'exprimer les problèmes de visibilité sous la forme atomique suivante : quel est le premier objet visible depuis un point dans une direction donnée. Toute question de visibilité se ramène à cette propriété atomique.

Quel est alors le problème des approches tridimensionnelles de visibilité ? Certaines effectuent pourtant des calculs sur les rayons, on a parlé du lancer de rayon par exemple. Les rayons n'y sont cependant pas étudiés en tant qu'entité propre, mais en tant qu'ensemble de points tridimensionnels. On s'intéresse aux objets qui ont une intersection avec les *points* le long du rayon ; on ne pose pas directement la question de l'objet *vu* par le rayon. Une représentation intermédiaire (ensemble de points de l'espace) a été introduite.

Une telle approche localise le problème de visibilité dans l'espace à trois dimensions : un point est-il dans un objet ? La seule structure ou cohérence qui peut être capturée est purement spatiale, liée à la proximité tridimensionnelle d'objets. Or, nous l'avons vu, il est difficile de parler de proximité spatiale dès lors que l'on aborde la visibilité.

La cohérence de visibilité

Par cohérence, nous exprimons l'idée que deux requêtes ou propriétés "proches" auront la plupart du temps une réponse "similaire". On en revient à la notion mathématique de continuité.

Prenons un exemple simple comme la vue depuis un point de l'espace. Cette vue exhibe en générale une grande cohérence. Si nous prenons deux points proches de l'image, il y a de fortes chances qu'ils correspondent au même objet de la scène. Pour décrire une telle cohérence, on dit que le voisinage d'un point voit le même objet que lui.

Déplaçons maintenant légèrement le point de vue. La vue sera très certainement similaire. On a là encore une cohérence. Il y aura discontinuité quand des objets apparaîtront ou disparaîtront. Un autre exemple de cohérence est donné par les ombres douces : le passage progressif de l'ombre totale à l'éclairage total traduit le fait que la partie visible de la source varie de manière *cohérente*. Cette cohérence d'une vue ou de la partie visible d'une source ont abondamment été étudiées en vision ou en synthèse d'image.

On semble cependant revenu à une proximité tridimensionnelle, puisque le voisinage dans l'image correspond à une proximité de points tridimensionnels. Toutefois la proximité existe entre les résultats de deux requêtes proches et non entre le résultat de la requête et le point de vue.

Espace des droites et objets transparents

Certaines approches [CEG+96, MO88, Pel97b, Pel90, Pel93, Tel92a] ont étudié la visibilité en considérant les droites de l'espace. Les droites y sont considérées comme entités et aucun recours à des techniques spatiales tridimensionnelles n'est utilisé. Une technique de dualisation est généralement employée pour faciliter les calculs et la conceptualisation.

Cependant, la visibilité est définie grâce à l'intersection des droites avec les objets de la scène. Toutes les intersections d'une droite sont prises en compte. La notion de première intersection, d'objet visible en fait, ne peut être exprimée. Les objets sont comme transparents, les occultations sont ignorées.

Il est indispensable d'utiliser la notion de l'origine d'un rayon. La technique de classification de rayons [AK87] offre cette possibilité, un rayon y est décrit comme une origine (un point tridimensionnel) et une direction. L'ensemble des objets que peuvent voir un groupe de rayons est calculé. Cependant, les calculs sont immédiatement ramenés à l'intersection du plongement tridimensionnel des rayons avec la scène : on est revenu au problème précédemment cité des approches spatiales, structure ou cohérence sont difficiles à dégager.

Les graphes d'aspect

Anticipons sur l'étude des travaux antérieurs. Les graphes d'aspect constituent l'exemple le plus abouti vers une compréhension des propriétés de visibilité d'une scène tridimensionnelle. Ils permettent de décrire toutes les vues possibles d'une scène, en les regroupant par "similarité". C'est une structure de données tout à fait fondamentale et nous reprendrons beaucoup des concepts sur lesquels elle repose.

Ils permettent notamment de décrire à quel moment la vue va changer de manière qualitative : apparition ou disparition d'une partie d'un objet principalement.

En tant qu'outil d'analyse de la visibilité, les graphes d'aspect se concentrent uniquement sur les vues d'un objet. De ce fait, ils ne permettent pas par exemple de décrire ou déterminer la visibilité mutuelle de points de l'espace.

Approche et buts

Nous décrivons maintenant l'approche que nous avons suivie pour aborder ces problème, puis les critères qui ont guidé nos travaux et qui illustrent les buts que nous nous sommes fixés.

Approche

Ce mémoire propose avant tout une analyse morphologique et phénoménologique de la visibilité.

On l'aura deviné à la lecture de cette introduction, notre étude de la visibilité sera basée sur la notion de rayon lumineux et sur l'expression d'une cohérence parmi ces rayons lumineux. Pour mieux qualifier la cohérence, nous allons nous intéresser aux endroits où cette cohérence est rompue.

Dans l'étude de toute question ou système, l'information n'est pas portée par les zones "uniformes", continues où par définition rien ne se produit. L'organisation ou la structure sont définies par les discontinuités, les frontières, les *catastrophes* (au sens de Thom [Tho72]). La plupart des méthodes actuelles de visibilité reviennent à découvrir un pays en en considérant quelques points. Nous pensons que la connaissance de ses frontières permet une compréhension plus globale, un plus grand recul, qui ne dispense cependant pas d'une étude des points intérieurs.

Nous allons nous attacher à comprendre pour quels rayons les propriétés de visibilité changent et pour lesquelles elles restent invariantes. C'est-à-dire quels rayons voient le même objet.

Notre approche est tout d'abord *topologique*. La topologie s'intéresse aux relations de voisinage. Elle est tout indiquée pour étudier de manière formelle la notion de cohérence. Elle nous permettra de décrire les relations entre les rayons qui voient différents objets. Nous n'aurons cependant recours à aucun formalisme mathématique topologique, même si les notions sous-jacentes sont bien présentes. Nous avons préféré un langage plus concret.

Le complexe de visibilité de Pocchiola et Vegter [PV96b] a été notre source d'inspiration initiale. Nous avons voulu étendre ces travaux à la troisième dimension.

Critères

Notre but est avant tout de développer un cadre qui décrive les propriétés de visibilité d'une scène tridimensionnelle.

Toute requête, toute propriété de visibilité doit pouvoir s'y exprimer naturellement, qu'elle soit simple (point vu par un rayon, vue depuis un point) ou plus globale (limites d'ombres, parties de la scène vues depuis un volume, ensemble des vues d'un objet).

Nous voulons expliciter la cohérence. Nous devons rendre compte de la similarité de deux requêtes, expliquer quand et comment un changement se produit.

Il est de plus souhaitable que cette meilleure compréhension théorique débouche sur des avancées pratiques, sous la forme d'une structure de données la plus générique possible. Nous voulons éviter de nous focaliser sur un problème particulier, même si nos travaux sont motivés par les applications que nous avons décrites.

Une validation d'un tel outil doit cependant être apportée par une confrontation à des problèmes pratiques et concrets. L'élégance et la complexité théorique d'une approche ne constituent pas un critère suffisant de son efficacité réelle.

Contributions

Les contributions de cette thèse peuvent être organisées en cinq points que nous reprenons ci-dessous. Les trois premiers sont le fruit d'une étude de la visibilité tridimensionnelle dans l'espace des droites. Nous décrivons tout d'abord une étude et une structure de données plutôt fondamentales et théoriques. Cela débouche sur le développement d'un outil pratique et générique de visibilité qui est appliqué au cas concret de la simulation de l'éclairage. Le point suivant consiste en un précalcul efficace de visibilité par rapport à des cellules volumiques pour l'affichage de très grosses scènes en se ramenant à des projections sur des plans. Pour finir, nous avons effectué un vaste tour d'horizon des travaux sur la visibilité.

– Nous avons développé le *complexe de visibilité 3D*, une structure de données qui décrit toutes les relations de visibilité dans une scène, qu'elle soit composée de polygones ou d'objets courbes. Il s'agit d'une structure dans l'espace des droites, ou plutôt comme nous le verrons, dans l'espace des segments libres maximaux. Intuitivement le complexe de visibilité regroupe les rayons lumineux qui "voient" le même objet. Les limites de ces groupes de segments correspondent aux changements de visibilité dans une scène. En particulier ils décrivent les limites d'ombre et de pénombre et les endroits où un observateur mobile voit apparaître les objets, ce que l'on nomme *événement visuel*.

  Nous analysons la complexité de cette structure de données, de manière classique par des bornes, mais aussi par une analyse probabiliste qui s'appuie sur un modèle simplifié de scène normale. Nous décrivons un algorithme de construction dont la complexité dépend de la taille du résultat. Nous présentons l'interprétation de nombreuses requêtes de visibilité grâce au complexe. Nous avons de plus généralisé cette approche au cas de scènes dynamiques.

– L'analyse du complexe de visibilité et la compréhension ainsi acquise nous ont permis de développer une structure de données plus simple que nous appelons le *squelette de visibilité*. Le squelette de visibilité est un graphe dans l'espace des droites qui décrit tous les *événements visuels* d'une scène.

  Nous décrivons un algorithme de construction qui évite le traitement direct des événements visuels, sources de complications algorithmiques et de problèmes de robustesse puisqu'ils décrivent des surfaces, pas toujours planes, dont il faut habituellement calculer l'intersection avec les objets de la scène. Nous calculons les *droites poignardantes extrêmes* qui sont les extrémités des événements visuels dans l'espace des droites. Les événements visuels sont simplement déduits grâce à un catalogue qui indique leurs adjacences avec les droites poignardantes extrêmes. La grande force de cette méthode est qu'elle ne nécessite que des calculs simples sur les droites. Elle est de plus très locale, ce qui est un gage de robustesse : une erreur ne peut entraîner des incohérences qu'au voisinage d'un événement, le reste du calcul n'est pas perturbé. Notre algorithme permet des calculs localisés, à la demande et n'impose pas un calcul de la structure complète.

  Cette méthode a été implémentée pour des scènes polygonales et des tests ont été réalisés sur des scènes comportant jusqu'à 1500 polygones. Nous avons également implémenté diverses requêtes de visibilité comme des calculs de parties visibles, de limites d'ombre, la liste des bloqueurs entre deux polygones

et les limites d'occultation causées par un objet. L'exécution de ces requête est très rapide une fois la structure construite, de l'ordre de quelques milli-secondes.

Une méthode pour mettre à jour cette structure après le mouvement d'un objet est également décrite.

– Nous avons appliqué le squelette de visibilité à la simulation de l'éclairage par la méthode de radiosité hiérarchique. Il permet un calcul exact et efficace de la quantité de lumière quittant un polygone qui arrive en un point. Nous l'utilisons également pour subdiviser le maillage utilisé pour la représentation de la fonction d'éclairage le long des limites d'ombre et de pénombre, en permettant de considérer n'importe quel polygone comme source, ce qui est crucial pour les scènes où l'éclairage indirect domine.

  Contrairement aux méthodes antérieures, notre représentation hiérarchique de la fonction d'éclairage n'est pas constante par morceaux mais linéaire par morceaux et continue. Nous avons pour cela développé l'utilisation d'ondelettes paresseuses avec des triangulations hiérarchiques pour traiter les maillages irréguliers dus aux limites d'ombre.

  L'information contenue dans le squelette de visibilité est également utilisée pour décider à quel niveau de la hiérarchie des triangles un échange doit être simulé. Nous utilisons une métrique perceptuelle. Les seuils que nous utilisons traduisent directement à quel point les artefacts dus à une approximation sont discernables pour un observateur humain, là où les méthodes classiques imposent le réglage de critères quelque peu arbitraires dont il est dur de prédire l'influence.

  Des comparaisons sur une série de scènes ont montré notre méthode meilleure qu'une des méthodes les plus récentes, que ce soit en temps de calcul ou en qualité. Le prix à payer est une grande consommation mémoire. Nous proposons cependant des solutions pour la limiter.

– Nous avons développé une méthode de précalcul pour l'affichage de scènes très complexes. L'espace dans lequel l'observateur peut se déplacer est subdivisé en cellules. Pour chaque cellule l'ensemble des objets potentiellement visibles est calculé de manière non exacte mais prudente.

  Notre méthode est basée sur une projection sur des plans. Nous définissons des opérateurs de *projection étendue* qui permettent des calculs prudents : un objet invisible peut être déclaré potentiellement visible, mais aucun objet visible ne doit être déclaré invisible. Notre méthode est la première qui permette dans un précalcul de prendre en compte l'occultation due à la conjonction de plusieurs bloqueurs dans des scènes quelconques.

  Nous décrivons de plus un *balayage d'occultation* qui consiste à balayer la scène avec un plan qui s'éloigne de la cellule et qui accumule les occultations dues à de petits objets comme les feuilles d'arbres dans une forêt.

  Ces méthodes ont été implémentées et tirent profit des cartes graphiques pour une efficacité accrue. Nous pouvons par exemple réaliser des promenade dans une ville de plus de deux millions de polygones contenant des voitures en mouvement à environ 25 images par seconde sur une Onyx2.

– Nous avons réalisé un vaste tour d'horizon des travaux qui traitent de visibilité dans de nombreux domaines. Après avoir exposé les situations dans lesquelles des problèmes de visibilité se posent, nous proposons une classification qui permet d'aborder les techniques avec un regard nouveau, indépendamment du domaine dans lequel elles ont été développées.

  Nous nous sommes efforcé de proposer un maximum de références et de souligner les similitudes. Nous essayons de fournir suffisamment de détails sur les techniques les plus marquantes pour ne pas réserver cette étude à ceux qui connaissent déjà les publications en question.

Organisation du document

Lors de notre synthèse des travaux antérieurs, la richesse de la littérature relative à la visibilité nous a vite conduit à dépasser la taille qu'il est raisonnable de consacrer à la bibliographie dans une thèse. Nous avons donc décidé d'en faire une partie indépendante après les contributions et d'inclure en plus un court résumé des travaux les plus pertinents pour notre recherche.

Le premier chapitre présente ainsi un résumé des méthodes antérieures qui traitent les problèmes de visibilité globale et les scènes complexes. Nous développons ensuite le *complexe de visibilité 3D* dans le chapitre 2, puis sa simplification, le *squelette de visibilité* au chapitre 3. Le chapitre 4 traite de l'application du squelette

aux calculs d'éclairage. Enfin le chapitre 5 décrit notre méthode de calcul d'occultations à l'aide de *projections étendues.*

Les travaux présentés dans cette thèse ont fait l'objets de publications. Nous signalons en introduction de chaque chapitre les ajouts et différences avec les versions publiées afin que le lecteur qui a déjà connaissance de nos travaux puisse se concentrer sur les derniers développements.

L'organisation du tour d'horizon proposé en seconde partie est présentée au chapitre 6, nous ne la reprenons donc pas ici.

Nous concluons par un résumé des contributions et par des pistes de recherche future.

Première partie

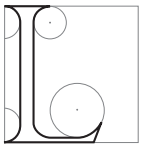
Contributions

CHAPITRE 1

Travaux antérieurs

1. Les lignes droites issues de l'œil franchissent des distances d'une grande longueur ; 2. La figure circonscrite par les rayons visuels est un cône qui a son sommet dans l'œil et sa base aux limites de ce qui est vu ; 3. On voit ce sur quoi tombent les rayons visuels, on ne voit pas ce sur quoi ils ne tombent pas.

EUCLIDE, *Optique*



A littérature dédiée à la visibilité est trop vaste pour être présentée ici de manière exhaustive. Le lecteur trouvera un plus large tour d'horizon des problèmes et techniques de visibilité dans la seconde partie de ce document. Dans ce qui suit, nous présentons des situations où des calculs de visibilité sont nécéssaires en synthèse d'images, en vision par ordinateur et en robotique, puis nous présentons des techniques importantes pour les résoudre. L'accent est mis sur les solutions récentes, les scènes très complexes et les calculs *globaux* de visibilité.

Par *globaux* nous entendons des problèmes ou méthodes où est nécessaire une information de visibilité par rapport à une entité plus grande qu'un point : visibilité depuis une surface, depuis un volume, ou par rapport à la scène toute entière (par opposition à la classique élimination des parties cachées qui considère la visibilité par rapport au point de vue).

Nous nous contentons de décrire succintement les méthodes. Le lecteur intéressé trouvera une description plus détaillée dans le tour d'horizon en seconde partie de ce mémoire.

Nous passons tout d'abord en revue les contextes dans lesquels surviennent des problèmes de visibilité. Dans la section 2, nous présentons les méthodes d'élimination d'occultation (*occlusion culling*) qui rejettent rapidement des portions d'une scène cachées depuis le point de vue, tandis que la visibilité par rapport à un volume est traitée dans la section 3. Les limites d'ombre sont couvertes dans la section 4. La section 5 présente les *graphes d'aspect*, une structure qui représente toutes les vues possibles d'un objet. Les méthodes qui opèrent dans l'espace des droites sont passées en revue dans la section 6 et la section 7 traite de la mise à jour de l'information de visibilité dans les scènes en mouvement. Nous concluons avec une discussion des problèmes qui demeurent.

1 Problèmes de visibilité

1.1 Synthèse d'images

L'un des grands et importants défis des débuts de la synthèse d'image était le problème de l'élimination des parties cachées, qui était alors synonyme de visibilité. Un tour d'horizon des techniques classiques peut être trouvé dans, par exemple, [FvDFH90, Rog97, SSS74], tandis que des résultats plus théoriques sont fournis dans [Dor94, Ber93].

Toutefois, malgré des implantations matérielles dédiées efficaces, la taille toujours croissante des bases de données graphiques rend problématique l'obtention d'un affichage en temps-réel. Des approches ont été développées qui diminuent la quantité de géométrie envoyée aux cartes graphiques [Cla76, HG94]. La dernière décennie s'est révélée particulièrement riche en travaux sur ces techniques. Elles incluent la simplification de maillages [HGar, Hop96, GH97a], la simplification à base d'images [SLSD96, SS96a, SDB97] et l'*élimination d'occultation* que nous introduirons dans la prochaine section. Cette dernière technique consiste à rejeter rapidement des parties de la scène qui sont cachées de manière "évidente", par exemple par un mur proche.

L'élimination d'occultation peut également être exploité dans des situations où la base de données de la scène est trop grande pour tenir en mémoire principale, ou si elle doit être chargée depuis le réseau [FST92, Fun96c, COZ98]. Seules les portions visibles du modèle, ou celles susceptibles de le devenir prochainement, sont effectivement chargées.

Le calcul des ombres, tout spécialement des ombres douces causées par les sources étendues, est toujours un problème critique [WPF90, Hec87][1]. Dans le dernier cas, on peut distinguer les points de la scène pour lesquels la source de lumière est complètement visible, partiellement visible ou cachée. Cela définit les zones éclairée, de pénombre et d'ombre. Les problèmes fondamentaux sont la détermination des limites des zones d'ombre et de pénombre, ainsi que la détermination efficace et robuste de la portion de la source visible depuis un point dans la région de pénombre.

L'illumination globale [CW93b, SP94, Gla95] simule la propagation de la lumière à l'intérieur d'une scène. Si l'éclairage indirect est traité, chaque surface de la scène est considérée comme une source et reflète une portion de la lumière qu'elle reçoit. La méthode de *radiosité* [CW93b, SP94] suppose que tous les objets de la scène sont *diffus*, c'est à dire que la lumière est reflechie uniformément dans toutes les directions. Les polygones de la scène sont subdivisés en éléments sur lesquels l'éclairement est supposé constant. L'interaction entre deux éléments est modélisée par un *facteur de forme* qui est la proportion de la lumière quittant un élément qui atteint l'autre. Le calcul de ces facteur de forme demeure la partie la plus coûteuse de la méthode à cause des calculs de visibilité intensif qu'il réclame [HSD94]. De plus, une subdivision régulière des polygones initiaux entraîne des artefacts, des marches d'escalier dans les ombres. Subdiviser selon les limites d'ombre est souhaitable pour améliorer la qualité des images.

1.2 Vision par ordinateur

En vision par ordinateur, la *reconnaissance d'objet basée sur les modèles* (*model-based object recognition*) essaye de mettre en correspondance les éléments d'une image avec les éléments de modèles d'objets. Cela nécessite une représentation efficace de la visibilité d'un objet depuis tout point de vue possible.

La position des senseurs a une grande influence sur bien des tâches de vision. La *vision active* [AAA+92, RM97, TAT95] adapte le placement des capteurs pour une tâche donnée. On recherche souvent la visibilité d'un élément ou de la scène entière, ce qui requiert une optimisation complexe qui dépend de la visibilité par rapport à de nombreux points de vue.

1.3 Robotique

Déduire la position d'un robot de sa vue courante [GMR95, SON96, TA96] est tout à fait semblable au problème de reconnaissance d'objets. Étant donnée une carte de l'environnement, le robot doit reconnaître la structure des murs visibles.

[1] Dans une discussion personnelle, Dan Wexler de Pacific Data Images déclare que les ombres dures posent toujours problème en production. L'aliassage et la précision numérique sont toujours des sources d'artefacts. De plus, si des "trucs" existent pour obtenir des ombres douces, une méthode qui permette des calculs robustes et efficaces de pénombre est ardemment souhaitée.

Récemment, on s'est posé le problème de trouver un intrus imprévisible à l'aide d'un robot mobile [GLLL98]. La trajectoire du robot doit être planifiée pour qu'il fouille chaque région de la scène et pour qu'un intrus ne puisse pas se déplacer d'une region pas encore visitée vers une region déjà visitée sans être vu.

2 Élimination d'occultation en ligne

L'idée de l'*élimination d'occultation* (*occlusion culling*) a été proposée dans les années 70 [Jon71, Cla76], mais c'est seulement récemment que la taille des bases de données 3D a imposé son utilisation en pratique.

Nous présentons ici les techniques *en ligne*, dans lesquelles les calculs sont effectués pour chaque image par rapport au point de vue courant. Le principe est semblable pour toutes ces méthodes, inspiré de l'algorithme efficace utilisé pour éliminer les objets qui sont en dehors du cône de vue [GBW90].

La scène est organisée en une hiérarchie de volumes englobants. Avant de rendre chaque nœud de la hiérarchie, la visibilité de sa boîte englobante est testée. Si le test échoue, tous ses enfants sont éliminés. Sinon, ils sont testés récursivement. Les méthodes diffèrent dans la façon dont les tests sont effectués, et selon que l'occultation due à tous les objets est prise en compte ou selon qu'un sous-ensemble de bloqueurs spéciaux doit tout d'abord être choisi.

2.1 Z-buffer hiérarchique et cartes d'occultation hiérarchiques

Greene *et al*. [GKM93, Gre96] proposent une extension du matériel dédié au z-buffer pour effectuer le test de visibilité dans l'espace image et prendre en compte toutes les occultations. Au fur et à mesure que l'image est générée, la boîte englobante de chaque nœud de la hiérarchie est testé avec les valeurs présentes dans le tampon de profondeur courant (mais aucun pixel n'est effectivement tracé). Ce test est optimisé grâce à l'utilisation d'une pyramide de profondeur qui évite la comparaison pour chaque pixel couvert par la boîte englobante. Cela explique le nom de la méthode, le z-buffer hiérarchique.

Les nœuds de la hiérarchie sont rendu grossièrement de l'avant vers l'arrière, et la pyramide de profondeur est mise à jour tandis que les objets sont rendus. Cette méthode prend en compte la *fusion de bloqueurs*, c'est à dire l'occultation causée par la conjonction de plusieurs objets, grâce à l'agrégation naturelle dans le bitmap.

Malheureusement, cela nécessite une modification du matériel graphique pour obtenir des requêtes efficaces de valeur de profondeur et pour mettre à jour la pyramide de profondeur. Zhang *et al.* [ZMHH97, Zha98b] proposent les *Cartes d'occultation hiérarchiques* (*Hierarchical Occlusion Map*) qui suppriment ces limitations (figure. 1.1). Un (grand) sous-ensemble de bloqueurs proches est tout d'abord rendu et une carte d'occultation est lue depuis le tampon graphique (une seule requête est faite au matériel graphique). Cette carte d'occultation est organisée selon une pyramide pour des tests efficaces d'occultation. Il faut cependant noter que le *test de recouvrement* est distinct du *test de profondeur*, c'est à dire qu'il est tout d'abord déterminé si la projection du nœud est recouverte par les projections des bloqueurs, puis si le nœud est derrière les bloqueurs. Cela est particulièrement important pour effectuer des calculs approximatifs, cf. [ZMHH97, Zha98b] pour plus de détails.

2.2 Bloqueurs convexes et événements visuels

Coorg et Teller [CT96, CT97b] ont choisi une approche dans l'espace objet qui traite les occultations dues à de gros bloqueurs convexes. Chaque bloqueur définit un *volume d'ombre* [Cro77] (pyramide tronquée infinie dont le sommet est le point de vue ou la source de lumière). Les nœuds de la hiérarchie de la scène sont classifiés par rapport à ces volumes d'ombres comme étant visibles, partiellement visibles ou occultés.

La *cohérence temporelle* est utilisée pour maintenir cette classification. Ils remarquent que pour un mouvement continu de l'observateur, les changements de visibilité ne se produisent que dans le voisinage des nœuds partiellement visibles. De plus, ces changements se produisent uniquement lorsqu'un *événement visuel* est traversé (cf. figure 1.2). Ils considèrent les événements visuels définis par les plans supports ou séparateurs, c'est à dire les plans tangents à un bloqueur et à un nœud.

Ils maintiennent une liste d'événements visuels, détectent quand le point de vue les traverse, et mettent à jour les statuts de visibilité et la liste d'événements en conséquence.

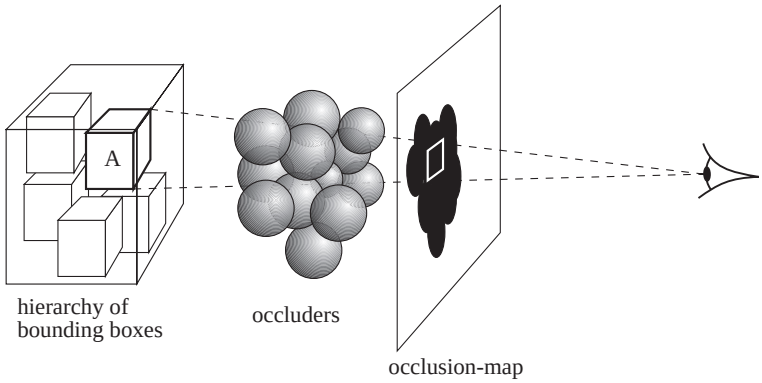


FIG. 1.1: Élimination d'occultation à l'aide de cartes d'occultation. Le nœud *A* de la hiérarchie des objets est testé par rapport à la carte d'occultation. Puisque sa projection se trouve dans la projection des bloqueurs, il est caché (s'il se trouve derrière).

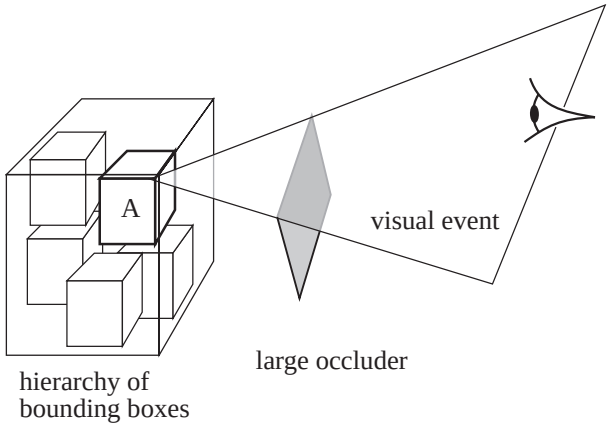


FIG. 1.2: Élimination d'occultation et événements visuels. Le nœud *A* de la hiérarchie des objets deviendra complètement occulté lorsque le point de vue traversera le coin de l'événement visuel.

Le principal défaut de cette approche est de ne pas prendre en compte la fusion de bloqueurs, puisque seuls sont éliminés les objets cachés par un bloqueur convexe unique. Des approches similaires sont décrites dans [HMC+97, BHS98, HZ81, HZ82].

2.3 Trace de pas de l'ombre des bloqueurs

Récemment, Wonka et Schmalstieg [WS99] ont proposé une méthode efficace pour les scènes comme les villes. Ils exploitent le fait qu'une cité peut être modélisée comme un champ de hauteur (ou un terrain), c'est à dire qu'elles peuvent être décrites comme une fonction $z = f(x,y)$. L'occultation causée par une maison se produit dans un volume d'ombre dont le sommet est le point de vue. Ce volume d'ombre peut également être décrit comme une fonction $z = f(x,y)$. Un objet est caché s'il se trouve en dessous du volume d'ombre des maisons (figure 1.3).

Ils utilisent le matériel graphique pour représenter et tester ces fonctions. Une vue orthographique depuis le dessus de la scène est utilisée, et le tampon de profondeur stocke la fonction $z = f(x,y)$. Les coins définissant les volumes d'ombre sont discrétisés et les nœuds de la hiérarchie de la scène sont éliminés si le test en z détermine qu'ils se trouvent en dessous des volumes d'ombre.

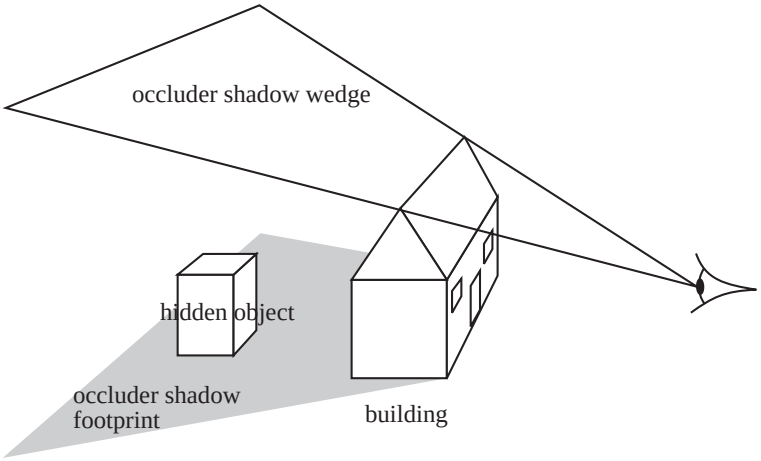


FIG. 1.3: Trace de pas de l'ombre des bloqueurs.

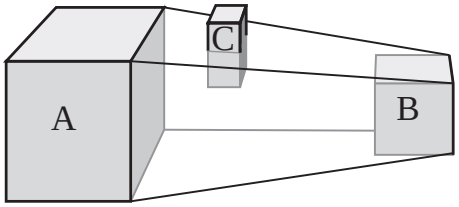


FIG. 1.4: Élimination par tubes. L'objet *C* intersecte le tube entre *A* et *B*, il est donc considéré pour les calculs de visibilité.

3 Tubes et séquences d'ouvertures

Comme nous l'avons vu dans la section précédente, la visibilité par rapport à un point est souvent caractérisée en utilisant des cônes ou volumes d'ombre. Nous présentons maintenant l'extension de ces concepts à la visibilité par rapport à une surface ou par rapport à un volume, où un sommet unique ne peut plus être utilisé pour définir les portions de l'espace où les occultations ont lieu.

3.1 Élimination par tubes

Haines et Wallace [HW91] ont développé l'élimination par tubes (*shaft-culling*) pour accélérer le calcul des facteurs de forme. Ils remarquent que les bloqueurs qui peuvent créer des occultations entre deux objets se trouvent dans l'enveloppe convexe approximative de ces objets, appelée *shaft* (tube) dans ce contexte (cf. figure 1.4). Ils ont développé un algorithme efficace pour rapidement déterminer les objets qui intersectent un shaft.

Zhao et Dobkin [ZD93] proposent également une méthode à base de tubes où les triangles de la scène sont pré-traités dans un espace multidimensionnel pour des tests d'intersection tube-triangle plus efficaces. Teller et Hanrahan [TH93] utilisent aussi des tubes pour maintenir des listes de bloqueurs dans un contexte de radiosité.

3.2 Environnements architecturaux et hublots

Airey [Air90] et Teller [Tel92b, TS91, TH93] ont effectué des précalculs prudents de visibilité dans des scènes architecturales. Le principe est de déterminer les parties de la scène qui sont visibles depuis chaque

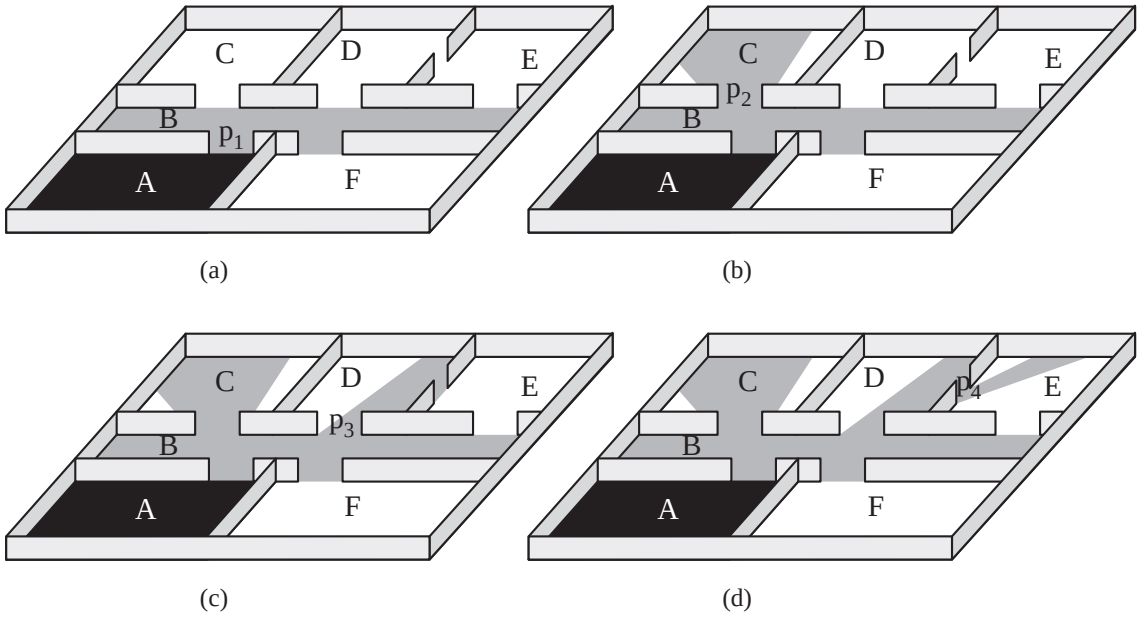


FIG. 1.5: Propagation de visibilité à travers de hublots. Les pièces visibles depuis la pièce *A* sont calculées à l'aide d'un parcours en profondeur des adjacences. La visibilité est propagée à travers les hublots p_i.

pièce (également appelées cellules). La visibilité est définie par rapport à n'importe quel point de la pièce.

Les *hublots* sont exploités : une cellule n'est visible depuis une autre cellule qu'à travers une séquence d'ouvertures ou hublots, c'est à dire uniquement s'il existe une ligne de vue qui traverse les hublots. La visibilité est propagée par un parcours en profondeur. La visibilité à travers une séquence de hublots peut être échantillonnée à l'aide de lancer de rayon ou d'un z-buffer [Air90], calculée à l'aide programmation linéaire si l'on utilise le plan au sol 2D de la scène [TS91], ou bien dans le cas de hublots généraux en 3D, en maintenant une liste de plans séparateurs et supports [TH93].

Cela fournit une information cellule-à-cellule (quelle cellule peut voir quelle cellule ?). Teller [Tel92b, TS91] raffine ensuite cette information pour obtenir une information cellule-à-objet en testant les objets à l'intérieur d'une cellule par rapport à une structure de tube définie parla séquence de hublots. Pour finir, à chaque cellule est stocké le PVS, l'ensemble des objets potentiellement visibles (*potentially visible set*). Il contient tous les objets dont on a déterminé qu'ils étaient potentiellement visibles depuis la cellule.

Lors du rendu temps-réel, les ensembles potentiellement visibles sont utilisés pour restreindre la géométrie envoyée au matériel graphique. Cette information peut être raffinée en utilisant la visibilité œil-à-objet qui se calcule avec une méthode similaire à celle décrite dans la section 2.2 et qui bénéficie de la séquence de hublots. L'information de PVS peut être utilisée pour du pré-chargement de base de données [FST92, Fun96c], ce qui est l'un des avantages clefs de la méthode.

Ce précalcul a également été appliqué aux calculs de radiosité [ARB90, TH93, TFFH94, Fun96b] où il permet de restreindre les calculs des intéractions lumineuses aux paires de polygones mutuellement visibles.

Luebke et George [LG95] proposent également une version en-ligne où la boîte englobante des hublots dans l'espace écran est utilisée pour éliminer la géométrie des pièces adjacentes.

3.3 Précalcul d'occultation pour des bloqueurs convexes

Cohen-Or *et al.* [COFHZ98, COZ98] proposent également un algorithme d'élimination d'occultation hors-ligne. Il n'est pas restreint aux scènes d'architecture, mais ne considère que l'occultation due à des bloqueurs convexes uniques. La région de l'observateur est subdivisée en cellules et pour chaque cellule la visibilité de tous les objets de la scène est testée.

Pour cela, un lancer de rayon est effectué entre toutes les paires de sommets de la cellule et de la boîte englobante de l'objet. Si tous les rayons sont bloqués par un même bloqueur convexe, alors par convexité le bloqueur cache l'objet depuis tout point de la cellule (cf. figure 1.6).

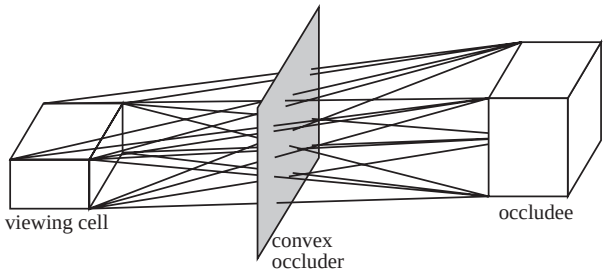


FIG. 1.6: Précalcul d'occultation pour des bloqueurs convexes. Si tous les rayons entre les sommets de la cellule et de l'objet sont bloqués parle même bloqueur convexe, alors l'objet est caché depuis tout point de la cellule.

Le défaut évident de cette approche est son incapacité à traiter l'occultation due à la conjonction de plusieurs bloqueurs. De plus, lancer autant de rayons entraîne un temps de calcul élevé.

4 Limites d'ombre et de pénombre

Le calcul des ombres douces requiert lui aussi des calculs de visibilté par rapport à une région étendue. Les méthodes que nous présentons ont tout d'abord été développées comme des extensions des volumes d'ombre. La différence par rapport à la section précédente est que le résultat attendu est ici la frontière exacte des ombres et pénombres des objets de la scène.

Ces limites sont utilisées dans deux buts : tout d'abord pour accélérer le calcul des ombres en n'effectuant des calculs complexes qu'à l'intérieur de la zone de pénombre, mais aussi pour subdiviser les polygones de la scène le long des limites d'ombre pour des ombres de haute qualité en simulation de l'éclairage.

4.1 Ombres et pénombres dues à un bloqueur convexe

Le travail de Nishita et Nakamae [NN85, NON85, NN83] étudie l'ombre causée par un bloqueur convexe par rapport à une source de lumière polygonale convexe. Ils remarquent que l'ombre est l'intersection des volumes d'ombres par rapport à tous les sommets de la source, tandis que la pénombre est l'enveloppe convexe de l'union de ces volumes d'ombre (cf. figure 1.7).

Ces volumes étaientutilisés pour accélérerle calcul d'ombres douces. Des constructions plus efficaces ont depuis été proposées [Cam91, YKSC98].

4.2 Maillage de discontinuité

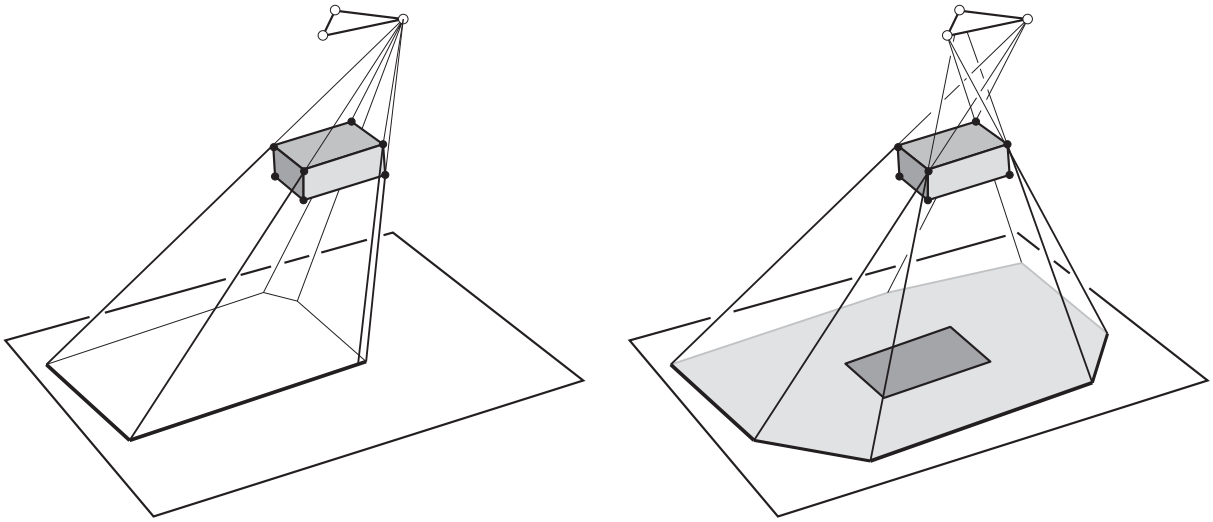
Campbell et Fussell [CF90] ont été les premiers à subdiviser un maillage le long des limites d'ombres. Ils approximent une source étendue en l'échantillonnant avec des points et utilisent ensuite des BSPs.

Heckbert [Hec92b, Hec92a] a introduit la notion de maillage de discontinuité. Il prouve que, le long des limites d'ombre, une discontinuité C^2 se produit dans la fonction d'éclairement. Des discontinuités C^1 peuvent également se produire pour des configurations dégénérées (mais courantes). Il considère les discontinuités engendrées par l'interaction d'un sommet et d'une arête qui appartiennent à la source primaire et à un bloqueur. On les appelle discontinuités EV (*edge-vertex* en anglais).

Des approches semblables peuvent être trouvées dans [LTG93, LTG92, Stu94]. Hardt et Teller [HT96] considère également certaines discontinuités dues à l'éclairage indirect.

4.3 Maillage de discontinuiité complet avec projections arrières

Teller [Tel92a], Drettakis et Fiume [DF94, DS96] et Stewartet Ghali [SG94] ont étendu le maillage de discontinuité pour traiter toutes les discontinuités C^2 possibles. Cela rend nécessaire le traitement de limites d'ombre engendrées par l'interaction de trois arêtes, également appelées EEE (cf. figure 1.8). Contrairement aux limites EV, les EEE ne sont pas des segments de droites mais des courbes coniques.



(a) (b)

FIG. 1.7: Limites d'ombre et de pénombre pour un bloqueur convexe (a) Volume d'ombre depuis l'un des sommets de la source. (b) L'ombre est l'intersection des volumes d'ombre, tandis que la pénombre est leur enveloppe convexe.

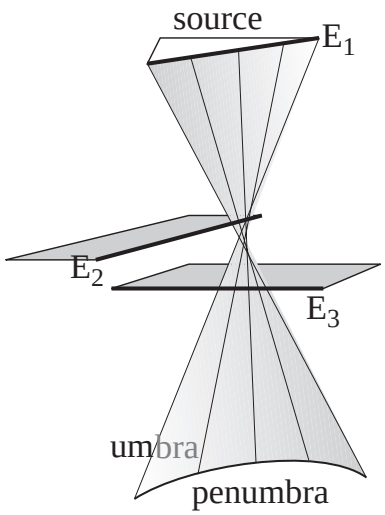


FIG. 1.8: Limite d'ombre EEE. L'interaction entre E_1, E_2 et $E3$ détermine la limite entre les régions d'ombre etde pénombre.

Traiter toutes les discontinuités permet également une classification plus précise de la région de pénombre. En effet, dans chaque région limitée par des événements visuels, la partie visible de la source est qualitativement invariante. La *projection arrière* (*backprojection*)

[DF94, DS96, SG94] stocke cette structure de la partie visible de la source. Elle permet des calculs d'éclairements rapides et précis.

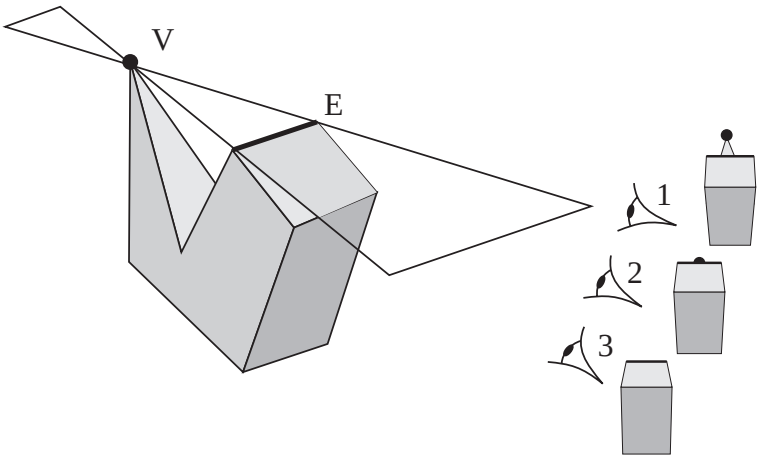


FIG. 1.9: Événement visuel EV. Au point de vue 2, le sommet V devient visible ou caché.

5 Le graphe d'aspect

Le *graphe d'aspect* a été développé en vision par ordinateur pour la reconnaissance à base de modèles [Kv76, Kv79, EBD92]. Il s'agit d'une représentation centrée sur l'utilisateur, ce qui signifie qu'elle représente toutes les vues possibles d'un objet plutôt que sa structure tridimensionnelle. Le principe est de partitionner l'espace des points de vue en régions où la vue est qualitativement invariante.

Si une projection orthographique est utilisée, une vue est déterminée par la direction de projection : l'espace des points de vue estla sphère des directions S^2. Dans le cas de la projection perspective, le point de vue peut être n'importe quel point 3D, l'espace des points de vue est donc $\mathbb{R}^3$.

Les changements de visibilité sont appelés *événements visuels*. Considérons le cas d'un objet polygonal. L'interaction d'une arête et d'un sommet est un exemple d'événement visuel ; un polyèdre A caché derrière un polyèdre B deviendra visible lorsqu'un sommet de A deviendra visuellement superposé à une arête silhouette de B (cf. figure 1.9). De la même façon, la conjonction de deux arêtes peut en cacher une troisième, ce qui impose de traiter des événements EEE. Les algorithmes de calcul de graphe d'aspect pour des polyèdres comptent par exemple [GM90, GCS91, PD90].

Ces événements sont les mêmes que ceux étudiés dans la littérature de maillage de discontinuité. Cela est intuitif car ils décrivent où la visibilité de la source change (les événements visuel ont été développés tout d'abord pour les graphes d'aspect etensuite adaptés au contexte des limites d'ombres).

Les objets courbes peuvent également être traités. Les événements visuels sont décrits par la théorie des singularités [Ker81, Rie92, Rie93, PPK92, Pet92].

Des méthodes semblables aux graphes d'aspect ont été utilisées pour la localisation de robot [GMR95, SON96, TA96] et pour les poursuites-évasions à base de visibilité [LLG+97, GLL+97, GLLL98].

6 Espace des droites

La visibilité peut être élégament exprimée en terme de droites ou de rayons. Nous présentons des approches qui effectuent des calculs directement dans l'espace des droites ou des rayons.

6.1 Approches discrètes

La classification de rayons [AK87] a été développée pour accélérer le lancer de rayons. L'espace des rayons est défini comme un espace à cinq dimensions : trois dimensions pour l'origine d'un rayon et deux pour sa direction. L'espace des rayons est subdivisé adaptivement. Un intervalle dans l'espace des rayons définit un faisceau dans l'espace objet (cf. figure 1.10 pour un équivalent 2D). Les objets qui intersectent potentiellement les rayons d'un intervalle sont ceux qui intersectent le faisceau. Lorsqu'un rayon est lancé, son intersection n'est calculée qu'avec les objets correspondant à son intervalle dans l'espace des rayons.

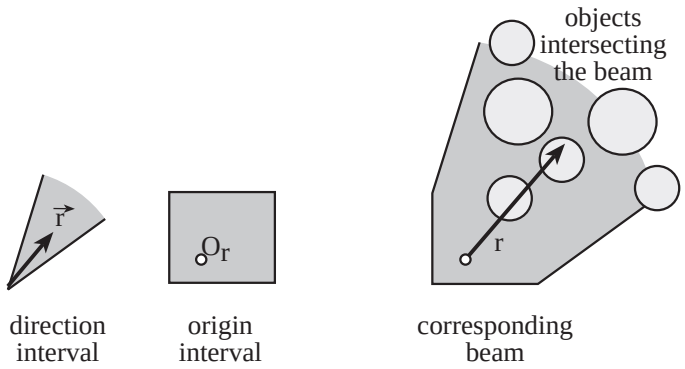


FIG. 1.10: 'Equivalent 2D de la classification de rayons. Le rayon r est défini par son origine O_r et sa direction $\vec{r}$. Il n'est testé qu'avec les objets qui intersectent le faisceau défini par l'intervalle contenant O_r et $\vec{r}$.

D'autres approches à base de discrétisation de l'espace des rayons ou des droites peuvent être trouvées dans [LMW90, CCOL98, WBP98].

6.2 Espace de Plücker

La paramétrisation de Plücker est une dualité puissante qui associe les droites à des points d'un espace à 5 dimensions. Notons que les droites de l'espace ne nécessitent que 4 dimensions, mais aucune paramétrisation 4D n'est possible sans singularité. L'ensemble des droites qui intersectent une droite donnée est un hyperplan, ce qui permet des calculs efficaces. Cependant, tous les points de l'espace de Plücker ne correspondent pas à une droite réelle. Les résultats obtenus dans l'espace de Plücker doivent être intersectés avec l'hypersurface de Plücker qui est le lieu 4D des droites réelles.

Les coordonnées de Plücker ont été utilisées en géométrie algorithmique, par exemple pour calculer les droites qui traversent un ensemble de polygones ou pour du lancer de rayons [Pel93, PS92, CEG+96]. Un tour d'horizon est disponible dans [Pel97b].

Teller [Tel92a] a développé et implémenté un algorithme qui calcule la partie de l'espace visible à travers une séquence d'ouvertures polygonales. Il construit un polytope dans l'espace 5D défini par le dual des arêtes des ouvertures et il intersecte ce polytope avec l'hypersurface de Plücker. Il obtient l'ensemble des événements EV et EEE qui bornent la région visible. Ils note cependant plus tard [TH93] que cette méthode souffre de problème numériques et de dégénérescences.

Teller et Hanrahan [TH93] utilisent également les coordonnées de Plücker pour classifier les objets comme visibles, partiellement visibles ou occultés. Ces calculs sont plus robustes parce qu'aucune construction n'est effectuée, seuls des prédicats sont évalués.

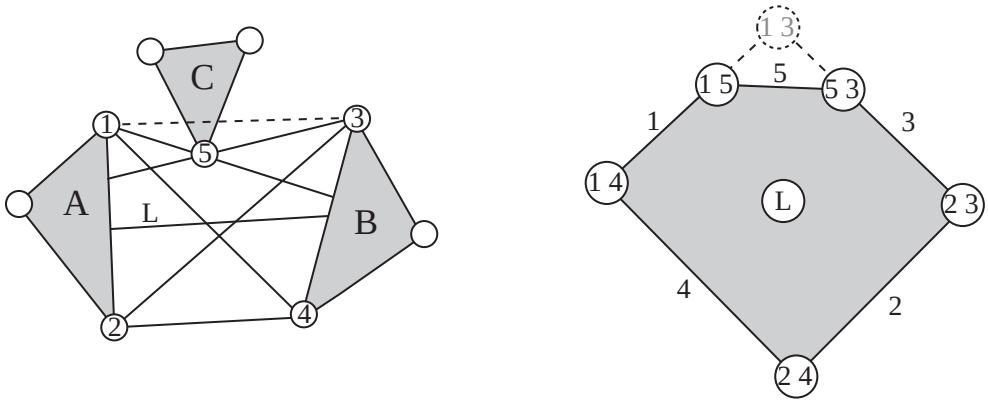
6.3 L'asp

L'*asp* a été développé par Plantinga et Dyer [PD90] comme structure intermédiaire pour la construction du graphe d'aspect pour des polyèdres. Ils définissent l'*asp* d'un polygone comme l'ensemble des droites qui l'intersectent. Ils notent que l'occultation correspond à la soustraction pour l'*asp* : si un polygone *A* occulte un polygone *B*, l'*asp* de *A* doit être soustrait de l'*asp* de *B*.

Deux définitions de l'*asp* existent. Dans le cas orthographique, l'espace 4D des droites est considéré, tandis que si la projection perspective est utilisée, l'espace 5D des rayons est considéré.

Les faces de dimension 1 de la décomposition cellulaire dans le cas orthographique et les faces de dimension 2 dans le cas perspective correspondent aux événements visuels requis pour la construction du graphe d'aspect. Aucune implémentation complète n'est reportée.

L'utilisation de l'*asp* pour la maintenance de vue a également été proposée [PDS90, Pla93], mais l'implémentation décrite est limitée à la rotation autour d'un seul axe.



(a) (b)

FIG. 1.11: Complexe de visibilité 2D. (a) Scène. (b) Représentation dans un espace dual de la face du complexe de visibilité 2D correspondant aux segments entre *A* et *B*.

6.4 Le complexe de visibilité 2D

Pocchiola et Vegter [PV96b, PV96a] ont proposé le *complexe de visibilité* pour les scènes 2D. Il est basé sur la notion de *segments libres maximaux* qui sont des segments dont les extrémités se trouvent sur les objets de la scène et qui n'intersectent l'intérieur d'aucun objet. Ils peuvent être vus comme des rayons qui "voient" dans les deux directions. Le complexe de visibilité est la partition des segments libres maximaux selon les objets à leurs extrémités. L'idée intuitive est de regrouper tous les rayons qui voient les mêmes objets.

Considérons l'exemple de la figure 1.11(a) qui montre une scène composée de trois triangles. Une dualité est utilisée qui associe les droites à des points (dans notre exemple, une droite $y = ax + b$ est associée au point (a,b)). Une face du complexe de visibilité 2D correspond à l'ensemble des segments qui ont deux objets donnés à leurs extrémités (parexemple les triangles *A* et *B* dans la figure 1.11(b)). Une telle face est bornée par des arêtes qui correspondent aux segments passant à travers un sommet de polygone (ou tangents à un objet dans le cas d'objets courbes). Ces arêtes sont adjacentes à des sommets qui correspondent aux droites passant par deux sommets de polygones.

Des algorithmes de construction optimaux ont été développés pour les objest courbes [PV96b, PV96a] et pour les polygones [Riv95, Riv97a]. Le complexe de visibilité 2D peut être utilisé pour le calcul et pour la maintenance de vues [Riv97c], ou pour la simulation de l'éclairage global en 2D [Ort97, ORDP96, DORP96].

7 Scènes dynamiques

Si les objets de la scène se déplacent continuement, l'information de visibilité varie de manière cohérente. C'est ce qu'on appelle la *cohérence temporelle*. La visibilité est localement invariante. Nous passons ici en revue les méthodes qui traitent cette cohérence.

7.1 Tubes et volumes de déplacement

Un *volume de déplacement* est semblable à un tube : il s'agit du volume de l'espace balayé par un objet en déplacement dans un intervalle de temps donné. Les volumes de déplacement ont été utilisés pour limiter les recalculs lors de la mise à jour de solutions de radiosité [BWCG86, Sha97], pour les balades virtuelles [SG96] et pour le placement de capteurs en robotique [AA95].

Shaw [Sha97] et Drettakis et Sillion [DS97] détectent les transferts lumineux qui requièrent un recalcul en testant les objets en mouvement par rapport aux tubes définis par les paires d'objets en interaction lumineuse.

7.2 Arbres BSP

Chrysanthou et Slater [CS92, CS95, CS97] mettent à jour des arbres BSP en supprimant les objets dynamiques, puis en le ré-insérant. Ils appliquent surtout leur technique au calcul d'ombres.

7.3 Lancer de rayon 4D

Des approches [Gla88, MDC93, BS96, Qua96, GP91] ont proposé d'accélérer le lancer de rayon pour des scènes animées où le mouvement des objets est connu à l'avance. Le temps est considéré comme une quatrième dimension et les intersection rayon-objet sont valides pour un intervalle de temps donné.

7.4 Graphe d'aspect et maillage de discontinuité

Eggert *et al.* [EB93] proposent une extension des graphes d'aspect pour les objets avecdes parties en mouvement. Ils notent que le graphe d'aspect change lorsqu'un événement visuel temporel a lieu, c'est à dire lorsque l'ensemble des événements visuels de la scène est modifié à cause d'une *configuration accidentelle* de l'objet. Leur analyse est toutefois restreinte à un cas simple et aucune implémentation n'est reportée.

Loscos et Drettakis [LD97] et Worall *et al.* [WWP95, WHP98] tirent partie de la cohérence pour la mise à jour de maillages de discontinuité. Ils notent que les limites de pénombre et d'ombre se déplacent continuement excepté à certains événements où une nouvelle ombre est projetée sur un objet ou si une ombre se déplace en dehors d'un objet. Le traitement de ces événements pour les événements *EV* est décrit.

8 Discussion

Malgré les nombreux travaux consacrés à la visibilité, certaines questions n'ont pas encore reçu de réponse satisfaisante. Nous avons identifié six problèmes majeurs qui entravent le traitement de la visibilité. Il s'agit de la compréhension des propriétés de visibilité, du manque d'outils génériques, des problèmes de visibilité par rapport à des volumes, du passage à l'échelle, de la robustesse et du traitement des scènes en mouvement. Nous développons ces problèmes ci-dessous à la lumière des travaux antérieurs.

– La visibilité 3D est mal comprise, la plupart des travaux théoriques sont restreints au cas 2D. Les travaux dans l'espace de Plücker, les graphes d'aspect et l'asp constituent des premiers pas remarquables pour la caractérisation des propriétés globales de visibilité dans une scène 3D. Malheureusement, le graphe d'aspect semble restreint à l'énumération de toutes les vues possibles – il n'est par exemple pas approprié pour l'extraction d'information de visibilité mutuelle – et sa consommation mémoire est énorme. L'asp pour la projection orthographique ne contient pas l'information concernant la visibilité à l'intérieur de la scène, tandis que la version perspective présente de la redondance puisque la visibilité ne varie en général pas le long de rayons colinéaires. La cohérence de visibilité n'est de plus pas véritablement explicite dans l'asp. Les coordonnées de Plücker nécessitent 5 dimensions pour paramétrer un espace à 4 dimensions. De plus, les occultations ne sont pas vraiment prises en compte puisque les méthodes que nous avons passées en revue considèrent toutes les intersections d'une droite.

– Bien que de puissantes techniques aient été développées pour des cas spécifiques, aucun outil générique n'a été proposé. Quelques travaux (avant tout théoriques) ont montré ques les techniques dans l'espace de Plücker ont des applications pour différents problèmes comme le lancer de rayon, la visibilité mutuelle ou le "poignardage" (*stabbing*). Mais aucune implémentation n'a été présentée excepté le travail de Teller sur l'antipénombre [Tel92a] et la classification de bloqueurs [TH93].

– Si la visibilité par rapport à un point a été beaucoup traitée, l'extension aux volumes et aux surfaces est plus problématique. Tandis que les techniques d'élimination d'occultation par rapport à un point traitent la fusion de bloqueur dans des environnements génériques, les méthodes à base de cellules sont restreintes aux environnements architecturaux ou aux occultations dues à un seul bloqueur convexe.

– Le coût élevé des calculs de visibilité fait de la question du passage à l'échelle un point critique. Une croissance quadratique, ou même linéaire n'est pas acceptable quand on considère l'explosion de la taille des scènes 3D. Le recours à des calculs approximatifs semble inévitable, ce qui pose la question cruciale

du contrôle de l'erreur. Des idées ont été proposées [SD95, SS96b, Sol98], mais le sujet n'en est qu'à ses balbutiements.

– Les techniques exactes de visibilité comme les graphes d'aspect ou les maillage de discontinuité requièrent des construction géométriques complexes sujettes aux problèmes de robustesse. L'une des principales question est le traitement des événements visuels, puisqu'ils définissent des surfaces qui peuvent être des quadriques réglées qui doivent être intersectées avec la scène. Les problèmes de robustesse et de précision numériques ne sont pas limités au seul domaine de la visibilité, ils concernent tous les calculs géométriques.
– La cohérence temporelle a reçu peu d'attention, surtout si l'on songe à ses bénéfices potentiels. Si tout le monde s'accorde à penser que des calculs pourraient être économisés, les méthodes qui ont été proposées n'ont pas véritablement concrétisé cet espoir. Elles consistent avant tout à localiser une portion de l'espace où les calculs doivent être mis à jour dans le cas du mouvement d'un objet, ou alors à utiliser le résultat de l'image précédente comme point de départ pour le calcul courant.

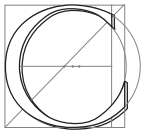
Lors de cette thèse, nous nous sommes attaqués à ces problèmes, en nous concentrant sur les trois premiers, bien que les autres point aient également été explorés à un degré moindre. Le chapitre 2 propose une interprétation des propriétés de visibilité d'une scène 3D. Une structure de données générique qui représente toute l'information globale de visibilité est présentée au chapitre 3 et appliquée à la simulation de l'éclairage au chapitre 4. Le chapitre 5 présente une méthode efficace pour effectuer de l'élimination d'occultation par rapport à un volume tout en prenant en compte la fusion de bloqueurs.

CHAPITRE 2

Le complexe de visibilité 3D

L'air est plein d'une infinité de lignes rayonnantes qui se coupent et se croisent sans se chasser l'une l'autre, et qui reproduisent sur tout ce qu'elles rencontrent la vraie forme de leur cause.

Léonard DE VINCI, *Codex Urbinas*



CE CHAPITRE propose une étude de la visibilité dans l'espace des droites. Les propriétés de visibilité les plus atomiques – visibilité mutuelle de deux points, lancer de rayon– s'expriment naturellement dans l'espace des droites. Nous étudions l'interprétation de l'intersection entre rayons et objets dans un espace dual approprié. Contrairement aux approches habituelles d'accélération du lancer de rayon, notre but n'est pas d'étudier la façon la plus rapide d'effectuer chaque requête, mais d'appréhender globalement les incidences entre une scène et l'ensemble des rayons, afin de caractériser la *cohérence* de visibilité.

Notre approche est de regrouper les rayons qui "voient" les mêmes objets. Cette définition naturelle de la cohérence pose la question des frontières de tels ensembles cohérents. Leur étude se révèle fructueuse pour la compréhension de bien des propriétés de visibilité. Nous attacherons une grande attention à la différence entre la visibilité selon les droites et selon les segments : doit-on prendre en compte toutes les intersections d'une droite, ou considérons-nous seulement le premier objet coupé par un rayon.

Cette étude aboutit à la définition d'une nouvelle structure, que nous appelons le *complexe de visibilité 3D*, qui décrit toute l'information de visibilité d'une scène tridimensionnelle. Cette approches est inspirée par l'équivalent bidimensionnel [PV96b, PV96a, DP95b], même si notre nouvelle structure a été entièrement développée avec, à l'esprit, les spécificités des problèmes tridimensionnels. Elle sera à la base du développement d'un outil pratique dans le chapitre suivant, qui sera appliqué à la simulation de l'éclairage dans le chapitre 4.

Ce chapitre regroupe des résultats publiés dans deux articles [DDP96, DDP97b]. Une discussion et une comparaison plus détaillées avec les approches antérieures ont été ajoutées (section 6). Le catalogue des adjacences entre faces a été inclus dans l'annexe A, ainsi qu'une description des événements visuels pour les objets concaves. Surtout, une étude probabiliste de la complexité pour les scènes "normales" est présentée section 2.3. Une extension du complexe aux scènes en mouvement est décrite à la section 3.

Nous présentons tout d'abord les intuitions qui mènent au développement de notre structure. La définition est ensuite formalisée pour des scènes composées de polygones et d'objets courbes. Nous effectuons ensuite

une analyse probabiliste de sa complexité, puis nous étendons notre étude au cas d'objets en mouvement. Un algorithme de construction dont la complexité dépend du résultat est présenté, ainsi que l'interprétation de requêtes de visibilité.

1 Introduction au complexe de visibilité 3D

Dans cette discussion, nous considérons tout d'abord des scènes composées d'objets convexes lisses disjoints. Nous étendrons ensuite notre approche au cas des objets polyédriques. Nous définissons la visibilité en terme d'intersection objet-rayon. Si nous considérons les objets transparents, un rayon n'est pas bloqué, et tous les objets rencontrés par une droite doivent être pris en compte. Cependant, si nous voulons prendre en compte les occultations, seul le premier objet coupé par un rayon est pertinent. Toutefois, les rayons entraînent une redondance de l'information, puisque beaucoup de rayons colinéaires "voient" le même objet : considérons un rayon r qui a pour origine A et dont la première intersection avec la scène est un point B. Tous les rayons colinéaires à r dont l'orgine est entre A et B "voient" le même point B, nous pouvons les regrouper en un segment.

Nous considérerons des *segments libres maximaux*, qui sont des segments qui n'ont pas d'intersection avec l'intérieur des objets et dont la longueur est maximale (leurs deux extrémités sont sur la frontière des objets ou à l'infini). Dans ce qui suit, nous les appellerons le plus souvent *segments*. Les segments peuvent êtres interprétés comme des rayons qui *voient* les deux objets à leurs extrémités. Une ligne droite peut être colinéaire à plusieurs segments, séparés par les objets qui l'intersectent. Dans ce chapitre, nous introduirons les concepts tout d'abord en termes de visibilité selon les droites (où tous les objets intersectés par une droite sont pris en compte), puis en terme de visibilité selon les segments (où les occultations sont traitées).

Nous voulons regrouper les segments (ou les droites) qui voient les mêmes objets. Nous avons donc besoin d'une partition de l'ensemble des rayons en composantes connexes selon les objets qu'ils voient. Comme les ensembles de segments ne sont pas des entités faciles à appréhender, nous proposons de les représenter dans un espace dual qui permettra une meilleure compréhension et une illustration plus simple.

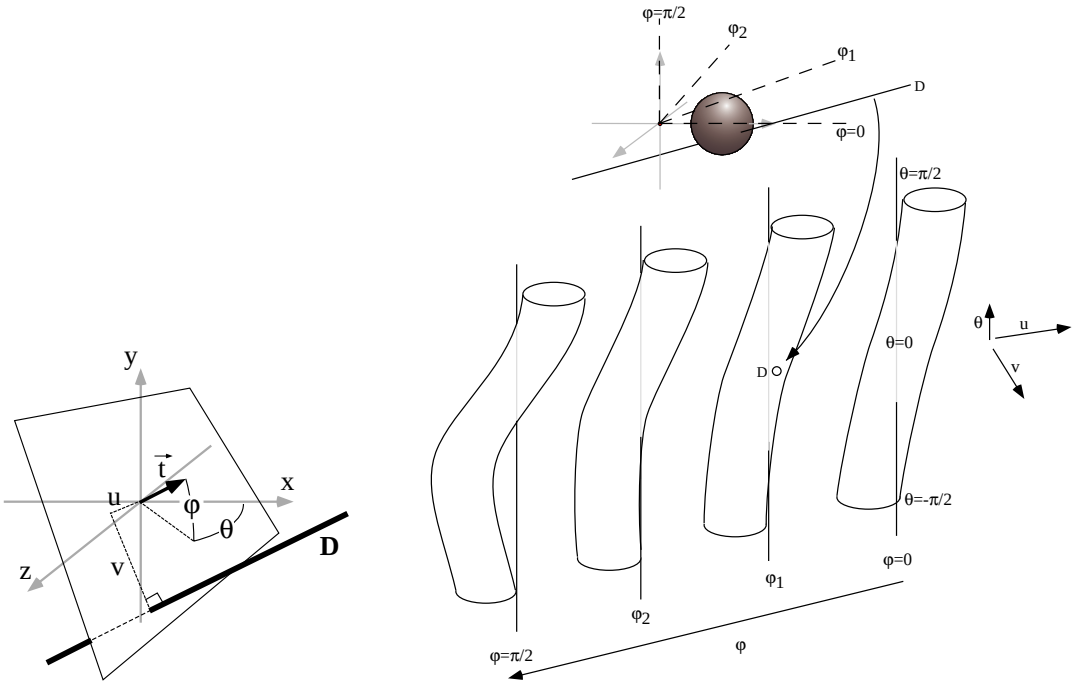
1.1 Dualité

Nous avons choisi de décomposer l'espace 4D des droites en deux composantes de direction (les coordonnées sphériques (θ, φ) du vecteur directeur) et une projection (u, v) sur le plan orthogonal à la droite passant par l'origine. Les axes de ce plan sont choisis tels que u soit selon $\vec{t} \wedge \vec{y}$[1]. L'intersection de la droite avec deux plans parallèles pourrait également être utilisée. Nous pensons toutefois que cette dernière approche rend plus difficile l'interprétation de droites ayant une même coordonnée. Nous verrons que certaines propriétés ne pourraient pas être décrites avec cette dualité.

Visualiser un espace 4D est très difficile. On peut le voir comme un espace 3D en mouvement où la quatrième dimension est le temps. Une approche est d'utiliser des tranches. Dans ce chapitre nous fixerons $\varphi = \text{cst}$, ce qui peut être vu comme des images au cours du temps. Une telle tranche sera appelée une φ-tranche. Comme une telle tranche est un espace 3D (θ, u, v), ils sera parfois utile de trancher une fois de plus, et de considérer φ et θ constants. Nous obtiendrons une tranche 2D, où u et v varient, composée de droites parallèles qui ont pour direction (θ, φ). Une telle tranche sera appelée une $\theta\varphi$-tranche. Ces tranches 2D sont plus faciles à appréhender car elles peuvent être interprétées comme des vues orthographiques de la scène.

Nous insistons sur le fait que cette dualité sera utilisée à des fins d'illustration avant tout. Les concepts que nous présentons sont indépendants de la paramétrisation des droites. Ils pourraient être présentés en utilisant uniquement des notions topologiques, bien que nous pensions qu'un espace dual aide grandement à leur compréhension.

[1] Des singularités se produisent pour $\varphi = \pm\frac{\pi}{2}$, mais comme nous utilisons cette dualité en tant qu'outil de présentation, nous pouvons les ignorer sans perte de généralité.



(a) (b)

FIG. 2.1: (a) Dualité. (b) Volume de tangence d'une sphère. L'axe θ ($u = 0$, $v = 0$) est tracé pour chaque φ-tranche afin d'offrir une meilleure visualisation 3D. Sur la φ-tranche de gauche, qui correspond à la singularité de notre dualité pour $\varphi = \frac{\pi}{2}$, le tube tourne simplement autour de l'axe des θ. La droite *D* intersecte l'objet, son dual est donc à l'intérieur du volume de tangence.

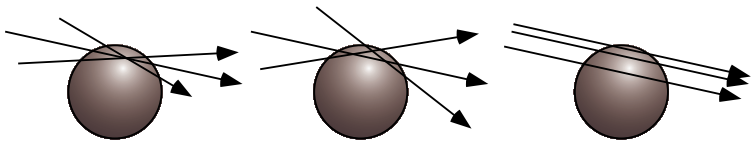


FIG. 2.2: Pour rester tangente à un objet, une droite possède trois degrés de liberté. (a) Rotation autour du point de tangence. (b) Rotation en avant ou en arrière. (c) Se décaler de côté.

1.2 Courbes de tangence

Visibilité selon les droites

La visibilité change quand une droite devient tangente à un objet. C'est-à-dire qu'une droite tangente à un objet constitue la limite entre des droites qui l'intersectent et des droites qui ne l'intersectent pas.

L'ensemble des droites tangentes à un objet est un ensemble 3D dans l'espace dual 4D. Cela signifie plus intuitivement qu'une droite a 3 degrés de liberté pour rester tangente à un objet (cf. figure 2.2). Nous appellerons le dual de l'ensemble des droites tangentes à un objet, le *volume de tangence* de cet objet.

La figure 2.1(b) est une représentation du volume de tangence d'une sphère. Pour chaque φ-tranche, le volume de tangence est une sorte de "tube" 2D, ce qui constitue un ensemble 3D dans l'espace dual 4D. Si l'on considère une θφ-tranche (tranche 2D horizontale dans la figure 2.1(b)), l'ensemble des tangentes qui partagent cette direction est un cercle. Ceci est général : de par la définition de *u* et *v*, l'ensemble des tangentes d'un objet dans une direction est la silhouette de cet objet dans cette direction.

Si une droite a son dual sur le volume de tangence, elle est tangente à l'objet. Si son dual est à l'intérieur de l'ensemble 4D borné par le volume de tangence, alors il intersecte l'objet, comme la ligne *D* de la figure 2.1(b).

Visibilité selon les segments

Intéressons-nous maintenant à la visibilité avec occultation. Une droite qui intersecte un objet est colinéaire à au moins deux segments : un devant et un derrière l'objet.

Considérons une $\theta\varphi$-tranche comme celle représentée en bas à gauche de la figure 2.3. Les ensembles de droites qui intersectent et qui n'intersectent pas l'objet sont séparés par la silhouette de l'objet. Pour la visibilité selon les segments, nous devons différencier les segments qui voient l'avant et l'arrière de l'objet. Puisque de tels segments sont colinéaires à une même droite, ils sont projetés sur le même point dans l'espace dual 4D. En conséquence, l'ensemble des segments qui voient l'arrière et ceux qui voient l'avant de l'objet ont la même position dans l'espace dual 4D, comme cela est montré sur la droite de la figure 2.3. La silhouette, qui correspond à l'ensemble des segments tangents à l'objet pour la direction (θ, φ) choisie, est incidente à trois ensembles de segments (avant, arrière et pas d'intersection). Cela signifie qu'un segment tangent à l'objet a dans son voisinage topologique des segments qui n'intersectent pas l'objet, des segments qui voient l'arrière, et des segments qui voient l'avant.

Pour différencier ces segments, nous ajoutons une pseudo-dimension. Il ne s'agit pas d'une véritable dimension continue, nous devons juste différencier les segments colinéaires. Si nous imposons $\theta = \text{cst}$, $\varphi = \text{cst}$ et $v = \text{cst}$, l'ensemble des segments peut être représenté par le graphe en bas à droite de la figure 2.3 (il s'agit en fait de la représentation d'un graphe puisque les points des arêtes ont aussi une signification). Chaque tangente correspond à un sommet du graphe. Ce graphe est une structure 1D plongée en 2D. De même, pour une $\theta\varphi$-tranche, l'ensemble des segments est représenté par une structure 2D plongée en 3D. Nous appelons la partition des segments ayant pour direction (θ, φ) selon leur visibilité le *complexe auxiliaire* pour (θ, φ) (voir aussi la figure 2.5). Le complexe auxiliaire peut être vu comme une vue orthographique généralisée où les objets visibles comme cachés sont organisés en couches.

De la même façon, une φ-tranche est une structure 3D plongée en 4D, et l'ensemble des segments est une variété[2] 4D plongée en 5D.

Le *complexe de visibilité 3D* est l'équivalent de l'arrangement dual pour la visibilité selon les segments. Il partitionne les segments en fonction des objets à leurs extrémités.

1.3 Bitangentes

Visibilité selon les droites

Considérons maintenant deux objets. Si une droite a son point dual à l'intérieur des deux volumes de tangence des objets, alors elle intersecte les deux objets. Les volumes de tangence induisent une partition de l'espace dual des droites 3D en fonction des objets qu'elles intersectent. Nous appelons cette partition l'*arrangement dual*. Ses *faces topologiques* sont des ensembles 4D de droites qui intersectent les mêmes objets. Leurs frontières sont des portions de volume de tangence qui sont 3D. L'intersection de deux volumes de tangence correspond à des droites tangentes aux deux objets (bitangentes). Une bitangente correspond à un *sommet-t* (*t-vertex*) dans une image (c'est-à-dire à l'intersection visuelle de deux contours d'objets).

Dans une φ-tranche l'ensemble des bitangentes forme une courbe dans l'espace (représentée en pointillés sur la figure 2.4 pour les deux tranches à droite). Il correspond à l'intersection de deux "tubes" qui sont les φ-tranches des volumes de tangence. La φ-tranche d'une face 4D est un volume, qui correspond à l'intersection de l'intérieur de deux tubes.

Visibilité selon les segments

Un complexe auxiliaire pour deux objets est représenté figure 2.5 pour une direction donnée. Il est toujours délimité par les silhouettes des objets, mais par exemple la silhouette de la sphère du haut n'a aucune influence sur l'ensemble B des segments qui voient l'arrière de la sphère du bas. Les deux bitangentes (en gras) sont incidentes à toutes les faces.

La figure 2.6 est une φ-tranche pour $\varphi = 0$ de toutes les faces du complexe de visibilité 3D pour la scène composée de deux sphère montrée figure 2.4. La vue dans une direction donnée est représentée à gauche des tubes, et nous considérons le complexe auxiliaire associé qui est dessiné six fois en haut du schéma. Chaque

[2] Une *n-variété* est un ensemble pour lequel le voisinage de chaque point est homéomorphique à $\mathbb{R}^n$. L'espace des segments est en fait non-variété à cause des branchements aux volumes de tangence, mais les non-variétés sont souvent appelées variétés par abus de langage.

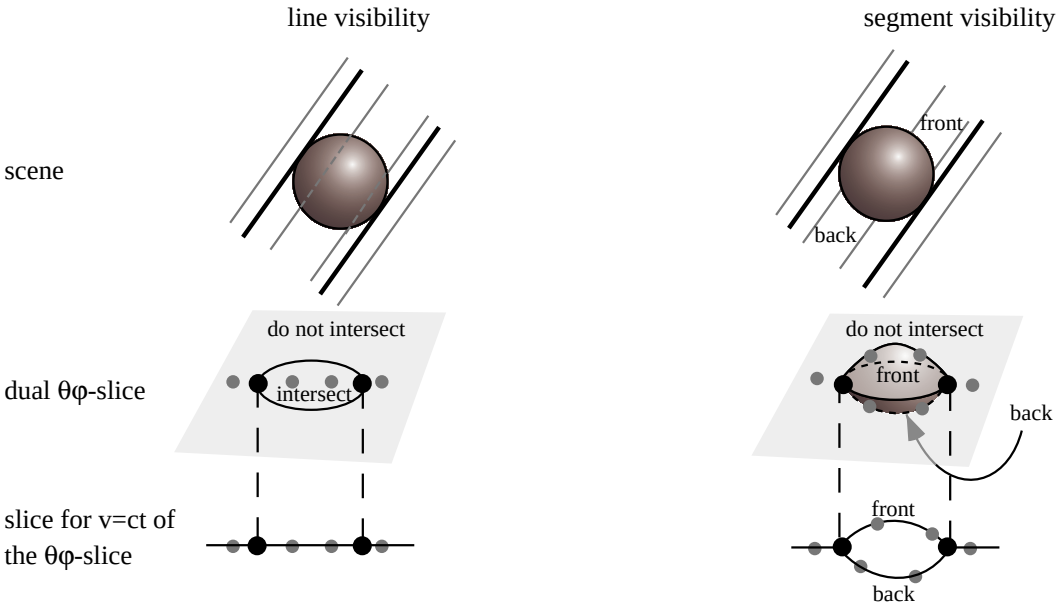


FIG. 2.3: Visibilité pour $\theta = cst$ et $\varphi = cst$. Si l'on considère les droites (à gauche), la visibilité peut être décrite par une structure planaire (en dessous). Par contre si l'on considère les segments (à droite), il y a plusieurs niveaux sur ce plan, en fonction du côté de l'objet vu. L'ensemble des segments qui n'intersectent pas l'objet, ceux qui en voient l'avant et ceux qui en voient l'arrière partagent une frontière commune : les tangentes à l'objet qui correspondent à sa silhouette. Rappelons que le complexe auxiliaire représenté en bas à droite est une variété 2D plongée en 3D, *i.e.*, il est "vide" puisque les points en dehors de la surface représentée n'ont aucune signification.

fois, une face est hachurée et un volume est représenté en dessous qui correspond à la φ-tranche d'une face du complexe de visibilité pour $\varphi = 0$. L'union de ces volumes représente plus que la tranche 3D en entier, puisqu'une φ-tranche de l'espace des segments est une 3-variété plongée en 4D qui a des branchements aux volumes de tangence.

1.4 Tritangentes

Considérons maintenant une scène avec trois objets. Une ligne tangente aux trois objets a son point dual à l'intersection de leurs trois volumes de tangence. Un ensemble connexe de tritangentes est un ensemble 1D dans l'espace dual. Dans une φ-tranche, c'est un point. L'ensemble des tritangentes peut aussi être interprété comme l'intersection de trois ensembles de bitangentes.

La figure 2.7 montre une partie du complexe de visibilité d'une scène composée de trois sphères. Sur la φ-tranche $\varphi = 0$ sont dessinées deux vues orthographiques en face de la valeur correspondante de $\theta : \theta = 0$ (vue 0) et $\theta = \theta_2$ (vue 2).

L'ensemble F des segments qui voient les deux sphères R et B est représenté pour deux tranches F_0 et $F_{\varphi 1}$. Il s'agit de l'intersection des volumes de tangence de R et B, moins le volume de tangence de G. Les tritangentes sont les points en blanc dans les φ-tranches. À cause de l'occultation due à G, certaines droites bitangentes à R et B ne correspondent à aucun segment bitangent. Ceci est illustré figure 2.8 qui est un agrandissement de la φ-tranche $\varphi = 0$. L'ensemble des bitangentes B_0 est coupé car des droites bitangentes telles que D coupent G et ne correspondent à aucun segment bitangent. On peut voir que les tritangentes T_0 et T_0' sont les intersections des φ-tranches des trois volumes de tangence, et sont aussi incidentes aux trois ensembles de bitangentes B_0, B_0' et B_0''.

Les tritangentes sont un exemple d'*événement visuel*. Les événements visuels décrivent les changements *qualitatifs* (topologiques) de visibilité. Considérons l'exemple de la figure 2.9. Quand le point de vue descend, la sphère A devient cachée par la conjonction de B et C. Cela se produit lorsque le point de vue se trouve sur une tritangente.

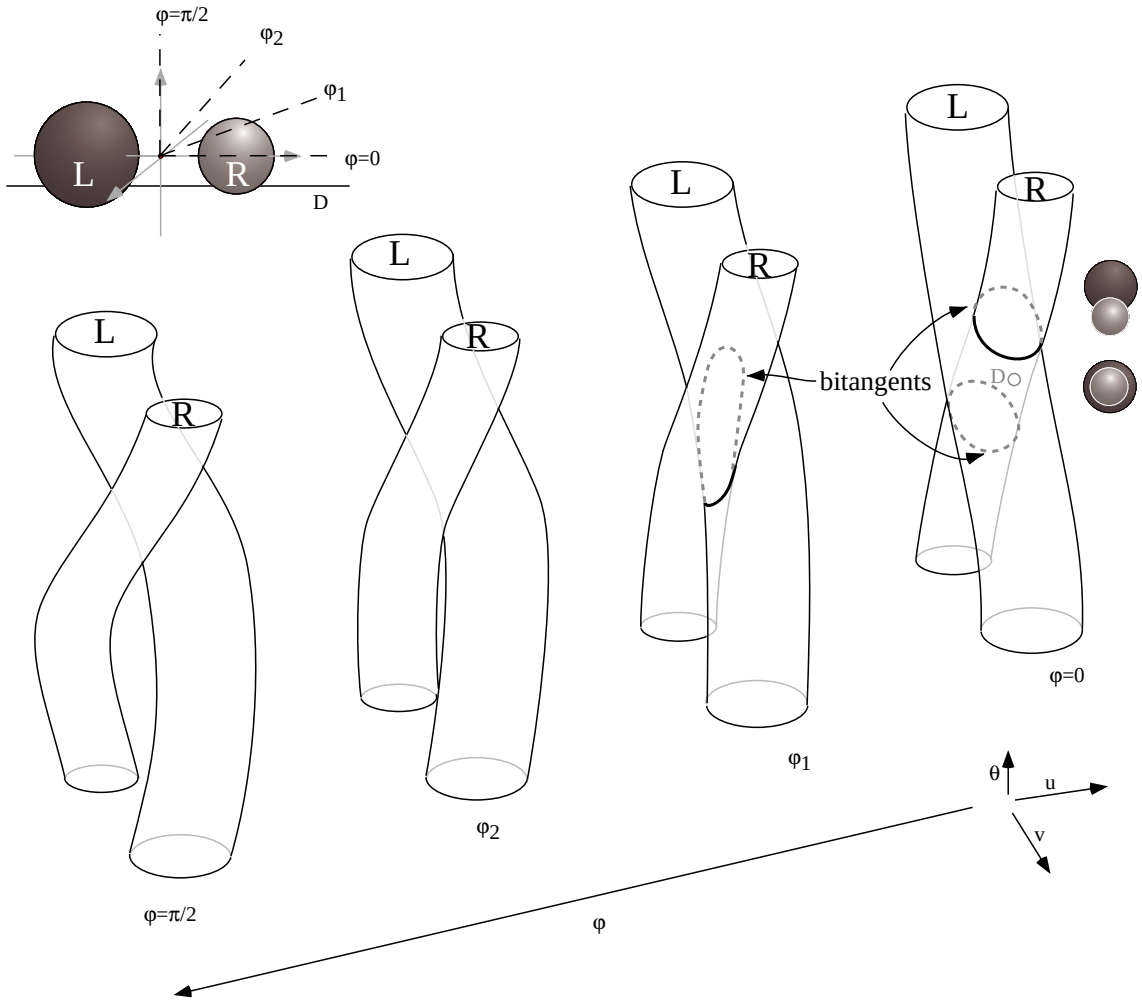


FIG. 2.4: Arrangement dual pour deux sphères.

Dans le cas général, une scène peut ne comporter aucune tritangente (c'est par exemple le cas figure 2.12(a)).

1.5 Croisement de tangence

Il existe une autre sorte d'événement visuel, nommé *croisement de tangence*. Il correspond à un plan tangent à deux objets, et à une ligne qui passe par les deux points de tangence correspondants. Considérons le cas d'une sphère cachée derrière une seconde sphère. Lorsque le point de vue se déplace, la sphère deviend visible quand les deux sphères sont visuellement tangentes. La droite qui passe par le point de vue et les points de tangence visuelle est contenue dans un plan bitangent.

Comment cela s'interprète-t-il dans notre espace dual ? Ces croisements de tangence sont bien entendu des sous-ensembles des bitangentes. Ils correspondent à des *extrema* par rapport à θ dans les φ-tranches. Pour comprendre pourquoi, il faut se rappeler qu'une θφ-tranche du complexe de visibilité correspond à une vue orthographique. Considérons une rotation à φ constant qui commence depuis une direction où les deux objets sont visuellement distincts. Des bitangentes (des sommets-t ou *t-vertices*) aparaîtront dans la vue orthographique lors d'un croisement de tangence. Les croisements de tangence correspondent donc à la première (ou à la dernière) valeur de θ pour laquelle les bitangentes existent.

Le même raisonnement prouve que les croisements de tangence sont aussi des extrema dans les θ-tranches. Bien entendu, si l'on considère la visibilité selon les segments, un croisement de tangence peut être supprimé à cause de l'occultation due à un troisième objet. Le minimum ou le maximum d'un ensemble de bitangentes

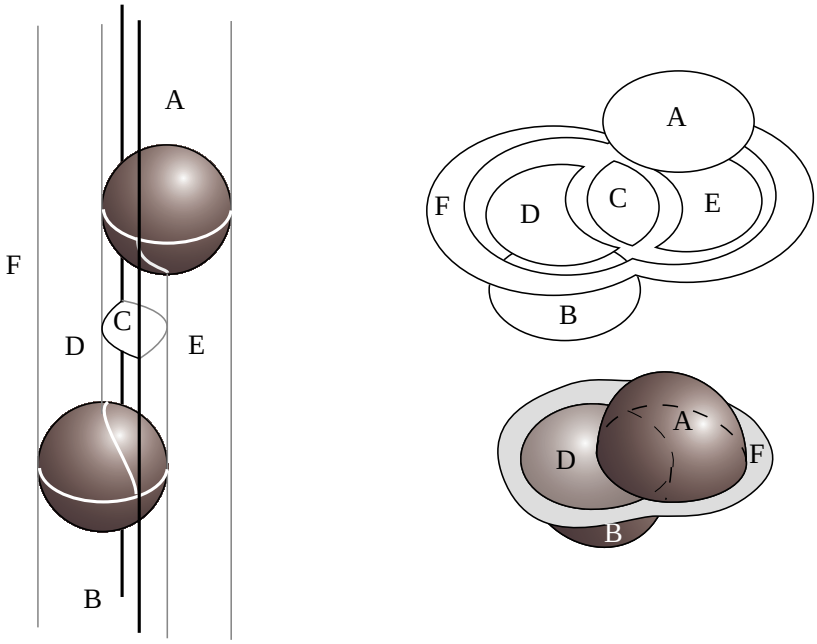


FIG. 2.5: Complexe auxiliaire pour deux sphères. Rappelons que le complexe auxiliaire est une structure 2D plongée en 3D. Dans la représentation du bas, seuls les point de la surface correspondent à des segments. Dans celle du haut, les faces du complexe auxiliaire ont été éclatées pour rendre leurs incidences plus simples à comprendre. La face F est infinie.

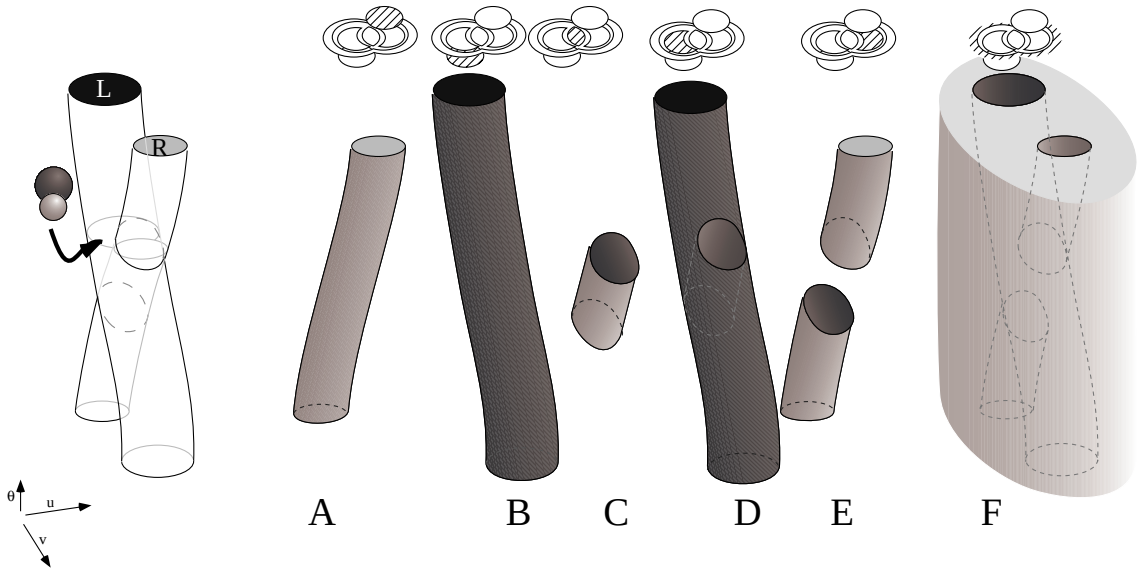


FIG. 2.6: φ-tranche pour $\varphi = 0$ des faces du complexe de visibilité pour la scène précédente. A est l'ensemble des segments qui voient l'avant de R. B est l'ensemble des segments qui voient l'arrière de L. C est l'ensemble des segments entre L et R. Il peut être interprété comme l'intersection de l'ensemble des droites qui voient L et de l'ensemble des droites qui voient R. Dans l'espace dual il a la forme de $A \cap B$. D est l'ensemble des segments qui voient l'avant de L. À cause de l'occultation due à R dans cette direction, D a la forme de $B - A$. De même, E est l'ensemble des segments qui voient l'arrière de R. Pour finir, F est l'ensemble des segments qui ne voient aucune des deux sphères. Il a la forme du complémentaire de $A \cup B$.

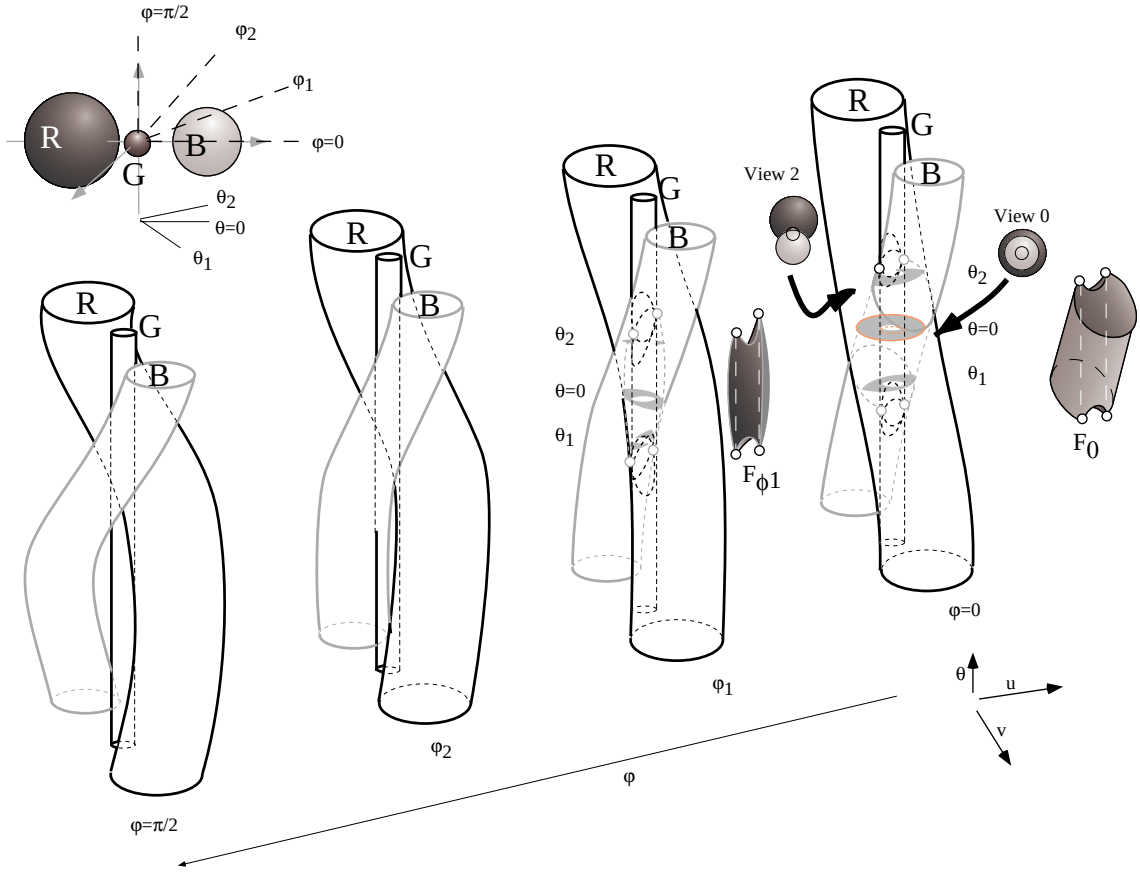


FIG. 2.7: Complexe de visibilité d'une scène composée de trois sphères.

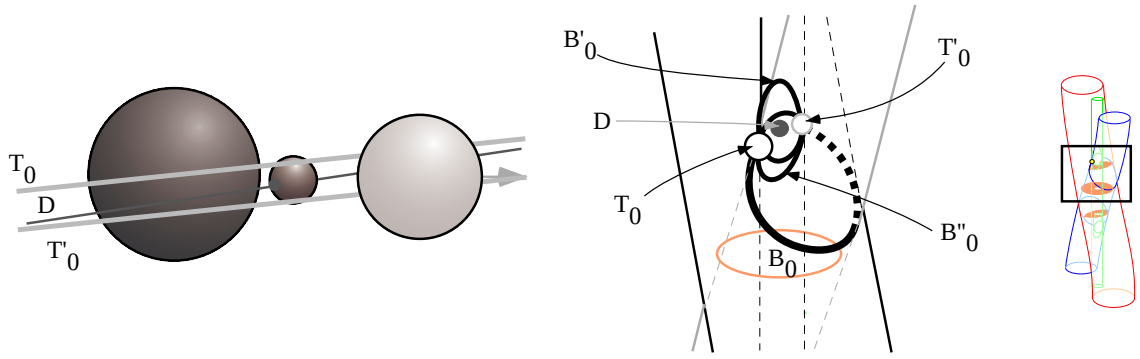


FIG. 2.8: Agrandissement de la φ-tranche $\varphi = 0$.

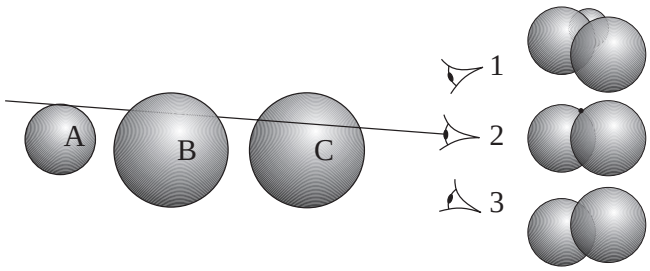
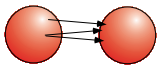
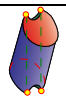
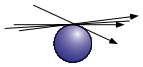
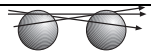
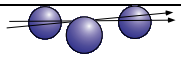
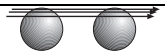
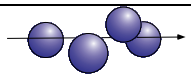


FIG. 2.9: Événement visuel de tritangence. Au point de vue 2, la sphère *A* devient cachée par les sphères *B* et *C*. Cela se produit lorsque le point de vue est sur un segment tangent aux trois sphères.

Dimension	Configuration	φ-tranche dans l'espace dual	Nom
4			face
3			face de tangence
2			face de bitangence
1			arête de tritangence
1			croisement de tangence
0			sommet

TAB. 2.1: Éléments du complexe de visibilité.

est alors une autre sorte d'événement visuel, comme c'est le cas dans la figure 2.8 où la tritangente T_0' est le minimum de l'ensemble B_0.

Les croisements de tangence, comme tous les événements visuels, ont dimension 1. Ces événements sont très importants pour la maintenance de vue, pour les graphes d'aspect, ou pour les maillages de discontinuité.

1.6 Le complexe de visibilité 3D

Nous avons défini l'arrangement dual comme la partition des droites de l'espace en composantes connexes en fonction des objets qu'elles intersectent. Il s'agit d'une structure 4D.

Le *complexe de visibilité 3D* est la partition des segments libres maximaux de l'espace en composantes connexes en fonction des objets qu'ils touchent. Il s'agit d'une structure 4D plongée en 5D. Les dimensions et adjacences des frontières de ses faces sont résumées table 1.6.

Les éléments du complexe de visibilité et ceux de l'arrangement dual ne sont pas les mêmes. Une droite peut être tangente à deux objets et ne correspondre à aucun segment bitangent à cause des occultations.

Dans le cas général, une scène peut avoir un complexe de visibilité sans sommet ni arête de tritangence.

2 Une définition pour les scènes composées de polyèdres et d'objets courbes

Nous nous intéressons maintenant aux scènes composées de polyèdres et d'objets algébriques lisses et convexes. Les objets concaves et lisses par morceaux seront discutés dans l'annexe A. Nous considérons des objets algébriques de degré borné (typiquement 3 pour les splines bicubiques). Dans ce qui suit, n représente la complexité totale de la scène, qui est le nombre total d'objets et de sommets de polygones. Nous supposons les objets en situation générique, les questions de dégénérescence ne seront pas traitées.

La nature algébrique des objets n'est nécessaire que pour dériver des bornes sur la complexité. Les concepts que nous décrivons sont valables pour tout objet lisse convexe.

2.1 Segments critiques

Un segment est dit en position générale s'il touche des objets seulement à ses extrémités. Un segment qui touche des objets en son intérieur est dit *critique*. À une telle intersection il y a un *événement local*. Si un segment touche plus d'un objet en son intérieur, nous disons qu'il y a un *événement multilocal*. Les segments critiques sont regroupés en *ensembles de segments critiques*. Nous avons vu que la dimension d'un tel ensemble est le nombre de degrés de liberté qui autorisent à conserver l'événement. On peut aussi s'intéresser à la *codimension* d'un tel ensemble, qui est le complément par rapport à la dimension de l'espace (le nombre de

degrés de liberté qui sont contraints)[3]. Les codimensions sont pratiques car elles sont additives comme nous le verrons.

Pour les scènes que nous considérons, il y a deux sortes d'événements locaux : la tangence et les sommets. L'objet tangent ou le sommet sont appelés *générateurs* de l'événement. Pour rester tangent à un objet, un segment possède trois degrés de liberté ; il a codimension 1. Le cas où un segment passe par l'arête d'un polygone est identique. Nous l'appelons un événement T pour tangence (E pour *edge* (arête) est employé quand uniquement des polygones sont considérés). Un segment qui passe par un sommet a deux degrés de libertés (rotation), et a donc codimension 2. Nous parlons d'événement V pour *vertex* (sommet).

La combinaison de plusieurs événements locaux engendre un événement multilocal, et les codimensions sont additionnées. Nous utilisons la notation $+$ pour décrire une telle combinaison. Par exemple un segment qui est tangent à un objet et qui passe par un sommet appartient à un ensemble de segments critiques $T+V$ de codimension $1+2=3$ (c'est donc un ensemble 1D).

Nous avons vu qu'à un *croisement de tangence* un segment est tangent à deux objets et appartient à un de leurs plans de tangence. Dans ce cas, le plan de tangence commun rajoute une contrainte et donc une codimension, et nous utilisons la notation $++$. Par exemple les segments critiques qui correspondent aux croisements de tangences sont notés $T++T$ et ont codimension $1+1+1=3$ (ensemble 1D)[4].

Chaque événement local correspond à une équation algébrique : une droite tangente à un objet algébrique ou passant par un sommet. Un ensemble de segments critiques peut donc être associé à l'ensemble des droites qui vérifient l'ensemble d'équations correspondant.

Les événements engendrés par les faces des polygones sont considérés comme des événements $T+T$ puisqu'ils correspondent à des segments passant par deux arêtes de la face. De la même façon, un segment passant par une arête correspond à un événement $V+V$. La raison pour laquelle les événement V ne sont pas vus comme des $T+T$ est qu'ils introduisent des discontinuités qui requièrent un traitement spécifique comme nous le verrons à la section 4.4. Il s'agit en fait de dégénérescence par rapport à des objets lisses.

2.2 Le complexe de visibilité pour des scènes de polyèdres et d'objets lisses

Le *complexe de visibilité 3D* est la partition des segments libres maximaux de l'espace en composantes connexes en fonction des objets qu'ils touchent. Ses faces de dimension 4 sont les composantes connexes maximales de segments en position générale qui ont les mêmes objets à leurs extrémités.

Les différentes faces du complexe correspondent aux segments critiques, comme c'est résumé à la table 2.2.

Théorème 1 *La taille du complexe de visibilité 3D est $\Omega(n)$ et $O(n^4)$ où n est la complexité de la scène.*

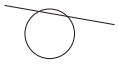
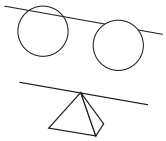
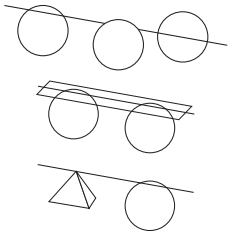
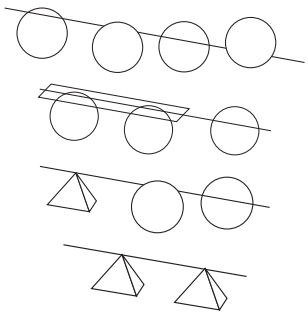
Preuve

Le nombre de faces de dimension $(k+1)$ adjacentes à une k-face est borné. Par exemple une 1-face $T_1+T_2+T_3$ est adjacente à cinq 2-faces : deux faces T_1+T_2 (il y a deux faces différentes car l'une des extrémités des segments peut se trouver sur l'objet tangent en T_3 ou pas, cf. figure 2.10), T_1+T_3 et deux T_2+T_3. Les autres adjacences sont résumées dans l'annexe A.

Chaque 4-face est adjacente à au moins une 3-face, une 3-face à au moins une 2-face et une 2-face à au moins une 1-face. Cependant, une 1-face peut n'être adjacente à aucune 0-face. Pour une face F donnée du complexe (composée de segments), nous considérons l'ensemble de droites associé S, c'est-à-dire l'ensemble des droites avec les mêmes propriétés de tangence. Une face T_1+T_2 est associée à l'ensemble de toutes les droites tangentes à T_1 et T_2, quelles que soient les occultations. Cet ensemble de droites est adjacent à un ensemble de droites S' avec une codimension de plus (l'une des droites tangentes à un objet est aussi tangente à un autre objet, l'une des droites tangentes à deux objets est contenue dans un plan bitangent, ou bien une droite passant par un sommet est également tangente à un objet). Considérons un chemin continu depuis l'une des droites associée à un segment s de F jusqu'à une droite de S'. Considérons également un chemin correspondant pour les segments (la notion de chemin correspondant est ambiguë à cause des branchements dus aux tangentes,

[3] Les codimensions que nous considérons sont différentes de celles étudiées dans la littérature des graphes d'aspect car la taille de l'espace complet y est différente. Dans le cas orthographique, l'espace des points de vue a dimension 2, les tangentes sont alors stables (dans n'importe quelle vue il y a une tangente) et ont codimension 0.

[4] On pourrait penser à l'exemple de deux cylindres parallèles pour lesquels les droites contenues dans l'un des deux plans de bitangence ont deux degrés de liberté. Nous ne considérons pas ce cas car ni la scène ni les objets ne sont génériques.

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	

TAB. 2.2: Faces du complexe de visibilité pour des scènes de polygones et d'objets lisses.

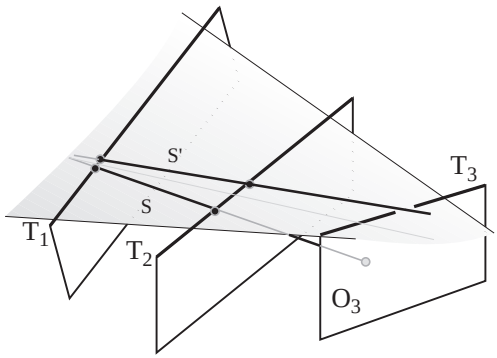


FIG. 2.10: Adjacences d'une face $T_1 + T_2 + T_3$. Deux ensembles différents de bitangentes $T_1 + T_2$ sont adjacents. Un avec pour extrémité O_3 (S par exemple) et un qui se trouve au dessus de T_3 (comme S').

mais cela n'est pas un problème dans notre cas puisque si une tangente est rencontrée, on a une codimension de plus). Si tous les segments de ce chemin ont la même extrémité, F est adjacent à la face associée à S', sinon, lorsque l'extrémité change il y a tangence et donc une codimension de plus.

Considérons par exemple une 3-face T, et un segment libre maximal s de cette face. Supposons de plus que s n'a pas d'extrémité à l'infini. Considérons un plan P contenant s. Définissons l comme la première droite (dans l'ordre polaire) tangente à l'objet en T et à l'objet O_1 de l'une des extrémités de s (figure 2.11). Considérons le chemin continu $l(t)$ dans l'espace des droites de s à l. Le chemin continu correspondant pour les segments est défini en considérant le segment $s(t)$ colinéaire à $l(t)$ et tangent en T. Si $s(t)$ a toujours les deux mêmes extrémités, alors ils appartiennent tous à la même face T et sont adjacents à la face $T+T$ qui correspond à l (figure 2.11(a)). Sinon, lorsque l'objet de l'une des extrémités change, le segment est tangent à deux objets, et il y a aussi une 2-face $T+T$ (figure 2.11(b)). Les adjacences pour les autres cas peuvent être démontrées de

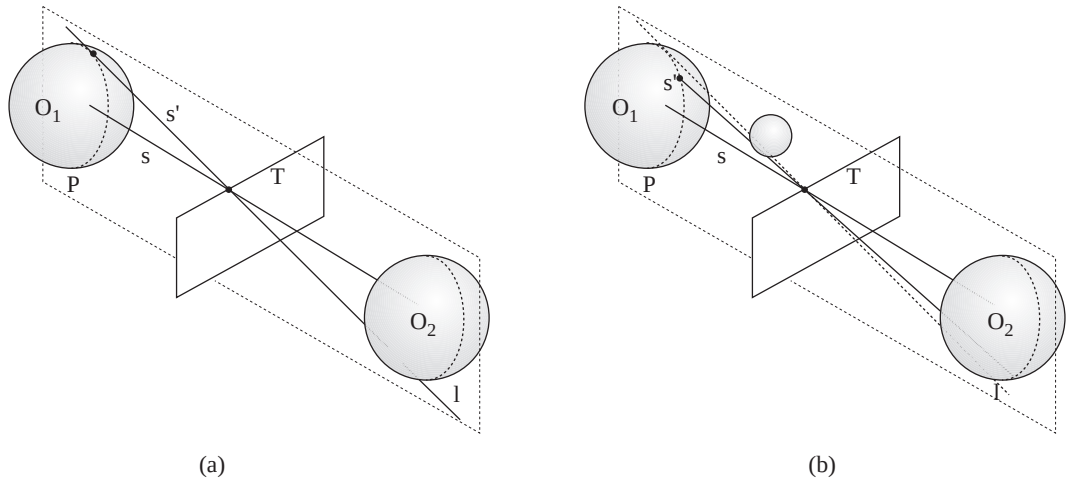


FIG. 2.11: Construction d'un segment $T+T$ appartenant à une 2-face adjacente à une 3-face T.

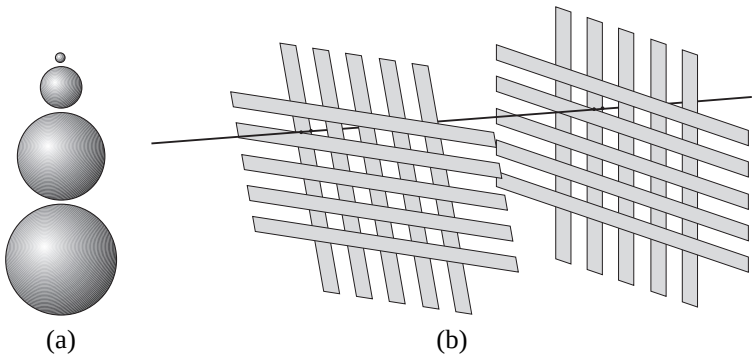


FIG. 2.12: Bornes inférieure et supérieure pour le complexe de visibilité. (a) Scène avec un complexe de visibilité de taille $O(n)$. (b) Scène avec un complexe de visibilité de taille $O(n^4)$. Un exemple de segment critique $T+T+T+T$est représenté.

manière similaire.

Observons qu'une 1-face peut par contre n'être adjacente à aucune 0-face (nous donnons dans ce qui suit l'exemple d'une scène sans 0-face).

La taille du complexe de visibilité est donc bornée par le nombre de 1-faces qui ne sont pas adjacentes à une 0-face plus le nombre de 0-faces. Le nombre de chaque sorte d'événement dépend du nombre de systèmes d'équations associés (puisque le degré est borné), et donc du nombre d'objets impliqués. Les événements $T+T+T+T$ sont donc potentiellement les plus nombreux avec $O(n^4)$.

La figure 2.12(a) montre un exemple de scène avec un complexe de visibilité de taille $O(n)$: il y a une face $T++T$ pour chaque paire de sphères voisines. Il n'y a aucune 0-face dans ce cas. La scène de la figure 2.12(b) est la même que celle exhibée par Plantinga et Dyer [PD90]. Elle a un complexe de visibilité de taille $O(n^4)$, et est composée de deux grilles formées d'ensembles orthogonaux de $\frac{n}{4}$ rectangles parallèles (une scène similaire peut être construite à l'aide d'ellipsoïdes très fins). Si l'on prend un rectangle dans chaque ensemble, il y a toujours un segment critique $T+T+T+T$ correspondant.

2.3 Complexité probabiliste

Dans bien des cas, les bornes théoriques ne donne que très peu d'indications sur la complexité pratique d'un problème ou d'une méthode pour des scènes "normales". Des études probabilistes utilisant un modèle de scène "raisonnable" dans l'esprit de de Berg *et al.* [dBKvdSV97] sont souhaitables.

Comme premier pas, nous proposons d'étudier le nombre de droites critiques $T+T+T$ dans une scène. La borne en $O(n^3)$ est fondée sur des objets potentiellement infiniment fins, et n'est pas réaliste comme nous

le verrons dans le chapitre suivant. Dans une scène typique, la taille des objets est bornée, et la complexité augmente en ajoutant de plus petits détails ou en plaçant plusieurs objets les uns à côté des autres, et non en ajoutant des lignes infinies entrelacées. Nous proposons un modèle simplifié probabiliste pour ces scènes "normales", et nous montrons que sous ces hypothèses le nombre d'événements $T+T+T$ est $O\left(n^{\frac{7}{3}}\right)$. Cette borne donne une meilleure intuition de la complexité réelle des scènes normales.

Nous considérons un modèle de scène où les objets ont une taille bornée r et sont uniformément distribués à l'intérieur d'une sphère de diamètre fini, R. La densité des objets est constante lorsque leur nombre varie. Si n est le nombre d'objets, on a $R = \Theta(\sqrt[3]{n})$. Nous étudions le nombre moyen d'événement $T+T+T$. Pour cela, nous étudions la probabilité, étant donnés deux objets, qu'un troisième engendre un événement $T+T+T$ avec eux. Nous montrerons que cela peut être exprimé comme un rapport de volumes. Nous intégrerons ensuite sur tous les objets pour obtenir une borne sur le nombre total moyen d'événements $T+T+T$. Pour simplifier nos calculs, nous utiliserons des sphères englobant les objets.

Probabilité pour deux objets donnés

Considérons deux objets A et B distants de x. Nous voulons obtenir une borne sur la probabilité $P_{AB,x}$ qu'un troisième objet C engendre un événement $T+T+T$ avec A et B. Cela se produira uniquement si la sphère englobant C intersecte un volume en forme de sablier formé de deux cônes et d'un cylindre tangents aux sphères englobant A et B (figure 2.13(a)). La probabilité qu'une sphère de rayon r intersecte ce volume est égale à la probabilité que son centre soit à l'intérieur du sablier dilaté de r. Puisque par hypothèse les centres de ces sphères sont uniformément distribués, cette probabilité est égale au rapport entre le volume du sablier dilaté et le volume de la sphère bornant la scène.

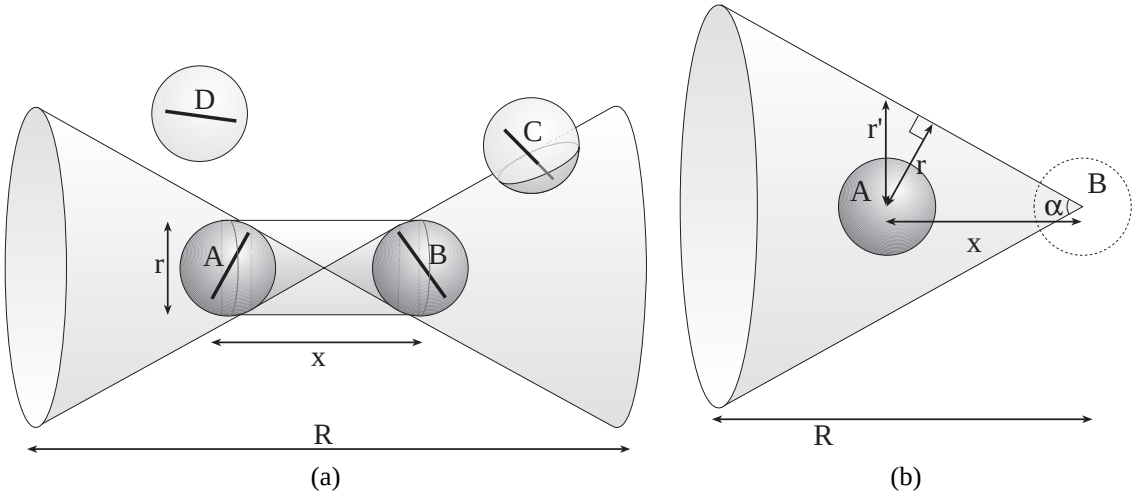


FIG. 2.13: Complexité probabiliste des événements $T+T+T$. (a) Volume en forme de sablier. (b) Construction d'un des cônes du sablier dilaté.

Calculons maintenant le volume du sablier dilaté. Rappelons que x est la distance entre les centres des sphères englobant A et B. Nous distinguons le cas où $x \leq 2r$ et le cas $x > 2r$. Dans le premier cas, nous bornons trivialement le volume par celui de la sphère, *i.e* $P_{AB,x\leq 2r} = 1$. Nous verrons que ce cas est asymptotiquement négligeable.

Dans le second cas, nous bornons le volume du sablier dilaté par la somme du volume d'un cylindre de longueur x et de diamètre $2r$, et et du volume de deux cônes de hauteur R définis figures 2.13(b). Le volume du cylindre est $vol_{cylindre} = \pi r^2 x$.
Bornons maintenant le volume d'un des cônes. De simples relations de trigonométrie donnent :

$$r' = x\tan(\arcsin\frac{r}{x})$$

r' est une fonction décroissante de x, nous la bornons par sa valeur pour $x = 2r$, qui est inférieure à $1.16r$. Le

volume du cône est alors :

$$vol_{cone} = \frac{\pi r'^2 x}{3} * \left(\frac{R}{x}\right)^3 < \frac{1.16^2 \pi r^2 R^3}{3x^2}$$

Le volume total du sablier dilaté est borné par :

$$\begin{aligned} vol_{sablier} &\leq 2vol_{cone} + vol_{cylindre} \\ &\leq 2\frac{1.35\pi r^2 R^3}{3x^2} + \pi r^2 x \end{aligned}$$

Nous divisons par le volume de la sphère englobant la scène pour obtenir la probabilité :

$$\begin{aligned} P_{AB,x} &\leq \frac{vol_{sablier}}{vol_{sphere}} \\ &\leq \frac{5.4r^2}{x^2} + \frac{6r^2 x}{R^3} \end{aligned}$$

Probabilité pour tous les objets

Nous avons maintenant une borne sur la probabilité, étant donnés une paire d'objets à une distance x, qu'un troisième objet C engendrent avec eux un événement $T+T+T$. Nous devons calculer le nombre total d'événements $T+T+T$, c'est-à-dire considérer tous les C et toutes les paires A,B à toutes les distances x possibles.

Nous multiplions par n pour obtenir le nombre moyen d'événements $T+T+T$ engendré par une paire A,B donnée avec tous les objets :

$$M_{AB,x}(T+T+T) = nP_{AB,x}$$

Jusqu'à maintenant nous avons considéré la distance x entre A et B fixée. Considérons maintenant une paire aléatoire d'objets A,B. Nous devons prendre en compte la probabilité qu'ils soient distants d'un x donné. Nous utilisons un résultat de géométrie intégrale ([San76] page 212) qui donne la distribution de probabilité de la distance entre paires de points dans une sphère. On a

$$\mu(x) = 12\lambda^2(1-\lambda)^2(2+\lambda)$$

où $\lambda = \frac{x}{R}$ (c'est-à-dire que la probabilité que deux points aléatoires dans une sphère soient à une distance comprise entre x et $x+dx$ est $\mu(x)\,dx$).

Pour avoir une intuition de cette formule, considérons qu'un point est choisi. Le second point est à une distance entre x et $x+dx$ s'il se trouve entre les deux sphères correspondantes (figure 2.14). La surface d'une telle sphère explique le terme en λ^2. Cependant, cette sphère doit être intersectée avec la sphère bornant la scène, ce qui explique le terme polynomial suivant.

Pour obtenir le nombre moyen d'événements $T+T+T$, nous intégrons sur x et multiplions par le nombre de paires d'objets n^2. Rappelons que nous avons distingué le cas $x \leq 2r$ où la probabilité est bornée par 1 et le cas $x > 2r$ où elle est donnée par un rapport de volumes.

$$M(T+T+T) < n^3 \left(\int_{x=0}^{2r} 1 * \mu(x)\,dx + \int_{x=2r}^{R} \left(\frac{5.4r^2}{x^2} + \frac{6r^2 x}{R^3}\right)\mu(x)\,dx \right) \tag{2.1}$$

Nous dérivons maintenant des bornes sur les deux intégrales de l'inéquation. 2.1. La première intégrale est :

$$\begin{aligned} \int_{\lambda=0}^{\frac{2r}{R}} 12\lambda^2(1-\lambda)^2(2+\lambda)\,d\lambda &= 2\left(\frac{2r}{R}\right)^6 - 9\left(\frac{2r}{R}\right)^4 + 8\left(\frac{2r}{R}\right)^3 \\ &= O\left(\frac{r^3}{R^3}\right) \end{aligned} \tag{2.2}$$

La seconde intégrale de l'inéquation 2.1 est bornée par :

$$\int_{x=2r}^{R} \left(\frac{5.4r^2}{x^2} + \frac{6r^2 x}{R^3}\right)\mu(x)\,dx \leq \int_{x=0}^{R} \left(\frac{5.4r^2}{x^2} + \frac{6r^2 x}{R^3}\right)\mu(x)\,dx \tag{2.3}$$

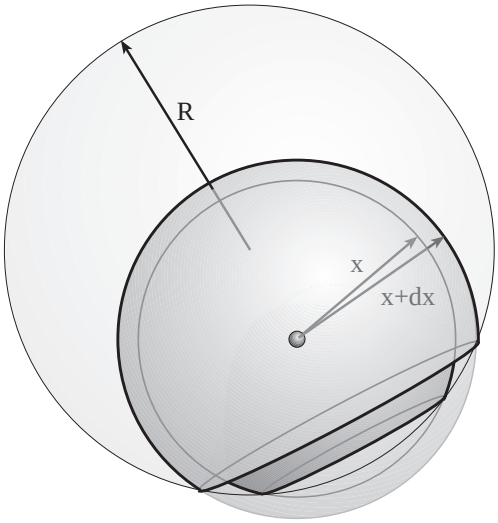


FIG. 2.14: Distribution de probabilité de distance entre paires de points dans une sphère.

Nous développons le premier terme à l'intérieur de l'intégrale :

$$\begin{aligned}\int_{\lambda=0}^{1} 12\frac{5.4r^2}{R^2}\lambda(1-\lambda)^2(2+\lambda)\,d\lambda &= \frac{5.4r^2}{R^2}\int_{\lambda=0}^{1} 12(1-\lambda)^2(2+\lambda)\,d\lambda \\ &= \frac{48.6r^2}{R^2} \\ &= O\left(\frac{r^2}{R^2}\right)\end{aligned} \tag{2.4}$$

Le second terme de l'inéquation 2.3 donne :

$$\begin{aligned}\int_{x=0}^{R} \frac{6r^2x}{R^3}\mu(x)\,dx &= \int_{\lambda=0}^{1} 12\frac{6r^2}{R^2}\lambda^3(1-\lambda)^2(2+\lambda)\,d\lambda \\ &= \frac{108r^2}{35R^2} \\ &= O\left(\frac{r^2}{R^2}\right)\end{aligned} \tag{2.5}$$

En combinant les résultats des équations 2.1, 2.2, 2.4 et 2.5 nous obtenons :

$$\begin{aligned}M(T+T+T) &= O\left(n^3\left(\tfrac{r^3}{R^3}+\tfrac{r^2}{R^2}+\tfrac{r^2}{R^2}\right)\right) \\ &= O\left(\tfrac{n^3}{R^2}\right) \\ &= O\left(\tfrac{n^3}{\sqrt[3]{n^2}}\right) \\ &= O\left(n^{\frac{7}{3}}\right)\end{aligned}$$

Le nombre moyen d'événements $T+T+T$ est $O\left(n^{\frac{7}{3}}\right)$ (c'est-à-dire $O\left(n^{2.33}\right)$) pour notre modèle au lieu de $O(n^3)$ lorsque des objets de tailles non bornée sont considérés. Nous verrons dans le chapitre suivant que sur les scènes sur lesquelles nous avons effectué nos tests, le nombre d' événements $T+T+T$ est à peu près quadratique, ce qui confirme notre borne probabiliste.

Des arguments similaires montrent que le nombre de 0-faces $T+T+T+T$ est $O\left(n^{\frac{8}{3}}\right)$ (*i.e.*, $O\left(n^{2.67}\right)$) au lieu de $O(n^4)$.

3 Extension à la visibilité temporelle

Dans cette section, nous décrivons l'extension du complexe de visibilité aux scènes dynamiques. Nous montrons qu'un cadre similaire permet de décrire les changements de visibilité lorsque des objets bougent de manière continue. Nous considérons n'importe quel mouvement continu, tous les objets peuvent être en mouvement et se déformer. Nous considérons la dimension temporelle continue, et non une succession régulière de pas de temps (même si une représentation à certains pas de temps nous sera indispensable pour l'exposition de notre propos).

3.1 Événements visuels temporels

Nous adoptons la même approche que pour le cas statique : nous essayons de déterminer quand la visibilité change dans une scène dynamique 3D. Nous nous intéressons aux changements topologiques du complexe de visibilité. Considérons la situation de la figure 2.15(a). La sphère A bouge vers le haut. Au temps t_1, il n'y a aucun segment qui a pour extrémités A et C à cause de l'occultation due à B. Au temps t_3, des segments voient A et C parce que A s'est déplacé vers le haut. La configuration limite se produit au temps t_2, où un plan est tangent aux trois sphères. Le segment $A++B++C$ correspond à ce plan tritangent. Il s'agit d'un *événement visuel temporel*. Il est similaire aux 0-faces du complexe de visibilité statique, avec une codimension en plus.

La figure 2.15(b) représente les 0 et 1-faces du complexe de visibilité statique pour plusieurs pas de temps. Au temps t_1 le complexe n'a que deux 1-faces et aucune 0-face. Au temps t_3, six 0-faces $T++T+T$ sont apparues ainsi que de nouvelles 1-faces de type $T++T$ et $T+T+T$. Il y a eu un changement topologique dans la structure du complexe de visibilité.

3.2 Le complexe de visibilité temporel

Nous ajoutons la dimension temporelle à l'espace des segments libres maximaux. Un segment temporel est défini comme un segment libre maximal de l'espace et une valeur de temps. L'espace des segments temporels est donc une 5-variété plongée en 6D.

Nous définissons le *complexe de visibilité temporel* comme la partition des segments libres maximaux temporels en fonction des objets qu'ils touchent.

Les k-faces du complexe de visibilité temporel correspondent aux (k-1)-faces du complexe de visibilité statique. Cependant, les codimensions sont conservées (puisque l'espace des segments temporels a une dimension en plus lui aussi). Le complexe de visibilité statique est en fait une tranche dans le temps du complexe de visibilité temporel. Les 0-faces du complexe temporel correspondent aux *événements visuels temporels*. Il s'agit de changements de la structure topologique du complexe de visibilité. Maintenir le complexe de visibilité lors du déplacement d'objets est équivalent à un balayage dans le temps du complexe de visibilité temporel.

La figure 2.15(c) montre les faces de dimension 0, 1 et 2 du complexe de visibilité temporel pour la scène de la figure 2.15(a). Lors de l'événement visuel temporel $A++B++C$, la tranche du complexe de visibilité temporel est modifiée : de nouveaux sommets $T+T++T$ apparaissent ainsi que de nouvelles 1-faces $T++T$ et $T+T+T$. Les événements visuels temporels pour les scènes de polygones et d'objets courbes convexes sont énumérés table 2.3. Ils sont très semblables aux sommets du complexe statique.

Notre définition est indépendante du mouvement des objets, pour peu qu'il soit continu. Des déformations peuvent survenir, aucune rigidité ou linéarité n'est imposée. Le traitement de la suppression ou de l'addition d'objets est cependant plus compliqué. Ils correspondent à des événements temporels dégénérés. Les objets concaves engendrent un catalogue d'événements plus complexe. En particulier, si des déformations sont considérées, il faut tenir compte des changements des propriétés différentielles des objets.

4 Balayage sensible à la taille du résultat

Nous présentons maintenant un algorithme de construction du complexe de visibilité statique dont la complexité dépend de la taille du résultat (c'est-à-dire du complexe). Notre algorithme est un double balayage avec une étape de précalcul. Tout d'abord, la scène est balayée par des plans horizontaux tandis qu'un complexe de visibilité 2D [PV96b] de la φv-tranche est mis à jour (figure 2.16(b)). Nous balayons ensuite selon φ (figure 2.16(c)) mais certaines 0-faces ne peuvent être détectées durant ce balayage et doivent être précalculées.

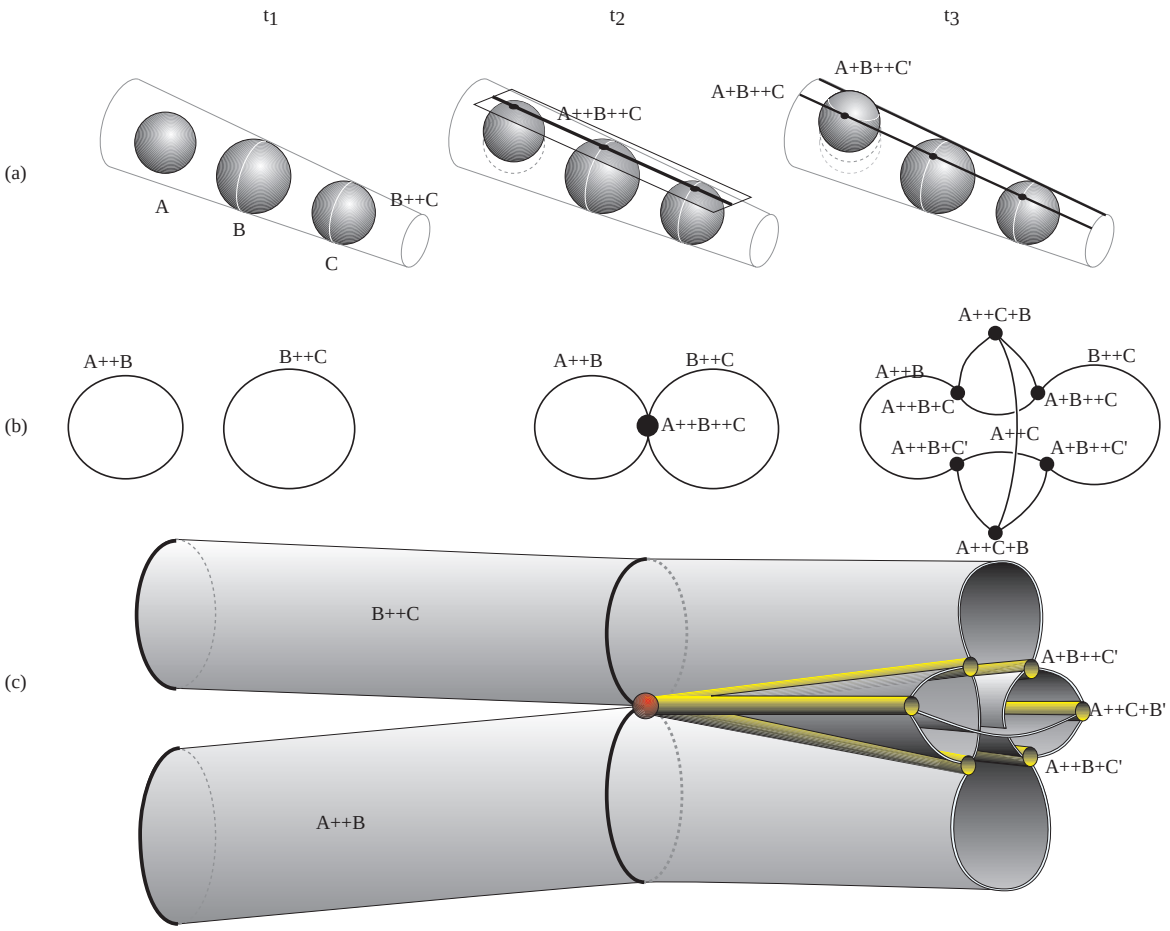
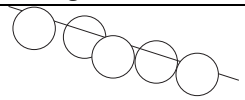
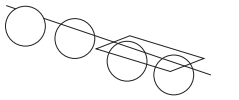
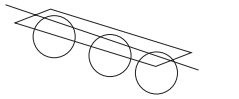
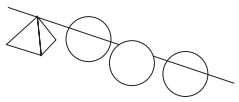
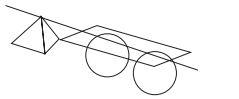
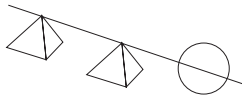


FIG. 2.15: Événement de visibilité temporel et complexe de visibilité temporel. (a) Situation dans l'espace 3D. Tandis que A se déplace vers le haut, il devient visible depuis C au temps t_2 où un segment $(A++B++C)$ se trouve dans un plan tangent aux trois sphères. L'enveloppe de $B++C$ est représentée comme un cône tronqué. Seulement deux 0-faces $T++T+T$ sont représentées, bien qu'il y en ait en fait 6. (b) Faces de dimension 0 et 1 du complexe de visibilité pour chaque pas de temps (il s'agit d'une projection et non pas d'une tranche : les ensembles 1D sont représentés comme des courbes). Au temps t_3 nous n'avons pas mis d'étiquette pour toutes les 1-faces. Remarquons que le graphe des 1-faces n'est pas planaire, pour le représenter nous ne pouvons éviter les croisements de $A++C$. (c) Structure d'une partie du complexe de visibilité temporel dans l'espace des segments temporels. Seules les faces de dimension 0, 1 et 2 sont représentées. Les faces du complexe de visibilité statique représentées en (b) sont des tranches du complexe de visibilité temporel. Les 1-faces du complexe temporel sont représentées comme de fins cylindres. Elles correspondent aux sommets du complexe statique et n'apparaissent qu'en t_2. Remarquons qu'à cause de la non-planarité des tranches, nous ne pouvons représenter les 2-faces du complexe temporel sans des intersections au milieu.

4.1 Balayage de la tranche initiale

Pour construire la φ-tranche initiale, nous commençons par maintenir une φv-tranche du complexe de visibilité, qui correspond au complexe de visibilité 2D [PV96b] du plan de balayage. Commençons par rapidement rappeler quelques points au sujet du complexe de visibilité 2D (voir aussi [PV96b, PV96a, Riv97a, DP95b]). Il s'agit de la partition des segments libres maximaux du plan en fonction des objets qu'ils touchent. Ses faces 2D sont des composantes connexes de segments qui touchent les mêmes objets à leurs extrémités (ce sont des φv-tranches des 4-faces du complexe de visibilité 3D). Elles ont pour frontières dans l'espace dual des arêtes qui correspondent à des segments tangents aux objets (φv-tranches des 3-faces T) et des sommets qui correspondent à des bitangentes (φv-tranches des 2-faces $T+T$). Puisqu'une vue correspond aux extrémités

Type	Configuration
T+T+T+T+T	
T++T+T+T	
T++T++T	
T+T+T+V	
T++T+V	
T+V+V	

TAB. 2.3: Sommets du complexe de visibilité temporel. Ces configurations peuvent survenir à cause du mouvement de n'importe lequel des objets, ou bien de tous.

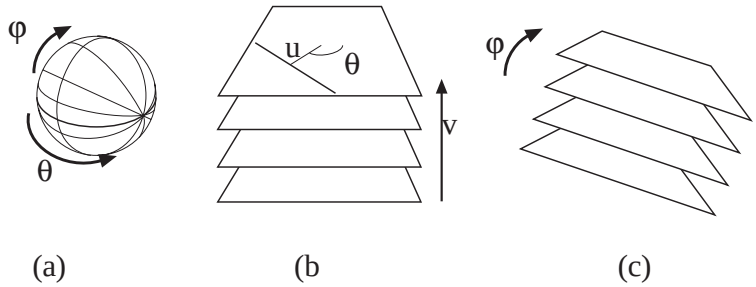


FIG. 2.16: (a) Paramétrisation des directions. (b) Balayage initial selon *v*. (c) Balayage selon φ.

des segments qui passent par le point de vue, cela revient à traverser le complexe de visibilité 2D le long du chemin 1D qui correspond à ces segments. L'objet vu change lorsque le chemin traverse une arête du complexe de visibilité 2D. Pour les scènes polygonales, la chaîne d'arêtes du complexe de visibilité 2D qui correspond à un sommet de la scène est la vue autour de ce sommet.

Lors de notre balayage, le complexe de visibilité 2D doit être mis à jour quand le plan de balayage est tangent à un objet, ou lorsqu'il contient un sommet de polyèdre, ou encore quand trois tranches d'objets ont une tangente commune en 2D.

Lorsque le plan de balayage commence à couper un objet, nous devons l'insérer dans le complexe 2D. Pour cela nous calculons la vue 2D autour du point de tangence ou autour du sommet en utilisant le complexe 2D courant. Cela peut être fait en temps $O(v \log n)$, où v est la taille de la vue, en utilisant les techniques décrites par Rivière [Riv97b]. Quand le chemin qui correspond à cette vue traverse une arête du complexe 2D, cela correspond à une nouvelle face $T+T$ ou $T+V$ du complexe de visibilité 3D. Dans le cas du premier sommet d'un polyèdre, la vue doit ensuite être restreinte pour chaque arête 3D du polyèdre, ce qui correspond en 2D aux vues depuis les sommets du polygone 2D qui est la tranche du polyèdre (figure 2.17).

De manière symétrique, lorsqu'un objet arrête de couper le plan de balayage, les faces correspondantes du complexe 2D doivent être supprimées. Ces faces sont celles situées le long de la chaîne d'arêtes de l'espace dual qui correspond aux segments tangents à cet objet. Leur suppression peut être effectuée en temps $O(v)$ où v est la taille de la vue 2D depuis le dernier sommet du polyèdre ou depuis le point de tangence.

Lorsqu'un sommet intermédiaire d'un polyèdre est balayé, les vues 2D autour des sommets 2D correspon-

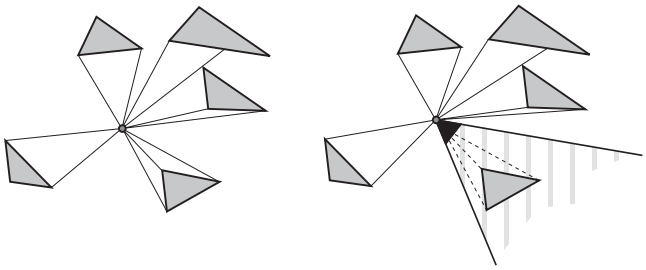


FIG. 2.17: Lorsque le premier sommet d'un polyèdre est balayé, la vue 2D correspondante est calculée dans le plan de balayage et est restreinte pour chacune des arêtes adjacentes au sommet balayé, en considérant l'angle formé par les deux polygones adjacents.

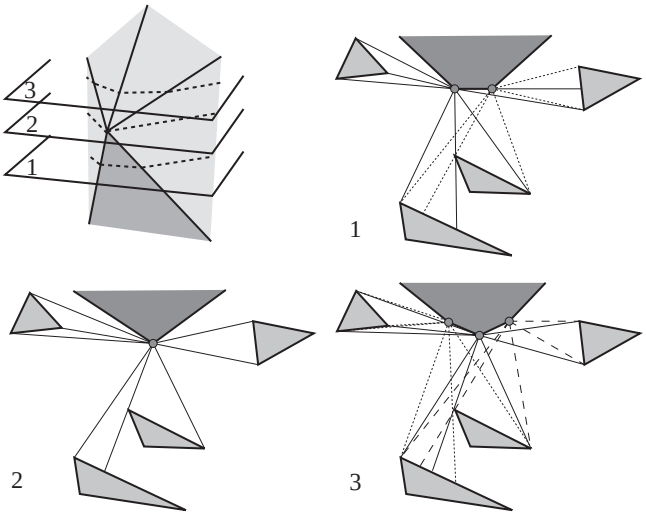


FIG. 2.18: Fusion-restriction d'une vue autour d'une arête lorsqu'un sommet est balayé

dant aux arêtes 3D en dessous du sommet sont fusionnées, et la vue autour du sommet doit être restreinte pour chaque arête 3D au dessus, de la même manière que pour les premiers sommets (figure 2.18). Chaque opération prend un temps linéaire en fonction de la taille de la vue autour du sommet (et du nombre d'arêtes adjacentes bien sûr).

Tandis que le plan se déplace, trois tranches d'objets peuvent avoir une tangente commune en 2D, ce qui correspond à une 1-face $T+T+T$ du complexe 3D. Dans ce cas, le complexe 2D est mis à jour en utilisant une méthode due à Rivière [Riv97b]. En résumé, pour chaque bitangente active on calcule la valeur de v pour laquelle un troisième objet deviendra tangent, et on stocke ces événements de balayage dans une queue de priorité, ce qui nécessite un temps $O(\log n)$ pour chaque nouvelle bitangente.

Pour finir, une bitangente 2D peut correspondre à un plan de tangence commun. Pour chaque bitangente, nous calculons donc la valeur de v pour laquelle elle correspond à un plan de tangence commun, et nous insérons cette valeur dans la queue d'événements de balayage. Bien entendu, cet événement doit être supprimé si la bitangente est supprimée à cause d'un événement $T+T+T$.

4.2 Principe du balayage en φ

Nous avons maintenant obtenu la φ-tranche initiale du complexe de visibilité 3D. Elle correspond à la partition des segments contenus dans des plans horizontaux. Dans cette φ-tranche, les 1-faces du complexe ont dimension 0, etc.

Au cours du balayag en φ, (figure 2.21(c)) nous maintenons cette φ-tranche ainsi qu'une queue de priorité d'événements de balayages. Dans ce qui suit, nous ne décrivons que la mise à jour des 1-faces du complexe de visibilité 3D. La mise à jour des faces de plus grande dimension est effectuée grâce au catalogue d'adjacences

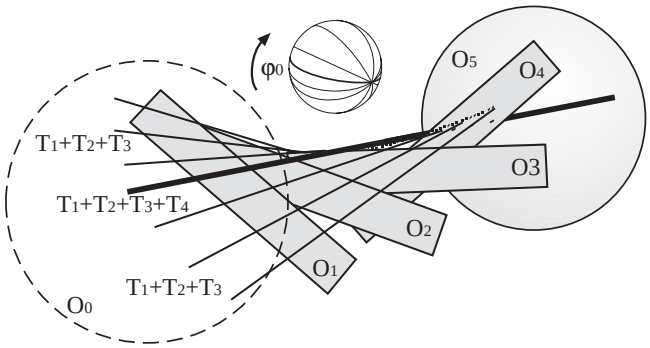


FIG. 2.19: Ensembles de segments critiques $T+T+T$ adjacent à un segment $T+T+T+T$.

qui, pour des raisons de place, est renvoyé en annexe A. Comme nous l'avons déjà signalé, le nombre de faces de dimension supérieure adjacentes à une face donnée est borné. Leur mise à jour ne modifie donc pas la complexité de l'algorithme.

Ce balayage selon φ peut également être appréhendé en projetant les 1-faces et les sommets du complexe 3D sur la sphère des directions S^2 représentée figure 2.16. Le balayage y correspond à la rotation d'un grand cercle. Cependant, certaines intersections de 1-faces sur S^2 ne correspondent à aucun sommet du complexe (la même différence existe entre un sommet-t (*t-vertex*) dans une vue et un véritable sommet de la scène). Cette représentation peut être utile pour se représenter le balayage car elle permet le parallèle avec la rotation d'une vue orthographique.

Nous prouvons tout d'abord que certains événements de balayage sont réguliers : une composante 1D de la φ-tranche disparaît quand ses deux extrémités fusionnent. Ces événements peuvent être facilement détectés en calculant pour chaque composante 1D de la φ-tranche, la valeur de φ pour laquelle ils disparaissent. Nous nous intéresserons ensuite au cas des événements irréguliers.

4.3 0-faces régulières

Considérons un segment $T_1+T_2+T_3+T_4$ qui a pour extrémités O_0 et O_5, et pour angle d'élévation φ_0 (figure 2.19). Considérons l'ensemble 1D de droites critiques $T_1+T_2+T_3$. Nous le paramétrons localement par φ et le nommons $l(\varphi)$. La surface réglée décrite par $l(\varphi)$ est tangente à O_4 en φ_0. Deux 1-faces du complexe sont associées à $l(\varphi)$: l'une pour $\varphi<\varphi_0$ et l'autre pour $\varphi>\varphi_0$; l'une a O_5 comme extrémité, l'autre O_4. La même chose est valable pour $T_2+T_3+T_4$. De plus, les deux 1-faces présentes avant φ_0 sont adjacentes à une même 2-face T_2+T_3. Dans la φ-tranche, cette 2-face est une composante 1D qui a pour limite les tranches de $T_1+T_2+T_3$ et $T_2+T_3+T_4$. Cet composante 1D disparaît à φ_0 : il s'agit d'un événement régulier de balayage. Il peut être détecté en considérant les faces $T+T+T$ adjacentes à une même $T+T$ dans la φ-tranche.

Les faces $T++T+T$ peuvent être traitées de la même façon car elles sont adjacentes à une paire de 1-faces $T++T$ et à une paire de $T+T+T$.

La projection d'un événement régulier sur S^2 est semblable à celle représentée figure 2.21(a).

4.4 0-faces irrégulières

Malheureusement, toutes les 0-faces ne sont pas des événements réguliers de balayage. Les événements $T+T+V$ et $V+V$ ne peuvent pas toujours être détectés de cette façon, car nous n'avons aucune garantie qu'une 1-face adjacente à une telle 0-face existe pour $\varphi<\varphi_0$. Par exemple, dans le cas représenté figure 2.20, les quatre $T+V$ adjacents au $V+V$ apparaissent à partir de φ_0. Cela correspond dans l'espace dual à la situation (b) de la figure 2.21[5].

Ces événements doivent être précalculés, en considérant toutes les paires de sommets, et tous les triplets objet-objet-sommet. Cela nous donne des 0-faces potentielles, dont on doit ensuite tester l'occultation lorsque leur angle φ_0 est balayé.

[5]Cela explique aussi pourquoi les cellules d'un graphe d'aspect ne sont pas nécessairement convexes, puisque ces événements irréguliers correspondent à des sommets rentrants sur S^2 ou à des arêtes rentrantes dans l'espace 3D.

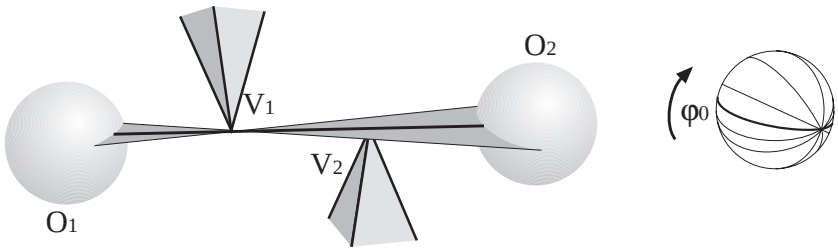


FIG. 2.20: Segment critique $V+V$ irrégulier. Aucun des ensembles de segments critiques $T+V$ adjacents à ce segment critique $V+V$ n'existe avant φ_0

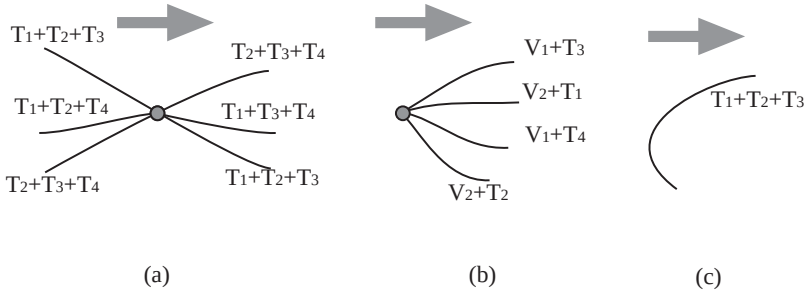


FIG. 2.21: Différents événements de balayage représentés dans l'espace dual. (a) Les événements $T+T+T+T$ sont réguliers (b) Les événements $V+V$ doivent être précalculés. (c) De même que les extrema des événements $T+T+T$ par rapport à φ.

Heureusement, il existe toujours au moins une 2-face V adjacente à un tel sommet avant son balayage, (face V_1 dans la figure 2.20), car les droites passant par un sommet couvrent toutes les directions. Une telle 2-face permet un test d'occultation efficace.

Cette 2-face peut être trouvée grâce à une structure de recherche sur les composantes 1D de la φ-tranche ordonnées par leurs générateurs. L'occultation de la 0-face potentielle est ensuite testée : on vérifie si le second générateur de la 0-face (V_2 dans notre exemple) se trouve entre les extrémités (O_1 et O_2) de la 2-face. La 0-face peut alors être insérée.

4.5 1-faces non monotones

il existe une autre sorte d'événements de balayage irréguliers. Une 1-face du complexe peut apparaître sans balayage d'une 0-face. Cela est bien évidemment le cas pour les événements $T++T$ puisqu'ils peuvent n'être adjacents à aucune 0-face, mais c'est aussi le cas des événements $T+T+T$. Considérons l'ensemble de droites critiques associé. Il n'est pas nécessairement monotone par rapport à φ (cf. figure 2.21(c)). Ces événements de balayage doivent eux aussi être précalculés et insérés dans la φ-tranche en utilisant une recherche sur les composantes 1D.

4.6 Complexité de l'algorithme

Théorème 2 *Le complexe de visibilité 3D peut être construit en temps $O((k+n^3)\log n)$ où n est la complexité de la scène et k le nombre de 0-faces du complexe.*

Lors du balayage initial selon v, chaque calcul de vue requiert au plus un temps $O(v\log n)$ où v est la taille de la vue. Une vue correspond au nombre de 3-faces du complexe 3D adjacentes à une 2-face qui apparaît ou disparaît. Le coût total de ces vues est donc borné par $O(k\log n)$. Chaque événement de tritangence demande un temps $O(\log n)$, là encore le coût total est borné par $O(k\log n)$.

Lors du balayage en φ, chaque événement régulier demande $O(\log n)$ pour la mise à jour de la queue de priorité.

Le précalcul des autres événements demande d'énumérer tous les triplets d'objets. Il coûte donc un temps $O(n^3 \log n)$ en comptant la recherche des 2-faces pour l'insertion et la queue de priorité.

Le fait que cet algorithme ait une complexité qui dépende de la taille du résultat est très important, car comme nous le verrons dans le chapitre suivant, le nombre de segments $T+T+T+T$ est bien inférieur à la complexité théorique de $O(n^4)$.

5 Applications de l'approche

5.1 Calcul de vue

La vue autour d'un point est définie par les extrémités des segments qui traversent ce point. L'ensemble des segments qui passent par un point est une surface 2D dans l'espace dual (u et v peuvent être exprimés comme des fonctions de $\sin(\theta)$ et $\sin(\varphi)$). La vue peut s'exprimer comme l'intersection du complexe de visibilité avec cette surface. Chaque face intersectée correspond à un objet vu. L'intersection avec un volume de tangence correspond à la silhouette de l'objet dans l'image. L'algorithme du lancer de rayon revient à échantillonner cette surface.

Dans la figure 2.22, la surface décrite par l'ensemble des droites passant par un point de vue V est représentée par ses φ-tranches qui sont des courbes. Les intersections de ces courbes avec les volumes de tangence sont les points du contour des objets dans la vue, comme D_1, D_2, D_3, D_4 et D_5. Cependant, toutes les intersections ne correspondent pas à des silhouettes car les objets ne sont pas transparents, et un point comme D' ne doit pas être pris en compte. Considérons la tranche $\varphi = 0$ et la tranche V_0 des droites passant par V. La figure 2.23 montre les φ-tranches des faces du complexe de visibilité et leur parcours. Nous traversons le complexe de haut en bas le long de V_0. Au départ, le segment ne voit rien, nous sommes dans la face F (qui est l'ensemble des segments dont les deux extrémités sont à l'infini). En D_1 nous quittons F, et devons choisir entre la face A et la face E au branchement dû à la tangence. Puisque V se trouve devant R, nous traversons A de D_1 à D_2. D' ne se trouve sur aucune frontière de la face A, il n'est donc pas considéré. Nous traversons ensuite la face D, et finalement à nouveau F. Une fois que la φ-tranche a été traversée, il faut maintenir les intersections avec les frontières des faces en balayant selon φ. La visibilité change quant V_φ rencontre une arête de bitangence ou un nouveau volume de tangence.

Pour une balade virtuelle, la vue peut être maintenue puisque les événements visuels décrivent les moments où la vue change : cela correspond à l'intersection de la surface des segments passant par V avec une 1-face du complexe. Coorg et Teller [CT96, CT97b] décrivent une approche similaire où des événements visuels prudents et locaux sont calculés à la demande.

La technique récente des *images à centre de projection multiple* (*multiple center of projection images*) [RB98] correspond en fait à intersecter le complexe de visibilité avec une 2-variété différente.

5.2 Facteurs de forme

Le facteur de forme[6] F_{ij} utilisé en radiosité est la proportion de lumière quittant l'objet i qui arrive en j. Il peut être exprimé comme la mesure des droites qui intersectent i et j divisé par la mesure des droites qui intersectent i. Dans l'espace dual, il s'agit de la mesure de la face F_{ij} divisée par la mesure de l'hyper-volume à l'intérieur du volume de tangence de i. Orti *et al.* [ORDP96, DORP96] donnent une interprétation similaire dans le plan à l'aide du complexe de visibilité 2D.

Malheureusement, bien qu'une formule simple existe pour calculer les facteurs de forme en 2D en présence d'obstacles, le seul résultat analytique connu en 3D ne traite que deux polygones en visibilité totale, et est déjà fort complexe [SH93]. Des calculs exacts peuvent néanmoins être réalisés entre un point et un polygone, comme nous le verrons dans le chapitre 4. Des approches stochastiques fondées sur la géométrie intégrale [San76, Sbe93] pourraient également être étudiées, en échantillonnant les 4-faces du complexe, ou en échantillonnant globalement l'espace dual.

[6] Nous utilisons la même notation pour le facteur de forme et pour la face du complexe entre i et j, bien que le facteur de forme soit un scalaire et la face un ensemble de segments.

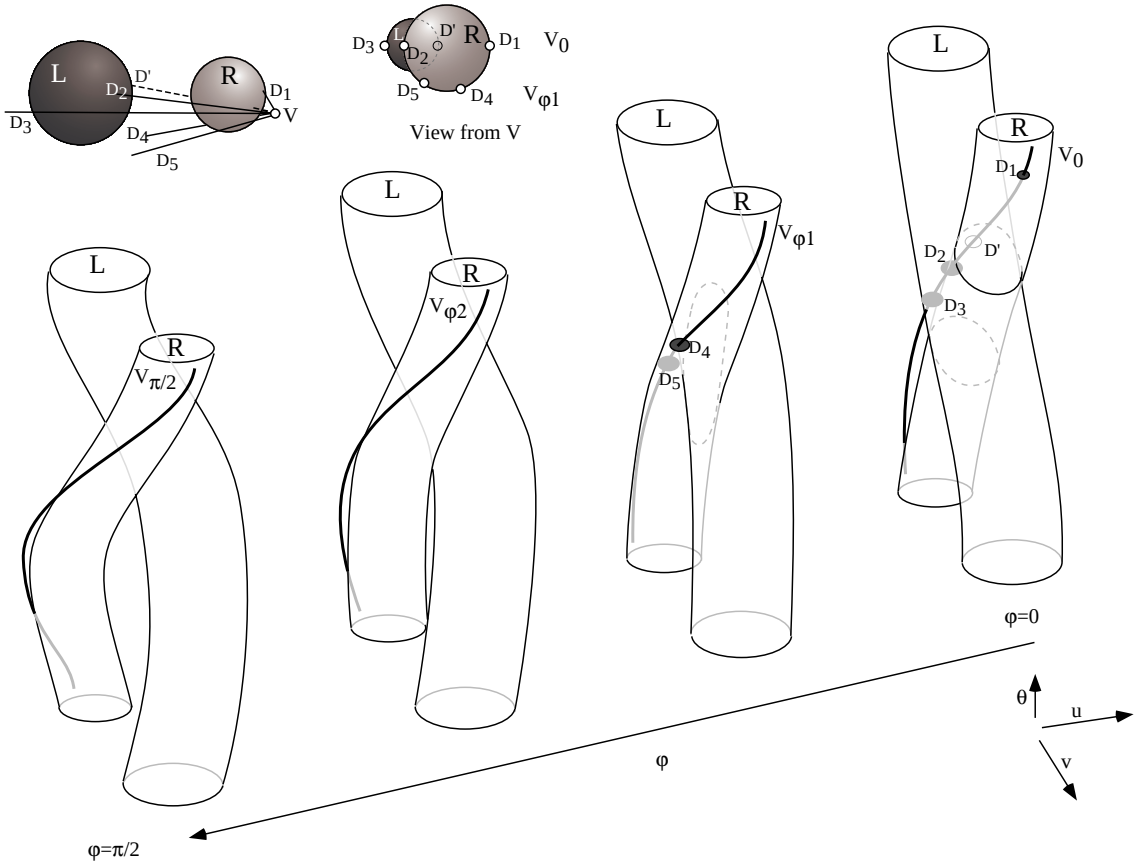


FIG. 2.22: La vue autour d'un point est l'intersection dans l'espace dual du complexe de visibilité et de la surface décrite par l'ensemble des segments passants par ce point.

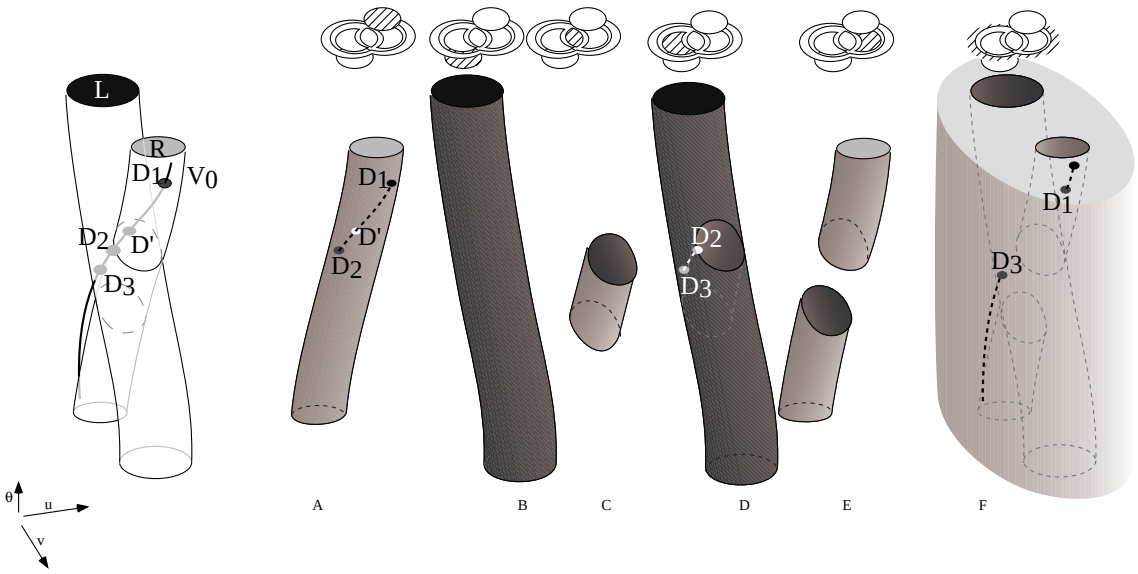


FIG. 2.23: Parcours de la φ-tranche $\varphi = 0$ du complexe de visibilité pour calculer la vue autour du point V.

5.3 Vision par ordinateur et robotique

Rappelons que le graphe d'aspect [PD90, GM90, EBD92] (chapitre 1 section 5) est une puissante structure de données centrée sur l'observateur, qui représente toutes les vues possibles d'un objet. Il est basé sur les événements visuels, qui partitionnent l'espace des points de vue en cellules où les vues sont qualitativement invariantes.

Les 1-faces du complexe de visibilité correspondent aux événements visuels du graphe d'aspect. Le complexe peut donc être utilisé pour construire le graphe d'aspect. Nous reviendrons sur ce sujet dans la section 6.1.

Les événements visuels interviennent aussi dans le calcul de l'*enveloppe visuelle* [Lau94] ou dans la planification de trajectoire avec visibilité pour des poursuites [GLLL98]. Le complexe pourrait être utilisé comme structure intermédiaire pour ces problèmes, pour lesquels les résultats connus sont principalement restreints au plan.

5.4 Ombre et pénombre

De la même façon, les 1-faces à l'intérieur du volume de tangence d'une source de lumière correspondent aux *surfaces de discontinuité* des méthodes de maillage de discontinuité (section 4.3 du chapitre 1), c'est-à-dire aux limites d'ombre et de pénombre. Le complexe de visibilité fournit tous les événements pour calculer un maillage de discontinuité où tous les objets peuvent être considérés comme des sources de lumière.

Dans le contexte de la radiosité hiérarchique [HSA91], si un lien entre deux objets i et j a besoin d'être raffiné, la face F_{ij} du complexe fournit toute l'information de visibilité pertinente pour cet échange énergétique. Cette information peut être utilisée pour effectuer du maillage de discontinuité à la demande, et pour améliorer la précision du calcul des facteurs de forme.

Dans le chapitre 4 nous introduirons une méthode efficace de simulation de l'éclairage basée sur une structure similaire au complexe de visibilité. Elle utilise la visibilité pour guider le raffinement, le maillage et le calcul des facteurs de forme, ce qui permet des simulations de haute qualité.

6 Approches liées

D'autres structures globales de visibilité ont été proposées dans la littérature, certaines d'entre elles basées sur des concepts similaires au complexe de visibilité. Nous les passons en revue, en insistant sur les similarités et les différences.

6.1 Le graphe d'aspect

Les événements visuels considérés dans la littérature consacrée aux graphes d'aspects correspondent aux 1-faces du complexe de visibilité : la topologie d'une vue change par exemple quand un sommet et une arête sont alignés depuis un point de vue. Le graphe d'aspect est en fait l'arrangement de ces événements dans l'espace des points de vue.

Cela explique sa plus grande taille par rapport au complexe. Intéressons-nous au cas polygonal. Il peut y avoir $O(n^3)$ événements $T+T+T$. Pour le complexe de visibilité, cela induit une taille en $O(n^4)$ car ces événements ne peuvent être coupés que par des sommets $T+T+T+T$. Dans le cas du graphe d'aspect pour une projection orthographique, la construction de l'arrangement 2D sur S^2 peut engendrer jusqu'à $O(n^6)$ intersections entre les paires d'événements $T+T+T$. Dans le cas perspective, l'espace des points de vue est $\mathbb{R}^3$, et la taille du graphe d'aspect $O(n^9)$. Le complexe de visibilité contient en fait implicitement l'information du graphe d'aspect.

Le graphe d'aspect pour une projection orthographique traite en fait lui aussi une information de dimension 4 : 2 pour l'espace des points de vue, et 2 pour chaque vue. Cependant, la séparation entre ces deux composantes rend difficile l'expression de propriétés comme la visibilité mutuelle de paires d'objets.

6.2 Asp

L'*asp* [PD90] (cf. section 6.3 du chapitre 1) tout comme le complexe de visibilité est une structure dans l'espace des droites. Il a été développé comme structure de données intermédiaire pour la construction de graphes d'aspects.

Deux définitions différentes existent en fonction de la projection, orthographique ou perspective. Dans le cas orthographique, les droites orientées sont considérées et l'*asp* est un complexe cellulaire 4D qui décompose l'espace des droites en fonction du premier objet rencontré. L'information de visibilité à l'intérieur de l'enveloppe convexe de l'objet n'est pas traitée.

Dans le cas de la projection perspective, l'*asp* est un complexe cellulaire 5D, qui décompose l'espace des rayons en fonction du premier objet intersecté. L'*asp* contient alors la même information que le complexe de visibilité, mais elle est redondante car de nombreux rayons colinéaires voient le même objet.

Le complexe de visibilité peut être vu comme offrant les avantages de l'*asp* pour les projections perspective et orthographique. Il traite la bonne quantité d'information grâce à la notion de segments libres maximaux. La cinquième dimension n'est utilisée que si nécessaire, c'est à dire le long des volumes de tangence. Il s'agit donc d'une structure 4D plongée en 5D. De plus, aucune version de l'*asp* n'a été proposée pour des objets courbes.

6.3 La fonction plénoptique et les champs lumineux

La *fonction plénoptique* [AB91] est une fonction dans l'espace 5D des rayons qui décrit la lumière traversant chaque point de l'espace dans chaque direction. Si la lumière qui parcourt l'espace libre est supposée invariante et si l'observateur est contraint à l'extérieur de la scène, cette fonction peut être simplifiée en une fonction 4D appelée champ de lumière (*light-field* [LH96] ou *lumigraph* [GGSC96]).

Ces fonctions correspondent en fait à l'*asp*, à part qu'une information de couleur est considérée.

Langer et Zucker [LZ97] utilisent des concepts topologiques similaires aux nôtres pour décrire le champ de lumière (*light field*) dans une scène dans un contexte de forme-à-partir-d'ombrage (*shape-from-shading*). Tout comme pour le complexe de visibilité, ils notent que la variété des rayons est en fait de dimension 4, avec des branchements. Ils ne proposent cependant pas d'analyse des changements de visibilité, ni l'expression de la cohérence gâce aux adjacences entre classes de rayons.

6.4 Coordonnées de Plücker

La paramétrisation de Plücker est une puissante dualité qui représente les droites de l'espace dans un espace à 5 dimensions où les hyperplans peuvent être utilisés pour caractériser les intersections droite-droite (cf. la section 6.2 du chapitre 1). Elle a été très utilisée pour les calculs de visibilité dans des scènes polygonales.

Remarquons que la cinquième dimension de l'espace de Plücker ne correspond pas à la pseudo dimension que nous avons introduite à la section 1.2. Les points de l'espace de Plücker ne correspondent pas tous à des droites réelles, uniquement ceux qui se trouvent sur l'*hypersurface de Plücker*. La cinquième dimension permet de s'affranchir des problèmes de singularité inhérents à toute paramétrisation 4D des droites (comme c'est aussi le cas pour une paramétrisation 2D de la sphère S^2). Par exemple notre paramétrisation (φ, θ, u, v) possède deux singularités aux pôles de la sphère des directions.

Des arrangements ont été décrits dans l'espace de Plücker qui sont très semblables à notre arrangement dual, par exemple [Pel93, Tel92b, Tel92a]. Un arrangement d'hyperplans est calculé en 5D et ensuite intersecté avec l'hypersurface de Plücker pour obtenir une structure 4D. Cette structure est exactement notre arrangement dual, exprimé dans une dualité différente. Cela illustre ce que nous disions en introduction de ce chapitre : les concepts que nous décrivons ne sont pas liés à une paramétrisation particulière des droites.

Néanmoins, les approches dans l'espace de Plücker ne se sont intéressées qu'à l'équivalent de l'arrangement dual. Elles ne considèrent que la visibilité selon les droites, et les occultations ne sont pas véritablement prises en compte.

6.5 Le complexe de visibilité 2D

La version bidimensionnelle du complexe de visibilité a été l'inspiration initiale de ce travail. Nous comparons maintenant les structures 2D et 3D, ce qui illustre les grandes différences entre la visibilité 2D et 3D.

Tout d'abord, l'espace des droites de l'espace 3D est de dimension 4. L'augmentation de la dimension des problèmes est 2 et non pas seulement 1. De même, la complexité théorique est $O(n^4)$ au lieu de $O(n^2)$. En 2D seules les bitangentes doivent être considérées, tandis qu'en 3D les segments tangents à quatre objets constituent les sommets du complexe.

La propriété qui explique que la visibilité 3D soit bien plus complexe est la *séparabilité*. En 2D, une droite sépare le plan en deux demi plans, ce qui n'est plus vrai en 3D car les droites n'y sont plus des hyperplans.

En conséquence, des propriétés de convexité ou de monotonicité qui étaient valables en 2D ne le sont plus en 3D. C'est le cas pour les faces du complexe : en 2D, en choisissant bien la dualité, les faces sont au moins monotones selon la direction. Cela implique des conséquences bien utiles, en particulier la possibilité d'effectuer des balayages optimaux ou des parcours efficaces le long du complexe.

En 3D, nous avons vu que certaines 0-faces sont irrégulières pour le balayage du complexe. Les faces ne peuvent être rendue monotones par rapport à un paramètre d'une paramétrisation. Elles peuvent de plus avoir un genre autre que 0 (elles peuvent avoir des trous, comme un tore). Cela rend notre algorithme non optimal. Cela explique également la difficulté de développer un algorithme efficace d'extraction de vue.

7 Conclusion

7.1 Résumé

Nous avons présenté une nouvelle approche pour les calculs de visibilité et décrit une puissante structure de données qui englobe toute l'information de visibilité d'une scène 3D. L'espace dual que nous utilisons permet une meilleure compréhension des événements visuels, qui ont été présentés en détail. Notre représentation donne de plus toutes les adjacences entre ces événements.

Nous avons introduit une structure de données unifiée, le *complexe de visibilité 3D*, qui décrit l'information globale de visibilité d'une scène 3D composée de polyèdres et d'objets lisses. Sa taille k est $\Omega(n)$ et $O(n^4)$, et nous avons présenté un algorithme de construction dont la complexité dépend de la sortie, puisqu'il est en temps $O((n^3+k)\log n)$.

En utilisant une approche probabiliste, nous avons montré que pour des scènes "normales" le nombre d'événements de tritangence est $O(n^{2,33})$.

Nous avons également défini le *complexe de visibilité temporel* qui décrit toute la visibilité d'une scène d'objets en mouvement.

Le complexe de visibilité est une structure de données très prometteuse pour de nombreuses applications en synthèse d'images : nous avons brièvement décrit son utilisation potentielle pour les calculs de vue, pour le calcul de facteur de forme, pour les limites d'ombre et de pénombre, et pour le calculs de graphes d'aspect.

Nous verrons dans le chapitre suivant que, simplifié, il mène à des solutions pratiques efficaces.

7.2 Discussion

Le complexe de visibilité est un cadre utile car il permet une description efficace de l'information de visibilité. Nous croyons qu'il s'agit d'un bon outil pour l'interprétation de problèmes de visibilité. En permettant un point de vue différent sur ces questions, nous espérons qu'il permettra une meilleure compréhension et le développement de nouvelle méthodes, ainsi que le montre le chapitre suivant.

Son intérêt pratique direct est plus discutable. L'implémentation d'une subdivision cellulaire 4D est une tâche ardue, et le parcours de ses adjacences est loin d'être trivial.

De plus, l'algorithme de construction est certes efficace, mais sujet à des problèmes de robustesse et de dégénérescences. Si un événement du balayage n'est pas correctement traité ou détecté, tout le reste de l'algorithme échoue car la cohérence est perdue.

Nos réflexions sur l'implémentation éventuelle du complexe de visibilité nous ont amenés à en extraire l'information la plus importante et à développer une structure de données plus simples, ainsi qu'un algorithme de construction plus robuste que nous présentons dans le chapitre suivant.

7.3 Travaux futurs

Les faces du complexe de visibilité traduisent directement la nature ardue de la visibilité 3D. Elles n'ont aucune propriété de convexité ou de monotonicité, comme le montre bien le problème des événements irréguliers lors de notre balayage. Forcer de telles propriétés serait possible en subdivisant de manière appropriée les faces du complexe. Les partitions cylindriques de Mulmuley [Mul91] ou les décompositions cylindriques algébriques de Collins [Col75] sont des exemples d'une telle approche. Malheureusement, cela augmenterait la complexité de la structure.

La distinction entre une composante directionnelle (θ, φ) et une composante spatiale (u, v) pour la paramétrisation des droites mériterait d'être étudiée plus avant. Comme nous l'avons vu, cela permet l'interprétation des événements $T++T$ dans notre espace dual. Nous pensons de plus que cela peut s'avérer utile pour obtenir des propriétés de monotonicité, car les volumes de tangence ont une structure de tube le long de la composante directionnelle : pour toute paire (θ, φ), il existe une paire (u, v) qui appartient à un volume de tangence donné, alors que l'inverse n'est pas vrai.

Les techniques de localisation randomisée [Cla87] pourraient permettre des requêtes efficaces. Cela demande de traiter des hyperplans, ce qui peut être obtenu pour des scènes polygonales en passant dans l'espace de Plücker. On se rapproche alors des travaux de Pellegrini [Pel93, Pel90, PS92, Pel94]. Cependant, on veut traiter la visibilité selon les segments et non selon les droites. Traiter la visibilité selon les segments est loin d'être trivial dans ce contexte, mais c'est le seul moyen d'éviter le coût fixe en $O(n^4)$.

La définition d'un algorithme efficace d'extraction de vue à l'aide du complexe doit être étudiée. Nous n'avons pu trouver une telle méthode à cause des problèmes de monotonicité mentionnés ci-dessus. La maintenance de vue pose le même genre de problèmes.

Le développement d'un algorithme de calcul de vue efficace est lié à la construction efficace du complexe. Pour les scènes polygonales, une autre approche consiste à balayer la scène par des plans tournant autour des arêtes des polygones, tout en maintenant des complexes de visibilité 2D. Il faut alors synchroniser les balayages, et ne considérer dans chaque plan que la partie du complexe 2D visible depuis l'arête. La définition d'un ordre topologique sur les 0-faces du complexe est une autre question à résoudre pour obtenir un algorithme de construction optimal dont la complexité dépende de la taille du résultat.

Dans le cas de scènes algébriques, les travaux de Petitjean [Pet95, Pet96] en utilisant la géométrie énumérative pourraient être adaptés pour obtenir des bornes fines sur la taille du complexe en fonction du degré des objets.

CHAPITRE 3

Le squelette de visibilité

> Une ligne droite toute seule n'a pas de signification, il en faut une seconde pour lui donner de l'expression.
>
> Eugène DELACROIX

IEN QU'IL décrive toute la visibilité d'une scène 3D, le complexe de visibilité possède des défauts qui rendent délicate son implémentation directe en tant qu'outil pratique. Les adjacences quadridimensionnelles de ses faces sont ardues à construire et à traverser. De plus, l'algorithme de construction par balayage proposé dans le chapitre précédent est sujet à des problèmes de robustesse et a un coût cubique fixe qui n'est pas acceptable en pratique. Qui plus est, le complexe de visibilité ne fournit pas vraiment un accès direct simple à l'information qu'il contient.

Dans ce chapitre, nous présentons une structure de données qui a été développée pour simplifier le complexe de visibilité, et transformer les concepts présentés dans le chapitre précédent en un outil pratique. Cela peut être vu comme une simplification ; en particulier le squelette de visibilité ne requiert la construction que des faces de dimension 0 et 1 du complexe de visibilité. Il peut aussi être vu comme une nouvelle structure développée à partir des événements visuels et des droites critiques. Nous adoptons cette seconde présentation, car elle rend le chapitre plus indépendant et parce qu'elle permet de revoir certaines notions liées au complexe de visibilité sous un nouvel angle.

Nous nous restreignons maintenant au cas de scènes polygonales. Nous mettons l'accent sur le développement d'un outil qui soit pratique et facilement implémentable. Les requêtes que nous présentons à la fin de ce chapitre, ainsi que l'utilisation du squelette de visibilité pour la simulation de l'éclairage proposée dans le chapitre suivant, montre que cette structure de données est polyvalente et efficace.

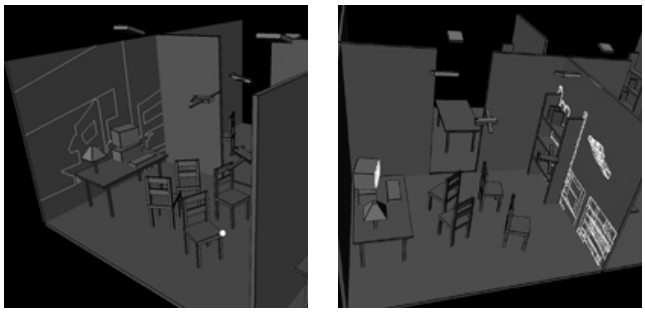
Une partie du travail décrit dans ce chapitre a été présenté à Siggraph'97 [DDP97c]. Nous avons ici inclus une optimisation de notre structure de données proposée dans un autre article [DDP99] et nous avons ajouté une description de la mise à jour pour des scènes d'objets en mouvement dans la section 7.

1 Motivation

Les méthodes antérieures n'ont pas fourni de structure de données efficace et robuste qui puisse répondre à des requêtes de visibilité globale pour des scènes typiques de synthèse d'images. Dans ce qui suit, nous

présentons une nouvelle structure de données qui fournit de l'information *exacte* de visibilité globale. Notre structure de données, que nous appelons le *squelette de visibilité*, est facile à construire puisque sa construction n'est basée que sur des algorithmes standards de synthèse d'images, comme le lancer de rayon et des intersections droite-plan. Il peut être utilisé pour résoudre de nombreux problèmes de visibilité globale, et pour finir, il est bien adapté à la construction *paresseuse* ou *à la demande* grâce à la localité de sa construction et à la structure elle-même. Ceci est particulièrement important dans le cas de géométries complexes.

Les composantes majeures du squelette de visibilité sont les *droites critiques* et les *droites poignardantes extrêmes* qui, comme nous le verrons, sont le lieu de tous les changements de visibilité d'une scène. Toute modification de visibilité dans une scène polygonale peut être décrite par ces droites critiques et par leurs adjacences. Nous présentons un algorithme de construction du squelette de visibilité, ainsi que son implémentation et plusieurs applications. Par exemple, la figure 3.1(a) représente une scène composée de 1500 polygones ; après la construction du squelette, des requêtes de visibilité peuvent être effectuées très rapidement. Nous montrons la partie du mur de gauche visible depuis un point qui a été obtenue en 1,4 ms, et dans la figure 3.1(b) nous montrons le maillage de discontinuité complet sur le mur de droite en considérant l'écran de l'ordinateur comme source de lumière, ce qui a requis 8,1 ms.



(a) (b)

FIG. 3.1: (a) Calcul exact de la partie du mur de gauche visible depuis le sommet vert sur la chaise. (b) maillage de discontinuité complet sur le mur de droite en considérant l'écran de l'ordinateur comme source de lumière.

Dans la section 2, nous fournissons une description complète de toutes les droites poignardantes extrêmes possibles, ainsi que des ensembles de droites critiques qui leur sont adjacents. Dans la section 3, nous présentons en détails nos structures de données, tandis que la construction est décrite en section 4. Les résultats de notre implémentation sont donnés à la section 5, avec en particulier la construction du squelette de visibilité complet pour une série de scènes de test. Nous montrons comment le squelette peut ensuite être utilisé pour fournir des informations exactes de visibilité depuis un sommet de la scène ; pour calculer un maillage de discontinuité entre toute paire de surfaces de la scène ; pour extraire la liste exacte des bloqueurs entre deux objets ; enfin pour obtenir les limites des occultations causées par un objet, ce qui pourrait être très utile pour une mise à jour dynamique de calculs d'éclairage en cas de déplacement d'un objet. La section 6 présente une construction à la demande qui permet de traiter les scènes complexes. Dans la section 7 nous décrivons brièvement un algorithme de mise à jour du squelette en cas de mouvement d'objet.

2 Le squelette de visibilité

Dans ce qui suit, nous ne considérons que des scènes polygonales.

2.1 Événements visuels

Nous avons vu dans le chapitre 1 que dans les méthodes antérieures de visibilité globale, en particulier pour les graphes d'aspect (*e.g,* [PD90, GM90, GCS91]) et pour l'antipénombre [Tel92a] ou pour les maillages de discontinuité [DF94, SG94], les changements de visibilité sont caractérisés par les *ensembles de droites critiques* et les *droites poignardantes extrêmes* (*extremal stabbing lines*) qui sont le lieu des *événements visuels*.

Nous adoptons la terminologie de Pellegrini [Pel90] et Teller [Tel92a], qui définit une droite poignardante extrême comme une droite passant par quatre arêtes de polygones. Il existe plusieurs sortes de droites poignardantes extrêmes, par exemple les sommet-sommet (VV pour *vertex-vertex*), les VEE (sommet-arête-arête ou *vertex-edge-edge*) et les quadruples-arêtes ($E4$). Comme nous le verrons dans la section 2.3, nous considérerons aussi des droites poignardantes extrêmes engendrées par les faces des polyèdres. Toutes ces droites extrêmes correspondent aux sommets du complexe de visibilité 3D.

Une *surface critique* (*critical line swath*) est la surface décrite par des droites extrêmes lorsque l'une des contraintes due à l'une des quatre arêtes est relâchée. La surface critique peut être plane (si les droites demeurent contraintes par un sommet) ou constituer une surface réglée, dont les trois lignes génératrices sont les droites des trois arêtes polygonales.

Nous appelons *générateur* un sommet ou une arête qui participe à la définition d'une droite poignardante extrême.

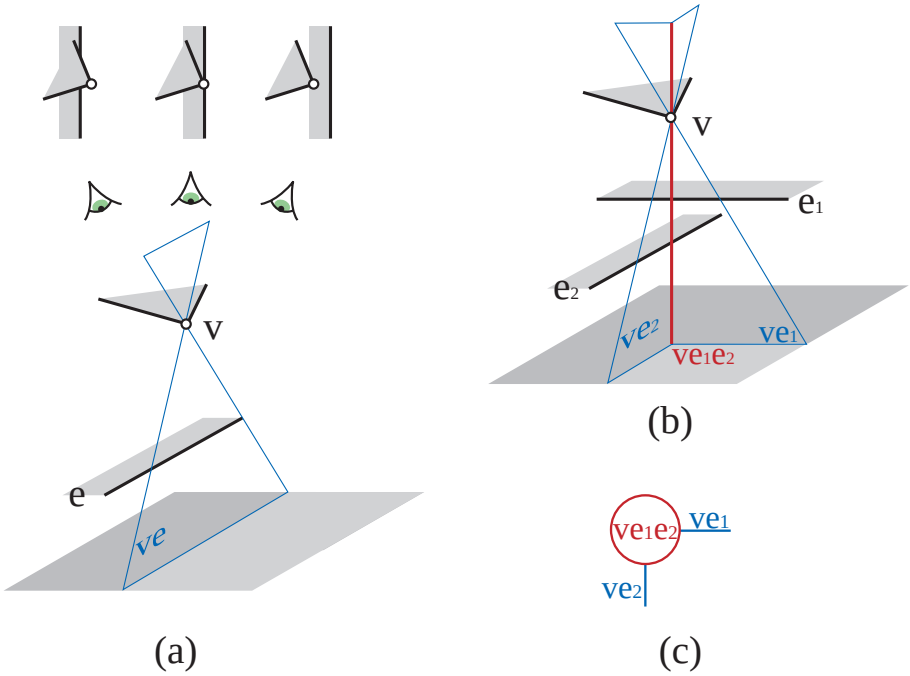


FIG. 3.2: (a) Tandis que l'œil traverse la surface critique VE, le sommet v passe devant l'arête e. (b) Deux ensembles de droites critiques se rencontrent à une droite poignardante extrême (c) et induisent une structure de graphe.

Commençons par un exemple : après avoir traversé la surface critique EV de gauche à droite dans la figure 3.2(a), pour l'observateur le sommet se trouve au dessus du polygone adjacent à l'arête et non plus au dessus du sol. C'est un changement de visibilité, *i.e.* un événement visuel, comme nous l'avons vu dans le chapitre précédent. La topologie de la vue est modifiée dès que le sommet et l'arête sont alignés, c'est-à-dire quand il existe une droite qui passe par l'œil, par l'arête e et par le sommet v.

Cet ensemble de droites critiques EV est un ensemble de dimension 1 de droites qui passent par le sommet v et l'arête e : il a un degré de liberté, par exemple un paramètre sur l'arête e. Quand deux surfaces EV se rencontrent comme sur la figure 3.2(b), une droite est définie par l'intersection des deux plans des surfaces EV. Cette droite est une droite poignardante extrême, elle n'a aucun degré de liberté.

Dans ce qui suit, nous développons des concepts pour éviter le traitement direct des surfaces critiques, car ces ensembles sont durs à traiter, entre autres parce qu'ils peuvent décrire des quadriques réglées. Tous les calculs seront effectués en intersectant des droites – ou des rayons– à travers la scène

Nous utiliserons les droites poignardantes extrêmes pour décrire toute l'information de visibilité, en stockant une liste de toutes les surfaces critiques adjacentes à chaque droite extrême. Dans notre premier exemple figure 3.2(b), la droite de type VEE ve_1e_2 est adjacente aux deux ensembles 1D ve_1 et ve_2 décrits ci-dessus. La droite ve_1e_2 est de plus adjacente à un autre ensemble de droites critiques engendré par l'interaction de ve_1 et

e_2 (figure 3.3(a)).

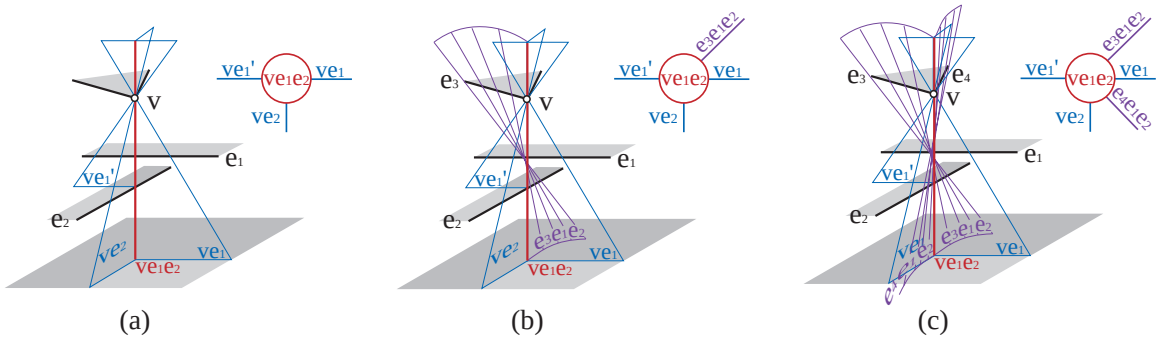


FIG. 3.3: (a) Un autre ensemble de droites critiques EV est adjacent à la droite poignardante extrême VEE, (b) (c) ainsi que deux ensembles EEE.

Pour compléter le catalogue d'adjacences d'une droite VEE, nous devons considérer les ensembles de droites critiques de type EEE engendrés par les arêtes e_1 et e_2, et les deux arêtes e_4 et e_3 adjacentes au sommet v (figure 3.3(b) et (c)).

La simple description ci-dessus introduit l'idée générale du squelette de visibilité : en déterminant toutes les droites poignardantes extrêmes de la scène, et en leur attachant les ensembles de droites critiques, on peut décrire toutes les relations de visibilité d'une scène 3D. Cette information sera stockée sous forme d'une structure de graphe (cf. figure 3.3), comme nous le verrons dans la section 2.3. Considérons l'exemple de la figure 3.3(a) : le nœud associé à la droite extrême ve_1e_2 est adjacent aux arcs associés aux ensembles de droites critiques ve_1, $ve_1\prime$ et ve_2, ainsi qu'à $e_3e_1e_2$ et $e_4e_1e_2$.

2.2 Le complexe de visibilité 3D, l'asp et le squelette de visibilité

Le *complexe de visibilité* introduit au chapitre précédent est une structure qui elle aussi décrit toute l'information de visibilité d'une scène 3D. Il est lui aussi basé sur les adjacences entre événements visuels et considère des ensembles de segments libres maximaux de la scène.

Les composantes de dimension 0 et 1 du complexe de visibilité sont effectivement les mêmes que les nœuds et les arcs que nous venons de voir, et que nous utiliserons pour la construction du squelette de visibilité. Des constructions similaires ont été présentées (mais pas complètement implémentées à notre connaissance) pour la structure appelée *asp* [PD90] utilisée pour la construction de graphes d'aspect.

Dans les deux cas, des faces de dimension supérieure sont construites. En particulier pour le complexe de visibilité, des ensembles de segments de dimension 2, 3 et 4 sont traités. Par exemple, l'ensemble des segments tangents à un objet a 3 degrés de liberté, etc.

Ces ensembles et leurs adjacences pourraient en théorie se révéler utiles pour certaines requêtes, comme le calcul de vue ou le maintien d'une vue lors du déplacement du point de vue, en particulier pour certaines scènes particulièrement complexes où tous les objets s'occultent les uns les autres, créant de nombreuses droites critiques, ce qui rend utile leur regroupement en faces de dimension supérieure.

Le complexe de visibilité et l'asp sont des structures de données compliquées et dont les algorithmes de construction sont ardus, puisqu'ils demandent de traiter des subdivision de dimension 4 ou 5. De plus, elles sont difficiles à traverser à cause de leurs nombreux niveaux d'adjacences. L'approche que nous proposons ici est différente : nous avons développé une structure qui est facile à implémenter et à utiliser.

Cela explique aussi le nom *squelette de visibilité*, puisque notre structure peut être vue comme le squelette du complexe de visibilité.

2.3 Catalogue d'événements visuels et de leurs adjacences

Le squelette de visibilité est une structure de graphe. Ses nœuds sont les droites poignardantes extrêmes, et ses arcs sont les ensembles de droites critiques (figure 3.3). Dans cette section et dans l'annexe B nous présentons une liste exhaustive de toutes les sortes d'arcs et de nœuds du squelette de visibilité.

Éléments 1D : arcs du squelette de visibilité

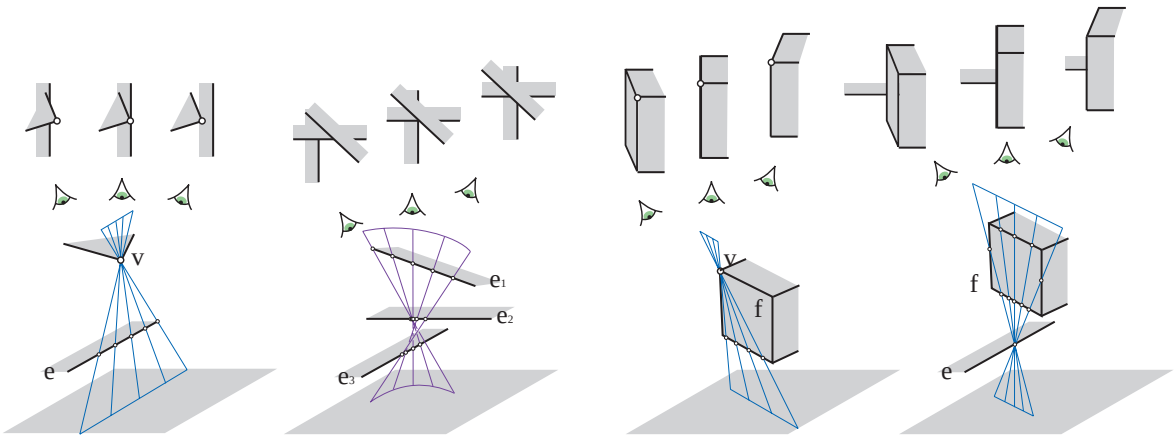


FIG. 3.4: (a) Idem figure 3.1(a) : événement EV. (b) Quand le point de vue traverse la surface critique EEE, l'ordre des intersections des arêtes e_1, e_2 et e_3 dans la vue est modifié. Sur la surface critique, les trois arêtes sont visuellement alignées. (c) À gauche de la surface critique FV, la face f est visible. Elle se réduit à une arête lorsque le point de vue est sur la surface critique, puis devient cachée. (d) Le cas FE est semblable au cas FV.

Sur la figure 3.4, on peut voir les quatre sortes d'éléments à une dimension du squelette : un ensemble de droites critiques EV, un ensemble EEE, et deux surfaces critiques Fv et FE liées aux faces des polyèdres (nous utilisons une minuscule pour les événements Fv car le sommet appartient à la face F génératrice). Dans le haut de la figure est représentée l'évolution d'une vue tandis que chaque surface critique est traversée.

Remarquons que l'interaction d'une arête e et d'un sommet v peut correspondre à plusieurs arcs ve dans le squelette de visibilité, qui ont des polygones différents à leurs extrémités. Ces arcs sont séparés par des nœuds. C'est le cas par exemple des arcs ve_1 et $ve_1\prime$ qui sont adjacents au nœud ve_1e_2 dans la figure 3.3(a).

Éléments 0D : nœuds du squelette de visibilité

Comme nous l'avons expliqué dans la section 2.1, l'incidence de deux surfaces critiques définit une droite poignardante extrême, ce qui correspond dans le squelette de visibilité à un nœud qui est la rencontre de deux arcs. Cette section présente une liste des configurations qui engendrent un nœud et les adjacences correspondantes. Nous donnons une figure pour chaque cas.

Le nœud le plus simple correspond à l'interaction de deux sommets, comme dans la figure 3.5(a).

L'interaction d'un sommet v avec deux arêtes e_1 et e_2 peut intervenir dans deux configurations, selon leurs positions respectives. Le premier de ces nœuds a déjà été présenté figure 3.3, et le second est montré figure 3.5(b).

L'interaction de quatre arêtes est présentée figure 3.6, ainsi que les six arcs EEE qui lui sont adjacents. Les nœuds liés aux faces des polygones sont renvoyés en annexe B : EFE, FEE, FF, E et Fvv (figure B.3 à B.4).

3 Structure de données

Étant donné le catalogue d'adjacences présenté dans la section précédente, nous pouvons présenter les détails de notre structure de données.

Préliminaires : Notre modèle de scène fournit les adjacences entre sommets, arêtes et faces des polyèdres. Avant de construire le squelette de visibilité, nous traversons tous les éléments de la scène et leur affectons un numéro d'identification unique. Cela nous sera utile pour identifier tous ces éléments facilement. De plus, nous considérons que toutes les arêtes sont orientées. Cette orientation est arbitraire et facilite la cohérence dans les calculs qui suivront.

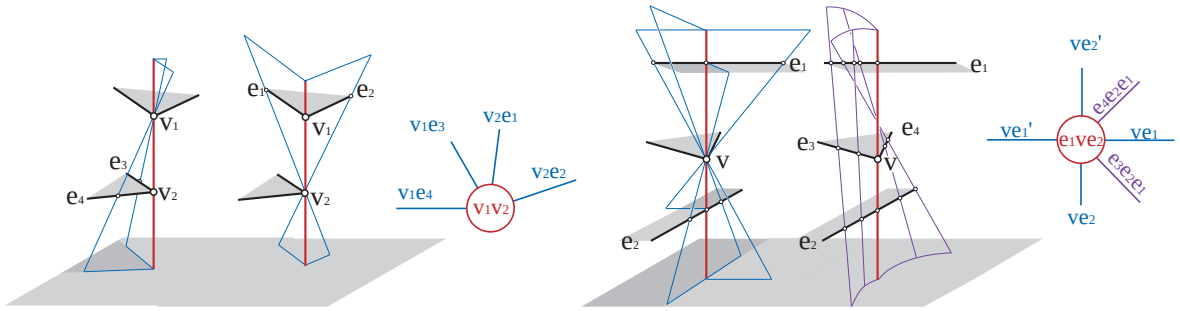


FIG. 3.5: (a) Un nœud VV, v_1v_2, est adjacent à quatre arcs EV engendrés par un sommet et une arête adjacente à ce sommet : v_1e_3 et v_1e_4, et e_1v_2 et e_2v_2. (b) Un nœud EVE, e_1ve_2 : chaque arête définit deux arcs EV avec le sommet v (en fonction du polygone à l'extrémité), et deux arcs EEE sont définis par e_1, e_2 et une arête adjacente à v.

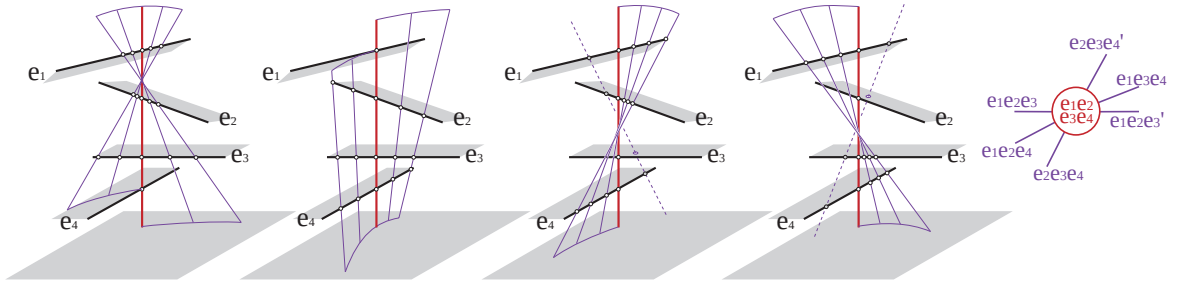


FIG. 3.6: Droite poignardante extrême $E4$ et ses six ensembles de droites critiques EEE adjacents.

3.1 Structure de données initiale

L'élément le plus simple de notre structure de données est un nœud. La structure d'un nœud (appelée *Node*) contient une liste d'arcs, et des pointeurs vers les polygones F_{up} et F_{down} (potentiellement nuls si le nœud a pour extrémité l'infini) qui bloquent la droite poignardante extrême à ses points d'extrémité P_{up} et P_{down} (figure 3.7).

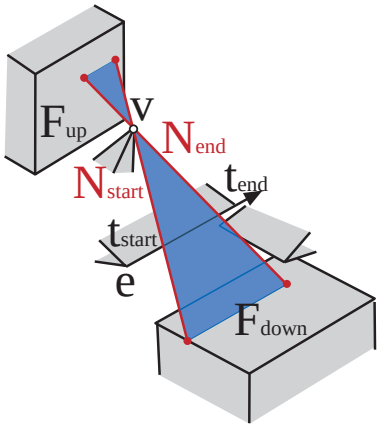
La structure pour un arc (*Arc*) est montrée figure 3.7 pour un arc EV. Il y a deux nœuds adjacents à chaque arc N_{start} et N_{end}. Cette information d'adjacence est stockée avec l'arc.

Les ensembles de droites critiques sont paramétrés selon un paramètre t qui est pris sur l'une des arêtes génératrices (on munit chaque arête d'un repère arbitraire). S'il y a ambiguïté à cause de la présence de plusieurs arêtes génératrices (cas des EEE), on choisit l'arête ayant le plus petit identificateur. Cette paramétrisation est très importante pour l'insertion des nœuds lors de la construction. Remarquons qu'une droite poignardante extrême a plusieurs paramètres, en fonction de l'arc que l'on considère. Chaque arc est caractérisé par les valeurs t_{start} et t_{end} de ses droites extrêmes adjacentes.

Les détails des structures *Node* et *Arc* sont donnés figure 3.7(b).

Pour accéder efficacement à l'information des arcs, nous maintenons un tableau d'arbres binaires de recherche pour chaque type de surface critique. Par exemple nous utilisons un tableau *ev* d'arbres stockant des arcs EV (cf. figure 3.7(b)). Ces tableaux sont indexés par l'identificateur unique des éléments de la scène aux extrémités des arcs. Il peut s'agir de faces ou de l'infini, mais aussi de sommets ou d'arêtes dans le cas de surfaces rentrantes, c'est-à-dire dont les droites traversent le polyèdre adjacent à l'un de ses générateurs.

Ces tableaux avec arbres de recherche permettent des requêtes efficaces aux arcs quand nous insérons de nouveaux nœuds ou quand nous effectuons une requête de visibilité. Des arbres binaires équilibrés sont utilisés, ils sont ordonnés par l'identificateur des générateurs et par la valeur de t_{start}. La figure 3.8(a) reprend notre structure de données initiale.

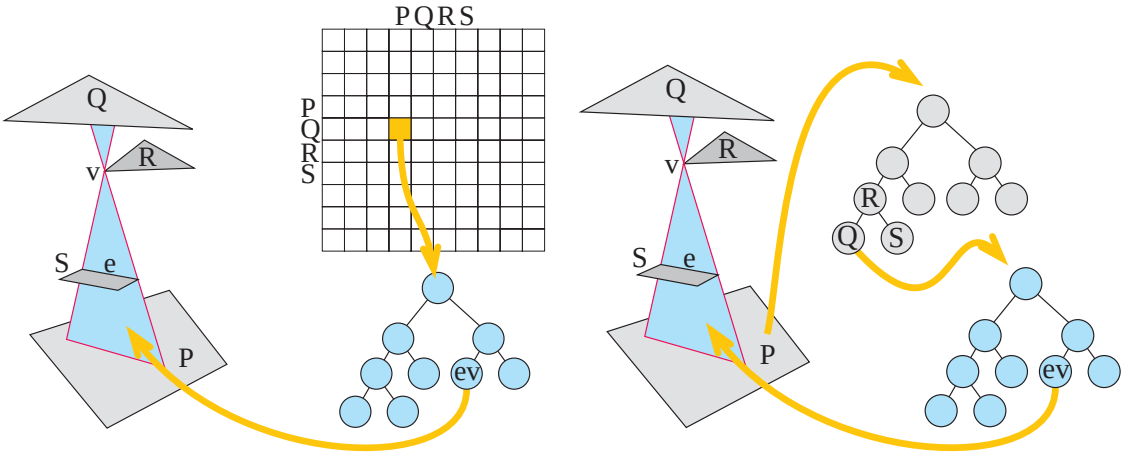



```
class Node {
    List<Arcs> adjacentArcs
    Face F_up, F_down
    Point3D P_up, P_down
}

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)

FIG. 3.7: Structures de base du squelette de visibilité.



(a) (b)

FIG. 3.8: (a) Résumé de la structure de données initiale du squelette de visibilité. Un tableau à deux dimensions, indexé par les polygones, stocke des arbres binaires de recherche contenant les arcs du squelette. (b) Résumé de notre structure de données pour le squelette de visibilité avec optimisation de la mémoire. Chaque polygone stocke un arbre binaire de recherche indexé par les polygones qu'il voit. Pour chaque paire de polygones mutuellement visibles, un arbre binaire de recherche des arcs est stocké.

3.2 Réduction du coût mémoire

Le stockage des arcs du squelette de visibilité dans un tableau à deux dimensions impose un coût $O(n^2)$ en mémoire. Dans les scènes que nous présentons en section 5, la moitié du coût mémoire du squelette est due à ce tableau, dont plus de 95% des cases sont vides ! C'est encore pire si la scène présente une grande densité d'occultation, comme dans le cas d'un bâtiment où chaque pièce ne peut voir qu'un nombre fixe de pièces : le nombre d'arcs du squelette est alors $O(n)$. De plus, pour les applications à la simulation de l'éclairage que nous présenterons dans le chapitre suivant, nous aurons besoin de subdiviser les polygones initiaux et de calculer et stocker de l'information de visibilité relative à ces sous-polygones, mais pas forcément vis à vis de tous les autres sous-polygones.

Pour ces raisons, nous proposons de stocker l'ensemble des surfaces critiques entre deux polygones au niveau des polygones eux-mêmes. Chaque polygone P stocke un arbre binaire de recherche ; chaque nœud de cet arbre contient l'ensemble des arcs entre P et un autre polygone Q. Cet ensemble est lui-même organisé en arbre de recherche (figure 3.8(b)). Chaque ensemble d'arcs est référencé deux fois : une fois sur P et une fois sur Q. De la même façon, chaque sommet v possède un arbre de recherche qui contient tous les arcs entre v et tout polygone Q (ce qui représente la vue de Q depuis v). Ces ensembles d'arcs sont liés à la notion de *lien* en radiosité hiérarchique, comme nous le verrons dans le chapitre suivant.

Pour nos scènes de test, cette approche nous a permis d'économiser environ 30% sur la taille totale du squelette de visibilité. De plus, nous avons effectué des expériences sur une série de scènes contenant la même pièce répliquée 2, 4 et 8 fois, et avons obtenu un accroissement mémoire grossièrement linéaire. L'utilisation d'arbres binaires de recherche à la place d'un tableau ajoute un coût d'accès en $O(\log n)$, mais cela n'était pas perceptible dans nos tests.

Nous verrons dans le chapitre suivant que le squelette de visibilité peut être utilisé pour la simulation de l'éclairage. Pour permettre la subdivision des surfaces nécessaire à la représentation de la fonction d'éclairement, de l'information de visibilité doit être calculée pour des triangles issus de la subdivision et les sommets intérieurs correspondants. Puisque l'information de visibilité est maintenant stockée au niveau des polygones et non plus dans un tableau 2D, la généralisation aux sous-polygones est immédiate. Sur chaque sous-polygone ou sous-sommet nous stockons l'information de visibilité pour les autres polygones avec lesquels il interagit.

4 Algorithme de construction

4.1 Détection des nœuds

Avant de présenter la construction proprement dite, nous évoquons brièvement la question de la visibilité locale. Tout comme cela a été proposé dans d'autres travaux (*e.g.* [GM90]), pour chaque arête adjacente à deux faces d'un polyèdre, l'intersection des deux demi-espaces négatifs des polygones est invisible localement depuis l'arête. Quand nous calculons les interactions d'une arête e, nous n'avons donc pas besoin de considérer tout autre arête e' qui est "derrière" les polygones adjacents à e. Cela permet d'éliminer un grand nombre d'événements potentiels.

Le principe général de la détection des nœuds consiste tout d'abord à trouver une droite poignardante extrême potentielle engendrée par un ensemble de générateurs (c'est-à-dire une droite qui passe par tous les générateurs). On teste ensuite si un objet se trouve entre les générateurs, auquel cas la droite est annulée à cause de l'occultation. C'est la différence entre la visibilité selon les droites et selon les segments telle que nous l'avons discutée dans le chapitre précédent.

Nœuds triviaux

Les nœuds les plus simples à traiter sont les VV, Fvv et Fe. Nous effectuons simplement une boucle sur les générateurs potentiels (sommets, arêtes et faces). Les droites correspondantes sont ensuite intersectées avec la scène en utilisant un simple lancer de rayon, afin de déterminer si un objet cause une occultation entre les générateurs. Si c'est le cas, la droite extrême est annulée, sinon un nouveau nœud est créé. Le lancer de rayon donne de plus les polygones aux extrémités de la droite poignardante extrême, et donc la place des arcs adjacents dans le tableau à double entrée.

Nœuds VEE et EEEE

Considérons deux arêtes de la scène e_i et e_j. Toutes les droites qui passent par ces deux segments sont à l'intérieur d'un tétraèdre étendu (ou double coin) représenté figure 3.9(b), défini par quatre plans. Chacun de ces plans est défini par une des arêtes et un sommet de l'autre arête.

Pour déterminer les sommets de la scène qui peuvent éventuellement engendrer une droite poignardante extrême VEE ou EVE, nous ne devons considérer que les sommets contenus dans ce tétraèdre étendu. Si un sommet de la scène est à l'intérieur du tétraèdre étendu, il y a alors une droite extrême VEE ou EVE potentielle.

Considérons ensuite une troisième arête e_k. Si e_k coupe l'un des plans du tétraèdre, un nœud VEE ou EVE potentiel est détecté : si l'arête e_k intersecte le plan défini par l'arête e_i et le sommet v de e_j, alors une droite potentielle ve_ie_k ou e_ive_k est créée (figure 3.9(b)).

Nous appelons le calcul des intersection de e_k avec le tétrahèdre étendu sa *restriction*. Il ne peut y avoir au plus que 3 segments à l'intérieur du double coin dans la restriction d'une arête, car il ne peut y avoir au plus que 4 intersections avec le double coin. De plus, les calculs des limites de la restriction d'une arête nous donnent également les droites VEE ou EVE potentielles engendrées par e_k, l'une des arêtes e_i ou e_j et un sommet de l'autre arête.

La restriction d'une arête e_k correspond également à la restriction de l'ensemble de droites critiques $e_ie_je_k$ par rapport à celui généré par les droites entières supports de e_i, e_j et e_k.

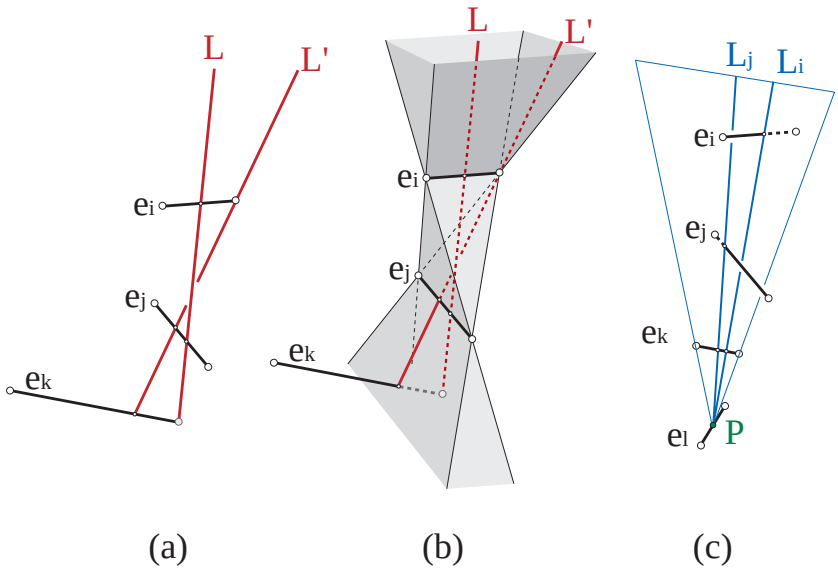


FIG. 3.9: (a) (b) Énumération des VEE et restriction des EEE. (c) Calcul des $E4$: il s'agit de trouver la droite passant par $e_je_ke_l$ qui passe aussi par $e_le_ke_i$.

Intéressons-nous maintenant au calcul des nœuds $E4$. L'intersection de e_k avec le double coin a restreint e_k. Pour obtenir une droite passant par e_i, e_j et e_k, nous ne devons considérer que les segments de e_k à l'intérieur du double coin. Ce processus de restriction est également appliqué à une quatrième arête e_l, qui est ensuite également restreinte par le double coin de e_i et e_k, puis par celui de e_j et e_k. Ces restrictions successives éliminent un grand nombre de candidats.

Une fois que la restriction est effectuée, nous avons deux ensembles critiques EEE : celui engendré par e_l, e_i et e_j, et celui engendré par e_l, e_i et e_k. Une simple recherche dichotomique est ensuite utilisée pour trouver (s'il existe) le point sur e_l par lequel passe une droite extrême de type $E4$ $e_ie_je_ke_l$. Cette dichotomie est faite sur un point P de e_l, en cherchant le zéro de l'angle formé par les deux droites L_i et L_j définies par l'intersection du plan (P, e_i) avec e_j et avec e_k (figure 3.9(c)).

Un algorithme plus robuste tel que celui présenté par Teller et Hohmeyer [TH91] pourrait être utilisé. Cependant, notre algorithme simple a semblé donner des résultats satisfaisants en pratique. C'est le cas principalement parce que nous ne cherchons pas des droites qui passent par des droites infinies, mais par des segments restreints au préalable. L'algorithme d'énumération des VEE et des $E4$ est décrit figure 4.1.

Accélération à l'aide d'une subdivision de l'espace

Nous avons développé une méthode d'accélération pour éviter l'énumération de tous les triplets d'arêtes. Pour chaque paire d'arêtes, nous rejetons la plupart des troisièmes arêtes potentielles rapidement grâce à une grille régulière. Au lieu de tester si chaque cellule de la grille est à l'intérieur du tétraèdre étendu, nous utilisons sa projection sur les trois plans alignés aux axes. Nous projetons le tétraèdre étendu (ce qui donne une forme de sablier) sur chacun des trois plans. Pour calculer cette projection, nous calculons en fait le sablier en 2D

```
enumeration des VEE et des E4 potentielles
{
foreach edge e_i from 1 to n
   foreach edge e_j from i+1 to n locally visible
      foreach edge e_k from j+1 to n locally visible
         compute the EEE restrictions e_i e_j e_k
      foreach edge e_k from j+1 to n locally visible
         foreach segment of its restrictions
            foreach edge e_l from k+1 to n locally visible
            foreach segment of its restrictions
               search for E4
}
```

(a)

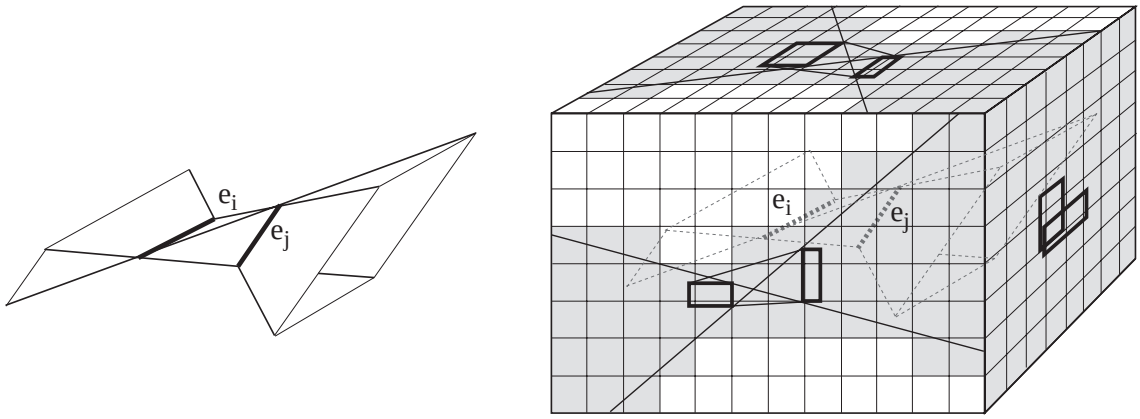
```
restriction des EEE
{
foreach of the 4 planes
   compute the intersection inter
   with the line of the edge
      if it inter on the edge
         propose a VEE
         restrict the edge
foreach of the edge endpoints s_ep
   if s_ep is inside the double wedge
      propose a VEE
      restrict the edge
}
```

(b)

FIG. 3.10: Énumération des $E4$ et des VEE potentielles.

à l'aide des boîtes englobantes des deux arêtes e_i et e_j. Le sablier 2D est défini par les droites séparatrices et supports des projections des boîtes englobantes. Ces sabliers 2D sont discrétisés à la résolution de la grille de manière prudente (tout pixel qui contient un point du sablier est marqué valide).

Nous calculons ensuite l'intersection arête e_k-tétraèdre étendu (la restriction) uniquement pour les arêtes qui se trouvent dans des cellules de la grille dont les trois projections sont à l'intérieur des sabliers. Ce processus est résumé figure 3.11).



(a) (b)

FIG. 3.11: Accélération à l'aide d'une grille régulière et de sabliers. (a) Deux arêtes et le tétraèdre étendu correspondant. (b) Nous utilisons la projection selon les trois axes de la grille régulière. La projection des boîtes englobantes des arêtes est utilisée pour calculer la forme du sablier. Les cellules valides sont celles qui se projettent sur les pixels (en gris) des sabliers sur chacun des trois plans.

Nœuds non triviaux engendrés par des faces

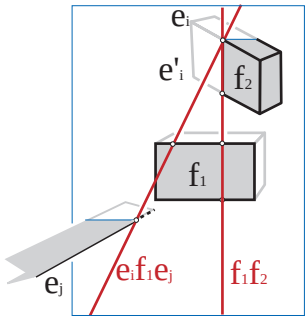
Pour calculer les nœuds non-triviaux engendrés par des faces (FvE, FF et FEE), nous commençons par calculer l'intersection du plan support de chaque face f_1 avec toutes les arêtes de la scène. Pour toute arête qui coupe ce plan, nous essayons de créer un nœud FvE (figure B.3). C'est-à-dire que nous effectuons le test d'occultation (y a-t-il un objet entre la face et l'arête ?), et si aucune occultation n'est détectée, nous créons effectivement le nœud.

Pour chaque paire d'arêtes intersectant le plan de la face, nous regardons si une droite extrême FEE potentielle existe. Pour ceci, nous déterminons si la droite joignant les deux intersections passe par la face f_1. Si c'est le cas, la droite FEE potentielle est ensuite passée au test d'occultation.

Pour finir, nous vérifions si des nœuds FF potentiels existent. Cela se produit si les faces adjacentes à une arête qui intersecte le plan engendrent une droite FF (face f_2 dans la figure 3.12 (b)). La construction des droites extrêmes FEE et FF est décrite figure 3.12.

```
Detection des noeuds lies aux faces {
foreach face f1 of the scene
   foreach edge e of the scene
      compute the intersection of the edge e with the plane of f1
      foreach intersection Pi
         create a FvE
         foreach intersection Pj
            if (PiPj) intersects f1
               create FEE
         foreach of the 2 faces f2 adjacent to the edge of Pi
            find Pj the intersection of a second edge of f2 with f1
            if (PiPj) intersects f1
               create FF
}
```

(a)



(b)

FIG. 3.12: Détection des nœuds liés aux faces.

4.2 Création des arcs

```
Creation d'un noeud du squelette de visibilite
{
   foreach adjacent arc n
      compute t
      foreach arc a with same extremities et same generators
         if a → t_start < t < a → t_end
         AjouteNoeudAArc(n, a)
      if no arc found
         create new Arc
}
```

```
AjouteNoeudAArc(Node n, Arc a)
{
   pos = decideStartOrEnd(n, a)
   if pos in a undefined
      set pos to n
   else
      split a into two parts
}
```

FIG. 3.13: Création d'un nœud.

La création des arcs du squelette de visibilité est effectuée simultanément à la détection des nœuds. Lorsqu'un nouveau nœud est inséré, nous traitons tous les arcs a adjacents grâce au catalogue présenté section 2.3. Si l'arc a déjà été créé (lors de la création d'un nœud précédent) nous le relions simplement au nouveau nœud, sinon nous le créons et le relions au nœud (il a donc son autre extrémité provisoirement pendante, dans l'attente de l'insertion d'un autre nœud).

Pour cela, pour chacun de ces arcs a, nous calculons le paramètre t (cf. section 3.1) qui correspond au nouveau nœud, puis nous faisons comme expliqué en figure 4.2. Nous accédons aux arcs a' du squelette de visibilité qui ont les mêmes extrémités que a (et donc qui se trouvent dans la même case du tableau ou dans le même arbre de recherche) et qui ont les mêmes générateurs (sommets, faces, arêtes). Si la valeur de t indique que que le nouveau nœud est contenu dans a', c'est que a avait déjà été créé et est égal à a'. Nous devons ensuite déterminer si le nœud est un nœud de début ou de fin pour cet arc (en fonction de la paramétrisation choisie). Cela sera expliqué plus en détails dans le paragraphe suivant. Si cette position est déjà occupée, nous coupons l'arc en deux, sinon nous attachons le nœud à l'extrémité correspondante de l'arc.

Nous avons vu ci-dessus que pour chaque arc adjacent à un nœud, nous devons savoir s'il s'agit d'un nœud de *début* (*start*) ou de *fin* (*end*) par rapport à la paramétrisation de l'arc. C'est-à-dire que l'on veut savoir si le nœud bloque l'arc dans le sens des t croissants ou décroissants. Dans certains cas, cette opération est triviale. Par exemple pour un nœud v_1v_2 et l'un des ses arcs adjacents v_1e, il suffit de déterminer si v_2 est le sommet de début ou de fin de e (rappelons que les arêtes sont arbitrairement orientées et que la paramétrisation t est basée sur un repère lié aux arêtes). Dans d'autres cas, cette détermination peut être plus ardue, en particulier pour les nœuds $E4$. Nous avons renvoyé la description des critères pour ces configurations dans la table 2 de l'annexe B.

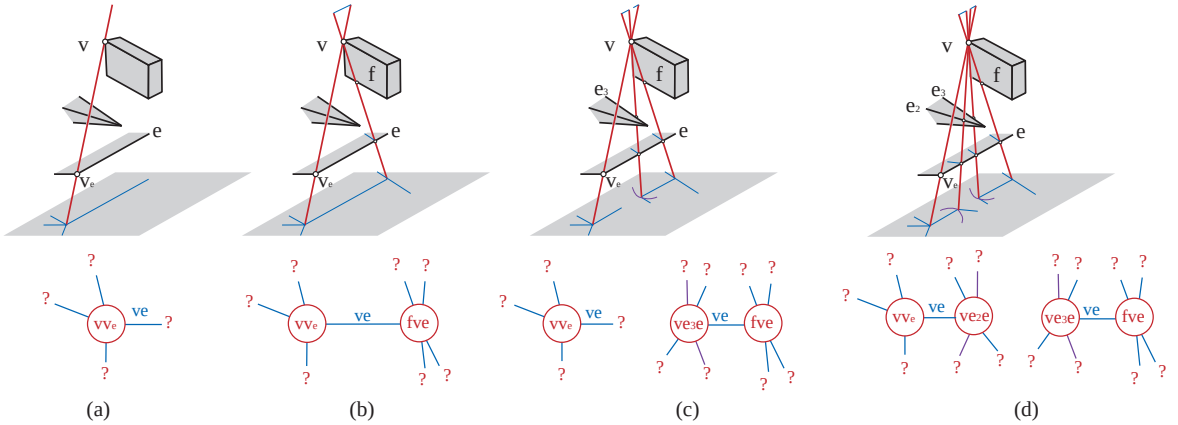


FIG. 3.14: Exemple d'insertion de nœuds. (a) Insertion du nœuds vv_e. (b) Insertion du nœuds f_ve. L'arc ve a maintenant deux nœuds extrémités. (c) Insertion du nœuds ve_3e. L'arc ve est coupé en deux. (d) Insertion du nœuds ve_4e, les deux arcs ve ont leurs deux véritables nœuds-extrémités.

Dans la figure 3.14, nous illustrons par un exemple notre algorithme de construction. Au début, un nœud trivial vv_e est créé. Le second nœud détecté est vfe, qui est adjacent à vv_e et vfe. Le troisième nœud est vee_3. Quand ce nœud est inséré, nous réalisons que le nœud de début de ve existe déjà, nous coupons donc ve en deux. L'arc ve issu de la subdivision qui est connecté à vv_e a un nœud extrémité indéfini. La dernière insertion que nous montrons est celle de ve_2e qui remplit ce vide.

4.3 Traitement des configurations dégénérées

La géométrie algorithmique fait souvent l'hypothèse que les scènes sont en configuration générique. Malheureusement, en synthèse d'images, les scènes sont la plupart du temps hautement dégénérées : de nombreux points sont alignés, des segments et des faces sont parallèles, coplanaires, et les objets se touchent les uns les autres.

Cela engendre des événement visuels dégénérés ; par exemple des droites poignardantes extrêmes VVV passant par trois sommets, ou des $E5$ qui passent par cinq arêtes (ces événement dégénérés interviennent également dans le traitement des scènes dynamiques discuté en section 7).

Ces configurations dégénérées sont la cause de la double définition de certaines surfaces critiques, et surtout d'instabilités numériques pour les tests d'occultation pour les droites extrêmes potentielles. Une droite extrême peut être supprimée de manière aléatoire, ce qui peut entraîner des incohérences dans le voisinage du nœud correspondant dans le graphe. Une politique cohérente doit donc être définie pour décider d'inclure ou non de tels nœuds et leurs arcs adjacents.

Tout d'abord, nous devons détecter ces problèmes. Quand un test d'occultation est effectué sur une droite extrême potentielle, nous vérifions aussi la présence d'objets rasants (en particulier des arêtes colinéaires à la droite ou des faces qui lui sont coplanaires). Cela nécessite une simple modification du test de d'appartenance à un polygone pour le lancer de rayon (une tolérance ε est donnée). Nous pouvons ainsi détecter l'intersection avec une arête ou un sommet silhouette.

Nous devons ensuite traiter la droite extrême qui a été déclarée dégénérée. Une première possibilité serait de créer explicitement un catalogue de configurations dégénérées et des adjacences appropriées. Mais à cause du trop grand nombre de cas différents, cette approche devient vite impossible à implémenter. Nous avons choisi de toujours considérer la situation proche non dégénérée la plus simple, c'est-à-dire celle pour laquelle on crée

le plus petit nombre d'événements visuels. Par exemple, si quatre arêtes E_1, E_2, E_3 et E_4 sont parallèles et dans cet ordre, nous considérons que E_3 cache E_2, etc. Les configurations à traiter sont donc plus simples et utilisent le catalogue d'adjacences standard du squelette de visibilité. Les problèmes de précision numérique sont gérés avec des ε cohérents.

Nous ne prétendons cependant pas que les problèmes de robustesse et de précision numérique ont ainsi reçu une solution définitive, générale et fiable, ainsi que nous en discuterons à la fin de ce chapitre. Notre implémentation a cependant permis le calcul du squelette de visibilité pour des scènes contenant des dégénérescences typiques en synthèse d'images.

5 Implémentation et résultats

Nous avons mené à bien une implémentation des structures de données et algorithmes décrits dans ce chapitre. Nous avons utilisé notre système sur une série de scènes de test : nous avons calculé le squelette de visibilité pour des scènes comportant jusqu'à 1500 polygones.

Dans ce qui suit, nous présentons tout d'abord les statistiques pour le squelette de visibilité sur nos différentes scènes. Nous démontrons ensuite la flexibilité de notre technique en présentant les réponses à différentes requêtes de visibilité globale.

5.1 Implémentation et statistiques de construction

Notre implémentation requiert pour l'heure des polyèdres convexes en entrée. Ceci n'est pas une limitation de l'approche, nous utilisons la convexité uniquement pour simplifier certains tests de visibilité locale.

Nous traitons les objets en contact en détectant ce genre de configuration lors d'un précalcul, et en modifiant légèrement notre lancer de rayon (pour éviter qu'un rayon ne "passe" entre deux objets en contact). Nous rejetons également les triplets d'arêtes coplanaires pour la restriction des EEE. D'autres dégénérescences comme le arêtes qui se coupent ou les interpénétrations d'objets ne sont pas encore traitées par notre implémentation.

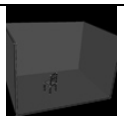
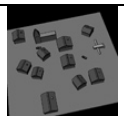
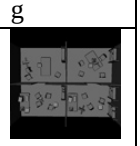
Nous présentons des statistiques sur la taille de notre structure de données et sur les temps de construction dans le tableau 5.1 et sur le graphique 3.15. Bien entendu, ces tests ne doivent être pris que comme une simple indication du comportement asymptotique de notre algorithme. cependant, nous observons une croissance quadratique de la mémoire, et une croissante super-quadratique du temps de calcul avec environ $n^{2,4}$ en moyenne, où n est le nombre de polygones.

Les nœuds VEE sont en fait les plus nombreux. Il y a environ cent fois moins de $E4$, bien qu'il puisse théoriquement y en avoir un ordre de grandeur de plus.

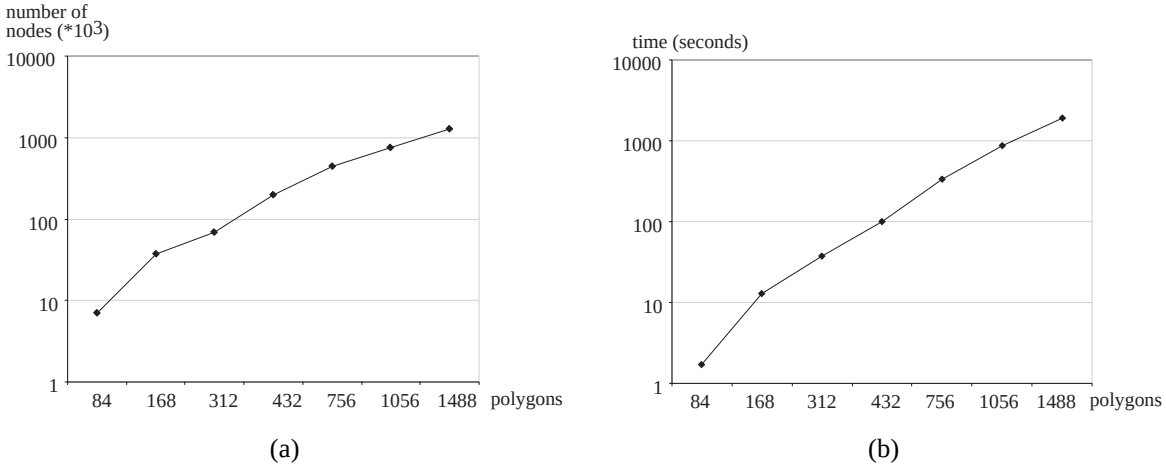
Avec la structure de données initiale (*i.e.* avec les tableaux 2D), un grand pourcentage de la mémoire utilisée est due à ces tableaux (pour la scène (d) du tableau 5.1, le tableau utilise 53.7Mo sur un total de 135Mo). Comme ces tableaux sont très vides (99.3% de cellules vides pour la scènes (d)), la mémoire utilisée est notablement réduite par notre structure améliorée qui remplace les tableaux par des arbres stockés au niveau des polygones. Un gain de 30% a été observé en moyenne pour nos scènes de test.

Dans le cas de scènes où l'occultation est dense, la mémoire requise croît plus lentement, plus proche de linéaire que quadratique par rapport au nombre de polygones en entrée. Par exemple, nous avons répliqué la scène (a) 2, 4 et 8 fois, pour obtenir des pièces isolées contenant une chaise chacune. La mémoire utilisée (sans le tableau) est alors 1,2 Mo, 2,8 Mo, 8,6 Mo et 17,3 Mo, pour respectivement 78, 150, 300 et 600 polygones. La structure améliorée présente une croissance linéaire.

Les bornes théoriques sont bien plus pessimistes, $O(n^4)$ en taille car chaque quadruplet d'arêtes peut engendrer deux droites $E4$ potentielles [TH91], et $O(n^5)$ en temps car chacune de ces droites extrêmes potentielles implique un lancer de rayon sur la scène entière. Cependant, comme nous l'avons vu dans le chapitre précédent, de telles bornes ne peuvent être atteintes que pour des scènes très peu courantes ("exotiques") comme les grilles orthogonales ou les droites infinies. Il est clair que dans ces cas, notre algorithme de construction se révélerait inefficace. Des algorithmes plus efficaces sont alors possibles (comme l'algorithme de construction du complexe de visibilité en entier présenté dans le chapitre précédent, qui tourne en temps $O(k + n^3 \log n)$ où k est la taille du complexe, qui est aussi celle du squelette). Mais ces approches souffrent des problèmes de robustesse décrits en section 2.2 ou dans le chapitre précédent.

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

TAB. 3.1: Statistiques de la construction du squelette réalisées sur une SGI Onyx 2 avec un R10000 à 195Mhz. La place mémoire correspond à la structure de données initiale (*i.e.* en incluant le tableau 2D).



(a) (b)

FIG. 3.15: (a) Graphique du nombre de nœuds (en double échelle logarithmique). La pente d'environ 2 indique une croissance quadratique. (b) Graphique du temps de calcul (en double échelle logarithmique). La pente d'environ 2,4 indique une croissance en $n^{2,4}$.

Quant à la robustesse de notre méthode, il faut remarquer que les techniques précédentes de graphe d'aspect ou de maillage de discontinuité dépendaient largement de la construction d'un arrangement global (pour le maillage ou sur la sphère des directions). Ce n'est pas le cas ici, toutes les opérations sont locales. Comme nous n'effectuons que du lancer de rayon et des intersections de plans, les problèmes numériques sont limités par rapport à l'intersection d'une surface (parfois quadrique réglée, pour les EEE) avec toute la scène. Des dégénérescences peuvent éventuellement poser des problèmes, mais, grâce à la localité de notre méthode, cela n'affectera pas la construction ailleurs. Des algorithmes plus efficaces à base de balayage sont bien plus sensibles à de telles instabilités, puisqu'une erreur à un moment donné se répercute sur toute la suite de l'exécution de l'algorithme et rend le résultat complètement incorrect, si tant est que le programme n'ait pas échoué en cours de route.

5.2 Facteur de forme point-polygone aux sommets

Le calcul de facteurs de formes est devenu central pour bien des méthodes de simulation de l'éclairage. Le facteur de forme entre deux surfaces est la proportion de lumière quittant l'une des surfaces qui arrive à l'autre. Dans bien des systèmes de simulation de l'éclairage global par la méthode de radiosité, des facteurs de forme entre points et surfaces sont utilisés comme approximation des calculs surface-surface [CG85, BRW89] ; dans d'autres l'éclairement est calculé directement aux sommets [WEH89].

Dans les études précédentes, qu'elle soient théoriques [LSG94] ou expérimentales [DS96], il a été montré que l'erreur dans le calcul de visibilité est prédominante. C'est particulièrement le cas lorsque le lancer de rayon est utilisé. Lischinski *et al.* [LSG94] ont développé une technique très prometteuse pour contrôler l'erreur

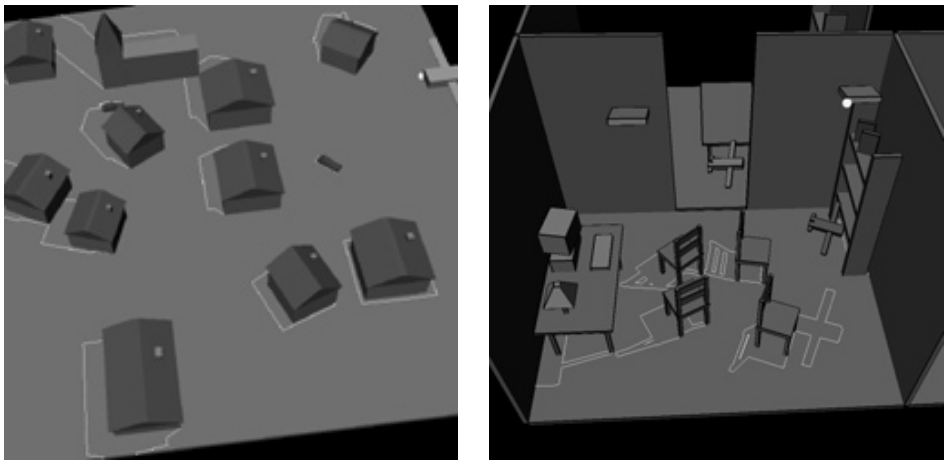
commise lors de la simulation d'un transfert lumineux pour la radiosité hiérarchique. Pour qu'il soit utile dans des environnements généraux, un accès à l'information exacte de visibilité entre un point et les surfaces avec lesquelles il interagit est nécessaire. Cette information est par essence globale, puisque toute paire d'éléments de surfaces de la scène peut être en interaction.

Le squelette de visibilité permet pour l'heure de répondre à cette requête de manière exacte pour tout sommet initial de la scène.

Pour calculer la vue d'une face polygonale f depuis un sommet v, nous accédons à tous les arcs EV relatifs à f. Il s'agit simplement de parcourir la ligne du tableau qui correspond à f (ou l'arbre de recherche stocké en f pour la structure améliorée). Pour chaque arbre de recherche correspondant à f, nous recherchons les arcs engendrés par v. Ces arcs correspondent exactement aux limites de la partie visible de f depuis v. Cette requête est effectuée efficacement car les arcs sont triés dans les arbres de recherche selon un ordre lexicographique dont les générateurs sont la première clef. En particulier, pour les arcs EV, nous avons choisi d'utiliser l'identificateur du sommet avant celui de l'arête.

La figure 3.16(a) et (b) montre des exemples de requêtes de partie visible depuis un sommet. Pour ces scènes, contenant respectivement 312 et 1488 polygones, une requête prend 1.2 ms et 1.5 ms (sans compter le temps pris pour l'affichage du résultat).

Dans le chapitre suivant, nous présentons un algorithme de simulation de l'éclairage global qui utilise le squelette de visibilité pour calculer des facteurs de forme point-polygone exacts.



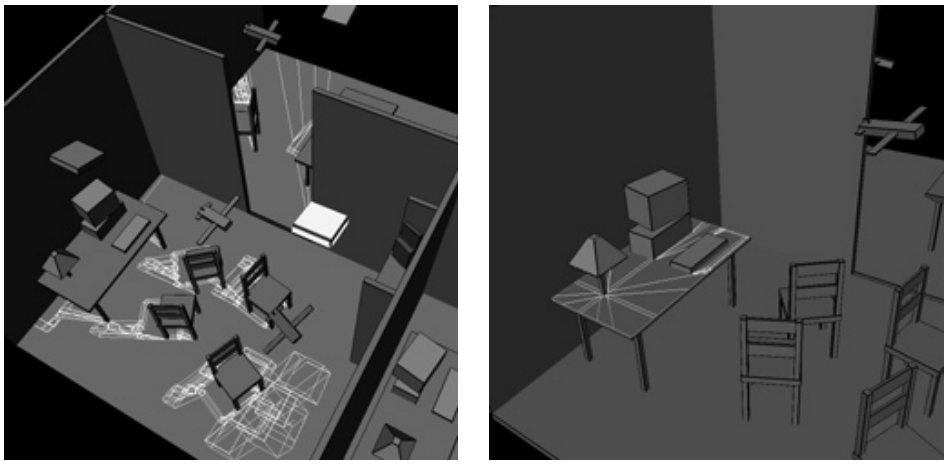
(a) (b)

FIG. 3.16: (a) Partie du sol visible depuis l'un des sommets de l'avion. (b) Partie du sol visible depuis l'un des sommets de la source de droite.

5.3 Maillage de discontinuité global et à la volée

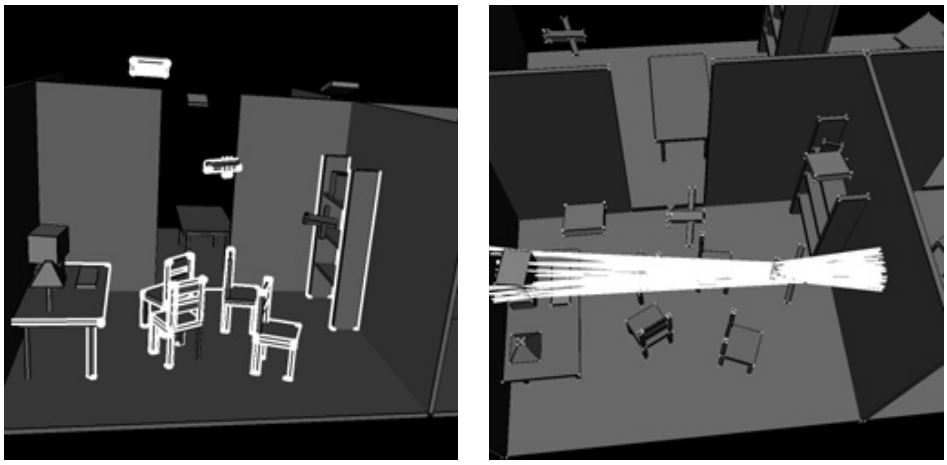
Dans les calculs de radiosité, il est en général bénéfique de subdiviser le maillage d'une surface selon certaines [Hec92a, LTG93] ou toutes [DS96] les discontinuité de la fonction d'éclairement, c'est-à-dire les surfaces critiques. La construction partielle [Hec92a, LTG92] ou complète [DF94, SG94] d'un tel maillage a été jusqu'alors restreinte à une source primaire de lumière (en général un petit polygone). Pour toutes les autres interactions lumineuses dans la scène, la complexité algorithmique et les problèmes inhérents de robustesse n'ont pas permis l'utilisation de ces techniques [Tam93].

Pour bien des transferts secondaires dans un environnement, la construction d'un maillage de discontinuité global (depuis toute surface émettrice ou réfléchissante vers toute surface réceptrice) peut améliorer la précision des calculs. Cela a été montré par Hardt et Teller [HT96] dans leur système de radiosité de haute qualité. Dans leur cas, les surfaces de discontinuité sont juste intersectées avec les polygones de la scène pour la subdivision, et l'information de visibilité portée par la surface critique n'est pas utilisée. De plus les surfaces critiques EEE ne sont pas traitées. Grâce au squelette de visibilité, le maillage de discontinuité global complet peut être calculé efficacement pour toute paire de surface.



(a) (b)

FIG. 3.17: (a) Maillage de discontinuité par rapport à la source de droite. (b) Maillage de discontinuité créé par la lampe sur la table.



(a) (b)

FIG. 3.18: (a) Liste des bloqueurs entre le sol et la lampe de gauche. Remarquons que les objets sur la table ne sont pas déclarés bloqueurs bien qu'ils appartiennent à l'enveloppe convexe du sol et de la lampe. (b) Limites de l'occultation causée par une partie de l'avion entre l'écran d'ordinateur et le mur de droite.

Considérons un polygone f_i comme source lumineuse et un polygone f_j comme récepteur. Les surfaces de discontinuité causées par f_i sur f_j correspondent, d'une part, aux arcs du squelette qui ont un générateur sur la source f_i et une extrémité sur f_j et d'autre part, à ceux qui ont une extrémité sur f_i et une sur f_j. Cette dernière catégorie correspond exactement aux arcs stockés dans la case du tableau indexée par f_i et f_j (ou dans le nœud de l'arbre de recherche correspondant à f_i et f_j pour la structure améliorée). La première catégorie correspond à des arcs stockés dans la ligne du tableau indexée par f_j. Ils peuvent être trouvés grâce aux arbres de recherche de manière similaire à l'extraction de vue.

Cependant, dans le cas du calcul de vue, un seul type de générateur est recherché (un sommet) et un arc EV ne peut être généré que par un seul sommet ; alors que pour le maillage de discontinuité, un arc EEE est engendré par trois arêtes. L'énumération des arcs pertinents ne se ramène plus à une simple recherche sur la première clef dans l'arbre de recherche. Des arbres de recherche multi-critères pourraient être utilisés, mais nous avons choisi une approche plus simple.

Nous ajoutons un tableau à deux entrées, $DM(i,\ j)$, qui stocke tous les arcs ayant un générateur sur f_i et une extrémité sur f_j. Pour extraire le maillage de discontinuité entre f_i et f_j nous accédons simplement à $DM(i,\ j)$ et à la case du tableau stockant les arcs ayant pour extrémités f_i et f_j, et nous considérons les arcs contenus. La

figure 3.17(a) montre le maillage de discontinuité entre la source et le sol (extrait en 28.6 ms). Le maillage créé par la petite lampe sur le bureau de la figure 3.17(b) est calculé en 1.3 ms (rappelons que l'arrangement n'est pas construit).

Le tableau supplémentaire $DM(i,\ j)$ est bien entendu remplacé par des arbres de recherche stockés au niveau des polygones dans le cas de la structure améliorée. Il se révélera particulièrement utile pour la simulation de l'éclairage, car il permet de conserver toute l'information pertinente pour un transfert de lumière au niveau des polygones concernés.

5.4 Liste exacte de bloqueurs, détection d'occultation

Pour traiter l'interaction entre deux faces f_i et f_j, il est courant d'avoir besoin de la liste des bloqueurs qui cachent une face depuis l'autre. C'est en particulier le cas de l'approche de maintenance de liste de bloqueurs de Teller et Hanrahan [TH93].

Le squelette de visibilité permet là aussi de répondre à cette requête de manière exacte et efficace. Nous utilisons la case du tableau initial contenant les arcs ayant pour extrémités f_i et f_j, ainsi que la case $DM(i,j)$, et parcourons les arcs correspondants. Tous les polygones contenant un générateur d'un tel arc sont des bloqueurs. Il est important d'observer que nous obtenons la liste *exacte* des bloqueurs, par opposition aux autres méthodes. Par exemple, la figure 3.18(a) montre les bloqueurs entre la lampe et le sol, extraits en 4 ms.

Les méthodes à base de "tubes" (*shaft*) [HW91] considéreraient tous les objets sur la table comme bloqueurs, alors qu'ils sont occultés par la table.

Lors de la construction du squelette de visibilité, nous calculons en fait également les paires d'objets mutuellement visibles : si deux objets se voient, il y aura au moins une droite poignardante extrême qui les touchent, eux ou leurs sommets ou arêtes. Ce résultat est très intéressant pour la radiosité hiérarchique puisque cela évite de simuler l'interaction entre objets mutuellement invisibles lors de la phase d'établissement des liens initiaux.

De même, le squelette permet la détection des occultations causées par un objet donné. Cela peut se révéler important dans le cas du mouvement d'un objet m en simulation de l'éclairage pour détecter les facteurs de forme qui doivent être recalculés. Pour savoir si un facteur de forme F_{ij} doit être mis à jour, nous effectuons une requête semblable au maillage de discontinuité entre deux polygones. Nous parcourons la cellule correspondante du tableau ainsi que $DM(i,j)$, et vérifions si un arc est généré par un élément de m. Cela nous donne les limites de l'occultation due à m entre f_i et f_j. De plus, en parcourant tous les arcs du squelette de visibilité, et en trouvant ceux engendrés par m, nous obtenons la liste exacte de tous les facteurs de formes à mettre à jour et non un surensemble. La figure 3.18(b) montre les limites des occultations causées par une partie de l'avion entre l'écran et le mur de droite. Ce calcul a nécessité 1.3 ms.

6 Gérer la complexité : construction à la demande

Nous proposons dans cette section une construction à la demande ou paresseuse qui permet d'obtenir l'information de visibilité uniquement là où elle est nécessaire. Par exemple, si l'on veut le maillage de discontinuité entre deux surfaces, nous n'avons besoin que des arcs du squelette entre ces deux surfaces. Pour cela, nous n'avons besoin de détecter que les droites poignardantes extrêmes entre ces deux faces.

La clef de cette approche est la localité de l'algorithme de construction du squelette de visibilité. Nous ne calculons les nœuds du squelette que là où ils sont nécessaires. Le fait que certains arcs puissent avoir des nœuds pendants ne pose aucun problème puisque aucune requête n'y sera faite. Ces nœuds manquants seront peut-être insérés plus tard, à l'occasion d'une autre construction locale due à une autre requête.

Deux problèmes se posent : la détermination de ce qui doit être calculé et la détermination de ce qui a déjà été calculé. Nous proposons deux approches, dont seule la première a été implémentée : un calcul guidé par les sources et une subdivision de l'espace des rayons dans l'esprit d'Arvo et Kirk [AK87].

Dans le contexte de la simulation de l'éclairage global, l'information liée aux sources est cruciale. La partie pertinente du squelette de visibilité à calculer est donc celle concernant des droites coupant ces sources. La détection des nœud doit être modifiée : si un double coin ou le plan d'une face ne coupe pas la source, nous abandonnons la paire d'arêtes ou la face. Si un nœud potentiel est détecté, nous commençons par vérifier que la droite extrême coupe la source avant d'effectuer le lancer de rayon.

Nous utilisons aussi notre accélération à base de grille régulière pour la phase de restriction des EEE. Pour chaque arête e_i, nous ne considérons de paire qu'avec les arêtes e_j contenues dans le tétraèdre étendu défini par e_i et la source. Une restriction similaire est utilisée pour la détection des VV.

Dans le cas où plusieurs sources sont considérées à la suite, nous devons également éviter de recalculer des nœuds (outre le surcoût de calculs inutiles, les doublons rendraient le graphe incohérent). Si les sources sont petites, seul le lancer de rayon final doit être modifié : si la droite extrême potentielle coupe une source précédente, nous l'annulons. Un calcul final peut être effectué en considérant toutes les droites qui ne coupent aucune des sources précédentes.

Pour la scène (g) de la table 5.1, la partie du squelette de visibilité liée à l'une des sources est calculée en 4 minutes 15 secondes au lieu de 31 minutes 59 secondes pour la scène entière.

Si le nombre de sources devient grand, l'approche précédente perd la plupart de son temps à vérifier si les droites potentielles intersectent les sources ou si elles ont déjà été calculées. Nous proposons d'utiliser la classification de rayon [AK87] avec la notion d'espace dual du chapitre précédent pour calculer le squelette de visibilité de manière flexible. Le principe (qui n'a pas encore été implémenté) est de paramétrer les droites de l'espace, par exemple par une direction et l'intersection sur un plan, ou par l'intersection avec deux plans parallèles. L'espace des droites est ensuite subdivisé (par exemple avec une grille ou à l'aide d'une subdivision hiérarchique), et nous calculons les nœuds du squelette uniquement pour les cellules qui nous intéressent.

7 Scènes d'objets en mouvement

Dans cette section nous présentons des résultats concernant le traitement des scènes contenant des objets en mouvement. Nous ne donnons que des ébauches d'algorithmes, aucune implémentation n'ayant été faite. Nous pensons cependant que l'importance de ces questions justifie une présentation, y compris sous cette forme préliminaire.

7.1 Événements visuels temporels

Lorsqu'un objet se déplace, la topologie du squelette de visibilité est modifiée si un *événement visuel temporel* se produit, comme nous l'avons présenté dans la section 3 du chapitre précédent.

Nous présentons ici l'exemple d'un événement VEV et montrons comment le squelette de visibilité est modifié. La section suivante montrera comment un tel événement peut être détecté. Les autres sortes d'événements visuels ($EEEEE$, $EEEV$, FFE, $FEEE$, FEV, ...) peuvent être traités de manière semblable.

Dans la figure 3.19 l'arête e se déplace de droite à gauche. Les deux arcs EV v_1e et ev_2 se rencontrent lors de l'événement visuel temporel v_1ev_2.

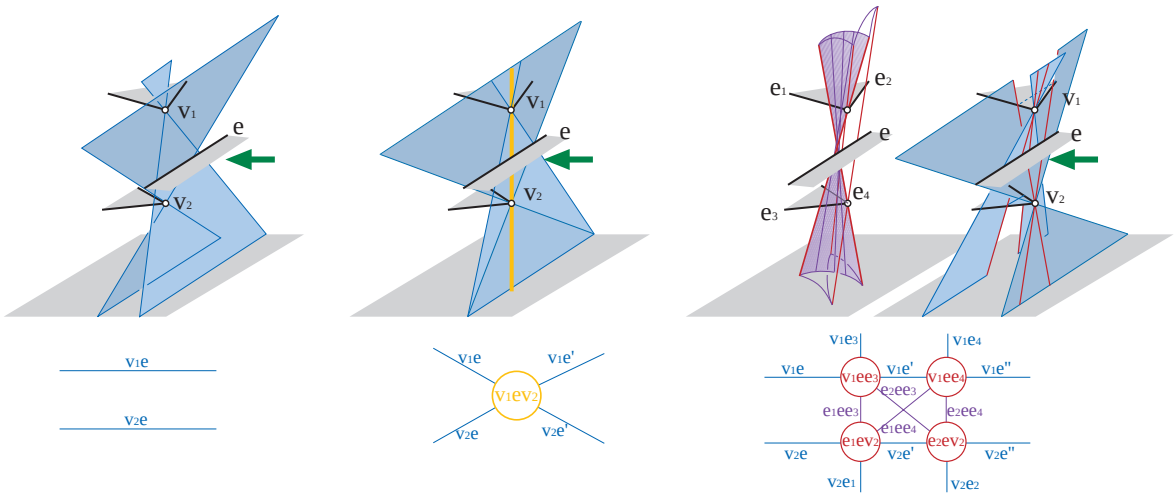
Les deux arcs EV sont coupés en trois, à cause de l'occultation due à l'autre sommet. Quatre nœuds v_1ee_3, v_1ee_4, e_1ev_2 et e_2ev_2 sont créés, ainsi que quatre arcs EEE e_1ee_3, e_1ee_4 e_eee_3 et e_2ee_4. Le changement réciproque se produit si l'arête se déplace dans la direction opposée.

Ces changements sont locaux pour le squelette de visibilité, même si les arêtes, sommets et faces impliqués sont distants dans la scène.

7.2 Détection des événement visuels temporels

La détection d'un événement VEV peut être faite comme suit. Les deux arcs EV qui se rencontrent ont les mêmes polygones à leurs extrémités. Ceci est toujours vrai pour des arcs qui se rencontrent lors d'un événement visuel temporel : ils ont toujours les extrémités de l'événement temporel. Ils ont un générateur commun, et l'un des arcs doit être engendré par l'objet dynamique. Une mise à jour dynamique du squelette de visibilité consiste donc à regarder, pour chaque arc lié à l'objet dynamique, quel est le prochain événement visuel temporel dans lequel il est impliqué.

Considérons l'arc v_1e lié à l'objet dynamique. Pour qu'il y ait rencontre, un autre arc doit avoir les mêmes extrémités (le sol et l'infini) et doit être engendré par e ou v_1. De tels arcs sont facilement trouvés en recherchant dans la case du tableau correspondante. Pour faciliter la recherche multicritère sur les générateurs, une structure de recherche adaptée doit être utilisée.



(a) (b) (c)

FIG. 3.19: Mise à jour du squelette de visibilité en cas de mouvement d'objet. L'arête e en se déplaçant de droite à gauche engendre un événement visuel temporel qui est la rencontre de deux arcs EV ayant les mêmes extrémités et un générateur commun (ici l'arête e). Quatre nœuds sont créés, les arc EV sont coupés en trois et huit arcs sont créés. Cet événement et les changements topologiques sont locaux dans le squelette de visibilité.

Tous les événements visuels sont ensuite triés et traités au fur et à mesure. Quand un événement se produit, le squelette de visibilité est localement mis à jour. La liste des événements temporels doit elle aussi être mise à jour. On vérifie si les arcs modifiés peuvent en rencontrer d'autres, et on supprime les événements liés aux arcs détruits.

Si la plupart des objets de la scène sont en mouvement, on recherche systématiquement dans chaque case de tableau d'arcs s'il y a des rencontres. Tous les arcs qui ont un générateur en commun (face, arête, sommet) sont confrontés pour une rencontre éventuelle.

7.3 Application possible : mise à jour de facteurs de forme

Un exemple d'application possible est la mise à jour de facteurs de forme pour la simulation de l'éclairage avec des objets en mouvement. Lorsqu'un objet se déplace, certains facteurs de forme doivent être recalculés : ceux qui concernent les faces de l'objet en mouvement, mais aussi les facteurs de forme en visibilité partielle pour lesquels l'objet est un bloqueur. De plus, certaines paires d'objets deviennent mutuellement visibles, ou complètement occultées, ce qui nécessite d'établir ou de supprimer la représentation du transfert lumineux correspondant. Le squelette de visibilité est un outil efficace pour détecter toutes ces modifications.

La visibilité mutuelle de deux polygones f_1 et f_2 est affectée par un objet si et seulement s'il y a un arc du squelette de visibilité lié à cet objet dans la liste d'arcs entre f_1 et f_2. Cela donne une caractérisation immédiate et exacte des facteurs de forme à recalculer.

La mise à jour d'un facteur de forme commencera ou arrêtera d'être nécessaire si et seulement s'il intervient un événement visuel temporel qui a un générateur ou une extrémité sur chacune des faces. Par exemple, dans la figure 3.19, le facteur de forme entre les faces adjacentes à v_1 et v_2 commence à nécessiter un recalcul à cause de l'occultation due à e. Le recalcul peut être fait localement à l'intérieur de la vue correspondante : la quantité à retirer au facteur de forme correspond à la surface qui devient cachée par e.

Nous renvoyons le lecteur aux travaux de Orti *et al.* [ORDP96] pour un développement équivalent dans le plan.

8 Conclusion

8.1 Résumé

Nous avons présenté une nouvelle structure de données appelée le *squelette de visibilité*, qui décrit toute la visibilité globale d'une scène polygonale 3D. Il s'agit d'un graphe dans l'espace des droites, dont les nœuds sont les droites poignardantes extrêmes engendrées par l'interaction de sommets, d'arêtes et de faces. Ces droites sont calculées grâce à des opérations simples de synthèse d'images, des intersections droite-plan ou du lancer de rayon. Les arcs du graphe sont les ensembles de droites critiques qui sont adjacents aux droites extrêmes. L'idée clef qui rend notre construction simple est de traiter explicitement uniquement les nœuds, et de déduire topologiquement les arcs grâce au catalogue d'adjacences que nous avons présenté. Nous avons ainsi proposé un algorithme complet de construction, qui comprend la détection et l'insertion des nœuds et des arcs qui leur sont adjacents.

Nous avons présenté une implémentation de l'algorithme de construction ainsi que différentes requêtes : partie d'un polygone visible depuis un sommet de la scène, maillage de discontinuité pour toute paire de polygones de la scène, liste exacte des bloqueurs entre deux faces et limites d'occultation.

Notre implémentation montre que, malgré une complexité asymptotique théorique élevée, la méthode reste en pratique utilisable pour la série de scènes que nous avons utilisées. De plus, nous avons présenté une approche pour la construction à la demande ou paresseuse qui est un premier pas vers des techniques progressives et hiérarchiques pour le squelette de visibilité.

8.2 Discussion et travaux futurs

Robustesse

La robustesse et le passage à l'échelle sont les deux problématiques majeures de travaux futurs. La localité de notre algorithme de construction limite l'effet d'éventuelles erreurs ; et le traitement des dégénérescences que nous avons implémenté permet de traiter la plupart des scènes de synthèse d'images. Cependant, l'implémentation a été fastidieuse et ardue, et nous ne pouvons prétendre que toutes les configurations sont traitées.

Nous ne pensons pas que les problèmes de robustesse doivent être traités en recourant à de l'arithmétique exacte (par opposition aux calculs sur les flottants). Les scènes rencontrées en pratique présentent de nombreuses dégénérescences qui ne sont pas le fruit du hasard, mais bien inhérentes à la scène, comme les contacts entre objets ou le parallélisme. Ces configurations doivent être prises en compte et non traitées comme si les objets étaient en position générique. Si deux objets se touchent, nous ne voulons pas calculer un événement visuel qui passent entre eux parce que la dixième décimale est légèrement différente. Nous voulons détecter que nous avons affaire à une configuration dégénérée, et qu'à cause du contact, aucune visibilité n'est possible. Cela ne dispense pas de l'utilisation d'arithmétique précise (si ce n'est exacte) pour détecter les cas dégénérés. Cependant, un traitement spécifique doit ensuite être appliqué.

Cela rend malheureusement l'implémentation fastidieuse car cela augmente beaucoup le nombre de cas différents à traiter. Pour résoudre ce problème, une définition générale de droite poignardante extrême pourrait être utilisée. Définissons une droite poignardante extrême comme passant par n arêtes (avec $n \geq 4$) (cf. figure 3.20). Les arcs adjacents peuvent ensuite être énumérés en considérant tous les triplets parmi ces arêtes. La configuration de chaque triplet d'arêtes doit être testée (en incluant un test de dégénérescence) pour décider si l'arc doit est créé ou annulé. Cela permettrait l'utilisation d'une seule procédure là où notre implémentation actuelle requiert du code spécifique pour chaque type de droite poignardante extrême. Du code spécifique doit cependant être écrit pour chaque type d'arc (EV, EEE, FE, ou Fv).

Considérons l'exemple de la figure 3.20. Les générateurs sont tout d'abord triés le long de la droite. Les arcs générés par V_1 sont traités en premier. Deux arcs V_1E_1 sont créés (avec une extrémité droite différente). V_1E_2 est annulé à cause de l'occultation due à E_1. Pour un triplet ou une paire de générateurs, la procédure consiste à tester l'occultation par un autre générateur qui se trouve entre eux, puis à trouver les extrémités (parmi les extrémités de la droite poignardante extrême et les faces adjacentes aux autres générateurs).

Une autre approche pour améliorer la robustesse de la construction est une réparation *a posteriori* du squelette. Comme nous l'avons déjà vu, une erreur n'entraîne des incohérences que localement. Un post-calcul pourrait être utilisé pour relier les arcs pendants ou pour supprimer les éléments inutiles du graphe de manière cohérente, si ce n'est exacte. Les problèmes de robustesse seront aussi abordés dans la section suivante.

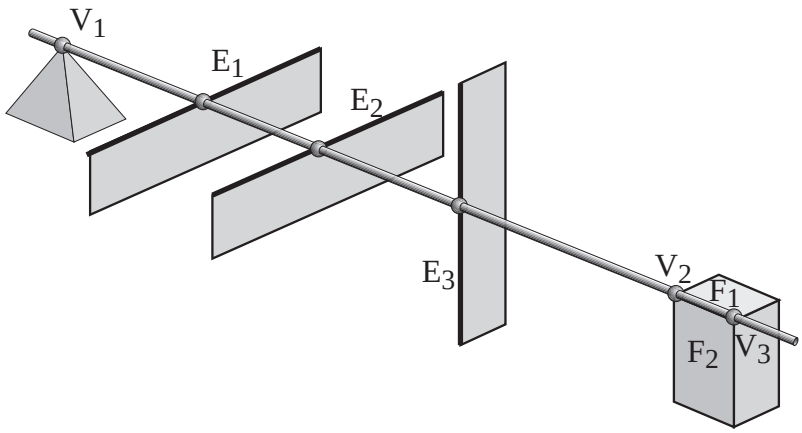


FIG. 3.20: Définition générale d'une droite poignardante extrême.

Passage à l'échelle

Bien que le comportement pratique de notre algorithme de construction soit bien meilleur que sa complexité théorique, un coût quadratique n'est toujours pas acceptable pour une utilisation pratique sur de grandes scènes. Nous pensons qu'une structure de visibilité doit être développée qui offre un compromis avec la précision quand les objets deviennent lointains. Une approche hiérarchique devrait être explorée, en calculant un squelette de visibilité exact à l'intérieur de regroupements d'objets, et une autre structure entre ces regroupements. Nous avons déjà étudié une telle approche en 2D [Dur95].

Néanmoins, il est difficile de définir une structure de visibilité approximative entre regroupements d'objets parce que la définition même de visibilité approximative est loin d'être immédiate. Un groupe d'objet n'est pas opaque, sa transmittance doit être prise en compte. Les travaux de Soler [Sol98, SS96b] constituent une contribution très intéressante dans ce sens.

Le squelette de visibilité est une structure de graphe. La littérature sur les graphes a sûrement beaucoup à nous apprendre. Quelles sont les propriétés du squelette de visibilité en tant que graphe ? Des techniques de compression de graphe devraient être essayées. Considérons l'exemple de deux triangles en visibilité totale. Cette situation pourrait être détectée et factorisée pour compresser l'information de visibilité.

La simplification du squelette à l'aide de suppression de sommet ou de fusion d'arête (*edge collapse*) est une autre voie intéressante. L'effet de telles opérations sur l'information de visibilité doit être analysé précisément, en particulier leurs conséquences globales. Par exemple, quand l'ombre d'un objet finement maillé est considérée, des événements pourraient être fusionnés. Cela supprime cependant l'information de visibilité liée aux sommets correspondants de l'objet.

Bien entendu, des restrictions liées à l'application peuvent aider à guider la simplification, de même que la définition de la visibilité approximative. Le maillage de discontinuité et la détermination de parties visibles pour la simulation de l'éclairage fournissent un cadre tout indiqué, puisque des approches hiérarchiques y ont été développées [HSA91, SAG94, Sil95].

De plus, nous pensons que les problèmes de passage à l'échelle et de robustesse gagneront à être abordés simultanément, puisqu'une approche hiérarchique permet de définir naturellement la notion d'échelle qui peut être utilisée pour définir des seuils pertinents pour la détection de configurations dégénérées. Le même ε peut être utilisé pour déterminer qu'une information de visibilité n'est pas perceptible et que deux éléments sont en configuration dégénérée.

Autres problèmes

Le squelette de visibilité peut être défini pour des scènes d'objets courbes comme les faces de dimension 0 et 1 du complexe de visibilité correspondant. Les événements visuels sont alors décrits par la théorie des singularités des applications continues (cf. annexe A). Malheureusement, étendre notre algorithme de construction n'est pas aussi simple. Le plus gros problème est celui de la paramétrisation des arcs, qui ne peut plus être basée sur la position sur une arête. Cela empêche de définir un ordre pour les arbres de recherche, en particulier

pour les objets concaves.

La mise à jour du squelette après l'ajout ou la suppression d'un objet est différente de la mise à jour en cas de mouvement d'un objet. Grâce à sa localité, notre algorithme de construction peut être adapté pour n'effectuer des calculs que là où c'est nécessaire.

Le squelette de visibilité permet en un sens d'émuler le complexe de visibilité grâce aux arbres de recherche. Les frontières de dimension 0 et 1 d'une 4-face du complexe sont regroupées dans un arbre de recherche du squelette. Pour une paire de polygones donnée, l'information relative aux 4-faces correspondantes est stockée dans la case du tableau indexée par les deux faces. Ces questions méritent une plus ample étude.

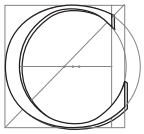
Dans le prochain chapitre, nous utiliserons le squelette de visibilité pour la simulation de l'éclairage par la méthode de radiosité hiérarchique. Nous montrerons qu'il permet une simulation efficace et précise, avec des ombres de haute qualité même dans des configurations difficiles comme des scènes éclairées par plusieurs sources ou par un éclairage principalement indirect.

L'application du squelette à d'autres techniques devrait être mise en œuvre. En particulier, en vision par ordinateur, les graphes d'aspect [Kv76, Kv79, EBD92], les enveloppes visuelles [Lau94, Lau95, Lau97, Lau99] ou le calcul de points de vue sans occultation [TTK96] pourraient bénéficier des événements visuels du squelette. En robotique, la planification de trajectoire avec visibilité pour des poursuites-évasions [LLG+97, GLL+97, GLLL98] demande aussi une partition de la scène selon les événements visuels. Néanmoins, le coût exponentiel de la recherche dans un graphe qui est ensuite nécessaire fait de l'extension des techniques 2D actuelles vers la 3D un véritable défi.

CHAPITRE 4

Radiosité hiérarchique guidée par la visibilité

> Les reflets des parties éclairées rejaillissent sur les ombres en face d'elles, et allègent plus ou moins leur obscurité selon leur distance et leur degré de clarté.
>
> Léonard DE VINCI, *Codex Urbinas*



E CHAPITRE décrit l'utilisation du squelette de visibilité pour répondre à des requêtes de visibilité rencontrées en simulation de l'éclairage. Les techniques récentes de simulation de l'éclairage global comme la radiosité hiérarchique [HSA91] et son utilisation combinée aux maillages de discontinuité [LTG93, DS96] ont permis des simulation de l'éclairage de haute qualité. Ces simulation sont *indépendantes du point de vue* et sont idéales pour les balades virtuelles (*walk-throughs*). La qualité de l'éclairement calculé est importante à tout endroit de la scène, car l'utilisateur peut par exemple s'approcher de l'ombre d'un objet et en regarder les détails.

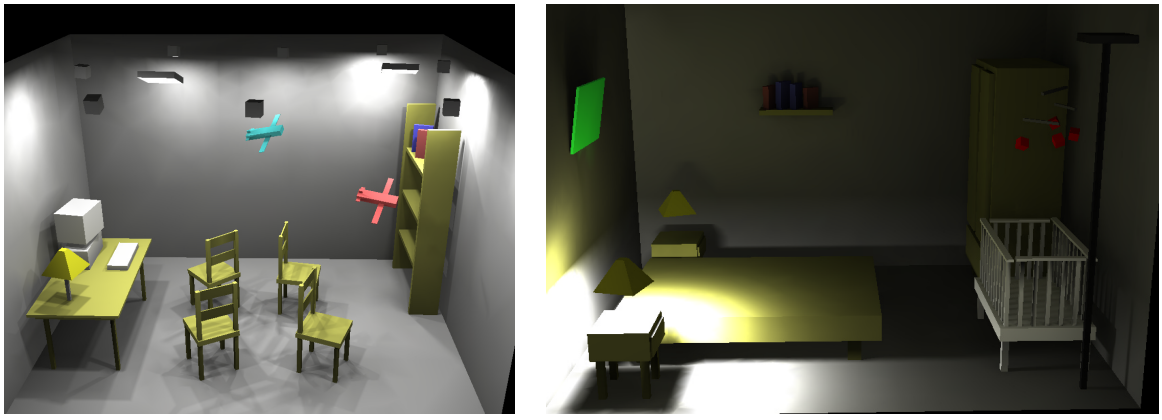
Malgré la grande qualité obtenue par les techniques existantes, certains aspects de ces algorithmes demandent à être améliorés. En particulier, déterminer quand un transfert lumineux doit être *raffiné*, et donc simulé avec plus de précision, est une décision difficile. Parmi les algorithmes actuels ([HSA91, LSG94, GH96] etc.), certains proposent un contrôle de l'erreur. La création d'un *maillage* pour la représentation des variations d'éclairement (en particulier pour les ombres) est une tâche difficile ; les *maillages de discontinuité* [LTG93, DS96] représentent une solution à ce problème, qui est cependant souvent limitée (à une source de lumière, à certains types de discontinuités). les approches récentes (par exemple [CSSD96, UT97]) évitent ce problème en ayant recours à une phase qui dépend du point de vue et effectue un lancer de rayon de collecte finale de l'énergie (*final gather*), et la possibilité d'affichage interactif et de balades virtuelles est sacrifiée. Des calculs de visibilité précis sont également difficiles par essence puisque nous devons considérer l'interaction de toutes les paires de polygones de la scène pour prendre en compte l'éclairage global.

Les trois problèmes que nous venons d'évoquer, la visibilité, le raffinement et le maillage sont accentués dans les configurations suivantes : les scènes éclairées par de nombreuses sources de lumières et celles où l'éclairage est avant tout indirect. Dans ce chapitre, nous présentons un nouvel algorithme qui permet de résoudre ces trois problèmes. Les approches précédentes manquaient d'une information précise sur les relations *globales* de visibilité. Cette information est fournie par le squelette de visibilité. Le squelette permet un

calcul rapide de facteurs de forme exacts point-polygone. De plus, toute l'information de visibilité (bloqueurs, surfaces de discontinuité) est disponible pour toute paire de polygones.

Cette information globale de visibilité nous permet de développer une stratégie pertinente de raffinement, puisque nous avons connaissance de l'information pour tous les transferts. Nous pouvons classer et sélectionner les discontinuités entre deux éléments du maillage, à l'aide de techniques basées sur des données perceptives [GH97b] ; de ce fait, seules les discontinuités qui sont visuellement importantes sont considérées. Un maillage approprié est créé à l'aide de ces discontinuités. L'éclairement est représenté de manière très précise, sur un maillage qui ne dépend pas du point de vue. Pour obtenir de tels résultats dans un contexte de radiosité hiérarchique, nous avons introduit des triangulations hiérarchiques. La radiosité est recueillie et stockée aux sommets, puisque le squelette permet de calculer exactement les facteurs de formes sommet-polygone. Une procédure de "pousser-tirer" (*push-pull*) multirésolution a été introduite pour maintenir une représentation cohérente. La qualité du maillage et la précision des calculs fournissent des solutions de haute qualité.

Notre approche est particulièrement adaptée au cas des sources multiples puisque la sélection des discontinuités opère simultanément sur tous les échanges d'énergie qui arrivent à un récepteur. L'éclairage indirect est également très bien traité puisque l'information de visibilité est disponible pour toute paire d'éléments, et donc le raffinement comme le maillage ou le calcul des facteurs de forme sont traités avec la même efficacité pour tous les échanges, *i.e.* directs et indirects. Des exemples sont donnés en figure 4.1. Dans la figure 4.1(a) on peut voir une pièce éclairée par dix sources de lumière qui projettent des ombres nettement visibles. Le maillage reste cependant gérable (cf. table 4.2 dans la section 6.2). Dans la figure 4.1(b) , nous montrons une pièce éclairée principalement par de l'éclairage indirect. On peut y voir des ombres de hautes qualité causée uniquement par éclairage indirect (par exemple les ombres des livres et de la lampe sur le mur du fond).



(a) (b)

FIG. 4.1: Images calculées avec notre nouvel algorithme de radiosité hiérarchique, fondé sur le squelette de visibilité et des triangulations hiérarchiques. (a) Pièce éclairée par de nombreuse sources. La construction du squelette a requis 2 minutes et 23 secondes, et la simulation de l'éclairage 8 minutes. (b) Scène éclairée principalement par éclairage indirect. La construction du squelette a requis 4 minutes 12 secondes et la simulation 6 minutes 58 secondes. Les ombres des livres et de la lampe sur le mur du fond sont uniquement due à de l'éclairage indirect par le lit et la table de gauche.

Le nouvel algorithme que nous présentons est en un sens une extension de la radiosité hiérarchique en utilisant une structure efficace de visibilité, une nouvelle technique de maillage et un raffinement guidé par des données perceptives. nous passons brièvement en revue les techniques précises de visibilité, les méthodes qui utilisent la visibilité pour le raffinement et les techniques perceptives pour la simulation de l'éclairage.

Le travail présenté dans ce chapitre a été accepté pour publication dans la revue *ACM Transaction on Graphics* [DDP99]. Nous y ajoutons la description de techniques pour décroître son coût mémoire dans la section 7.

Le chapitre est organisé comme suit. Après un présentation des travaux antérieurs nous donnons les détails de notre structure multirésolution dans la section 2. Dans la section 3 nous présentons notre algorithme de radiosité guidé par la visibilité ainsi que de nouvelles structure de liens hiérarchiques point-polygone et polygone-

polygone, Dans la section 4 nous présentons les procédures de raffinement et de mise à jour de l'information de visibilité. La section 5 décrit le subdivision des polygones et les critères de raffinement fondés sur la perception. Nous présentons une discussion des résultats obtenus avec notre implémentation ainsi que des améliorations possibles. Nous concluons ce chapitre avec une discussion des limitations et des avantages de notre approche.

1 Motivation et travaux antérieurs

1.1 Radiosité hiérarchique.

L'algorithme de radiosité hiérarchique [HSA91] autorise des simulations efficaces d'éclairage global. Les calculs d'éclairage sont limités à un niveau de précision fixé par l'utilisateur, grâce à la subdivision des polygones de la scènes en arbres quaternaires (*quadtrees*), et à l'établissement de "liens" (*links*) de transfert lumineux au niveau approprié de la hiérarchie. Dans la méthode originale de radiosité hiérarchique [HSA91], la radiosité est considérée constante sur les éléments du quadtree. La nature rectangulaire des quadtrees, et l'hypothèse de radiosité constante rendent nécessaire une subdivision très fine pour créer des images de qualité.

Des méthodes utilisant des représentations d'ordre plus élevé (non constantes) ont été utilisées, notamment dans le contexte de solutions à base d'ondelettes [GSCH93]. Les solutions de radiosité à base d'ondelettes présentées à ce jour utilisent des bases discontinues, ce qui crée des artefacts visibles à l'affichage (*e.g.*, [CSSD96]). Zatz [Zat93] utilise une méthode de Galerkin et des masques d'ombres (*shadow-mask*) pour améliorer la qualité des ombres. Pour éviter le problème des représentations discontinues, une étape de "collecte finale" (*final gather*) a été introduite [Rei92], et appliquée aux méthodes par ondelettes (*e.g.*, [CSSD96]). La collecte finale consiste à utiliser le lancer de rayon pour créer l'image, et à recalculer les transferts lumineux entre chaque pixel de l'image [UT97] et tous les éléments de maillage. Cette approche permet des images de qualité à partir d'une simulation initiale grossière de l'éclairage, à un coût souvent élevé. La solution dépend de plus du point de vue, et il n'est plus possible de se balader dans la scène.

Récemment, Bekaert *et al.* [BNN+98] ont présenté un algorithme efficace qui combine la radiosité hiérarchique et la radiosité de Monte-Carlo. Cependant, la nature stochastique de cette approche rend difficile le raffinement le long des limites d'ombre.

1.2 Précision de la visibilité et qualité des images

La précision des calculs de visibilité est essentielle pour la simulation de l'éclairage : la précision numérique du résultat tout comme la qualité visuelle des images en dépendent. Les calculs exacts entre deux éléments ou entre un élément et un point nécessitent le traitement des événements visuels étudiés dans les chapitres précédents. Dans le cas d'une source de lumière, ces événements correspondent aux limites d'ombre et de pénombre. Des algorithmes qui utilisent certaines de ces limites pour guider le maillage (irrégulier) permettent d'améliorer la qualité et la précision des images (*e.g.*, [Hec92a, LTG92]).

La détermination de la partie visible d'une source étendue en synthèse d'image correspond exactement au calcul de son aspect (au sens des graphes d'aspect [PD90, GM90, EBD92]). Des algorithmes permettent d'effectuer cette opération en construisant un maillage de discontinuité complet ainsi que les "projections arrières" (*backprojections*) qui décrivent l'aspect qualitatif de la source (*e.g.*, [Tel92a, DF94, SG94]). Le maillage de discontinuité complet et les projections arrières permettent le calcul exact de facteurs de forme entre n'importe quel point et la source. Cependant, ces méthodes souffrent de problèmes numériques dus aux calculs d'intersection entre les surfaces de discontinuité et les polygones de la scène, aux structures de données complexes nécessaires pour représenter les maillages irréguliers. Elles ont de plus un coût élevé.

Nous avons choisi d'utiliser le squelette de visibilité à cause de sa facilité d'utilisation et de sa robustesse (comparée à celle des maillages de discontinuité complets).

1.3 Stratégies de raffinement fondées sur la visibilité pour la radiosité

Dans l'algorithme de radiosité hiérarchique, la subdivision du maillage se fait à travers le raffinement de liens. L'algorithme initial utilise un critère *BFV* (radiosité multipliée par le facteur de forme, modulé par un terme de visibilité *V* pour les liens en visibilité partielle). Les maillages qui en résultent sont souvent trop fins dans les régions non occultées, mais ne représentent pas bien les fins détails d'ombre.

Les stratégies de raffinement fondées sur le contrôle de l'erreur [LSG94, GH96] ont amélioré la qualité des maillages et du raffinement. Les précalculs prudents de visibilité pour les scènes architecturales [TH93], permettent de déterminer de manière précise si un lien est en visibilité pleine, nulle, ou partielle. Ce tri guide la subdivision, et autorise un raffinement plus fin dans les régions éclairées partiellement.

Le maillage de discontinuité améliore la qualité visuelle des images crées par la simulation de l'éclairage [LTG92, Hec92a, DF94], puisque le maillage utilisé pour représenter l'éclairement suit effectivement les limites d'ombres, au lieu de les approcher par un maillage quadtree. L'un des problèmes est le grand nombre de discontinuités. Tampieri [Tam93] a essayé de limiter le nombre de discontinuité insérées en échantillonnant l'éclairement le long des discontinuités et en insérant uniquement celles dont la variation dépassait un seuil donné. Dans le contexte de la radiosité avec raffinement progressif, Stuerzlinger [Stu94] n'insère les discontinuités qu'à un second niveau dans un maillage de type quadtree, une fois que la région a été déclarée importante par une étape de lancer de rayon. Les discontinuités insérés sont celles dues aux bloqueurs identifiés par le lancer de rayon.

De nombreuses méthodes qui combinent la radiosité hiérarchique avec les maillages de discontinuité ont été présentées [LTG93, DS96, HT96, BP95]. Hardt et Teller [HT96] présentent une approche dans laquelle les discontinuités dues à toutes les surfaces sont considérées, sans calculer leur intersection avec les autres bloqueurs. Les discontinuités potentielles sont triées, et celles jugées les plus importantes sont insérées au niveau le plus fin d'un quadtree. Drettakis et Sillion [DS96] utilisent l'information du maillage de discontinuité complet, ce qui entraîne un grand nombre de triangles très fins. Les facteurs de formes exact avec la source primaire sont calculés aux sommets de ces triangles grâce aux projections arrière. Les triangles sont ensuite regroupés dans un quadtree, et une phase standard de radiosité hiérarchique [HSA91] est ensuite utilisée. Les problèmes déjà mentionné des maillages de discontinuité, le regroupement coûteux et le fait que des triangles couvrent différents éléments du quadtree limitent cependant l'utilisabilité de cette méthode. La seule autre méthode de radiosité hiérarchique qui effectue une collecte aux sommets est celle de Martin *et al.* [MPT97], qui utilise une valeur de radiosité à chaque sommet, et une procédure complexe de "pousser" (*push*) pour maintenir la représentation multirésolution.

L'algorithme de Lischinski *et al.* [LTG93] est le plus complet et le plus proche de notre travail. Cette approche sépare la phase de simulation et la phase de rendu. L'idée est dans son esprit similaire à celle de collecte finale, mais de manière indépendante du point de vue. Une première passe globale est effectuée qui crée des arbres BSP 2D sur les polygones de la scène, en les subdivisant le long de discontinuité importante causées par les seules sources primaires. Les arbres BSP requièrent de longues subdivision, et donc de fins triangles qui ne sont pas adaptés pour une représentation fine de l'éclairement. La deuxième passe, locale, recalcule l'éclairement aux sommets de la subdivision due aux BSP. Le coût de cette phase est en fait plus élevé que celui de la simulation elle-même (si l'on ignore l'établissement des liens initiaux).

1.4 Raffinement fondé sur la perception

Récemment, des métriques perceptives ont été utilisées pour réduire le nombre d'éléments requis pour représenter de manière précises l'éclairement (*e.g.*, [GH97b, HWP97]). Un opérateur de reproduction de nuance (*tone-reproduction*) [TR93, War94] est utilisé pour associer les valeurs de radiosité calculées à des valeurs d'affichage qui reproduisent un effet semblable à celui que ressentirait un observateur dans un monde réel. Puisque les dispositifs d'affichage (écrans, imprimantes) ont une dynamique très limitée par rapport aux valeurs de luminance du monde physique, le choix de cet opérateur est important. L'opérateur de reproduction de nuances Tumblin et Rushmeier [TR93] et celui de Ward [War94] dépendent de deux paramètres ; un niveau d'adaptation du monde qui correspond grossièrement à la luminance moyenne à laquelle l'œil d'un observateur placé dans la scène s'adapterait, et une adaptation de l'affichage, qui correspond à la luminosité affichée à l'écran. Le choix de ces paramètres affecte ce qui sera affiché, et surtout les différences de radiosité qui seront effectivement perceptibles dans l'image finale. En particulier, on peut définir une "différence tout juste perceptible" (*just noticeable difference*) à l'aide cet opérateur. Dans le contexte de la simulation de l'éclairage, la différence tout juste perceptible correspond à la plus petite valeur de radiosité qui, une fois convertie en couleur d'affichage, sera perceptible par un observateur. L'adaptation d'affichage est typiquement une valeur constante, comme la moitié de la luminance maximale affichable par l'écran [War94], tandis que l'adaptation du monde peut être choisi de différentes façons. L'adaptation statique [GH97b] utilise une valeur moyenne qui est indépendante du point de vue, tandis que l'adaptation dynamique (qui est plus réaliste) utilise une valeur qui dépend de la

direction dans laquelle l'observateur regarde. Gibson et Hubbold [GH97b] utilise la reproduction de nuances pour guider la subdivision pour la radiosité progressive, ce qui permet de raffiner un calcul uniquement si le résultat sera "tout juste perceptible".

De manière similaire, Hedley *et al.* [HWP97] utilisent un opérateur de reproduction de nuances pour déterminer si une discontinuité doit être insérée dans un maillage d'éclairement. Le calcul est effectué en échantillonnant le long de la discontinuité (comme Tampieri [Tam93]), mais aussi perpendiculairement. Cela permet une diminution importante du nombre de lignes de discontinuité insérées sans perte de qualité.

1.5 Discussion

Les méthodes antérieures de radiosité fournissent une simulation de l'éclairage indépendante du point de vue qui est acceptable dans bien des situations. En particulier, la subdivision de type quadtree fournit des solutions rapides de qualité raisonnable, y compris pour des scènes éclairées principalement par éclairage indirect. Cependant lors de balades virtuelles, l'observateur s'approche souvent de régions dans l'ombre, et dans ce cas le manque de précision ou de qualité des ombres pose problème. Les approches précédentes fondées sur le maillage de discontinuité résolvent cette question en subdivisant le maillage le long des limites d'ombre pour l'éclairage direct, mais elles deviennent inutilisable dès lors que de nombreuses sources sont présentes ou que l'éclairage indirect domine. Ces limitations sont dues au grand nombre de discontinuités qui doivent être prises en compte pour le calcul de l'éclairage indirect et de la complexité des maillages qui en résulteraient. Ces problèmes sont discutés par Lischinski [LTG92] et Tampieri [Tam93]. Les solutions à ce jour ont limité l'utilisation des informations de discontinuité aux sources primaires [LTG93, DS96] ; pour les réflexions suivantes de la lumière, des approches approximatives par lancer de rayon sont utilisées pour le calcul des transferts lumineux.

Le nouvel algorithme présenté dans ce chapitre permet de calculer des ombres de haute qualité pour une classe plus générale de scènes, incluant celles contenant de nombreuses sources et celles où l'éclairage indirect domine. Le squelette de visibilité nous permet de sélectionner et insérer les lignes de discontinuité pour tous les transferts lumineux, et de calculer de manière exacte et rapide les facteurs de forme point-polygone. Ces choix nous ont amenés à développer un nouvel algorithme de radiosité hiérarchique où l'éclairement est recueilli aux sommets. Il est fondé sur des *triangulations hiérarchiques* qui autorisent le maillage à suivre les lignes de discontinuité. Nous décrivons de plus un critère de raffinement fondé sur l'information précise de visibilité et sur une métrique perceptive qui évite le réglage fastidieux de seuils imprévisibles.

2 Triangulations hiérarchiques et ondelettes paresseuses

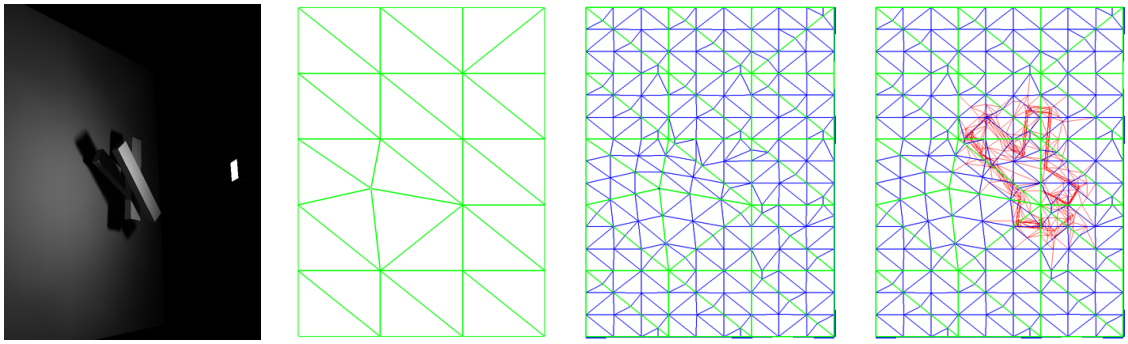
Dans les travaux antérieurs [Hec92a, LTG92], il a été montré que la création d'un maillage adapté aux discontinuités permet des images de haute qualité. L'incorporation de tels maillages irréguliers dans un algorithme de radiosité hiérarchique présente néanmoins des difficultés importantes. Comme nous l'avons signalé dans le chapitre 1, la plupart des algorithmes [LTG93, DS96] qui traitent la question sont restreints aux discontinuités dues à l'éclairage direct.

Le cœur du problème est la présence de deux buts antagonistes : avoir une hiérarchie simple qui permette des manipulations faciles, et avoir un maillage irrégulier qui suive les limites d'ombre. Le premier but est habituellement atteint grâce aux quadtrees [HSA91], tandis que pour le second on a recours aux arbres BSP [LTG93].

Dans les approches précédentes, l'information de discontinuité et de visibilité était utilisée avec un algorithme de radiosité hiérarchique reposant sur des éléments constants par morceaux. Avec l'utilisation du squelette de visibilité, cela serait perdre le bénéfice du calcul exact de facteur de forme point-polygone. La collecte aux sommets introduit cependant des complications : contrairement aux éléments, dont le niveau est bien défini dans la hiérarchie, les sommets sont partagés par plusieurs niveaux.

Pour résoudre ce problème, nous introduisons l'utilisation de triangulations hiérarchiques et d'ondelettes paresseuses pour la radiosité hiérarchique. Notre approche a deux avantages majeurs par rapports aux méthodes précédentes : (i) elle s'adapte bien aux maillages complètement irréguliers, et ceci de manière locale (des triangulations contenues dans des triangulations) et (ii) elle permet la collecte aux sommets grâce à des ondelettes

paresseuse (ou du sous-échantillonnage, cf. le livre de Stollnitz *et al.* [SDS96] pp 102–104 et 152–154). L'approximation linéaire de la radiosité est conservée lors du "pousser" (*push*) et lors de la collecte.



(a) (b) (c) (d)

FIG. 4.2: Triangulation hiérarchique. Les triangles sont bien adaptés aux détails. (a) Géométrie de la scène : le polygone de gauche est éclairé par la source étendue à droite. (b) Premier niveau de subdivision quasi régulier pour le polygone de gauche (en vert). (c) Deuxième niveau (en bleu). (d) Troisième niveau (en rouge).

Cependant, la représentation linéaire par morceaux ne sera utilisée que lors de l'opération de "pousser" et lors de l'affichage. La collecte et le "tirer" seront effectués sur des valeurs moyennes aux triangles. La simulation est tout de même améliorée puisque les valeurs calculées à un plus haut niveau sont linéairement interpolées pendant l'opération de "pousser", ce qui permet une représentation plus lisse aux niveaux les plus fins.

2.1 Triangulations hiérarchiques

Nos triangulations hiérarchiques sont largement inspirées de celles de de Floriani et Puppo [DP95a]. Le principe en est très simple : il s'agit d'une hiérarchie de triangulations, où chaque triangle d'un niveau peut être subdivisé en une triangulation.

Nous commençons par une triangulation initiale qui est une triangulation de Delaunay contrainte. Les triangulations de Delaunay contraintes permettent l'insertion d'arêtes *contraintes* qui sont marquées inamovibles, elle ne peuvent être modifiée pour satisfaire le critère de Delaunay. Chaque triangle de la triangulation initiale peut ensuite être subdivisé par l'ajout de nouveaux points et de nouvelles arêtes contraintes qui définissent une nouvelle triangulation de ce triangle, et ainsi de suite récursivement sur les sous-triangles. A chaque niveau, une triangulation de Delaunay contrainte est maintenue.

Un exemple est donné figure 4.2. Il montre que les triangulations possèdent des triangles qui ont une bonne forme (pas de triangles trop longs et fins), tout en permettant de représenter de fins détails dans les zones où c'est nécessaire. Cela entraîne une subdivision irrégulière dans les niveaux les plus fins de la hiérarchie.

Nos triangulations hiérarchiques sont "en correspondance" (*matching*), c'est-à-dire qu'une arête est subdivisée aux mêmes points dans tous les triangles qui la partagent. À la fin de chaque subdivision, une étape d"'ancrage" (*anchoring*) est effectuée qui ajoute les points manquant dans chaque triangle voisin. Cela permet d'avoir une représentation cohérente et continue de la fonction d'éclairement, en particulier pour l'opération de "pousser" (*push*) de l'information multirésolution.

Comme nous l'avons déjà signalé, les sommets sont partagés par différents niveaux de la hiérarchie de triangulations. Le niveau initial de la hiérarchie est un *HPolygon*, qui contient des enfants *HTriangulation* une fois subdivisé (le préfixe *H* signifie la nature hiérarchique de la structure).

Pour transmettre l'information d'adjacence entre niveaux (pour la correspondance), nous utilisons une structure que nous appelons *HEdge*. Un *HEdge* est partagé par plusieurs niveaux de la hiérarchie, et par toutes les arêtes qui partagent le même segment. Il contient des pointeurs vers ses sous-*HEdge* lorsqu'il est subdivisé Pour assurer la correspondance lorsque nous insérons un point sur une arête, nous déterminons si l'arête a déjà été subdivisée. Nous ajoutons alors les points de cette subdivision initiale, et ajoutons le nouveau point. Les triangles adjacents peuvent identifier les nouveaux sous-*HEdge* grâce au *HEdge* partagé. Par exemple, dans la figure 4.3, après la subdivision du triangle en bas à gauche (figure 4.3(a)), le *HEdge* partagé entre les deux

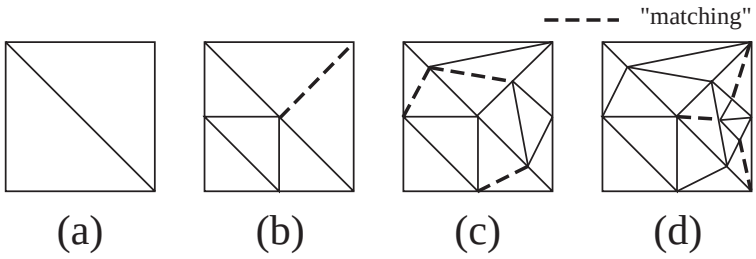


FIG. 4.3: Contrainte de "correspondance" pour les triangulations hiérarchiques. Nous montrons une séquence d'insertions d'arêtes. Les lignes pointillées montrent les arêtes dues à la contrainte de correspondance entre triangles adjacents à une arête.

triangles signale au triangle en haut à droite que l'arête a été coupée, ce qui simplifie la mise en correspondance montrée figure 4.3(b).

2.2 Représentation multirésolution linéaire par morceau

Le second avantage, la représentation linéaire par morceaux de l'éclairement, nécessite l'utilisation d'ondelettes paresseuses. Les ondelettes paresseuses fournissent un formalisme élégant pour une approche simple : une approximation linéaire par morceaux est raffinée grâce à l'ajout de nouveaux points d'échantillonnage. Le principe des ondelettes paresseuses sur une triangulation hiérarchique est illustré figure 4.4.

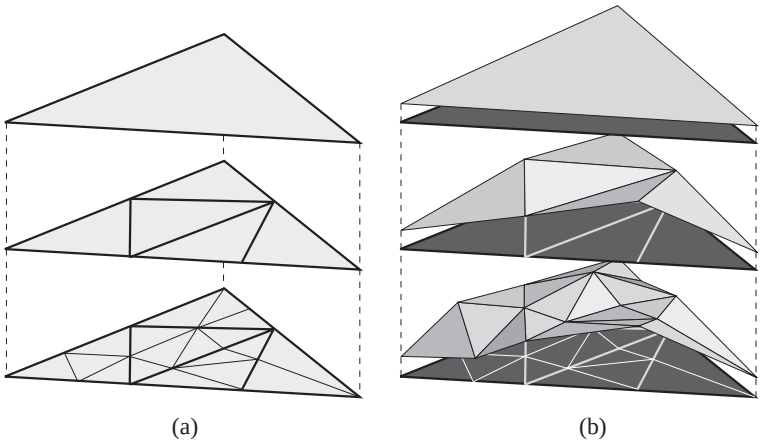


FIG. 4.4: Principe de notre représentation multirésolution. (a) Triangulation hiérarchique (le triangle de base est en haut et les maillages plus fins sont représentés en dessous). (b) Fonction d'éclairement représentée à l'aide d'ondelettes paresseuses. Des détails sont ajoutés aux sommets provenant de la subdivision des triangles.

Comme signalé ci-dessus, les sommets sont partagés par plusieurs niveaux dans nos triangulations hiérarchiques. De ce fait, les procédures habituelles de "pousser-tirer" (*push-pull*) ne peuvent être appliquées directement.

Pour comprendre pourquoi, considérons l'exemple 1D représenté figure 4.5. le segment $v_a v_b$ est éclairé par deux sources lumineuses S_1 et S_2. Supposons que les deux transferts soient tout d'abord raffinés, et que le sommet v_1 soit inséré. On obtient la configuration du premier niveau représentée figure 4.5). Le transfert avec S_2 est à nouveau raffiné avec l'insertion de v_2 qui coupe le segment $v_1 v_b$ à droite. Pour finir, le transfert avec S_1 est raffiné à gauche avec l'insertion de v_3.

Pour déterminer la contribution de S_1 dans l'intervalle $[v_1, v_b]$, nous interpolons la contribution $S_1 \to v_1$ et $S_1 \to v_b$, qui sont représentées au niveau 1. Toutefois, pour la source S_2, nous devons interpoler dans le sous-intervalle $[v_1, v_2]$ en utilisant $S_2 \to v_1$ et $S_2 \to v_2$ et dans le sous-intervalle $[v_2, v_b]$ en utilisant $S_2 \to v_2, S_2 \to v_b$, qui sont tous représentés au niveau 2. v_1 est partagé entre les niveaux 1 et 2. La procédure habituelle de pousser-tirer ne peut être appliquée, puisqu'elle nécessite qu'un élément appartienne à un niveau bien défini.

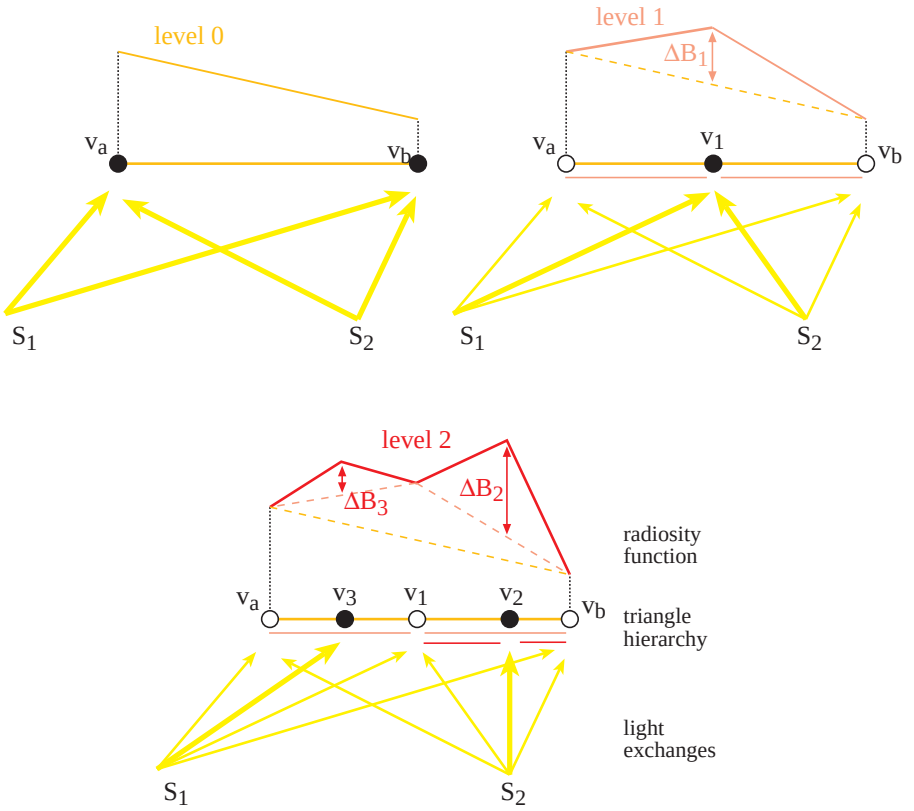


FIG. 4.5: Représentation multirésolution cohérente grâce aux ondelettes paresseuses. Au lieu de stocker des valeurs de radiosité, nous stockons des différences de radiosité aux sommés raffinés.

Une solution naïve consisterait à dupliquer le sommet v_1 pour faire la distinction entre les échanges simulés à différents niveaux de la hiérarchie. Cela n'est malheureusement pas suffisant, l'opération de "pousser" (*push*) ne peut alors pas être définie. Supposons qu'il y ait une représentation des échanges arrivant à v_1 au niveau 1, et une pour le niveau 2. Comment dès lors traiter les échanges $S_1 \rightarrow v_1$ et $S_2 \rightarrow v_1$? Si $S_1 \rightarrow v_1$ est stocké au niveau 1, l'interpolation est correcte sur l'intervalle $[v_1, v_b]$ pour le "pousser" sur v_2. Cependant, la valeur n'est plus disponible au niveau 2 pour l'interpolation entre v_3 et v_1. De manière symétrique, $S_2 \rightarrow v_1$ est nécessaire pour l'interpolation au niveau 1 pour l'intervalle $[v_a, v_1]$ et le "pousser" sur v_3, et au niveau 2 pour l'interpolation dans $[v_1, v_2]$. Quand la collecte est effectuée au sommet, parler de transfert à un niveau n'a plus guère de sens.

Nous utilisons les ondelettes paresseuses pour résoudre ces problèmes. Au lieu de considérer des valeurs effectives de radiosité, nous utilisons des *différences de radiosité* aux sommets raffinés (figure 4.5). Cela correspond à la différence de radiosité au niveau courant et la valeur interpolée à partir des valeurs des parents.

La procédure de "pousser" est alors immédiate : pour calculer la radiosité totale à un sommet, nous interpolons les valeurs de ces parents et ajoutons la différence de radiosité. On obtient ainsi la valeur totale à ce sommet, qui est ensuite poussée à travers la hiérarchie par un parcours en profondeur.

Cette technique est directement applicable en 2D, en utilisant les coordonnées barycentriques (ou bilinéaires pour les quadrilatères) pour l'interpolation. Il est donc simple d'effectuer une opération de "pousser" sur une triangulation hiérarchique avec collecte aux sommets.

Il est toutefois plus difficile de calculer la différence de radiosité d'un transfert lumineux que le transfert total. La section 3.2 résoudra cette question grâce à l'utilisation de liens négatifs.

L'opération de "tirer" (*pull*) est plus simple, puisque nous extrayons des valeurs moyennes sur les triangles. A chaque triangle feuille de la hiérarchie, la valeur est simplement la moyenne des valeurs aux sommets (après le pousser). Les nœuds intermédiaires reçoivent la moyenne pondérée par l'aire de leurs enfants, comme dans la radiosité hiérarchique classique.

Grâce à cette approche, nous pouvons utiliser une représentation multirésolution de la radiosité sur une tri-

angulation hiérarchique tout en collectant aux sommets. De plus, le pousser maintiens la représentation linéaire des contributions calculées aux plus hauts niveaux.

le résultat obtenu est plus lisse que le post-traitement de lissage habituellement utilisé, puisque nous effectuons des interpolations à tous les niveaux de la hiérarchie. Un artefact classique dû au lissage rend la structure de quadtree apparente par la présence de discontinuité légèrement lissées le long des droites centrales verticales et horizontales. Cela est dû à un transfert avec une source S simulé au niveau deux de la hiérarchie, tandis que le polygone est subdivisé plus finement pour d'autres échanges. L'interpolation ne se fait qu'aux feuilles, et la contribution de S est interpolée uniquement pour les éléments feuilles adjacents aux lignes centrales, alors que l'interpolation devrait être faite sur tout le polygone.

3 Radiosité hiérarchique guidée par la visibilité : algorithme et structures de données

Les triangulations hiérarchiques et les ondelettes paresseuses comptent parmi les outils nécessaires pour notre méthode de radiosité hiérarchique guidée par la visibilité. Le squelette de visibilité fournit quant à lui une information *exacte* et *globale* de visibilité, qui permet de calculer des facteurs de formes exacts entre points et polygones pour tout transfert (direct ou indirect). L'information fournie par les arcs du squelette (les événements de visibilité) permettent de plus le développement de critères pertinents de raffinement, une fois de plus pour tout échange.

Dans ce qui suit nous présentons notre nouvel algorithme qui utilise le squelette et les triangulations hiérarchiques pour un raffinement efficace et une simulation précise.

3.1 Présentation de l'algorithme

Notre nouvel algorithme est esquissé figure 4.6. Il commence avec la création du squelette de visibilité de la scène, en utilisant la structure améliorée à base de lien stockés aux polygones en lieu et place du tableau 2D (cf. section 3.2 du chapitre 3). Après cette étape, nous avons toute l'information pour calculer les facteurs de forme entre chaque sommet initial et chaque polygone. L'information de visibilité pour toute paire polygone-polygone est également disponible, ce qui évite les liens initiaux entre polygones occultés. Après le calcul des facteurs de forme initiaux, une étape de "collecte" est effectuée aux sommets, suivie d'un "pousser-tirer". En pratique, nous faisons un nombre fixé d'itérations pour obtenir un équilibre initial ; il serait toutefois possible d'attendre la convergence puisque ces itérations sont peu coûteuses.

Pour commencer la subdivision, nous insérons les maxima des fonctions d'éclairement dues aux sources les plus importantes pour chaque gros polygone (procédure *insertMaxima()*, cf. section 5.2).

```
RadiositeHierarchiqueGuideeParLaVisibilite
{
   computeSkeleton()          // calculer le squelette
   computeCoarseLighting()  // 3 collectes pousser-tirer
   insertMaxima()             // insertion des maxima
   while( !converged() ) do
      subdividePolygons()    // Raffinement des polygones a l'aide de l'info de visibilite
      refineLinks()             // Raffinement des liens a l'aide de l'info de visibilite
      gatherAtVertices()      // Collecte aux sommets des triangulations hierarchiques
      pushPull()
   endwhile
}
```

FIG. 4.6: Radiosité hiérarchique guidée par la visibilité

Une fois que le système a ainsi été initialisé, nous commençons la subdivision guidée par les discontinuités (*subdividePolygons()*) et le raffinement des liens (*refineLinks()*). Grâce à l'information globale de visibilité, nous pouvons subdiviser les surfaces en suivant les discontinuités qui sont "importantes". À la fin de

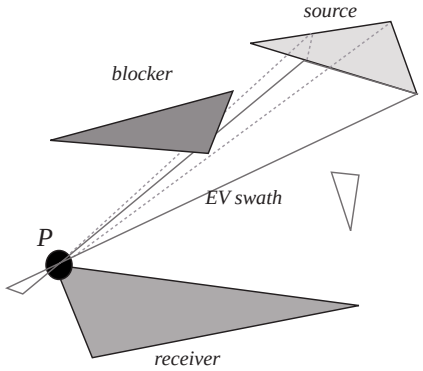
chaque étape de raffinement, une collecte et un pousser-tirer sont effectués, afin de maintenir la cohérence de la représentation.

Dans ce qui suit, nous utiliserons les termes “source” et “récepteur” pour simplifier la discussion. Une source peut cependant être n’importe quel polygone de la scène, qu’il s’agisse d’une source primaire ou non.

Dans le reste de cette section, nous présentons les structures de données de liens et traitons les questions liées au calcul des facteurs de forme et à la représentation multirésolution des liens. Dans la section 4 nous décrivons le raffinement des liens, et les mises à jour de visibilité ; dans la section 5 nous présentons notre stratégie de subdivision des polygones et les critères de raffinement perceptifs.

3.2 Structures de liens et facteurs de forme

Les structures de lien sont centrales dans notre méthode, à la fois pour les transferts et pour les raffinements. Par opposition aux méthodes antérieures, nous utilisons deux sortes de liens distinctes : les liens point-polygone sont utilisés pour collecter l’éclairage aux sommets, tandis que les liens polygone-polygone permettent les décisions de raffinement et la mise à jour de l’information de visibilité lors des subdivisions.




```
class LinkPtPoly {
    List<Arcs> Arcs
    Polygon Src
    float FF
}
```

(a) (b)

FIG. 4.7: (a) Un lien point-polygone utilisé pour collecter l’énergie lumineuse au sommet P. Tous les arcs du squelette entre P et le polygone *source* sont stockés dans le lien, comme la surface critique EV qui est représentée. (b) La structure de données correspondante.

Lien point-polygone

Comme nous l’avons déjà écrit, le squelette de visibilité fournit toute l’information requise pour calculer le facteur de forme exact entre tout polygone de la scène et tout sommet. Grâce à la mise à jour de vue que nous allons décrire dans la section, 4.3, cette possibilité sera étendue aux sommets issus de la subdivision.

Il y a de nombreux avantages à calculer l’éclairement aux sommets. Nous l’avons vu, le lissage est bien meilleure que lorsque une radiosité moyenne sur les éléments est considérée. De plus, il est moins coûteux de calculer des facteurs de forme exacts aux sommets qu’aux polygones. Le calcul de facteurs de forme exacts sommet-polygone a été introduit par Wallace *et al.* [WEH89] pour la radiosité progressive. Pour la radiosité hiérarchique, le fait que les sommets soient partagés entre niveaux rend la chose plus problématique.

Un lien point-polygone et la structure de données correspondante sont représentés figure 4.7. Les liens point-polygone sont stockés aux sommets des triangulations hiérarchiques. Ils contiennent la valeur du facteur de forme, ainsi que les arcs du squelette de visibilité (les événements visuels de la vue) entre le point et le polygone.

Le facteur de forme point-polygone est calculé de manière analytique en utilisant une formule classique (*e.g.* [BRW89]). Considérons la configuration de la figure 4.8.

$$F_{P,source} = \frac{1}{2\pi}\vec{N} \bullet \sum \gamma_i \frac{\vec{R}_i \times \vec{R}_{i+1}}{\| \vec{R}_i \times \vec{R}_{i+1} \|}$$

La somme est calculée à l’aide des arcs du squelette stockés au lien. $\vec{R}_i$ et $\vec{R}_{i+1}$ correspondent aux deux nœuds (droites poignardantes extrêmes) de l’arc.

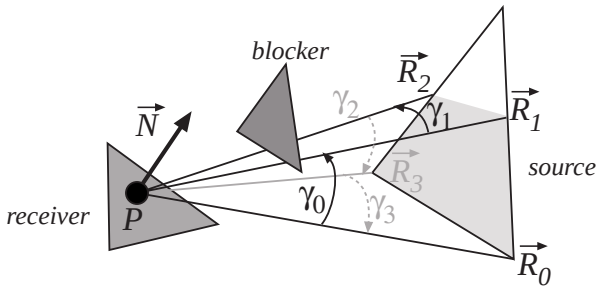
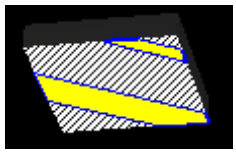


FIG. 4.8: Géométrie pour le calcul du facteur de forme.

La figure 4.9 donne un exemple de calcul de facteur de forme à l'aide du squelette de visibilité ; le calcul est exact. Nous le comparons avec un calcul par lancer de rayon avec échantillonnage perturbé (le noyau et la visibilité sont tous deux évalués par une méthode de Monte-Carlo). L'erreur moyenne relative est donnée, 36 rayons sont nécessaires pour que l'erreur numérique sur le facteur de forme soit en dessous de 10%. Comme on peut s'y attendre de la part d'échantillonnage stratifié, la convergence est en $O(n^{-\frac{3}{4}})$ [Mit96], puisque la fonction à intégrer est seulement continue par morceaux à cause du terme de visibilité. Dans la section 6.2 nous montrerons l'effet de cette erreur sur la qualité des images.



	Squelette	4 rayons	16 rayons	36 rayons	64 rayons	100 rayons
temps	0,07 ms	0,5 ms	1,7 ms	3,8 ms	6,7 ms	10,4 ms
erreur	0	50%	20 %	9,6%	7,6%	4,6%

FIG. 4.9: Exemple de calcul de facteur de forme depuis le point en blanc vers la source étendue en utilisant le squelette de visibilité et le lancer de rayon avec échantillonnage perturbé. La partie cachée de la source a été hachurée. Les temps donnés pour le squelette ne comprennent pas la mise à jour de l'information de visibilité (environ 0,13 ms par lien en moyenne pour cette image). Les temps de calculs ont été mesurés sur une Onyx2 avec un R10k à 195Mhz.

Représentation multirésolution des liens

Pour maintenir la représentation multirésolution de la radiosité sur la hiérarchie de triangulations, il est nécessaire de représenter la différence ΔB ainsi que nous l'avons décrit en section 2.2 pour le "pousser de la phase de "pousser-tirer".

Lorsqu'un nouveau sommet est inséré dans un polygone récepteur, des liens "négatifs" sont créés et stockés à son niveau. Ils correspondent aux échanges entre la source et les trois sommets parents (en pratique il s'agit de pointeurs vers ces liens). Ces liens permettent le calcul de ΔB de la façon suivante :

$$\Delta B = B_l - \sum_{i=0..2} c_i \, B^i_{nl}, \tag{4.1}$$

où B_l est la radiosité collectée le long du lien positif, B^i_{nl} la radiosité le long des liens négatifs et c_i les coordonnées barycentriques du point P_i. Un exemple de liens négatif est montré figure 4.10(b).

L'opération de collecte et pousser-tirer est donnée figure 4.11. Le champs *child* d'un *HPolygon* est sa triangulation associée.

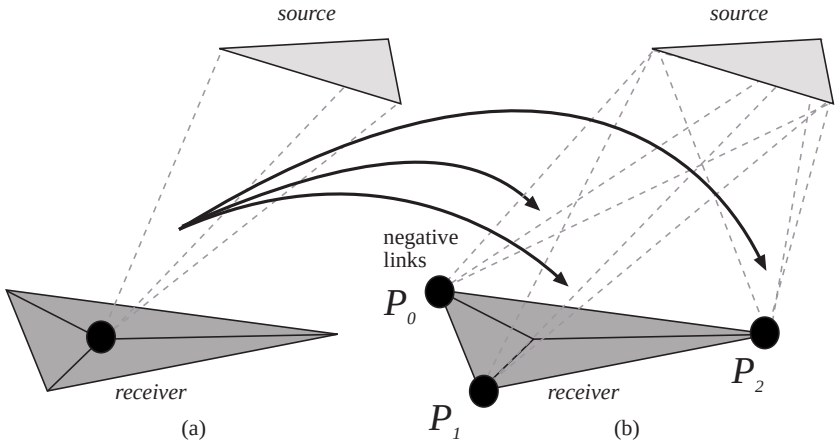


FIG. 4.10: Liens négatifs. (a) Un point nouvellement inséré (en noir) et le lien point-polygone vers la source. (b) Le sommet pointe vers trois liens négatifs vers la source utilisé pour le calcul de ΔB.

```
Collecte ()
{
   for each vertex v
      ΔBv = gather(positive linksv) - gather(negative linksv)
}
Pousser (HPolygon poly)
{
   if child(poly) == NULL return
   for each vertex v in triangulation child(poly)
      Bv = ΔBv+ interpolation (poly, v)
   for each triangle t in triangulation child(poly)
      Push(t)
}
Tirer (HPolygon poly)
{
   if child(poly) == NULL
      Bpoly = average of Bv, for all v vertex of poly
      return
   for each triangle t in triangulation child(poly)
      Pull(t)
   Bpoly =average of Bt, for all t in child(poly)
}
```

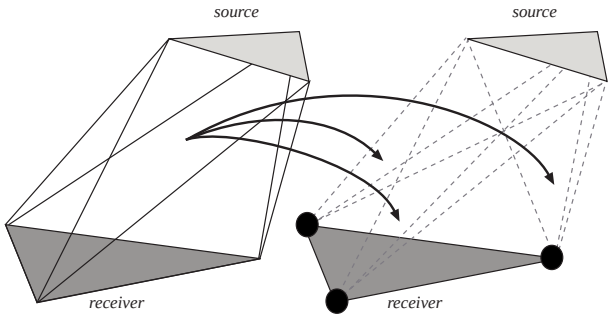
FIG. 4.11: Collecte et pousser-tirer

Lien polygone-polygone

Les liens polygones-polygones sont utilisés principalement pour déterminer à quel point le transfert entre deux polygones est bien simulé. Cette information est utilisée dans le raffinement que nous décrivons dans la section suivante.

Un lien polygone-polygone stocke l'information de visibilité grâce à des pointeurs sur les liens point-polygones correspondant (deux groupes de trois pointeurs pour une paire de triangles). Il stocke également l'ensemble des arcs du squelette dont les deux polygones sont les extrémités, ainsi que ceux qui ont une arête génératrice sur l'un des polygones, et une extrémités sur l'autre (figure 4.12).

Quand il y a subdivision, tous les liens polygone-polygone de triangles adjacents partagent des liens point-polygone.



(a) (b)

```
class LinkPolyPoly {
   List<LinkPtPoly> ptPolyLinks
   List<Arcs> Arcs
   Polygon Src
}
```

(c)

FIG. 4.12: (a) Un lien polygone-polygone est utilisé pour estimer le transfert entre deux polygones des triangulations. (b) Le lien polygone-polygone contient des pointeurs vers les six liens polygones-polygones correspondants, trois d'entre eux (*source* $\rightarrow$ *récepteur*)sont représentés. (c) Structure de données correspondante.

4 Raffinement des liens

Les structures de données pour les liens étant décrites, nous pouvons décrire en détail l'algorithme de raffinement des liens. L'opération est plus compliquée que dans le cas de la radiosité hiérarchique standard [HSA91], parce que la subdivision n'est pas régulière et à cause de l'existence de deux types de liens. Cette section décrit comment la mise à jour des liens est effectuée.

4.1 Aperçu du raffinement

Considérons un échange lumineux d'un polygone source vers un polygone récepteur. Puisque nous effectuons la collecte aux sommets, deux types de raffinements sont nécessaires.

– *Le raffinement de la source* si la radiosité varie trop sur la source ;

– *Le raffinement du récepteur* si l'échantillonnage est trop grossier sur le récepteur.

L'algorithme de raffinement est très simple : pour chaque polygone et chacun des ses liens polygone-polygone, on teste si le lien doit être raffiné. Si le test le décide, le lien est raffiné et de nouveaux liens point-polygone et polygone-polygone sont créés, et l'information de visibilité est mise à jour. Le test de raffinement utilise un critère de raffinement basé sur l'information de visibilité contenue dans le squelette de visibilité (section 5.4). Remarquons que puisque nous utilisons des facteurs de forme point-polygone exacts, le raffinement d'une source ne peut être causé par une imprécision dans le calcul. Il ne peut se produire que si la radiosité de la source n'est pas uniforme, c'est-à-dire si un raffinement du polygone source a été effectué lors d'un autre échange où il est considéré comme récepteur.

4.2 Raffinement de la source et raffinement du récepteur

Le premier type de raffinement est celui des sources. Si la représentation de la radiosité sur la source est insuffisante pour le transfert courant (c'est-à-dire si la radiosité varie trop sur la source), le liens est raffiné. La subdivision géométrique de la source a été effectuée lors d'une itération précédente, par exemple à cause d'une ombre. De nouveaux liens polygone-polygone sont créés entre le récepteur original et les sous-triangles de la source. De nouveaux liens point-polygone sont créés pour chaque sommet et pour chaque sous-triangle de la source ; l'information de visibilité est mise à jour comme nous le décrirons dans la section 4.3.

Le second type de raffinement est celui des récepteurs. Par exemple, figure 4.13, un point est ajouté au récepteur. La triangulation est mise à jour et trois nouveaux liens polygone-polygone sont créés. Nous en montrons un figure 4.13(b). De plus, un nouveau lien point-polygone est créé entre le nouveau point et la source (figure 4.13(c)).

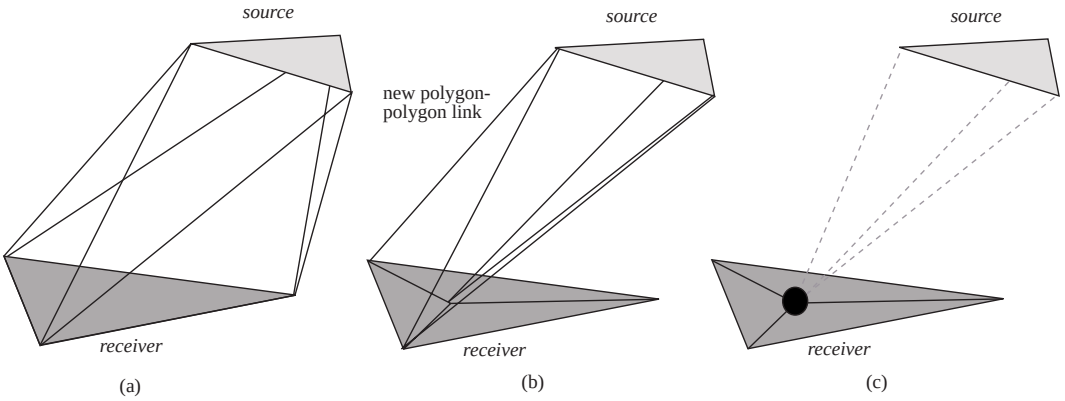


FIG. 4.13: Raffinement d'un récepteur. (a) Lien polygone-polygone original. (b) Insertion d'un point sur le récepteur. Nous représentons l'un des trois liens polygone-polygone créés. (c) Le liens point-polygone supplémentaire vers la source.

4.3 Mise à jour de la visibilité

Chaque opération de raffinement requiert la mise à jour de l'information de visibilité correspondante. Nous distinguons là aussi le cas du raffinement des sources et des récepteurs.

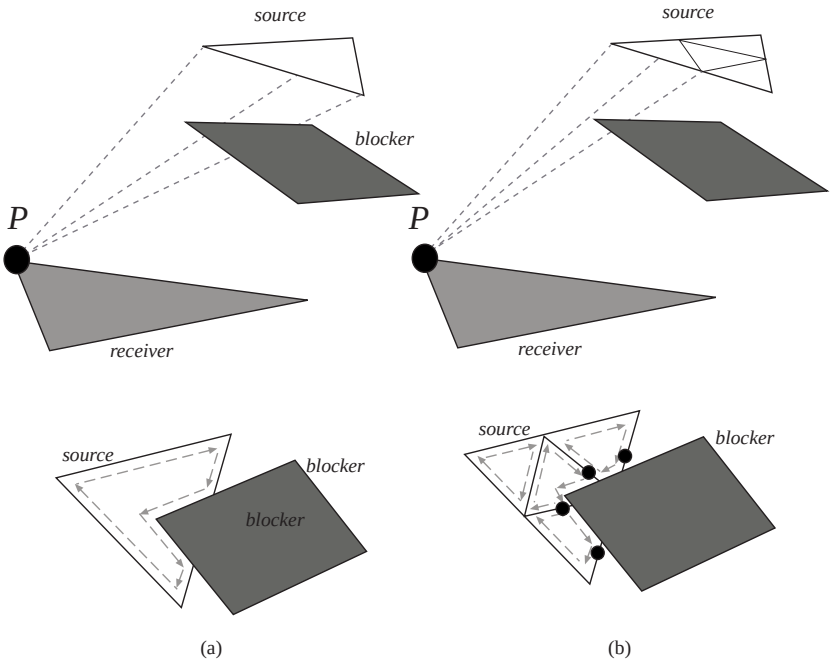


FIG. 4.14: Mise à jour de l'information de visibilité pour la source. Les flèches pointillées (en bas) représentent les limites de la partie visible de la source (utilisée pour le calcul des facteurs de forme). (a) Le lien point-polygone avant subdivision, avec en dessous la vue correspondante de la source. (b) L'un des 4 nouveaux liens point-polygone dû à la subdivision et les 4 nouvelles vues de la source. Les cercles en noir correspondent à de nouveaux nœuds du squelette.

Dans le cas du raffinement d'une source, nous devons mettre à jour l'information de visibilité contenue dans les nouveaux liens point-polygone. Puisque l'information de visibilité d'un de ces liens correspond à la vue de la source depuis un point du récepteur, nous devons restreindre cette information à chacun des 4 sous-triangles. Par exemple, dans la figure 4.14(a) la vue originale depuis le sommet du récepteur est représentée en bas. Lors de la subdivision de la source, 4 nouvelles vues sont calculées par rapport à chacun des sous-triangles, ainsi que

nous le montrons en bas de la figure 4.14(b). Les nouveaux liens point-polygone contiennent une référence aux arcs du squelette qui correspondent aux parties de la vue affectée. Ainsi, le sous-triangle de gauche est complètement visible depuis *P*, et aucun arc n'est stocké. Pour les autres, l'intersection des arcs existants et des limites des sous-triangles crée de nouveaux nœuds du squelette (les cercles en noir). Les arcs correspondant sont subdivisés et liés à ces nouveaux nœuds. Toutes ces mises à jour sont en fait des opérations 2D.

Le raffinement d'un récepteur est plus complexe. Lorsqu'un nouveau point est inséré, de nouvelles vues doivent être calculées. Nous utilisons les algorithmes de construction du squelette qui est robuste. La seule différence est que nous utilisons les listes de bloqueurs définies par les arcs stockés dans le lien polygone-polygone initial, au lieu d'utiliser la scène entière. Puisque leur nombre est relativement restreint, le coût d'une nouvelle vue est faible. Nous donnons un exemple figure 4.15(a)-(b), où le point *P* est inséré sur le récepteur. Sur la figure 4.15(b) on peut voir le nouveau lien point-polygone et sur la figure 4.15(c) la nouvelle vue est représentée. Les cercles noirs correspondent aux nouveaux nœuds du squelette.

Le cas de visibilité totale est détecté en utilisant l'information contenue dans le lien polygone-polygone. La mise à jour de visibilité est alors optimisée : aucun arc n'est calculé ni stocké, le facteur de forme sans occultation est utilisé, ce qui permet de gagner du temps et de la mémoire.

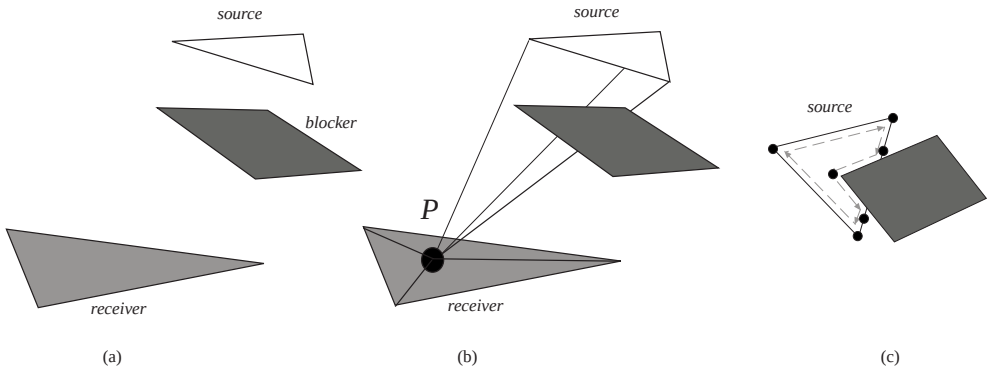


FIG. 4.15: Mise à jour de la visibilité pour le récepteur. (a) configuration initiale. L'information sur les bloqueurs est contenue dans le lien polygone-polygone. (b) Un point est inséré sur le récepteur, ce qui crée un lien point-polygone. (c) La nouvelle vue de la source depuis *P* est calculée. Ce calcul permet également de mettre à jour la liste de bloqueurs.

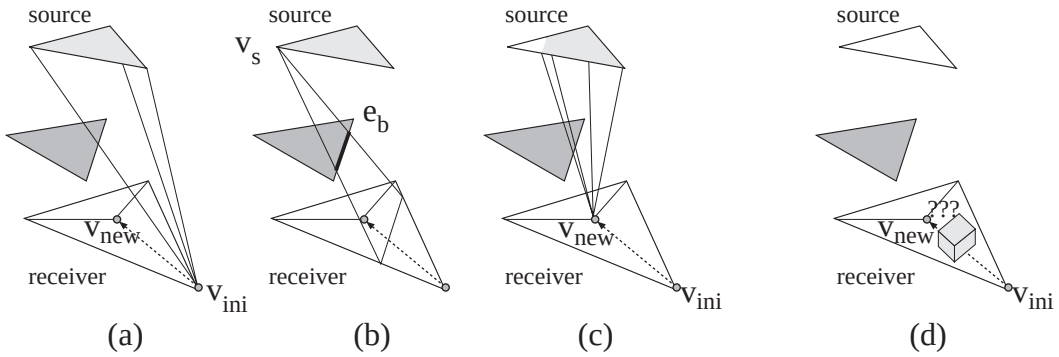


FIG. 4.16: Raffinement de l'information de visibilité sur le récepteur à l'aide des événements visuels (cette solution n'a pas été implémentée). (a) Nous commençons avec la vue depuis l'un des sommets initiaux. (b) Nous marchons sur le récepteur en direction du nouveau sommet. Nous traversons un événement $v_s e_b$. v_s commence à être occulté par le bloqueur. (c) Nous obtenons ainsi la vue depuis le nouveau sommet. (d) Dans le cas de contact d'objets, l'information ne peut être maintenue lorsque l'on traverse l'interface entre les deux objets.

4.4 Autre possibilité pour la mise à jour des vues

La mise à jour de la visibilité pourrait être effectuée en utilisant l'information stockée dans le squelette de visibilité. On peut partir d'un des sommets initiaux, marcher en direction du nouveau sommet et mettre à jour la vue chaque fois qu'un événement visuel est traversé (cf. figure 4.16). Cette méthode est cependant difficile à mettre en œuvre et elle souffre de problèmes de robustesse si les événements visuels ne sont pas traversés dans un ordre cohérents (si ce n'est exact). De plus, le cas d'objets en contact complique encore le problème, puisqu'aucune information ne peut être maintenue quand on passe "en dessous" d'un objet.

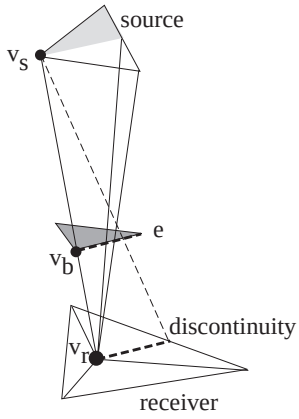


FIG. 4.17: Dégénérescence due a maillage de discontinuité. Le récepteur est subdivisé le long de la discontinuité $v_s e$, ce qui cause une droite extrême dégénérée $v_s v_b v_r$.

4.5 Traitement des dégénérescences

La subdivision le long des discontinuités entraîne des points de vue dégénérés. Par exemple, dans la figure 4.17, nous subdivisons le récepteur le long de la discontinuité $v_s e$. La vue depuis v_r présente une droite extrême dégénérée $v_s v_b v_r$. Pour la traiter de manière cohérente, nous stockons avec chaque sommet de la triangulation un pointeur (potentiellement nul) vers la droite extrême qui l'a créé. Cette solution est plus simple que notre lancer de rayon modifié pour les objets rasants décrit section 4.3. Le traitement de la dégénérescence est ensuite effectué de la même façon.

Une autre possibilité aurait été de perturber légèrement la position des points pour éviter ces dégénérescences. Nous ne l'avons pas fait pour deux raisons. Tout d'abord, le maillage de discontinuité nous permet de délimiter des régions de visibilité totale, des régions de pénombre et des régions d'ombre. Si nous perturbons les points, des régions qui auraient dû être dans l'ombre totale auront une petite partie dans la pénombre et requerront plus de subdivision. De plus, la perturbation des points causerait des problèmes de précision numérique.

5 Subdivision des polygones

Nous avons expliqué comment les liens et l'information de visibilité sont mis à jour. Bien évidemment, ces mises à jour résultent d'une décision de *raffinement* basée sur un critère approprié. Nous avons choisi d'utiliser un critère basé sur une métrique *perceptive*. Dans ce qui suit, nous passons tout d'abord en revue les bases de la reproduction de nuance que nous utiliserons pour notre oracle. Nous présentons ensuite le processus de subdivision des polygones et détaillons le raffinement que nous avons utilisé.

5.1 Différence tout juste perceptible

Les travaux dans le domaine de la perception nous fournissent deux résultats importants. Tout d'abord, ils permettent la conversion de données radiométriques en couleurs affichables, tout en préservant l'impression

subjective qu'un observateur ressentirait en voyant la scène. De plus, ils nous permettent d'utiliser des seuils qui sont liés à l'erreur qu'un observateur est capable de percevoir, ce qui rend les réglages bien plus simples.

Le système visuel humain peut percevoir une large gamme de luminosités, tandis que les affichages informatiques sont habituellement limités à un intervalle de 1 à $100 cd/m^2$ [GH97b]. L'œil s'adapte en fonction de la luminosité de la scène qu'il regarde. Cela explique que nous soyons capable de voir des scènes de nuit et des scènes ensoleillées très lumineuses. L'opération de reproduction de nuance (*tone-mapping*) permet de convertir des quantités radiométriques à grande dynamique en couleurs de dynamique plus restreinte, en essayant de préserver la sensation ressentie par l'observateur. Une méthode simple consiste à diviser toutes les quantités par la radiosité maximale de la scène. Le problème de cette approche est que si l'intensité des sources est divisée par deux, l'image affichée sera exactement la même, alors que l'on souhaiterait qu'elle paraisse plus sombre.

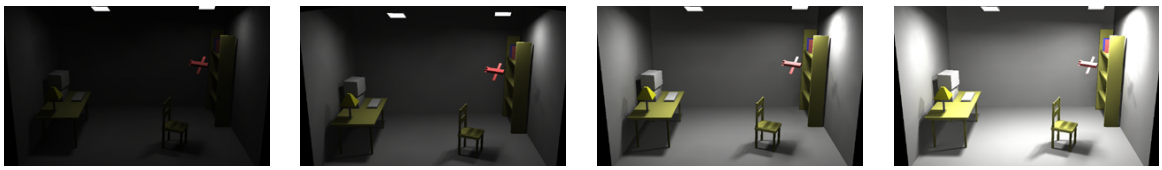


FIG. 4.18: Effet de la reproduction de nuance de Ward quand l'intensité des sources varie. Les détails restent perceptibles et l'impression de luminosité ou d'obscurité est préservée.

L'opérateur de reproduction de nuance de Ward [War94] permet de préserver le contraste. Un simple facteur d'échelle sf est utilisé pour l'ensemble de la scène. Il dépend de la luminance maximale affichable L_{dmax} , et du niveau d'adaptation du monde L_{wa} qui est en général la moyenne logarithmique de la luminosité de la scène sans les sources primaires. Dans ce qui suit, toutes les intensités sont exprimées en $candelas/meter^2$ (un *candela* est un $lumen/steradian$ [War94]). Le facteur sf est donné par :

$$sf = \frac{1}{L_{dmax}} \left[\frac{1.219 + (L_{dmax}/2)^{0.4}}{1.219 + L_{wa}^{0.4}} \right]^{2.5}$$

La figure 4.18 montre l'effet de cet opérateur pour une scène donnée lorsque l'on modifie l'intensité des sources de lumière.

Nous utilisons une technique similaire à celle de Gibson *et al.* [GH97b] pour calculer le niveau d'adaptation. Nous utilisons un niveau d'adaptation statique qui est la moyenne de la radiosité dans la scène. Cependant, comme nous utilisons la radiosité hiérarchique et non la radiosité progressive, nous évitons l'utilisation d'une estimation basée sur la réflectance moyenne. Au lieu de cela, nous utilisons à tout instant la valeur moyenne de radiosité des polygones de la scène. C'est pourquoi nous commençons par plusieurs itérations de collecte-pousser-tirer pour obtenir une première estimation de l'équilibre énergétique.

L'utilisation d'un niveau d'adaptation global n'est qu'une approximation de l'adaptation du système visuel. Ainsi que l'ont montré Gibson *et al.* cela donne cependant une bonne estimation du niveau d'adaptation dynamique [GH97b]. Des solutions plus élaborées pourraient être explorées, comme l'utilisation de niveaux d'adaptation locaux calculés au voisinages des objets. Des opérateurs plus évolués pourraient également être utilisés [TR93, FPSG96].

Une fois que l'opérateur de reproduction de nuance a été appliqué, l'erreur admissible peut être exprimée comme un pourcentage de l'intensité maximale affichable L_{dmax}. Des études psycho-visuelles [GH97b, Mur87] ont montré que l'œil humain est cabale de distinguer une différence de 2% : il s'agit de la *différence tout juste perceptible*. Nous appellerons l'erreur autorisée ε_{percep}, et utiliserons $\varepsilon_{percep} = 2\%$ pour tous nos critères de raffinement.

5.2 Subdivision des polygones

Nos expériences ont montré que subdiviser le long des discontinuités au cours des premiers raffinements causait la création de triangles ayant un mauvais rapport d'aspect (triangles trop fins), ce qui entraîne des artefacts visibles. C'est pourquoi nous subdivisons les polygones en deux temps :

– *Lors des deux premières subdivisions :* les polygones sont subdivisés de manière régulière selon une grille.

```
subdivisePolygone() {
   for each polygon r et each poly-poly link s to r
      if shouldRefine Link(s, r)
         refine source
   for each polygon r et each poly-poly link s to r
      if shouldRefine Link(s, r)
         if iteration < 3
            regularSubdivision( r ) // subdivision en grille
         else
            find et insert discontinuities in r
   complete subdivision at this level // creation des sous-triangles
}
```

FIG. 4.19: Subdivision des polygones

– *Lors des subdivisions suivantes :* les discontinuités sont insérées.

Cette approche est similaire à celle de Stuerzlinger [Stu94] et Hardt et Teller [HT96]. L'algorithme de subdivision des polygones est esquissé figure 4.19. Contrairement à la radiosité hiérarchique classique, nous ne pouvons subdiviser les polygones au vol quand un lien a besoin d'être raffiné, car notre subdivision n'est pas uniforme et doit être effectuée le long des discontinuités. C'est pourquoi nous considérons d'abord tous les liens d'un polygones afin de décider s'il nécessite une subdivision et pour déterminer quelles discontinuités doivent être insérées. Les discontinuités sont alors insérées et la triangulation de Delaunay contrainte est mise à jour.

5.3 Insertion des maxima

Si la radiosité provenant d'une source est considérée comme une fonction sur le récepteur, il a été montré que la subdivision du maillage au maximum de cette fonction peut accroître la précision de la simulation [DF93]. Nous calculons le maximum des fonctions de radiosité sans occultation pour les sources de lumière avant le premier raffinement.

Les maxima sont considérés uniquement pour les transferts lumineux importants (ce que nous estimons à l'aide de la formule disque-disque [HSA91] et de la métrique perceptive). Étant donné un polygone source et un récepteur, nous utilisons une descente de gradient pour localiser le maximum. Nous calculons alors la contribution de la source en ce point ; si elle est supérieure à ε_{percep}, le maximum est stocké pour être ensuite inséré dans le maillage. La radiosité du récepteur est mise à jour pour tenir compte de ce maximum. Nous effectuons une collecte avant même la création d'un lien afin d'obtenir une meilleure estimation de la distribution de lumière.

Les maxima sont insérés lors de la subdivision initiale. Les points de la subdivision régulière qui sont trop proches d'un maximum ne sont pas insérés. Un exemple est donné figure 4.2(b), où le maximum correspond au point en bas à gauche qui n'est pas exactement aligné avec les autres. Nous obtenons ainsi un maillage quasi-régulier avec des triangles qui ont un bon rapport d'aspect.

La recherche des maxima est appliquée de manière itérative pour prendre en compte l'éclairage indirect. L'insertion du maximum des sources indirectes est très important, par exemple figure 4.24, où la table (éclairée par la lampe) est la source de lumière principale pour la partie supérieure du mur de gauche.

5.4 Critère de raffinement

Nous distinguons deux critères de raffinement (ou oracles) : un critère radiométrique qui prend en compte la variation de la radiosité sans occultation et un critère de visibilité (ou discontinuité). Le critère de visibilité guide aussi le choix des courbes de discontinuité à insérer.

L'oracle radiométrique estime si l'interpolation linéaire du transfert énergétique est suffisamment précise. Nous échantillonnons le facteur de forme sans occultation ([BRW89] et section 3.2) au centre du polygone et aux centres des arêtes, et comparons cette valeur à celles interpolées. Si la valeur (après transformation perceptive) est supérieure à ε_{percep}, nous effectuons la subdivision.

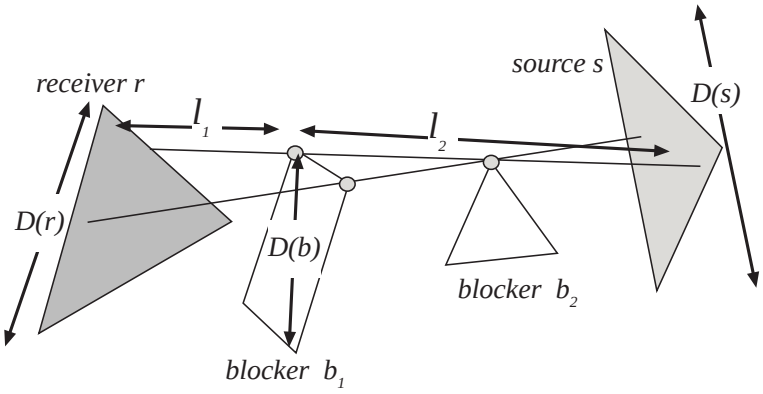


FIG. 4.20: Géométrie pour le critère de raffinement

Le principe de notre oracle de visibilité est d'estimer (comme un pourcentage de l'intensité maximale affichable) la "quantité d'ombre" causée par les bloqueurs, c'est-à-dire la quantité de lumière supplémentaire qui serait échangée sans le bloqueur. Notre critère de raffinement est en trois étapes : estimation sans occultation, estimation de la quantité d'ombre et estimation de la dureté d'ombre.

Considérons un récepteur et une source. Rappelons qu'une source peut être n'importe quel polygone de la scène qui est considéré comme une source lors de cette étape du raffinement. Dans ce qui suit, nous utilisons les quantités géométriques définies figure 4.20.

Tout d'abord, nous estimons le transfert énergétique sans occultation B_{unoc} en utilisant la formule disque-disque [HSA91]. Comme précédemment, si cette valeur est inférieure à ε_{percep}, le lien n'est pas subdivisé.

Ensuite, nous considérons chaque événement visuel entre la source et le récepteur et estimons la quantité d'ombre. Pour ceci, nous estimons la partie de la source potentiellement cachée par le bloqueur, en utilisant le diamètre du bloqueur projeté sur la source :

$$D_{proj}(b) = D(b) * \frac{l_2}{l_1}$$

Le pourcentage d'occultation estimé est alors :

$$occlu = \frac{\frac{\pi}{4} D_{proj}(b)^2}{Area_{source}}$$

(limité à 1). La quantité d'ombre est :

$$shadow = B_{unoc} * occlu$$

Si *shadow* est inférieur à ε_{percep}, l'événement visuel est ignoré.

Pour finir, nous estimons la "dureté" de l'ombre. L'étendue de la zone de pénombre est approximée en projetant le diamètre de la source sur le récepteur :

$$D(penumbra) = D(s) * \frac{l_1}{l_2}$$

Si la taille du récepteur est plus grande que la zone de pénombre, il peut contenir des zones où la source est complètement visible et des régions où le bloqueur se projette complètement sur la source. Dans ce cas, la fraction d'occultation est maximale et approximée par *occlu*. *shadow* fournit alors une approximation de la variation de radiosité sur le récepteur. Dans l'autre cas, nous faisons l'approximation qui considère que la radiosité varie linéairement dans la pénombre ; la variation de radiosité est alors :

$$\Delta(B) = \begin{cases} shadow & si\ D(penumbra) > D(r) \\ shadow * D(r)/D(penumbra) & sinon \end{cases}$$

Tous les liens qui possèdent des événements visuels tels que $\Delta(B) > \varepsilon_{percep}$ seront subdivisés. Comme nous l'avons expliqué ci-dessus, les deux premières subdivisions sont régulières. Dans les subdivisions suivantes,

nous insérons les discontinuités ayant les plus grandes valeurs de $\Delta(B)$. C'est la phase de classement des discontinuités, similaire à celle de Hardt et Teller [HT96].

$\Delta(B)$ est calculé en utilisant les valeurs de l_1 et l_2 aux deux nœuds extrémités de l'arc et en prenant le maximum. L'évaluation de ces oracles est très rapide car les liens et les événements sont éliminés dès que nous pouvons déterminer qu'ils ne nécessitent pas de raffinement.

6 Implémentation et Résultats

6.1 Implémentation

Nous avons utilisé l'implémentation en C++ du squelette de visibilité décrite dans le chapitre précédent, avec l'optimisation pour le stockage des arcs au niveau des polygones.

Les triangulations hiérarchiques utilisent l'implémentation de la méthode de Guibas et Stolfi [GS85] par Dani Lischinski [Lis94], qui permet des triangulations contraintes.

Sur nos scènes de test, l'algorithme passe la majeure partie de son temps pour la mise à jour de visibilité, en particulier pour le calcul des nouvelles vues lors du raffinement des récepteurs. Par exemple, dans la scène de bureau de la figure 4.21, pour la dernière itération, le calcul des critères de raffinement prend 15 seconds, la mise à jour de visibilité 64 secondes, et la collecte-pousser-tirer 2,5 secondes.

Nous avons choisi de ne pas utiliser de textures pour nos exemples, car elles masquent généralement la précision de la simulation.

6.2 Résultats

Nous présentons des résultats pour quatre scènes différentes. La première scène (*Bureau*) est un simple bureau contenant 438 polygones et deux grandes sources de lumière (figure 4.21). Cette scène permet d'illustrer les caractéristiques générales de notre méthode. La deuxième scène contient la même géométrie, mais possède 8 petites sources puissantes additionnelles. La luminosité des deux grandes sources a été diminuée. Nous appelons cette scène "Nombreuses" (cf. figure 4.22). Cette scène montre comment notre algorithme traite les sources de lumière multiples. La troisième scène a été choisie pour montrer les performances de notre méthodes pour les scènes où l'éclairage indirect domine. Nous avons choisi un exemple courant de chambre illuminée uniquement vers le bas par une petite lampe de chevet. La plupart de la scène est éclairée de manière indirecte. Nous appelons cette scène "Lit" (figure 4.24). Pour finir, afin de montrer un type de scène résolument différent, nous montrons les résultats de notre approche sur un scène de "Village" contenant des maisons et des voitures (figure 4.27).

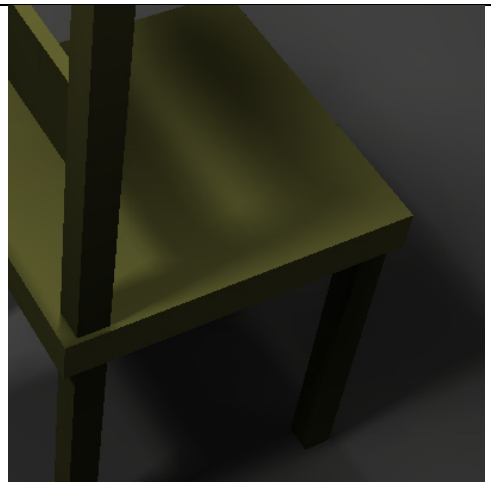
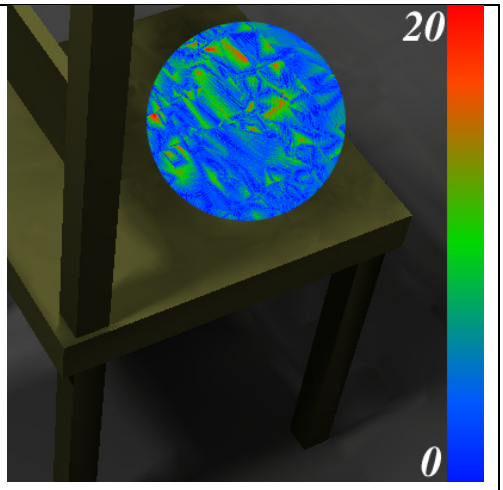
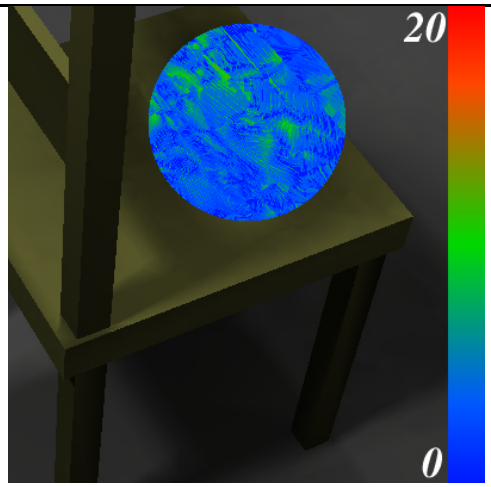
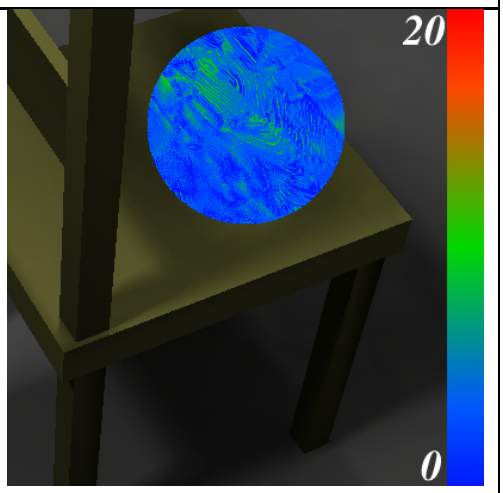
Dans ce qui suit, nous présentons diverses statistiques ainsi que des comparaisons avec un algorithme de radiosité hiérarchique à base de quadtrees, mais utilisant un critère de raffinement amélioré à l'aide d'une stratégie de bornes d'erreurs ([GH96, LSG94]).

Tous les temps de calculs ont été mesurés sur une station de travail Silicon Graphics Onyx 2 avec un processeur R10 000 à 195 MHz.

Avant de présenter les résultats pour l'algorithme complet, nous donnons des statistiques sur l'importance de la précision du calcul des facteurs de forme.

Importance de la précision

Nous avons effectué des tests pour évaluer l'impact des calculs exacts de facteurs de forme sur la qualité des images (table 4.1). Nous avons légèrement modifié notre implémentation pour calculer les facteurs de forme à l'aide de lancer de rayon sur un échantillonnage de la source selon une grille perturbée (*jittered grid sampling*), tandis que le raffinement est le même dans tous les cas, basé sur les événements visuels et le squelette de visibilité. Le coût de la mise à jour du squelette n'est pas inclus dans les temps du lancer de rayon, qui ne couvrent que le calcul du facteur de forme. Comme nous l'avons déjà mentionné (cf. figure 4.9), les calculs inexacts des facteurs de forme introduisent des erreurs significatives. Leur conséquence visuelle peut être vue dans la table 4.1. De plus, le surcoût n'est pas négligeable si une bonne qualité est requise, car au moins 64 rayons sont alors nécessaires.

Image		
Méthode	exacte (Squelette)	16 rayons
Temps total	1min 19	1min 17
Image		
Méthode	36 rayons	64 rayons
Temps total	2min 02	3min 04

TAB. 4.1: Importance de la précision des facteurs de forme pour une petite scène de 246 polygones. Le nombre de rayons pour l'éclairage indirect est fixé à 4, tandis que leur nombre varie pour l'éclairage direct. Nous montrons en encart en fausses couleurs le différence avec le calcul exact à l'aide du squelette de visibilité, dans l'espace de couleur perceptivement uniforme *CIE L*a*b**.

L'effet sur les images est particulièrement fort car la subdivision due au maillage de discontinuité n'est pas uniforme. Les fins triangles entraînent des artefacts très visibles. Ces résultats confirment ceux de Drettakis et Sillion [DS96].

Solution générale

Les images de la figure 4.21 montrent les différentes étapes de notre algorithme. La figure 4.21(a) montre le résultat des trois étapes de collecte-pousser-tirer sur la scène avant subdivision. À ce stade, nous possédons déjà une approximation grossière de la distribution globale de l'éclairage dans la scène. La figure 4.21(b) montre la première subdivision qui est une grille quasi régulière avec insertion des maxima. Les figures 4.21(c) et (d) montrent la suite de notre algorithme. La figure 4.21(e) montre les discontinuité que nous avons effectivement insérées. Cela comprend des discontinuités pour tous les transferts lumineux (directs et indirects) ; leur nombre est bien plus faible que pour une approche standard de maillage de discontinuité (dans ce cas, 40% des discontinuités causées par les sources primaires sont insérées).

Scène	*Pol*	*Squel*	*1ere*	*2eme*	*3eme*	*Total*	*Mem(ini/tot)*	*liens*	tris
Bureau	444	2min 08	22s	16s	1min 14	4min	40/200Mo	378K	46K
Nombreuses	492	2min 23	2min 38	55s	4min 27	10min 23	47/365Mo	1546K	104K
Lit	534	4min 12	1min 25	58s	4min 35	11min 10	56/400Mo	383K	43K
Village	312	45s	12s	7s	24s	1min 28	15/43Mo	134K	28K

TAB. 4.2: Temps de calcul et utilisation mémoire pour les scènes de test. Les statistiques d'utilisation mémoire comprennent la mémoire initialement requise par le squelette de visibilité de la scène en entrée (avant subdivision) et le total à la fin de la simulation de l'éclairage.

La table 4.2 donne les statistiques des simulations d'éclairage de nos scènes de test. Pour la scène "Bureau", on peut voir que la simulation complète nécessite 4 minutes. La qualité de la solution est très bonne et présente des ombres bien définies sur toutes les surfaces. Le nombre total de liens point-polygone est 378 746 et le nombre de triangles feuilles est 46 058.

Traitement de nombreuses sources

Notre approche se comporte particulièrement bien pour les scènes contenant de nombreuses sources de lumière. C'est ce que montre notre deuxième scène, qui contient 10 lampes et la même géométrie que "Bureau". La figure 4.22(a) montre un aperçu de la scène rendue grâce à notre approche et la figure 4.22(c) propose un gros plan sur le sol. Les ombres dues aux multiples sources de lumière sont bien représentées là où il faut. Notre algorithme de classement des discontinuités basé sur une métrique perceptive a correctement choisi les discontinuités importantes, puisque l'influence combinée des différentes sources est prise en compte, comme le montre le (relativement) faible nombre de discontinuités insérées dans la figure 4.22(d). La table 4.2 montre que le nombre de liens pour cette scène est 1,5 millions et que le temps de calcul total est de 10 minutes et 23 secondes. Seulement 10% des discontinuités directes sont insérées.

Nous proposons une comparaison informelle avec une implémentation de l'algorithme de radiosité hiérarchique avec regroupement (*clustering*) proposé par Gibson et Hubbold [GH96] en utilisant le principe de propagation des bornes d'erreurs de Lischinski *et al.* [LSG94]. Pour la scène "Nombreuses", le calcul utilise alors 1 million de liens et le temps de calcul est de 2 heures (table 4.3). De plus, la qualité du résultat est plus faible, les ombres sont moins nettes, ou même manquantes (cf. figure 4.23). Un plus grand nombre de liens serait nécessaire pour obtenir une image de qualité similaire à celle de la figure 4.22. Bien que cette méthode utilise approximativement le même nombre d'éléments (110K contre 104K pour notre méthode) la qualité des images n'est pas aussi bonne.

Scène	*Poly*	*1ere*	*2eme*	*3eme*	4eme	*Total*	*Mem*	*liens*	elems
Nombreuses	492	1 hr 25	22 min	10 min	-	1 hr 57	147 Mo	1098K	110K
Lit	534	11 min	37 min	6 min	25s.	54 min	94 Mo	903K	32K

TAB. 4.3: Comparaison des temps de calcul et de la mémoire utilisée pour nos scènes de test en utilisant la radiosité hiérarchique avec bornes d'erreur de Lischinski [LSG94, GH96].

Éclairage indirect

Les scènes où l'éclairage indirect domine sont un autre défi pour la simulation de l'éclairage. C'est pour ce type de scène que notre méthode de calcul de facteur de forme et de traitement des discontinuités se révèle puissante. Les approches antérieures requièrent un temps de calcul bien plus grand pour obtenir un niveau de précision similaire pour la simulation de l'éclairage indirect.

Ceci est illustré par les figures 4.24 et 4.25, dans lesquelles la lampe de chevet dirigée vers le bas (aucune lumière n'arrive par les côtés ni par le haut). De ce fait, toute la partie de la scène au dessus de la lampe est éclairée de manière indirecte.

Notre algorithme utilise un nombre de liens point-polygone relativement faible (383 715) et réussit cependant à représenter fidèlement les ombres dues à l'éclairage indirect. Par exemple, les ombres causées par la lampe de droite ou par les livres sont dues à la propagation de la lumière réfléchie par la table de chevet et par le lit.

Une autre comparaison informelle est présentée avec le même algorithme [GH96, LSG94] que pour la section précédente. La simulation par radiosité hiérarchique utilise pratiquement un million de liens et prend un peu moins d'une heure pour produire des résultats de moins bonne qualité (figure 4.26(a) et (b)).

De plus, les avantages de notre représentation à base d'ondelettes paresseuses linéaires sont bien illustrés sur la vue générale de la solution de radiosité hiérarchique. La partie gauche du mur du fond est plus lumineuse que la partie droite, et une forte discontinuité au milieu révèle le quadtree sous-jacent. La faute en incombe à l'interpolation qui n'est appliquée qu'au niveau le plus fin du maillage comme un post-calcul après la simulation, les échanges simulés à un niveau supérieur ne sont donc pas correctement interpolés.

Scène de village

Pour finir, nous présentons une scène de village dans la figure 4.27, afin de montrer que notre algorithme peut être utilisé pour d'autres type de scènes. Le village est éclairé par un rectangle dans le ciel et par les phares des voitures.

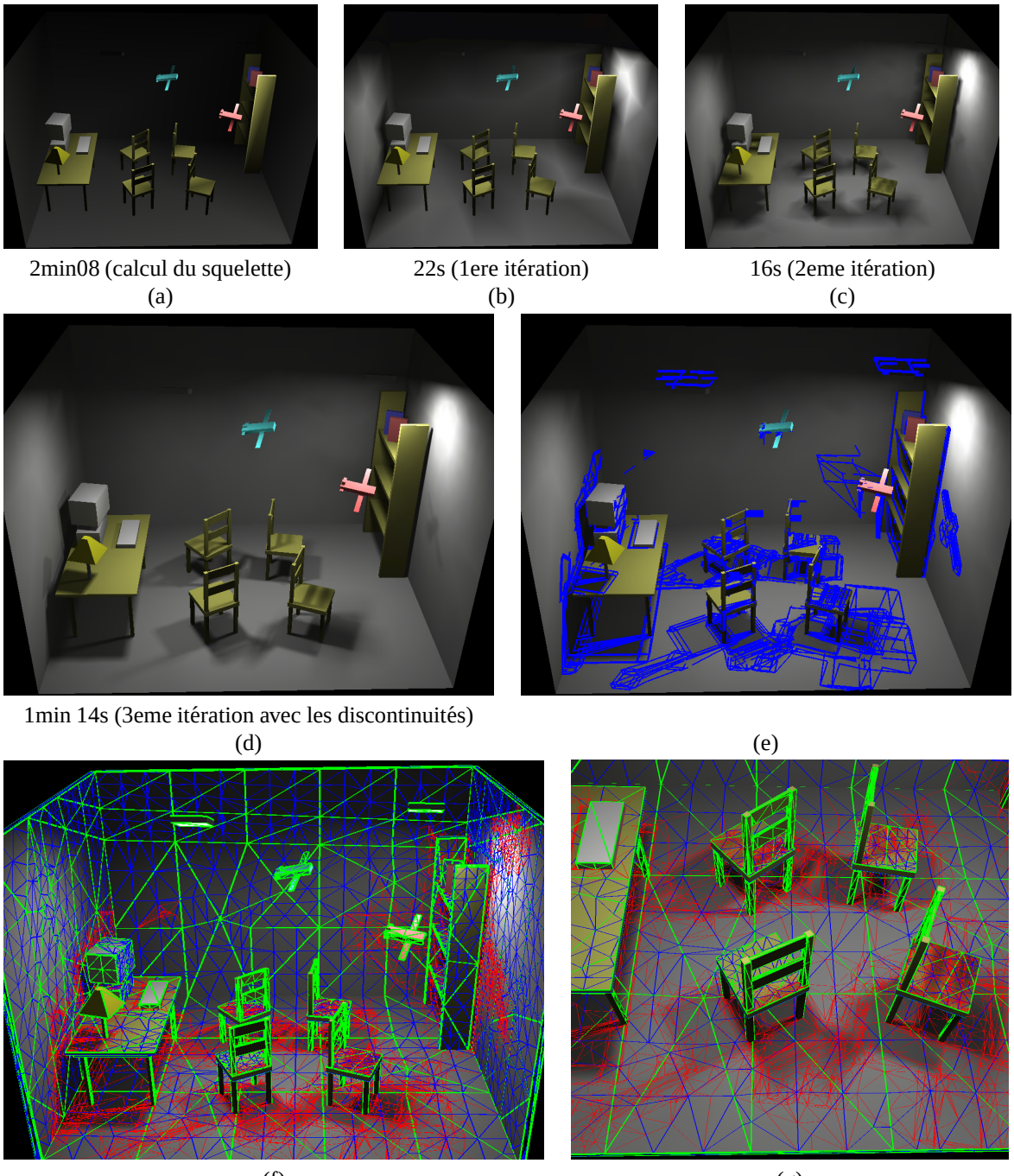
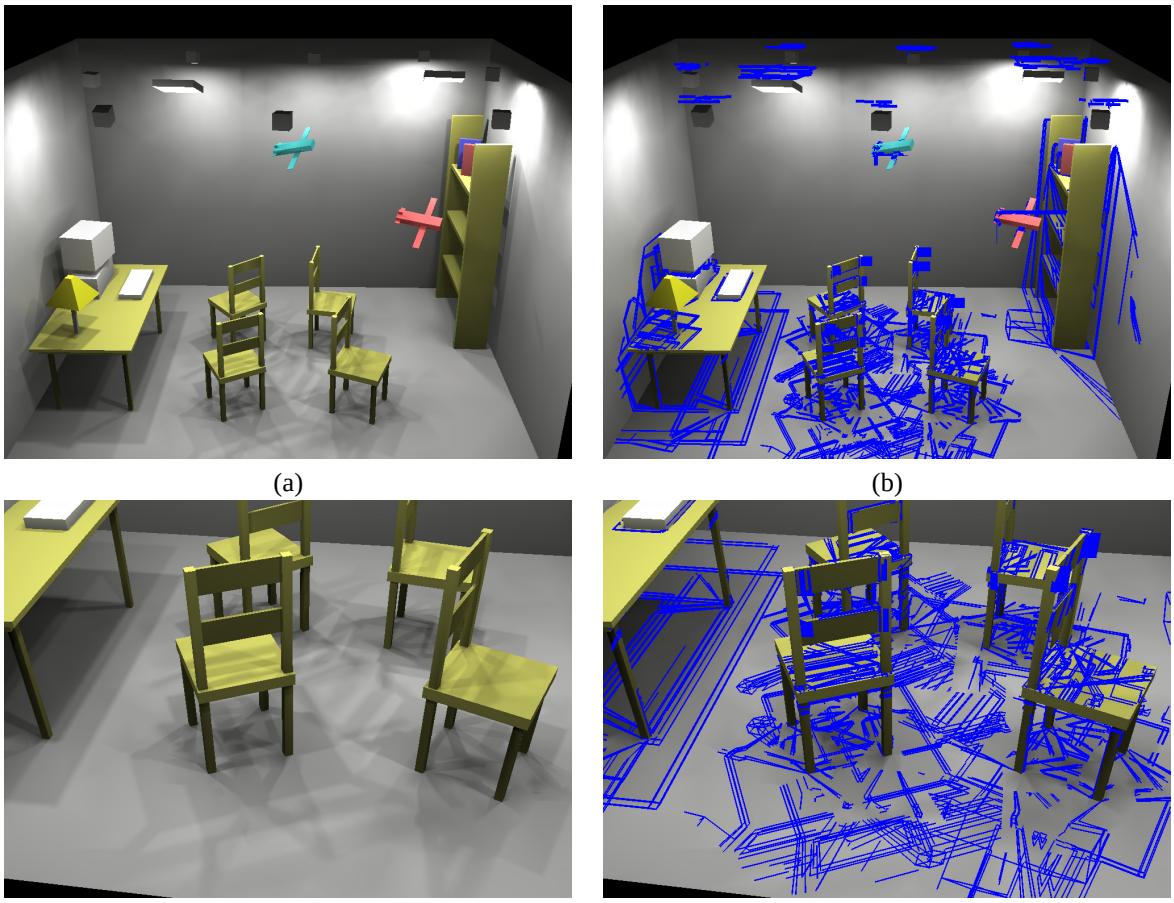


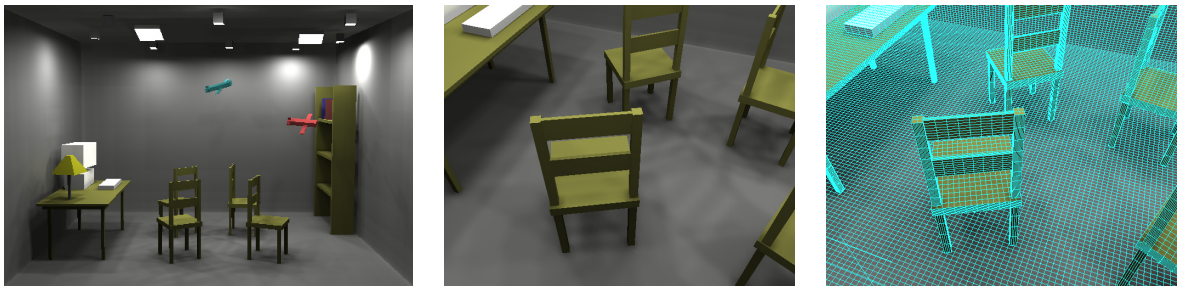
FIG. 4.21: Scène de bureau. (a) Scène sans subdivision. (b) Première subdivision selon une grille régulière et les maxima. (c) Deuxième itération. (d) Résultat de la troisième itération, avec maillage de discontinuité. (e) discontinuités effectivement insérées. (f) et (g) Maillage hiérarchique correspondant (premier niveau en vert, deuxième en bleu et troisième en rouge).



(a) (b)

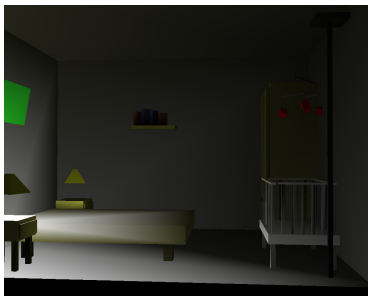
(c) (d)

FIG. 4.22: Scène avec de nombreuses sources de lumières. (a) Image finale. (b) Discontinuités effectivement insérées. (c) et (d) Gros plan sur le sol.

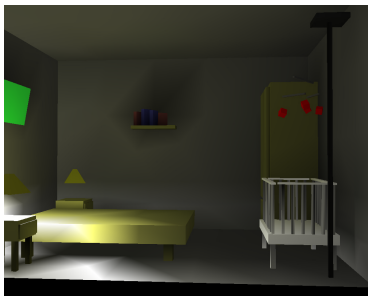


(a) (b) (c)

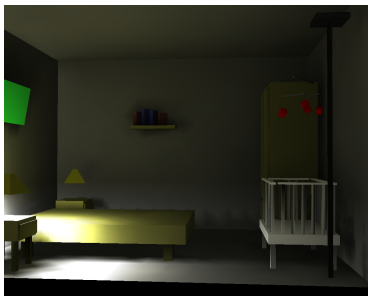
FIG. 4.23: Comparaison informelle avec la radiosité hiérarchique pour la scène “Nombreuses”. (a) Maillage final. (b) vue générale de la scène à la fin de la simulation. (c) Gros plan sur le sol.



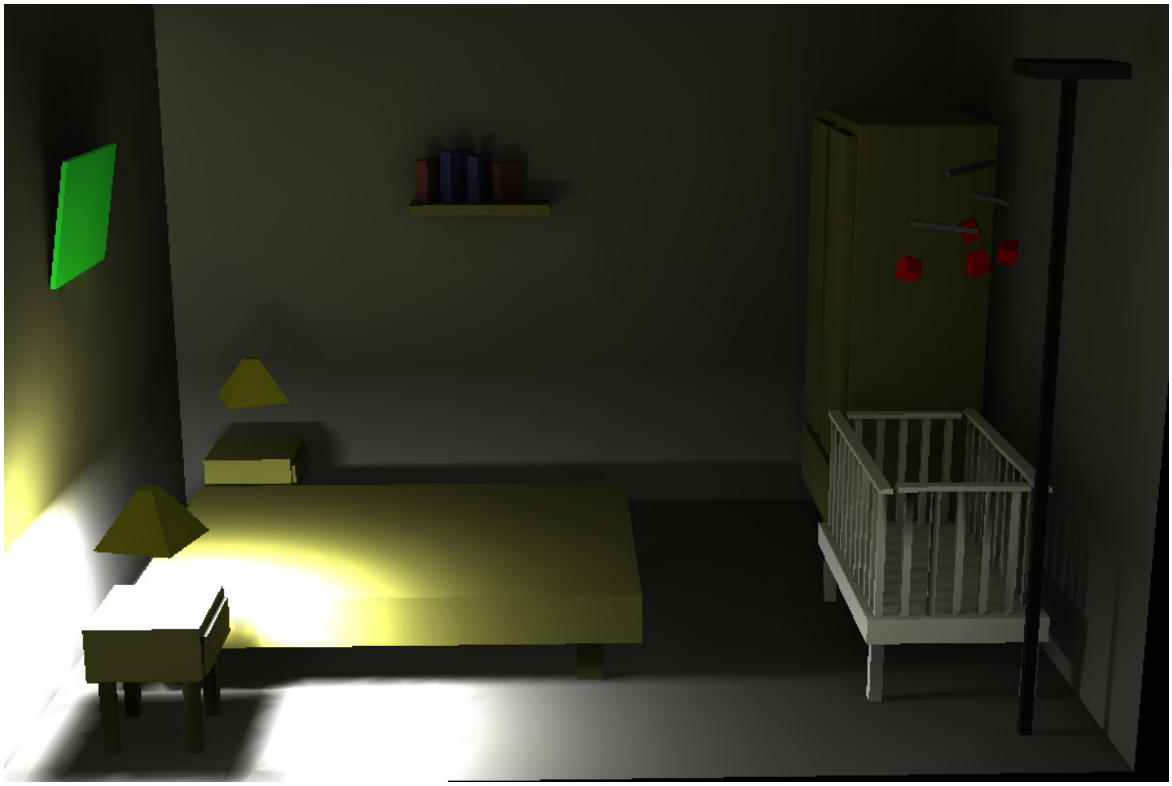
(a)



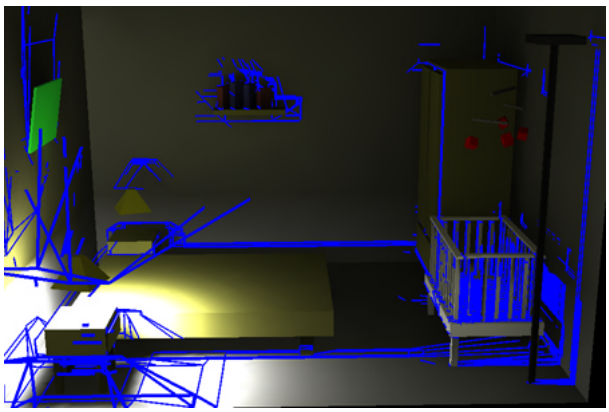
(b)



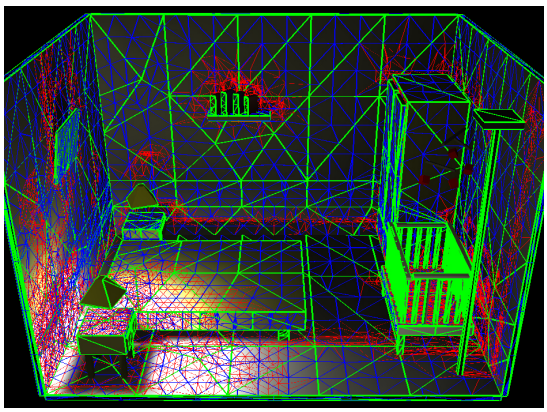
(c)



(d)

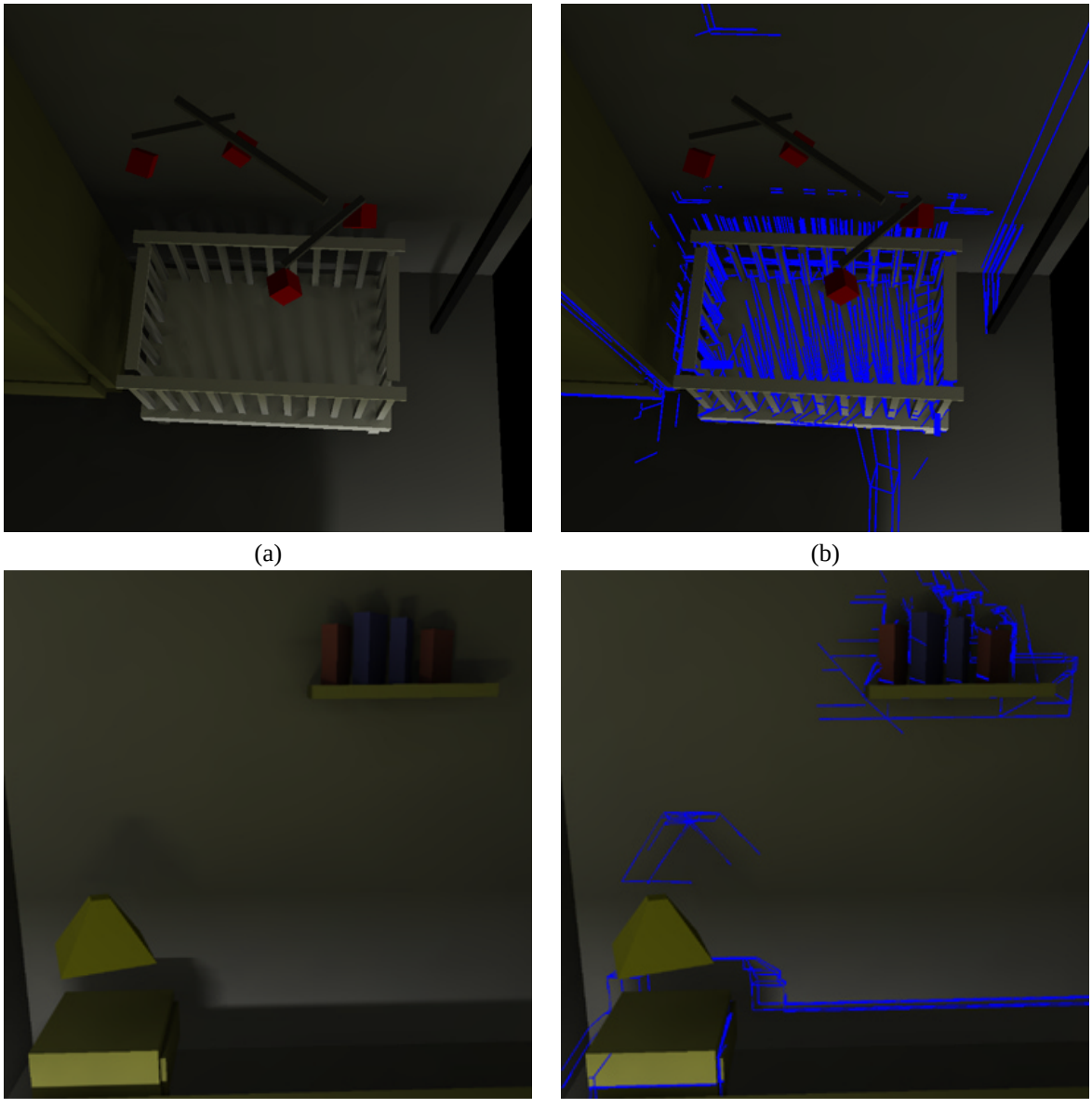


(e)



(f)

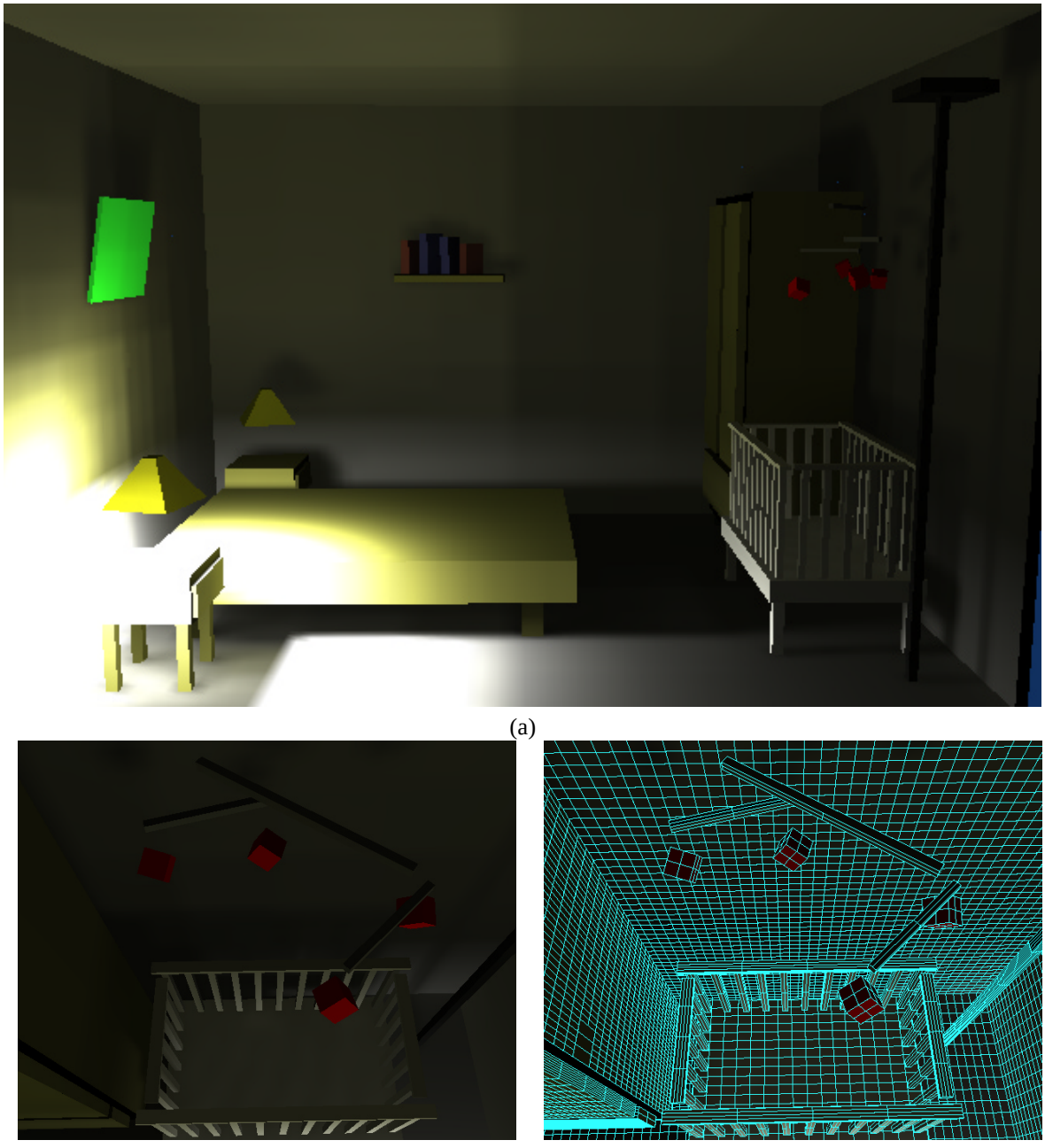
FIG. 4.24: Scène avec éclairage indirect. (a) Solution initiale. (b) Première itération. (c) Deuxième itération. (d) Image finale. (e) Discontinuités insérées (les discontinuités insérées sur le mur de devant sont affichées bien que le mur lui même ne le soit pas car sa normale tourne le dos au point de vue) (f) Triangulation hiérarchique.



(a) (b)

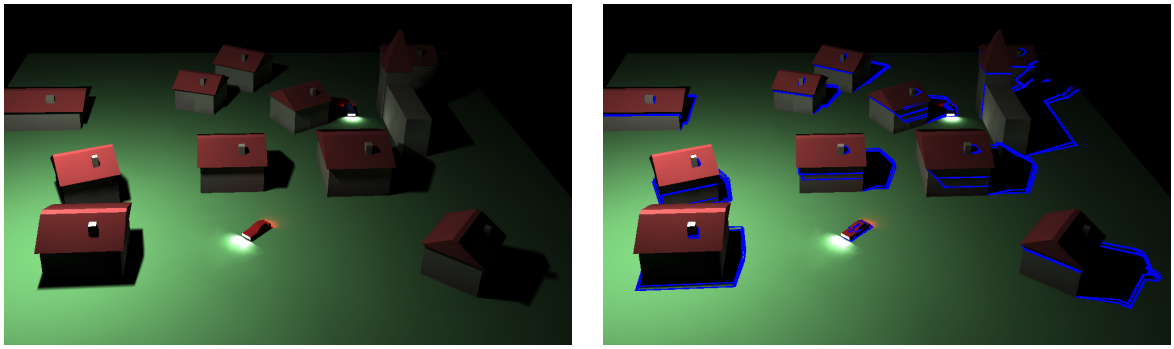
(c) (d)

FIG. 4.25: Scène avec éclairage indirect. (a) et (b) Gros plan sur le mur de droite.(c) et (d) Gros plan sur le mur du fond. La partie basse des murs est éclairée de manière directe par la lampe de gauche (invisible dans cette image), tandis que la partie supérieure est éclairée de manière indirecte par la table de gauche. On peut voir les ombres indirectes projetées par les livres et la lampe de droite.



(a) (b) (c)

FIG. 4.26: Comparaison avec la radiosité hiérarchique pour la scène avec éclairage indirect. (a) Image finale. (b) et (c) Gros plan sur le mur de droite.



(a) (b)

FIG. 4.27: Scène de village. (a) Image finale. (b) Discontinuités effectivement insérées.

7 Améliorations

Le grand coût mémoire de notre méthode provient principalement de l'information de visibilité due à la subdivision des polygones[1]. Dans cette section, nous proposons des améliorations pour diminuer ce coût. Elles n'ont pas encore été implémentées, mais nous pensons que leur importance pratique justifie leur exposition, Nous présentons tout d'abord une méthode qui économise du temps et de la mémoire pour certains raffinements. Nous proposons ensuite un compromis temps-mémoire pour tous les liens.

7.1 Optimisation pour les projections arrière invariantes

Comme nous l'avons observé dans la section 4.3, lorsqu'un raffinement de récepteur est effectué sur un lien en visibilité totale, aucune information n'a besoin d'être recalculée. Dans cette section, nous étendons ceci aux raffinements où la visibilité est qualitativement invariante.

Nous nous basons sur les maillages de discontinuité complets avec projection arrière [DF94, SG94]. Rappelons qu'un maillage de discontinuité complet subdivise les polygones de la scène en cellules où la partie de la source visible est qualitativement invariante. La projection arrière représente la structure de cette partie visible, (la chaîne d'arêtes et de sommets qui la bordent). À l'intérieur des cellules de ce maillage, la visibilité ne néccessite aucune mise à jour. Ces cellules sont bornées par les événements visuels contenus dans le squelette de visibilité, puisqu'ils représentent le lieu des changements de visibilité.

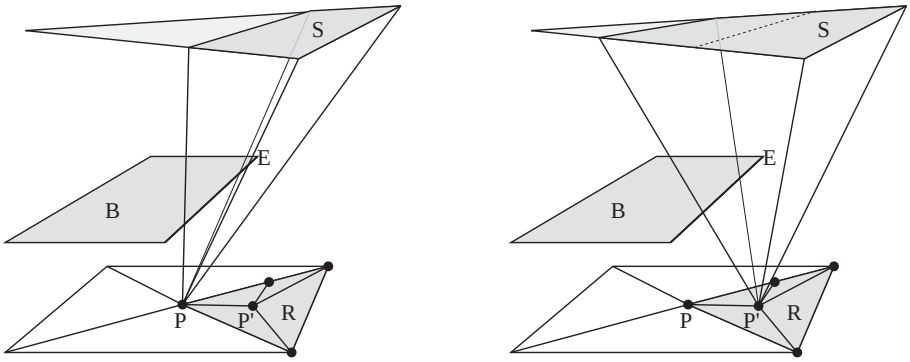


FIG. 4.28: Optimisation pour les projections arrière invariantes. Le lien entre le triangle R et la source S ne contient aucun événement visuel. La vue de S depuis P est qualitativement équivalente à celle depuis P'.

Le squelette de visibilité contient de manière implicite l'information d'un maillage de discontinuité complet. De plus, cette information est disponible pour toute paire de polygones. Considérons un lien polygone-polygone où un polygone S est la source et R est le récepteur. La partie de S visible depuis les points de R est invariante qualitativement si aucun événement visuel lié à S ne se trouve dans R ; c'est exactement l'information fournie par le lien polygone-polygone (cf. figure 4.28).

Tester si la visibilité est invariante entre deux (sous)-polygones revient donc à vérifier si le lien polygone-polygone correspondant contient un événement visuel.

Si aucun événement n'est trouvé, la visibilité est invariante et nous devons simplement calculer la valeur numérique du facteur de forme pour les nouveaux sommets issus de la subdivision. Nous utilisons l'information contenue dans les sommets du niveau supérieur (P' dans notre exemple). Nous utilisons la formule de la section 3.2 et de la figure 4.8. Les valeurs des angles des arêtes sont différentes de la configuration du sommet initiale et doivent être recalculées.

Cette méthode diminue à la fois le temps dévolu au calcul des nouveaux arcs, et la place mémoire nécessaire pour leur stockage.

[1]Notre implémentation utilise de plus une structure de maillage très coûteuse, parce que nous avons utilisé du code provenant de différentes sources, ce qui nous amène à utiliser une information de maillage très redondante.

7.2 Compromis temps-mémoire

Nous montrons maintenant comment diminuer le coût mémoire même si la visibilité de la source varie sur le récepteur. Commençons par une simple observation, illustrée figure 4.29 où un lien entre deux polygones R et S est représenté. Un premier raffinement subdivise R. L'un des triangles créés, R_i, est à nouveau subdivisé. Dans notre méthode initiale présentée section 4.2, la visibilité entre chaque R_j et S, ou depuis chaque nouveau point P est déduite de l'information de visibilité entre R_i et S. Cependant, R_j peut aussi être considéré comme un sous-élément de R, et P est aussi un point à l'intérieur de R. La visibilité peut être calculée à partir du lien polygone-polygone entre R et S, comme si le premier raffinement avait directement causé les sous-éléments les plus petits.

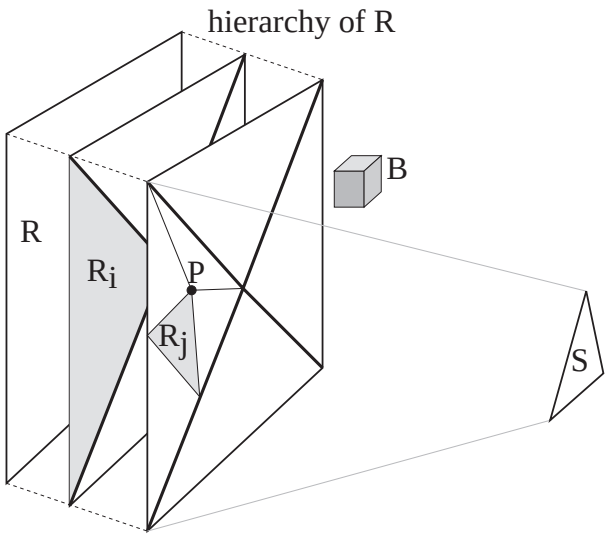


FIG. 4.29: Compromis temps-mémoire. L'information entre R_j et S peut être déduite de l'information entre R_i et S, ou de l'information entre R et S. Cette dernière solution est plus coûteuse en temps puisque le bloqueur B est inutilement considéré. Cependant, moins de mémoire est utilisée puisque l'information entre R_i et S n'a pas besoin d'être stockée.

Cette dernière méthode a le désavantage de considérer plus de bloqueurs que nécessaires, puisqu'ils sont en général plus nombreux entre R et S qu'entre R_i et S. La mise à jour de visibilité sera donc plus coûteuse. Cependant, la mémoire requise pour stocker l'information entre R_i et S est sauvée. Il s'agit là d'un compromis temps contre mémoire.

Dans les scènes que nous avons testées, nous estimons par exemple qu'il est utile de stocker l'information pour lors du premier raffinement d'un lien pour un mur ou pour le sol, puisque de nombreux bloqueurs sont présents. Lors des raffinements suivant, le nombre de bloqueurs est en revanche plus faible, et l'information du niveau supérieur de la hiérarchie pourrait être utilisée sans causer un trop grand surcoût en temps.

Nous proposons d'utiliser comme critère pour stocker ou non l'information, la différence du nombre de bloqueurs entre le lien parent et le lien enfant. Ce critère est en fait très lié à la méthode de la section précédente : si la visibilité est invariante, le nombre de bloqueurs est constant, et aucune information n'est stockée. Un budget mémoire peut aussi être allouée : une fois qu'il a complètement été utilisé, on arrête de stocker l'information.

La question des critères de raffinement présenté section 5.4 est alors plus délicate, puisque l'information de visibilité n'est alors plus disponible dans son intégralité. Les événements visuels sont toujours stockés, mais à un niveau supérieur de la hiérarchie des liens, il n'existe aucune garantie qu'ils affectent le sous-polygone courant. Le critère peut alors être calculé pour tous les événements visuels stockés au niveau supérieur, et si l'un d'entre eux indique qu'un raffinement est nécessaire, on vérifie qu'il affecte effectivement le sous-triangle en question.

Nous pensons que ces deux optimisations abaisseront grandement la consommation mémoire de notre méthode. La première solution diminuera aussi le temps de calcul et nous croyons que la seconde n'imposera pas une trop grande pénalité.

8 Discussion

8.1 Résumé

Nous avons présenté un nouvel algorithme de radiosité hiérarchique qui utilise le squelette de visibilité, Nous avons introduit un algorithme de mise à jour des vues qui permet d'obtenir les vues depuis les sommets dus à une subdivision.

Le squelette de visibilité permet le calcul de facteurs de forme exacts point-polygones pour toute paire sommet/polygone de la scène. Il fournit également une information de visibilité détaillée entre toute paire (sous)polygone-(sous)polygone.

Nous avons introduit un nouvel algorithme hiérarchique basé sur une représentation multirésolution sous forme d'ondelettes paresseuses. Elles s'appuient sur des triangulations hiérarchiques, qui consistent en une hiérarchie de triangulations de Delaunay contraintes. En utilisant des différences de radiosité aux sommets issus de subdivisions, nous avons introduit une opération de "pousser" linéaire, qui produit une reconstruction de la radiosité de meilleure qualité aux feuilles. Un nouveau critère de raffinement perceptif guidé par la visibilité autorise une subdivision des surfaces qui s'adapte mieux aux variations d'ombre. Les résultats montrent que notre approche produit des solutions indépendantes du point de vue de haute qualité de manière efficace. Ils montrent également que notre méthode traite bien les configurations traditionnellement difficiles comme les scènes comportant de nombreuses sources ou bien éclairées principalement de manière indirecte.

Le coût mémoire est la principale limitation de notre approche, mais des solutions proposant un compromis temps/mémoire ont été présentées, basées sur la détection de configurations où la visibilité ne change pas du tout ou très peu.

8.2 Limitations

Nous pouvons identifier deux limitations majeures à notre algorithme : sa grande consommation mémoire (même avec les améliorations) et les problèmes de robustesse.

L'utilisation mémoire du squelette de visibilité est grande et présente en général une croissance quadratique en fonction du nombre de polygones en entrée. Même pour des scènes très simples, notre méthode utilise une grande quantité de mémoire (cf. table 4.2). Pour que cette approche soit utilisable pour de grandes scènes, il est évident qu'une des stratégies suivantes doit être adoptée : construction paresse ou méthode par regroupement avec représentation multirésolution de la visibilité. Des idées dans ce sens peuvent être trouvées à la section 8.4.

La robustesse numérique et le traitement des cas dégénérés sont des problèmes épineux. Comme nous l'avons dit au chapitre précédent, malgré la simplicité de notre algorithme, les configurations dégénérées peuvent causer des erreurs dans la construction du squelette. De plus, à cause de la subdivision, de nombreux événements visuels coïncident, ce qui crée des problèmes pour l'étape de lancer de rayon (pour la création des nœuds) et pour la détermination des adjacences. Ces problèmes sont particulièrement évidents dans le cas de la mise à jour de vues, dont la mise au point a été fastidieuse. Un traitement cohérent des dégénérescences est prévu, mais il constitue un sujet de recherche en lui-même, et dépasse largement le cadre de cette thèse. L'insertion des points dans le maillage est elle aussi loin d'être simple, à cause des problèmes d'imprécision numérique.

8.3 Avantages

L'algorithme de radiosité hiérarchique guidé par la visibilité que nous avons présentś possède plusieurs avantages. Pour commencer, il produit des ombres de qualité et utilise des facteurs de forme exacts pour l'éclairage direct comme indirect. Notre nouvelle représentation multirésolution et les deux types de liens permettent une reconstruction linéaire continue de la radiosité sur des maillages irréguliers. Le traitement global de la visibilité et des discontinuités autorise des oracles de raffinements pertinents et efficaces. Grâce à une estimation perceptive de l'importance des ombres, notre algorithme de raffinement s'est révélé très efficace dans des configurations traditionnellement difficiles, comme les scènes comportant de nombreuses sources ou qui possède un éclairage indirect prédominant. Nos comparaisons informelles ont montré un net avantage pour notre méthode, que ce soit en terme de qualité ou de temps de calcul.

Des approches comme celle de Lischinski *et al.* [LTG93] qui utilisent le maillage de discontinuité sont limitées à un petit nombre de sources, puisque le nombre de discontinuité devient très vite ingérable. Cela a des conséquences néfastes sur le temps de calcul et sur la robustesse de la méthode. Pour des raisons similaires, aucun algorithme de simulation hiérarchique ne traite les discontinuités dues à l'éclairage indirect.

8.4 Travaux futurs

Passage à l'échelle

Les améliorations que nous avons proposées pour décroître le coût mémoire doivent être implémentées et testées.

L'avantage de notre algorithme de construction du squelette est sa localité, qui le rend particulièrement bien adapté à la construction paresseuse ou à la demande. En utilisant des critères à définir, il est possible de ne construire que la partie du squelette liée aux transferts lumineux "importants". Cette information pourrait être ensuite effacée pour réduire le coût mémoire.

L'approche hiérarchique discutée dans les travaux antérieurs du chapitre précédent pourrait être appliquée directement à la simulation de l'éclairage. Le cadre de la radiosité hiérarchique avec regroupement [SAG94, Sil95] est l'extension naturelle de la radiosité hiérarchique. Il pourrait bénéficier d'une structure de donnée multirésolution et précise.

Contrôle de l'erreur

L'information contenue dans le squelette de visibilité permet également le calcul du gradient ou des dérivées supérieures de la fonction d'éclairement [Arv94, HS95, HS99]. Une telle information pourrait se révéler utile pour développer de nouveaux critères de raffinements, ou des interpolants de degré supérieur pour la fonction de radiosité. Les techniques de contrôle de l'erreur pour la radiosité [ATS94, LSG94, HS99] font en général l'impasse sur la question de la visibilité, jugée trop difficile à gérer. Le squelette de visibilité contient l'information nécessaire, ce qui rend possible le contrôle précis de la simulation.

Un contrôle total de l'erreur est cependant rarement nécessaire. Toutefois, ce genre d'approche peut être simplifié à l'aide d'approximations pour obtenir des critères plus globaux. Les oracles que nous avons implémentés n'estiment l'erreur que localement pour chaque interaction. L'influence est pourtant plus globale, puisque l'énergie lumineuse (et donc l'erreur) est propagée à travers toute la scène.

Maillage

La qualité du maillage est cruciale pour les simulations par éléments finis. Les triangles fins en particulier doivent être évités. Les triangulations de Delaunay, surtout quand elles sont contraintes, ne garantissent pas une telle propriété. Des techniques classiques de maillage (cf., *e.g.*, [She96]) pourraient être utilisées pour optimiser nos maillages, par exemple en insérant des sommets sur les arêtes contraintes qui ne respectent pas la contrainte de Delaunay.

De plus, une triangulation de Delaunay ne fournit pas toujours la meilleure interpolation d'une fonction. Considérons l'exemple de la figure 4.30 qui représente l'interpolation d'une fonction sur un carré échantillonné à ses quatre coins. Deux triangulations sont possibles, en fonction de la diagonale retenue. Le critère ne devrait pas être basé sur une contrainte de Delaunay (les deux diagonales sont équivalentes), mais devrait prendre en compte le gradient de la fonction [ARB90, Sch93].

Critère perceptif

Les critères que nous avons développés peuvent également être adaptés à la radiosité hiérarchique classique, sans l'information du squelette de visibilité. En particulier, la séparation en trois phase (radiométrique, estimation de la quantité d'ombre, estimation de la variation d'ombre) est une approche prometteuse pour des raffinements efficaces. L'utilisation d'une métrique perceptive permet de plus éviter le réglage de seuils imprévisibles et arbitraires.

Des métriques plus sophistiquées pourraient être utilisées [Mys98, BM98, RPG99, FPSG97, PFFG98]. En particulier, la composante fréquentielle spatiale devrait être prise en compte, puisque l'œil humain est moins sensible à l'erreur dans les rgions où l'image varie beaucoup, comme sur le motif complexe d'un tapis.

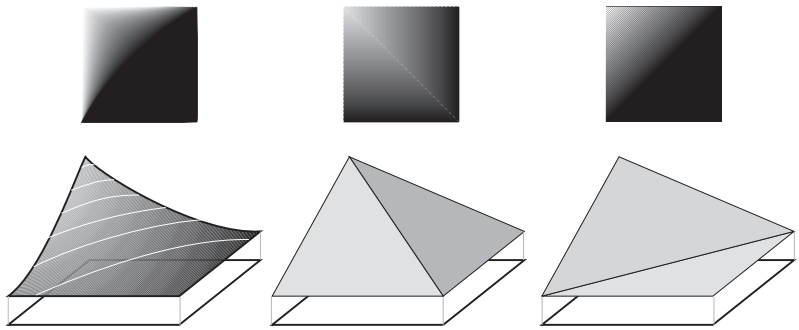


FIG. 4.30: Optimisation d'une triangulation pour représenter une fonction (a) Fonction exacte. (b) Mauvaise triangulation. (c) Bonne triangulation.

Nous pensons que l'utilisation d'une métrique perceptive est une technique puissante pour optimiser les calculs en synthèse d'images. Les méthodes d'animation pourraient également bénéficier de ces critères pour allouer plus de ressources dans les parties de la scène ou du mouvement les plus visibles par l'observateur.

Autres questions

Le squelette de visibilité pourrait également être utilisé pour des méthodes stochastiques. Pour les méthodes classiques de Monte-Carlo, la nature purement aléatoire de l'échantillonnage rend difficile la prise en compte d'une éventuelle cohérence. Cependant, une approche plus récente comme l'échantillonnage de Metropolis [VG97] pourrait tirer partie du squelette de visibilité pour une meilleure exploration de l'espace des chemins lumineux et pour trouver de bons chemins initiaux.

L'extension du squelette de visibilité aux scènes en mouvement appelle naturellement son utilisation pour la mise à jour de l'éclairage.

CHAPITRE 5

Calculs généraux d'occultations à l'aide de projections étendues

> La perspective n'est rien d'autre que la vision d'une scène derrière une vitre plane et bien transparente, sur laquelle on marque tous les objets qui sont de l'autre côté de cette vitre.
>
> Léonard DE VINCI, *Codex Urbinas*

ISUALISER des environnements complexes est l'un des buts majeurs des systèmes de rendu interactifs. Malgré les progrès impressionnants de la vitesse du matériel graphique, il est toujours difficile, voire impossible, de visualiser des modèles très complexes (des millions de polygones) en temps réel, même sur des machines très haut de gamme. Nous avons identifié trois points qui empêchent l'affichage de très grosses scènes : (i) la vitesse limitée des cartes graphiques, *i.e.* le nombre de polygones affichables ; (ii) la difficulté de développer des algorithmes valables pour différentes classes de modèles (*e.g.* architecturales, avec des objets concaves, etc.) ; (iii) la taille mémoire requise par la base de données graphique, qui empêche parfois son chargement en mémoire centrale, ou sa transmission complète *via* réseau.

La nouvelle approche que nous proposons est un *précalcul* pour les scènes complexes qui décompose l'espace où l'observateur se déplace en cellules. Nous introduisons un nouvel algorithme qui utilise des *projections étendues* pour calculer une *carte de profondeur étendue* par rapport à tous les points de vue d'une cellule. Cette nouvelle approche traite l'effet combiné de plusieurs obstacles (la *fusion de bloqueurs* ou *occluder fusion*) et peut traiter des objets concaves. Les cartes de profondeur étendues sont utilisées pour un calcul prudent[1] de la géométrie potentiellement visible depuis la cellule. La construction des cellules est adaptative ; une subdivision récursive et des tests hiérarchiques sont utilisés pour déterminer les objets cachés ; le résultat est ensuite utilisé par un programme interactif de visualisation de scènes complexes. Notre algorithme permet de s'affranchir des limitations mentionnées ci-dessus, et d'afficher interactivement des scènes très complexes et génériques. Les applications potentielles de notre technique comprennent les jeux vidéo et le rendu à travers le réseau, pour

[1] Dans ce chapitre, nous utilisons le terme *prudent* (*conservative*), bien qu'une discrétisation soit ensuite employée qui limite la précision des calculs.

lesquels la diminution de la taille de la géométrie envoyée au matériel graphique est primordiale.

Ce chapitre commence par une présentation générale de notre méthode. Nous présentons ensuite dans la section 2 des méthodes pour calculer les projections étendues, avant de proposer dans la section 3 une amélioration importante pour certains types de bloqueurs. Puis nous introduisons dans la section 4 un opérateur de reprojection et un balayage d'occultation. La section 5 présente le précalcul adaptatif, tandis que la section 6 décrit notre implémentation et les résultats obtenus. Pour finir, la section 7 comporte une discussion et des propositions de travaux futurs.

1 Projections étendues

Nous présentons une méthode de précalcul qui subdivise l'espace où l'observateur se déplace en cellules de vue. Pour chaque cellule de vue, nous calculons un ensemble d'objets qui sont potentiellement visibles depuis les points à l'intérieur de la cellule (le *PVS, potentially visible set*). Pour cela, nous effectuons un test prudent d'occultation pour chaque objet de la scène. Notre test d'occultation utilise une représentation des occultations dues aux bloqueurs qui est fondée sur des *projections étendues* sur un plan.

1.1 Principe

Notre méthode peut être vue comme une extension aux volumes des méthodes de calcul d'occultation par rapport à un point. Dans ces méthodes qui travaillent dans l'espace image [GKM93, ZMHH97], les bloqueurs (*occluders*) et les objets (*occludees*) sont projetés sur le plan de l'image. L'occultation est détectée en testant si la projection de l'objet es contenue dans la projection des bloqueurs (test de recouvrement ou *overlap test*) et si l'objet est derrière les bloqueurs (test de profondeur ou *depth test*). Dans certaines méthodes [GKM93], les deux tests sont effectués en une seule étape en utilisant une carte de profondeur.

Nous étendons les méthodes pour un point de vue unique à des cellules de vue volumiques. Dans la section suivante, des *opérateurs de projection étendue* sont définis pour les objets et pour les bloqueurs. Un objet est caché depuis tous les points de vue de la cellule si : (i) la projection étendue de l'objet est contenue dans la projection étendue des bloqueurs et (ii) si l'objet est derrière les bloqueurs. Les opérateurs de projection étendue sont différents pour les bloqueurs et pour les objets.

Nos opérateurs de projection étendue consistent à sous-estimer la projection pour les bloqueurs et à la sur-estimer pour les objets. Pour les bloqueurs, la projection étendue par rapport à la cellule est une sous-estimation de la projection depuis chaque point de vue.

Bien que nous ne décrivions notre méthode que pour un seul plan de projection, six sont en fait nécessaires pour détecter les occultations dans toutes les directions. La question importante du choix des plans de projection ne sera pas traitée avant la section 5.2.

Nous définissons une vue comme la projection perspective depuis un point sur un plan de projection. Cependant, dans ce qui suit, le plan de projection sera partagé par tous les points de vue d'une cellule, ce qui entraîne des perspectives "penchées", comme lorsque l'on se déplace dans une pièce en décrivant ce qui est vu à travers une fenêtre fixe. La visibilité est cependant équivalente au cas où un plan de projection perpendiculaire à l'axe optique est utilisé, et ces projections peuvent être traitées à l'aide de matrices classiques de projection.

1.2 Projections étendues

Définition 1 *Nous définissons la projection étendue (ou* Projection*) d'un* bloqueur *sur un plan par rapport à une cellule comme l'*intersection *de ses vues depuis chaque point de la cellule.*

Définition 2 *La projection étendue (ou* Projection*) d'un* objet *est définie comme l'*union *de ses vues depuis tous les points de la scène.*

La figure 5.1 illustre le principe de nos projections étendues.

Dans ce qui suit nous utiliserons simplement le terme *Projection* pour parler de projections étendues. La projection standard depuis un point sera appelée *vue*.

Cette définition des Projections permet des test d'occultation prudents. Pour le prouver, considérons tout d'abord le cas d'un seul bloqueur. Supposons que l'objet est derrière le bloqueur. Il est déclaré caché si sa Projection est contenue dans la Projection du bloqueur. Cela signifie que l'union de ses vues est contenue dans

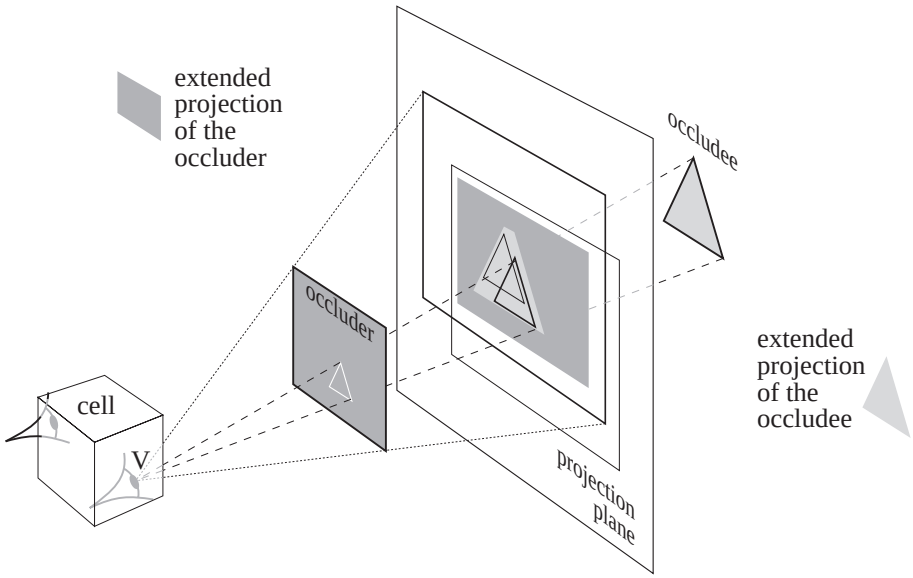


FIG. 5.1: Projections étendues d'un bloqueur et d'un objet. La vue depuis le point V est représentée en gras. Les vues depuis un autre point de vue sont représentées en traits fins. La projection étendue d'un bloqueur sur un plan est l'*intersection* de ses vues. Pour un objet, il s'agit de l'*union* de ses vues.

l'intersection des vues du bloqueur. Depuis tout point V de la cellule, la vue de l'objet est contenue dans la vue du bloqueur. Pour résumer, on a :

$$vue_V^{objet} \subset \bigcup_{W \in cellule} vue_W^{objet} \subset \bigcap_{W \in cellule} vue_W^{bloqueur} \subset vue_V^{bloqueur}$$

L'objet est donc caché, si le test de profondeur est satisfait (cf. figure 5.1 pour une illustration).

Considérons maintenant le cas d'un objet dont la Projection est contenue dans la Projection combinée de plusieurs bloqueurs. Cela signifie que depuis tout point de vue V à l'intérieur de la cellule, la vue de l'objet est contenue dans la vue combinée des bloqueurs. On a :

$$vue_V^{objet} \subset \bigcup_{W \in cellule} vue_W^{objet} \subset \bigcup_{bloqueurs} \bigcap_{W \in cellule} vue_W^{bloqueur} \subset \bigcup_{bloqueurs} vue_V^{bloqueur}$$

L'objet est donc là aussi caché (figure 5.2). Nos opérateurs de projection étendue traitent la *fusion de bloqueurs*, c'est-à-dire l'occultation due à la conjonction de plusieurs bloqueurs.

Malheureusement, il n'y a pas de correspondance directe entre les points de la Projection et les points de l'objet, contrairement à une vue classique. Considérons la situation de la figure 5.3(a). En fonction du point de vue V, un point P de la Projection du bloqueur correspond à la vue de différents points du bloqueur.

Nos définitions ne sont pas limitées aux objets convexes, tout comme dans le cas des calculs d'occultations par rapport à un point. Cependant, nous verrons qu'il est plus simple de calculer les Projections dans le cas convexe et, dans la section 3, nous montrons que l'efficacité des Projections peut être accrue si les bloqueurs sont convexes ou planaires.

On peut donner une autre interprétation de ces opérateurs dans le cas où le plan de projection est derrière les objets et les bloqueurs. La Projection d'un bloqueur est alors son *ombre* si la cellule est considérée comme une source de lumière. Un point P dans la Projection d'un bloqueur A est dans la vue de A depuis n'importe quel point de vue V de la cellule (cf. figure 5.3(a)). Donc pour tout point V dans la cellule, le rayon quittant P en direction de V coupe le bloqueur A. P est dans l'ombre de A sur le plan de projection. De même, la projection d'un objet correspond à sa *pénombre*.

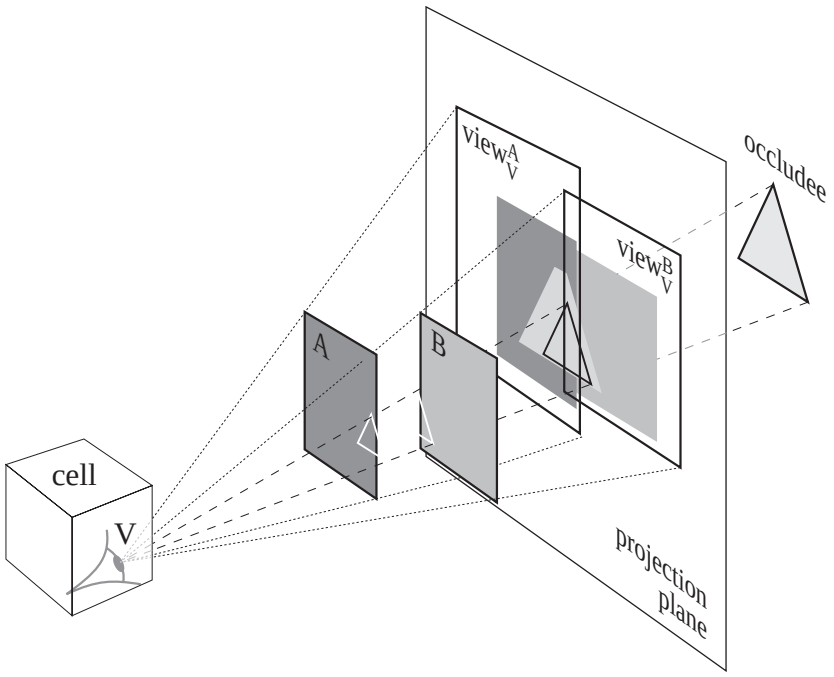


FIG. 5.2: Les projections étendues prennent en compte la *fusion des bloqueurs* pour deux bloqueurs A et B. Nous montrons un exemple de vue depuis le point V. La vue de l'objet est contenue dans la vue cumulée des deux bloqueurs, comme le prédit le test d'occultation utilisant les projections étendues. On remarquera qu'un seul des deux bloqueurs ne peut cacher l'objet.

1.3 Profondeur

Le test de recouvrement des Projections ne suffit pas pour conclure qu'un objet est caché ; un test de profondeur doit être effectué. Il est cependant plus complexe que dans le cas où un seul point de vue intervient, puisque la distance entre le point de vue et un point projeté ne peut être définie simplement. Ceci parce qu'un ensemble volumique de points de vue est considéré, et aussi parce que les points de la Projection ne correspondent pas à un point projeté unique, comme nous l'avons vu figure 5.3(a).

Notre définition de la profondeur doit être cohérente avec les propriétés d'occultation : pour tout rayon quittant un point de la cellule en direction du plan de projection, la profondeur doit être une fonction croissante de la distance au point de vue. La définition la plus immédiate de la profondeur est selon la direction orthogonale au plan de projection. Nous choisissons le sens positif qui s'éloigne de la cellule, et plaçons le zéro sur le plan de projection. Les objets entre le plan de projection et la cellule ont donc une profondeur négative. Les bloqueurs ont une profondeur plus faible que celle des objets qu'ils cachent.

Nous avons défini une fonction de profondeur cohérente pour tous les points de vue de la cellule. Considérons un point P dans la Projection d'un bloqueur (figure 5.3(b)). Il correspond à plusieurs points sur le bloqueur (par exemple A_1 et A_2), qui appartiennent à l'intersection de toutes les droites passant par la cellule et ce point.

Définition 3 *Nous définissons la profondeur étendue (ou* Profondeur*) d'un point dans la Projection d'un bloqueur comme le* maximum *des profondeurs des points projetés correspondants.*
*De même, nous définissons la profondeur étendue (*Profondeur*) d'un point dans la Projection d'un objet comme le* minimum *des profondeurs des points projetés correspondants.*

Dans ce qui suit, nous appellerons cette quantité simplement *Profondeur*. Si la Profondeur d'un point dans la Projection d'un bloqueur est plus petite que la Profondeur dans la Projection d'un objet, tous les points correspondants de l'objet sont derrière le bloqueur par rapport à tout point de la cellule. Notre définition permet donc des tests prudents.

Considérons maintenant le cas de plusieurs bloqueurs. Nous voulons considérer l'occultation causée par le bloqueur le plus proche de la cellule. Pour effectuer le test de profondeur, nous considérons donc le bloqueur qui a la Profondeur *minimale*. Rappelons que la Profondeur d'un point P est la profondeur *maximale* des

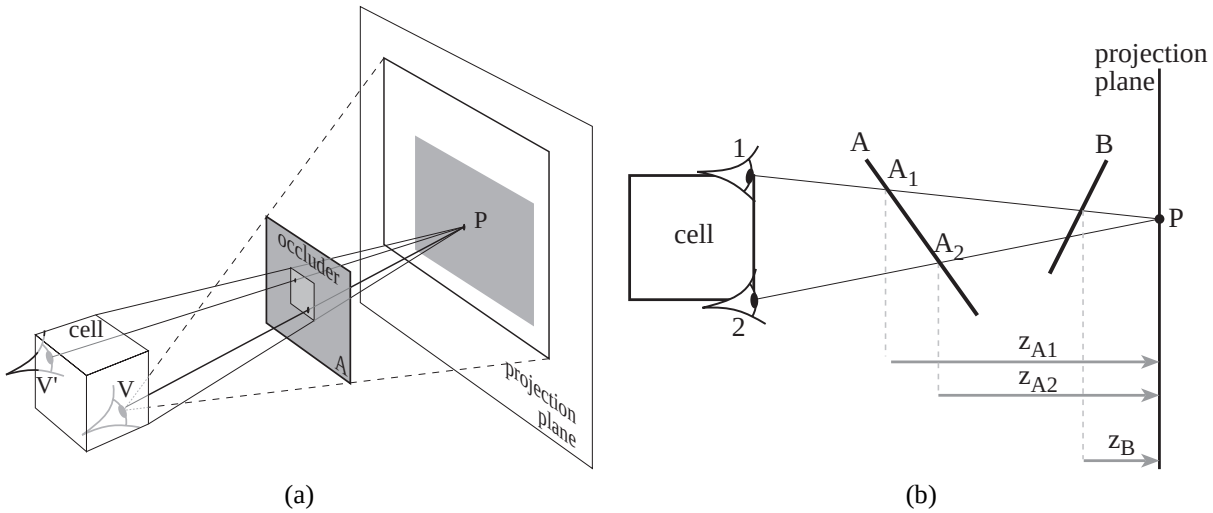


FIG. 5.3: (a) Inverse d'un point *P* dans une Projection. Nous montrons le point du bloqueur qui correspond à un point *P* dans la vue depuis *V*. Pour le deuxième point de vue *V′*, *P* correspond à la vue d'un autre point du bloqueur. La région claire du bloqueur correspond à l'ensemble des points qui se projettent en *P*. (b) Nous considérons la profondeur le long de la direction perpendiculaire au plan de projection (dans cette figure, toutes les profondeurs sont négatives car le zéro est choisi sur le plan de projection). La Profondeur d'un point *P* dans la Projection d'un bloqueur *A* est le maximum de la profondeur des points projetés correspondants (z_{A2} ici). Dans le cas de bloqueurs multiples, le bloqueur le plus proche de la cellule est considéré (*A* ici).

points projetés correspondants. La Profondeur d'un point dans la Projection de plusieurs bloqueurs est donc le *minimum des maxima* des profondeurs des points projetés.

Pour traiter ces profondeurs, nous définissons la carte de profondeur étendue (*Carte de Profondeur*). Pour chaque point *P* du plan de projection, la valeur de la Carte de Profondeur est la Profondeur du bloqueur le plus proche de la cellule qui se projette sur *P*. Si aucun bloqueur ne s'y projette, une valeur "infinie" est utilisée.

Une version simplifiée du test de profondeur peut être utilisée. Si les bloqueurs sont tous devant le plan de projection et les objets tous derrière, alors le test de recouvrement suffit à déterminer l'occultation. Ceci se révélera utile pour la reprojection des bloqueurs. Nous n'utiliserons pas cette possibilité avant la section 4. D'ici là, nous considérons l'utilisation d'une valeur de Profondeur pour chaque point dans la Projection d'un bloqueur.

1.4 Aperçu de la méthode

Nous présentons maintenant notre méthode. Nous commençons par calculer les Projections des bloqueurs sur un plan de projection. Ensuite, pour chaque objet, nous testons si sa Projection recouvre celles des bloqueurs et s'il se trouve derrière eux, c'est-à-dire s'il a une plus grande Profondeur. Pour chaque point à l'intérieur de la Projection d'un objet, nous devons tester si sa Profondeur est supérieure à celle de Projection d'un bloqueur.

De manière plus algorithmique, notre méthode peut être décrite comme suit : pour chaque point de la Projection de l'objet, tester si sa Profondeur est supérieure à la valeur correspondante de la Carte de Profondeur.

Si l'on utilise le test de profondeur simplifié (si le plan de projection est entre les bloqueurs et les objets), on utilise une carte d'occultation étendue (*Carte d'Occultation*). On associe à chaque point du plan de projection une valeur booléenne qui est vraie si et seulement si ce point appartient à la Projection d'au moins un bloqueur.

1.5 Choix d'implémentation

Jusqu'ici, nos définitions ont été générales et indépendantes de la cellule, du plan, des objets et de la façon dont les tests sont effectués en pratique. Nous présentons maintenant les choix que nous avons faits pour notre implémentation.

Les cellules de vue sont des boîtes englobantes non-alignées aux axes. Les plans de projection seront restreints aux trois directions des faces de la cellule (qui peuvent être différentes pour chaque cellule). Les objets sont organisés en une hiérarchie de boîtes englobantes alignées aux axes.

Nous utiliserons une représentation discrète (*bitmap*) des Cartes de Profondeur. Il s'agit là de notre seule concession à la prudence (on peut retirer cette concession, comme nous le verrons plus tard). Cela permet d'utiliser le matériel graphique, ce qui simplifie la plupart des calculs et évite les problèmes de robustesse inhérents à tout calcul géométrique.

Nous stockons une valeur de Profondeur pour chaque pixel. Comme nous l'avons expliqué, nous utilisons le minimum des Profondeurs des bloqueurs pour ce pixel. La fusion des bloqueurs est traitée grâce à l'agrégation naturelle dans la Carte de Profondeur. Tout comme Greene [GKM93], nous organisons la Carte de Profondeur en une pyramide pour des tests plus efficaces ; nous l'appelons la Carte de Profondeur Hiérarchique. De même, nous utiliserons des Cartes d'Occultation Hiérarchiques dans l'esprit de Zhang *et al.* [ZMHH97].

2 Calcul des projections étendues

Nous avons défini des *opérateurs de projection étendue* (Projection) ainsi qu'une *profondeur étendue* (Profondeur) qui permettent des tests prudents d'occultation sur un plan de projection. Ces tests prennent en compte la *fusion de bloqueurs*. Nous représentons les occultations sur le plan de projection à l'aide de *Cartes Hiérarchiques de Profondeur*. Nous décrivons maintenant comment effectivement calculer les Projections des bloqueurs et des objets.

Nous présentons tout d'abord une méthode approximative mais prudente pour la Projection des objets. Nous donnons ensuite une méthode pour la Projection des bloqueurs convexes, puis une pour les bloqueurs concaves.

2.1 Projection des objets

Rappelons que la Projection d'un objet est l'union de ses vues. Nos cellules de vue sont convexes, tout comme les boîtes englobantes des objets. La Projection de ces boîtes se ramène donc par convexité à l'enveloppe convexe des vues depuis les sommets de la cellule.

Ce calcul peut être implémenté en utilisant un algorithme géométrique classique, en calculant les vues, leur enveloppe convexe et en la discrétisant dans la Carte de Profondeur. Nous utilisons toutefois une approximation prudente qui simplifie le calcul et la phase de discrétisation.

Nous utilisons le rectangle englobant la Projection dans le plan de projection, ainsi que le montre la figure 5.4. Notre but est de sur-estimer la Projection de l'objet. Pour ceci, nous séparons le problème en deux problèmes 2D plus simples. Nous projetons sur deux plans orthogonaux au plan de projection et parallèles aux faces de la cellule. La projection 2D de la cellule est un rectangle, tandis que la projection 2D de la boîte englobante de l'objet est un hexagone en général (la figure 5.4 représente un cas spécial pour plus de clarté).

Nous calculons ensuite les droites supports et séparatrices du rectangle et de l'hexagone. L'intersection de ces droites avec le plan de projection définit un segment 2D englobant. Un rectangle englobant est défini par le produit cartésien des deux segments (cf. figure 5.4). Nous utilisons ce rectangle englobant comme une sur-estimation de la Projection de l'objet. Les droites séparatrices sont utilisées si l'objet se trouve entre le plan de projection et la cellule, tandis que l'on considère les droites supports lorsqu'il se trouve derrière le plan.

Cette méthode de calcul des Projections des objets est générale et toujours valide, mais peut parfois être trop pessimiste. Dans la section 3, nous en présenterons une amélioration pour certaines configurations.

2.2 Projection de bloqueurs convexes à l'aide d'intersections

La Projection de bloqueurs convexes possède des propriétés semblables à celle des boîtes englobantes des objets. Par convexité de la cellule et du bloqueur, l'intersection de toutes les vues depuis les points de la cellule se ramène à l'intersection des vues depuis les sommets de la cellule (cf. figure 5.5(a)). Cette Projection peut être calculée à l'aide d'algorithmes géométriques classiques.

Nous avons cependant développé une méthode efficace qui est optimisée pour le matériel graphique. Il s'agit d'une méthode *multi-passes* qui utilise le tampon de stencil (*stencil buffer*). Le tampon de stencil peut

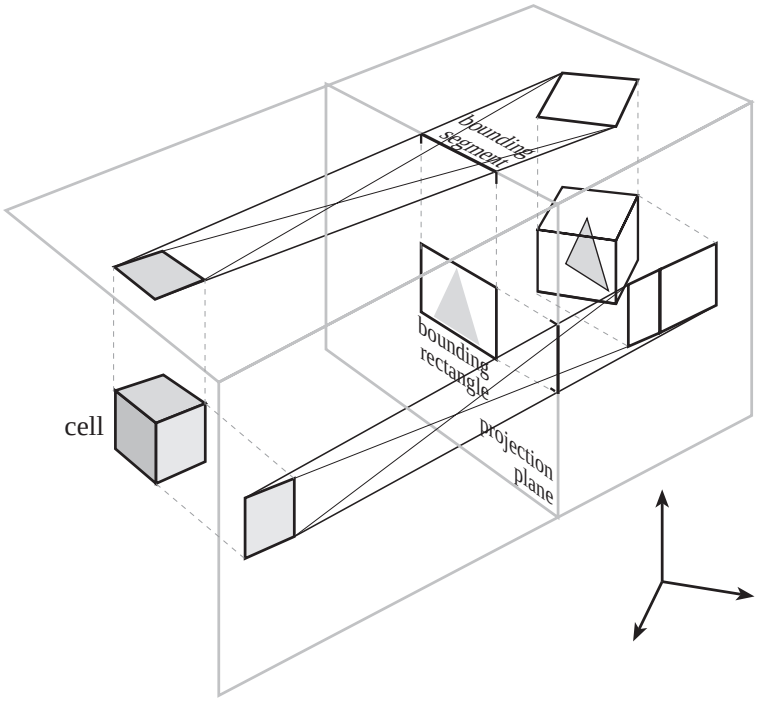


FIG. 5.4: La projection étendue d'un objet se réduit à deux problèmes 2D.

être écrit et comparé à une valeur de test pour un rendu conditionnel. Il a été introduit pour interdire le rendu sur les pixels représentant le cockpit d'un avion par exemple [MBGN98].

Le principe de notre méthode est de projeter les bloqueurs depuis chaque sommet sans écrire dans le tampon d'affichage mais en incrémentant le tampon de stencil des pixels projetés. Les pixels de l'intersection des vues sont donc ceux qui ont pour valeur de stencil le nombre de sommets de la cellule (cf. figure 5.5(b)).

Le traitement cohérent des valeurs de Profondeur (comme défini section 1.3) nécessite une attention particulière avec cette approche et est décrit dans l'annexe C.

2.3 Tranchage des bloqueurs concaves

Les maillages polygonaux concaves peuvent être traités avec la méthode précédente, en projetant chaque triangle. Cependant, des interstices apparaîtront alors entre les Projections ; l'information de connexité du bloqueur est perdue. Nous présentons maintenant une première méthode pour traiter les bloqueurs concaves en tant qu'entités. D'autres approches seront esquissées à la section 7.3.

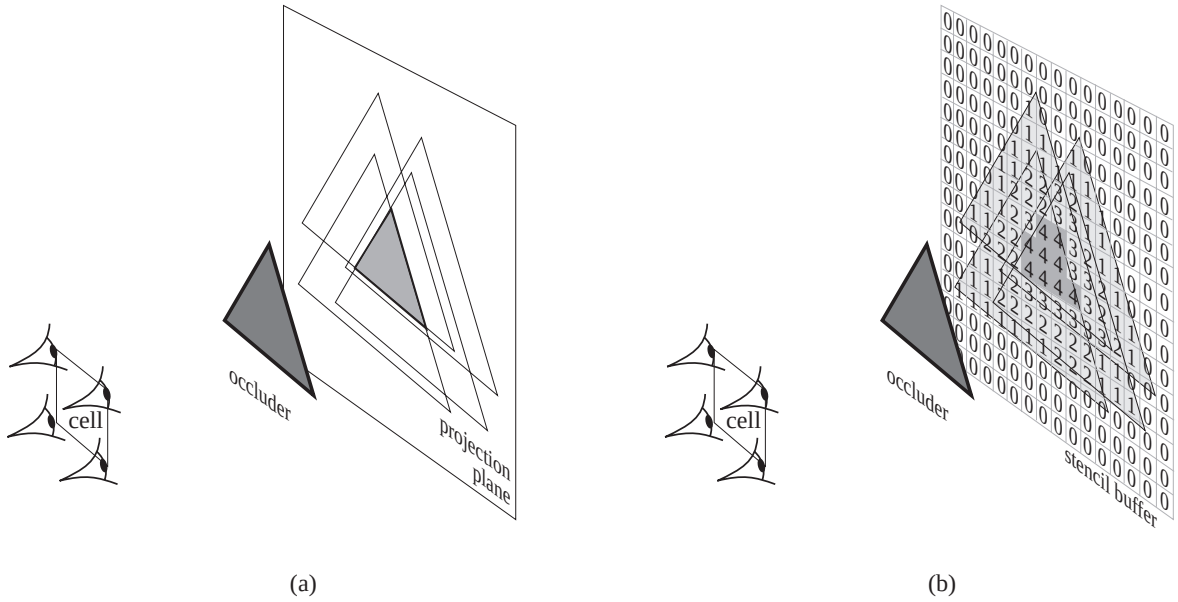
Notre méthode est fondée sur cette simple observation : la Projection d'un bloqueur qui se trouve dans le plan de projection est le bloqueur lui-même. Nous utilisons donc l'intersection des bloqueurs concaves avec le plan de projection, ce que nous appelons une *tranche* du bloqueur.

Trancher les bloqueurs concaves est simple : le bloqueur est intersecté avec le plan, et un contour 2D est trouvé. Il est important que ce contour soit fermé. Nous le discrétisons ensuite dans la Carte de Profondeur, avec la valeur de Profondeur 0, ou alors nous le remplissons en noir dans la Carte d'Occultation.

Bien entendu, un bloqueur complexe peut être tranché à différentes positions. La combinaison prudente des Cartes d'Occultations requiert une *reprojection* prudente sur un nouveau plan, ce qui sera abordé dans la section 4.

3 Amélioration de la projection étendue des objets

Les opérateurs de Projection que nous avons présentés sont *prudents*, ce qui signifie qu'ils ne peuvent jamais identifier par erreur un objet comme invisible alors qu'il est visible. Cependant, le contraire peut se



(a) (b)

FIG. 5.5: Projection étendue d'un bloqueur convexe. (a) La Projection d'un bloqueur convexe est l'intersection des vues depuis les sommets de la cellule (nous avons représenté une cellule carrée pour plus de simplicité). (b) Calcul à l'aide du tampon de stencil. Les valeurs du stencil indiquent le nombre de vues qui se projettent sur chaque pixel.

produire, un objet peut être déclaré potentiellement visible alors qu'il est en fait caché depuis tous les points de la cellule.

Dans cette section, nous présentons une amélioration dans le cas de bloqueurs convexes ou planaires, pour des configurations dans lesquelles notre Projection initiale entraîne des tests trop prudents. Nous présentons tout d'abord des propriétés d'occultation qui permettent notre optimisation. Nous décrivons une Projection améliorée en 2D où le problème est plus simple. Nous expliquons ensuite pourquoi ce résultat ne peut être trivialement étendu en 3D, et comment on peut tout de même s'en servir pour optimiser les calculs 3D.

3.1 Quelques propriétés des ombres

Dans la figure 5.6(a), nous montrons une situation 2D pour laquelle notre Projection est trop restrictive. La Projection de l'objet n'est pas contenue dans la Projection du bloqueur, alors qu'il est caché de manière évidente depuis tous les points de la cellule. Cette configuration se produit parce que l'objet se trouve entre le plan de projection et le bloqueur, et parce que le bloqueur est convexe. Si l'objet est derrière le plan, la Projection que nous avons présenté dans la section 1.2 donne des résultats satisfaisants.

Tester la pénombre de l'objet et l'ombre du bloqueur n'est pas suffisamment restrictif. Comme nous allons le voir, on peut remplacer la pénombre par l'ombre de l'objet dans cette configuration pour obtenir la réponse souhaitée.

Pour ceci, nous allons définir une Projection améliorée pour les objets. Elle ne sera valable que si les bloqueurs sont convexes ou planaires, mais la fusion des bloqueurs sera toujours prise en compte. Remarquons que, bien que la restriction porte sur la classe des *bloqueurs*, c'est la Projection des *objets* qui est améliorée. Nous fondons notre Projection améliorée sur quelques propriétés des ombres des objets convexes que nous allons démontrer.

Dans ce qui suit, nous utilisons le terme *cône* pour désigner un cône général défini par un sommet et un ensemble de points (pas nécessairement circulaire). Le terme "pyramide" pourrait aussi être utilisé. Nous utilisons également le mot "cône" pour signifier un coin 2D, ce qui rend la terminologie plus cohérente entre la 2D et la 3D. De la même façon, nous utiliserons toujours l'expression "plan de projection" en 2D, alors qu'une ligne est en fait utilisée.

Avant de définir notre Projection améliorée, introduisons deux propriétés importantes que nous utiliserons

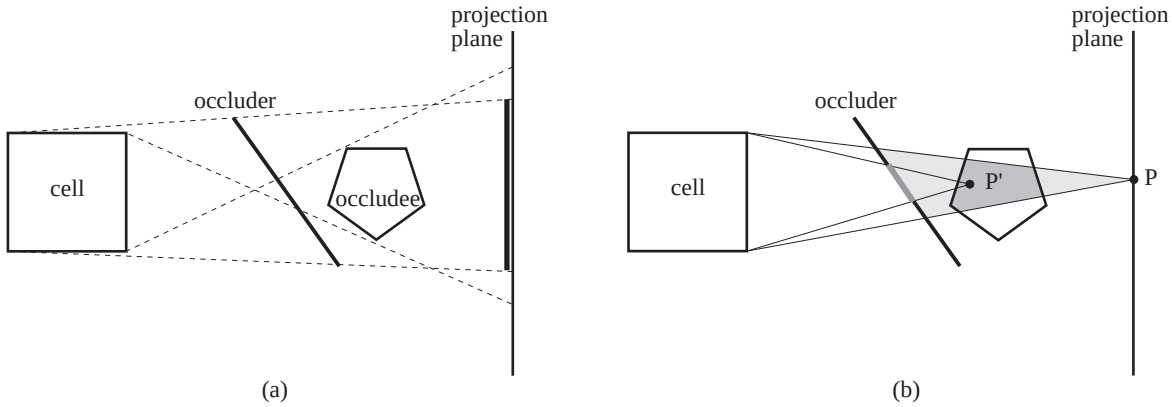


FIG. 5.6: (a) Configuration dans laquelle les projections étendues initiales sont trop restrictives. La Projection de l'objet n'est pas contenue dans la Projection du bloqueur, alors qu'il est de manière évidente caché. (b) Tout point P' à l'intérieur du cône défini par P et la cellule et qui est derrière le bloqueur est caché.

pour démontrer sa validité.

Propriété 1 *Pour un point P donné à l'intérieur de la région d'ombre d'un bloqueur convexe ou planaire par rapport à une cellule, tous les points P' qui se trouvent derrière le bloqueur et qui sont également dans le cône défini par P et la cellule, sont également cachés depuis la cellule.*

Cette propriété est illustrée figure 5.6(b). La preuve est évidente pour les bloqueurs convexes, puisque le cône défini par P' et la cellule est contenu dans le cône défini par P et la cellule. La section du bloqueur qui occulte P est un sur-ensemble de la section qui occulte P'. Remarquons que cette propriété n'est pas valable pour des bloqueurs quelconques.

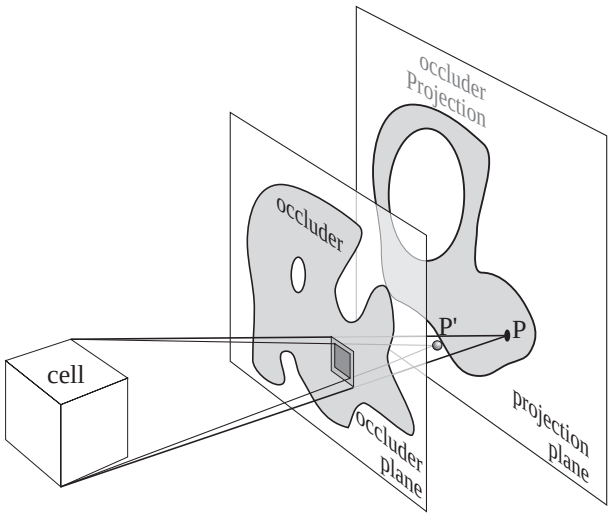


FIG. 5.7: Cas d'un bloqueur planaire concave. L'intersection du bloqueur et du cône (défini par le point P et la cellule) est en gris sombre. Elle est convexe et égale à l'intersection avec le plan du bloqueur.

Toutefois, le cas 3D de bloqueurs concaves et planaires est similaire au cas convexe. Considérons la situation de la figure 5.7. Si un point P est dans l'ombre d'un tel bloqueur, alors l'intersection du bloqueur avec le cône défini par P et la cellule doit être égal à l'intersection du plan du bloqueur avec le cône (autrement il y a un "trou" et P n'est pas dans l'ombre). Cette intersection est donc convexe. L'intersection du cône défini par P' et la cellule est un sous-ensemble de cette intersection, P' est donc caché. Le plan du bloqueur n'est pas nécessairement parallèle au plan de projection, et le bloqueur peut être concave ou avoir des trous.

Propriété 2 *Pour pouvoir effectuer des tests valides, la* Projection améliorée *d'un objet doit avoir la propriété suivante : l'union des cônes définis par les points de la Projection améliorée et la cellule doit contenir l'objet.*

Considérons la situation de la figure 5.6(b). Les points de l'objet à l'intérieur du cône défini par P et la cellule sont cachés à cause de la propriété 1. Donc si l'objet est contenu dans l'union des cônes définis par la cellule et les points de la Projection améliorée, tous les points de l'objet sont dans l'un de ces cônes, et ils peuvent donc être déterminer cachés grâce à la propriété 1.

Remarquons que la fusion des bloqueurs est toujours prise en compte, puisque tous les points P qui définissent les cônes n'ont pas besoin d'être cachés par le même bloqueur convexe ou planaire.

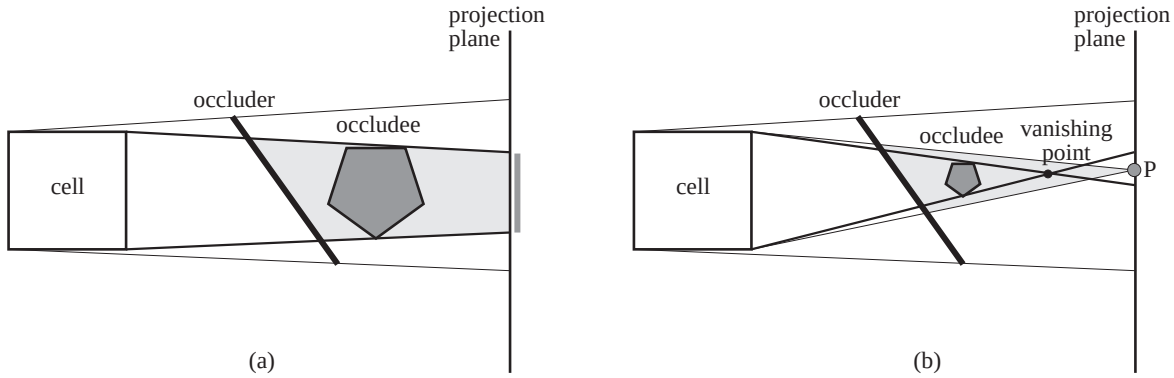


FIG. 5.8: (a) En 2D, la Projection améliorée est définie par l'intersection du plan de projection avec les droites supports de la cellule et de l'objet. (b) lorsque le point de fuite de l'ombre de l'objet est devant le plan de projection, tout point entre les intersections des droites supports peut être utilisé.

3.2 Projection étendue améliorée en 2D

Une Projection améliorée qui respecte la propriété 2 peut être définie en 2D en considérant les droites supports de la cellule et de l'objet, comme cela est illustré figure 5.8(a). L'intersection de ces droites avec le plan de projection définit les limites de la Projection améliorée de l'objet.

Cependant, si l'objet est trop petit, les droites supports s'intersectent avant le plan de projection au *point de fuite*. Dans ce cas, tout point P entre les intersections des droites avec le plan de projection satisfait la propriété 2 (cf. figure 5.8(b)). Tout cône défini par un de ces points et la cellule contient le point de fuite, et donc l'objet. En pratique, nous utilisons le milieu des deux intersections pour nos calculs.

Remarquons que le calcul est alors le même si l'objet est devant ou derrière le plan de projection, les droites supports sont utilisées dans les deux cas. Par opposition à la Projection standard définie section 1.2, nous ne nous intéressons plus aux droites séparatrices. Dans le cas où l'objet est entre le plan de projection et la cellule, nous avons substitué l'ombre à la pénombre.

Résumons notre *Projection améliorée* en 2D : pour tout objet (qu'il soit devant ou derrière le plan de projection), nous calculons l'intersection avec le plan de projection des droites supports haute et basse de la cellule et de l'objet. Si le plan de projection est derrière le point de fuite (c'est-à-dire si l'intersection de la droite basse est au dessus de l'intersection de la droite haute), nous utilisons le milieu des deux intersections, sinon nous utilisons le segment tout entier. Nous illustrons notre Projection améliorée dans la figure 5.8. La Projection améliorée d'un objet est un point ou un segment, selon que le plan de projection est devant ou derrière le point de fuite.

3.3 Projection étendue améliorée en 3D

En 3D malheureusement, les plans supports ne peuvent être utilisés aussi simplement, et le point de fuite est mal défini. Même si le volume d'ombre d'un objet intersecte le plan de projection, l'union des cônes définis par la cellule et les points du plan dans l'ombre ne contient pas nécessairement l'objet.

C'est pourquoi nous projetons le problème sur deux plans orthogonaux au plan de projection et parallèles aux faces de la cellule (cf. figure 5.9). Sur chacun de ces plans, nous utilisons la Projection améliorée. Le produit cartésien de ces deux Projections 2D améliorées définit notre Projection 3D améliorée.

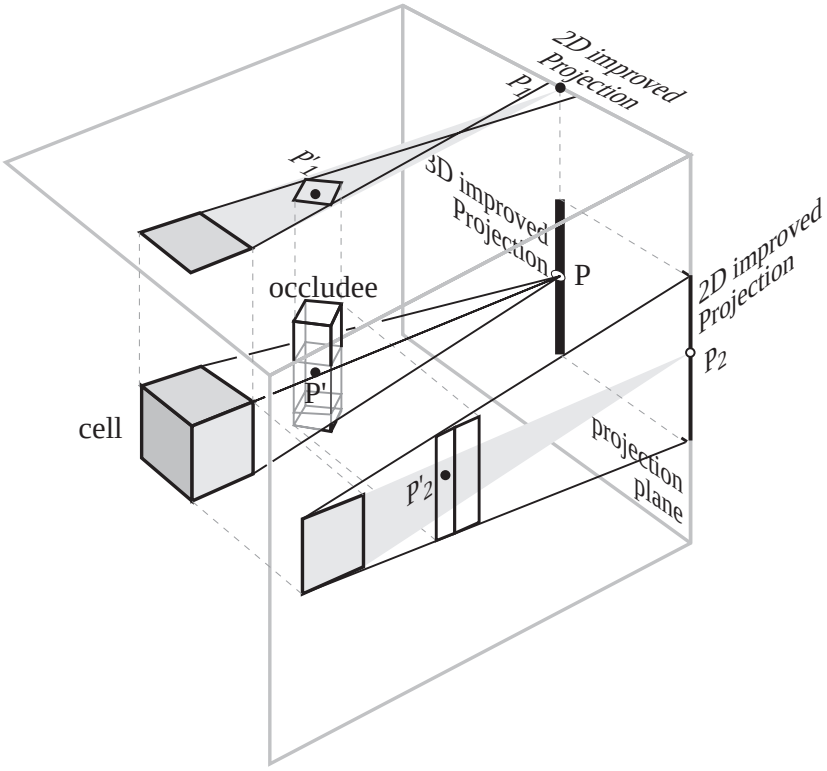


FIG. 5.9: La Projection 3D améliorée est le produit cartésien de deux Projections 2D améliorées. Tout point P' de l'objet est contenu dans un cône défini par la cellule et un point P de la Projection améliorée. Ce cône peut être défini en considérant les deux projections 2D correspondantes.

Cette Projection 3D améliorée vérifie la propriété 2. Considérons un point P' de l'objet. Sa projection P'_i sur chacun des deux plans est à l'intérieur d'un cône défini par la projection de la cellule et par un point P_i de la Projection 2D améliorée. Le point 3D P' est donc contenu dans le cône défini par la cellule et le produit cartésien P de P_1 et P_2 (cf. figure 5.9).

Cela est vrai avant tout parce que la cellule est le produit cartésien de ses projections 2D, puisqu'il s'agit d'une boîte et que les plans sont parallèles à ses faces. Un cône défini par un point P et par la cellule est donc le produit cartésien des cônes définis par les projections de la cellule et du point P sur les deux plans.

La Projection 3D améliorée est le produit cartésien des Projections 2D améliorées qui sont des points ou segments. Il s'agit donc d'un rectangle (segment $\times$ segment), d'un segment (segment $\times$ point), ou d'un point (point $\times$ point).

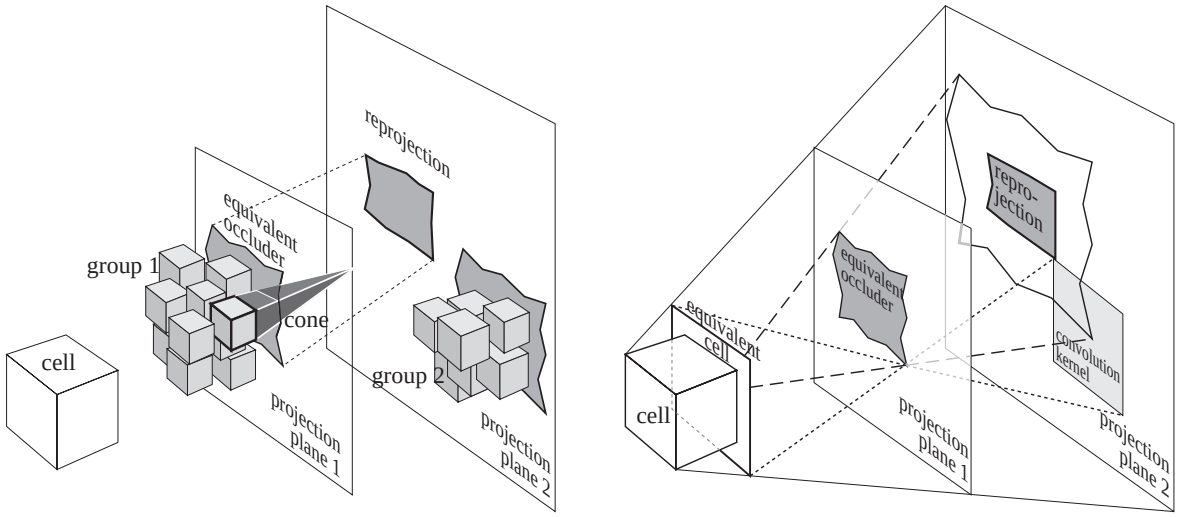
Notre Projection 3D améliorée permet des tests d'occultation prudents dans le cas de bloqueurs convexes ou planaires. Comme nous l'avons déjà mentionné pour le cas 2D, la fusion des bloqueurs est toujours prise en compte puisque tous les cônes contenant l'objet ne doivent pas nécessairement être occultés par le même bloqueur.

4 Reprojection des bloqueurs et balayage d'occultation

Le choix du plan de projection a une grande influence sur l'efficacité des tests d'occultations à l'aide de Projections, en particulier pour la fusion des bloqueurs (nous en discuterons plus longuement dans la section 5.2). Des emplacements contradictoires peuvent être requis pour prendre en compte la fusion de différents

groupes de bloqueurs, comme cela est illustré figure 5.10. Pour régler ce problème, nous proposons d'utiliser un premier plan de projection pour calculer une Carte d'Occultation où les bloqueurs fusionnent, puis de re-Projeter cette Carte d'Occultation sur un nouveau plan de projection. L'agrégation dans la première Carte d'Occultation crée un plus grand *bloqueur équivalent* dont l'occultation peut être prise en compte dans le deuxième plan. Le même procédé s'applique pour le tranchage des bloqueurs concaves. Dans un certain sens, notre approche peut être vue comme une agrégation et simplification des bloqueurs à base d'image.

Nous commençons par justifier pourquoi et quand la re-Projection est valide. Nous expliquons ensuite comment les Cartes d'Occultation peuvent être efficacement re-Projetée sur de nouveaux plans de projection. Pour finir, nous étendons ces concepts et présentons un *balayage d'occultation* où un plan balaye la scène en partant de la cellule tout en agrégeant les occultations.



(a) (b)

FIG. 5.10: (a) Si le plan de projection 2 est utilisé, les occultations dues au groupe de bloqueurs 1 ne sont pas prises en compte. Le cône d'ombre de l'un des cubes montre bien que sa Projection serait vide, puisque son point de fuite est devant le plan 2. Les mêmes contraintes s'appliquent pour le plan 1 et le groupe 2. Il est souhaitable de Projeter le groupe 1 sur le plan 1, puis de *re-Projeter* le bloqueur équivalent sur le plan 2. (b) Re-Projection des bloqueurs. La Carte d'Occultation du plan 1 est projetée sur le plan 2 depuis le centre de la cellule équivalente. Elle est ensuite convoluée avec l'image inverse de la cellule équivalente.

4.1 Validité

Nous montrons maintenant que la Projection de plusieurs bloqueurs peut être utilisée comme un bloqueur équivalent, c'est-à-dire que tout objet caché par cette Projection est également caché par les bloqueurs. Nous prouvons le propriété suivante qui est plus générale.

Propriété 3 *Considérons une source de lumière étendue, un objet A quelconque (convexe ou concave) et U le cône d'ombre de A. Alors l'ombre de tout sous-ensemble U' de U est contenue dans U.*

Pour le démontrer, considérons un point P dans l'ombre de U' (figure 5.11). Tout rayon r passant par P et la source coupe U'. Soit P' un point d'intersection. Puisque $P' \in U' \subset U$, alors P' est dans l'ombre de A. Donc tout rayon (r par exemple) passant par P' et la source coupe A. Nous avons montré que tout rayon passant par P et la source coupe A. P est donc dans l'ombre de A.

Observons que la propriété 3 ne requiert aucune propriété de convexité ou de planarité de la part de A.

Si la cellule est considérée comme une source de lumière, cela prouve que tout sous-ensemble de l'ombre de bloqueurs est une version prudente de ces bloqueurs. Comme nous l'avons vu, la Projection d'un bloqueur qui se trouve devant le plan de projection est son ombre sur ce plan. Cette Projection peut donc être re-Projetée

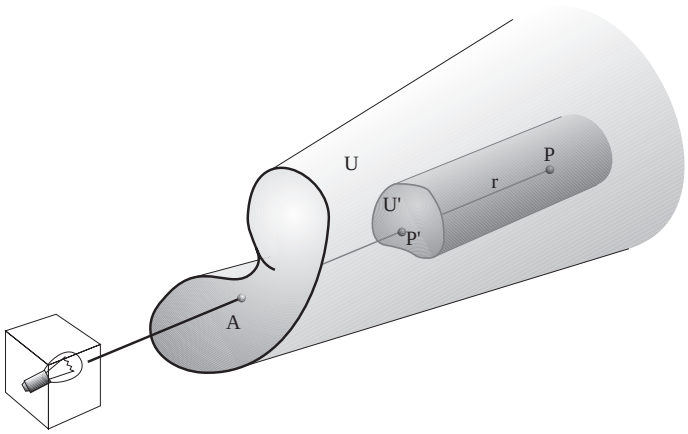


FIG. 5.11: Ombre d'un sous-ensemble d'ombre. Le point P est dans l'ombre de U' qui est un sous-ensemble de l'ombre U de A. Il est donc lui aussi dans l'ombre de A.

comme un nouveau bloqueur. Si le bloqueur se trouve derrière le plan, sa Projection ne se trouve pas dans l'ombre parce qu'elle se trouve avant le bloqueur. La propriété 3 ne s'applique alors pas.

Nous ne considérons donc plus que des bloqueurs qui se trouvent devant le plan de projection, et ne considérons donc plus dans le plan de projection initial que des Cartes d'Occultation et non des Cartes de Profondeur. Une Carte d'Occultation définit un bloqueur planaire qui peut être utilisé avec notre Projection améliorée. La reprojection sera effectuée vers des plans qui se trouvent derrière le plan de projection initial. Cependant, la profondeur du plan initial peut être utilisée pour la Carte de Profondeur du plan de Projection final.

L'opérateur de reprojection que nous allons définir est en fait un opérateur de projection étendue pour le cas spécial de bloqueurs planaires parallèles au plan de projection.

4.2 Reprojection

Notre technique de reprojection s'appuie sur les travaux de Soler et Sillion [SS98a, Sol98] sur les calculs d'ombres douces. Ils montrent que dans le cas de bloqueurs planaires parallèles à la source et au récepteur, le calcul d'ombres douces est équivalent à la convolution de la projection des bloqueurs avec l'image inversée de la source. Cette convolution peut être calculée de manière très efficace à l'aide de transformées de Fourier rapides (FFT). Ils étendent cette approche au cas général en projetant la géométrie sur trois plans parallèles, en calculant les ombres par convolution, puis en reprojetant l'ombre sur le véritable récepteur. Ce calcul est une approximation, mais il donne des résultats impressionnants.

Nous nous trouvons quasiment dans le cas idéal pour l'algorithme de Soler et Sillion : le bloqueur (la Carte d'Occultation sur le plan initial) et le récepteur (le plan de projection final) sont parallèles. Cependant, notre source de lumière (la cellule) est un volume. Nous définissons une cellule équivalente qui est parallèle aux plans de projection et qui permet des tests prudents sur le nouveau plan. Sa construction est simple et est illustrée figure 5.10(b). Nous utilisons le fait que notre plan de projection n'est pas infini mais borné (c'est un rectangle). Notre cellule équivalente est le rectangle défini par la face la plus proche des plans de projection, et par les plans supports de la cellule et du plan (rectangle) de projection final.

Tout rayon qui passe par la cellule et par le plan de projection coupe également la cellule équivalente. Donc tout objet caché depuis la cellule équivalente est également caché depuis la cellule.

La méthode de convolution calcule des niveaux de gris continus. Pour obtenir une Carte d'Occultation binaire, nous effectuons un seuillage. En pratique, comme nous utilisons la convolution câblée sur des entiers, nous utilisons un facteur d'échelle et le seuillage (multiplier par 255 revient à mettre à 255 toutes les valeurs non nulles).

La figure 5.10 montre que la reprojection des Projections depuis le plan 1 vers le plan 2 permet de traiter l'occultation cumulée des deux groupes et la fusion des bloqueurs. Rappelons qu'une Carte de Profondeur peut être utilisée sur le plan final. La Profondeur de la reprojection du bloqueur équivalent est la profondeur du plan initial.

4.3 Balayage d'occultation

Pour traiter l'occultation due à de nombreux bloqueurs concaves ou à de petits bloqueurs, nous généralisons la reprojection en un *balayage d'occultation*. Il s'agit d'un balayage[2] de la scène par des plans de projection parallèles quittant la cellule où les occultations sont agrégées grâce à la reprojection.

La figure 5.12 illustre son principe. Nous Projetons les bloqueurs convexes qui se trouvent devant le plan de projection courant P sur P. Nous calculons également les tranches des bloqueurs concaves qui coupent P. Le résultat est une Carte d'Occultation contre laquelle les objets sont testés.

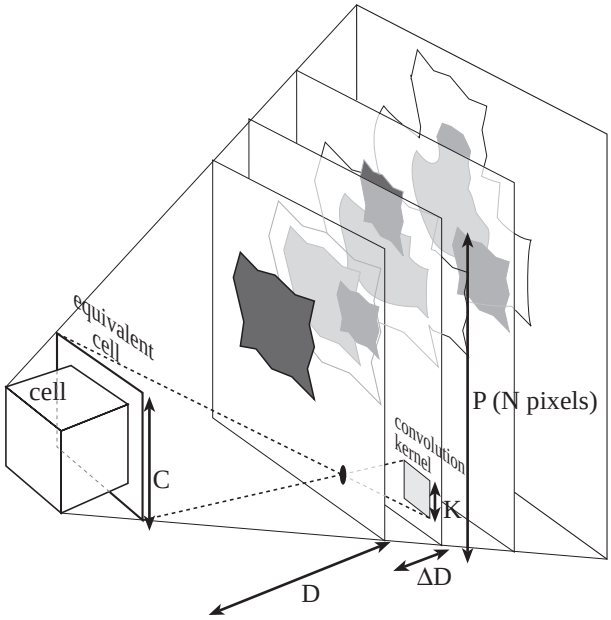


FIG. 5.12: Balayage d'occultation. La scène est balayée par un ensemble de plans parallèles quittant la cellule. Les bloqueurs et les objets ne sont pas représentés pour simplifier la figure. Sur chaque plan de projection nous montrons en gris clair la reprojection de la Carte d'Occultation du plan précédent (la projection avant convolution est représentée avec des traits fins). Les nouvelles Projections sont en gris sombre.

Nous avançons ensuite au plan de projection suivant. Nous reprojetons la Carte d'Occultation du plan précédent en utilisant la technique de convolution décrite dans la section précédente. Nous calculons de nouvelles tranches, et Projetons les bloqueurs qui se trouvent entre les deux plans. Cela définit une Carte d'Occultation sur le nouveau plan. Nous testons à nouveau les objets, et le balayage peut suivre son cours.

La distance ΔD entre deux plans de projection est choisie pour optimiser l'utilisation de la convolution discrète. Nous souhaitons que la taille du noyau de convolution soit un nombre entier K de pixels. Cela signifie que nous souhaitons que la projection inverse de la cellule équivalente sur le nouveau plan soit de taille K (figure 5.12). Soient D la distance entre le plan initial et la cellule équivalente, C la taille de la cellule équivalente, N la résolution de la Carte d'Occultation et P la taille du plan de projection. En appliquant le théorème de Thalès on obtient :

$$\Delta D = \frac{D(K-1)P}{CN}$$

On utilise $(K-1)$ car un noyau discret de taille K "retire" effectivement $\frac{K-1}{2}$ pixels autour de chaque pixel. En pratique nous utilisons $K = 5$ pixels. Remarquons que les plans ne sont alors pas équidistants, ce qui traduit naturellement le fait que les occultations varient plus rapidement près de la cellule.

Il n'est pas nécessaire de tester les objets à chaque plan de projection. Nous ne testons que les objets les plus proches pour chaque plan, les autres sont testés moins souvent (par exemple tous les 3 ou 5 plans).

[2]Notre utilisation du terme balayage est différente de celle en vigueur en géométrie algorithmique ou de celle du chapitre 2. Nous ne calculons aucun événement de balayage et l'emplacement des plans est indépendant de la scène.

Cette méthode de balayage demeure cependant plus coûteuse que notre approche initiale de Projection avec un seul plan de projection, car de nombreuses reprojections doivent être effectuées, et parce que les objets doivent être testés plusieurs fois. Cependant, elle permet de traiter des configurations très difficiles comme l'occultation due à l'agrégation des feuilles dans une forêt.

5 Système

Certains aspects de notre méthode restent vagues. Dans ce qui suit, nous décrivons quels objets sont choisis comme bloqueurs et comment les plans de projection sont choisis. Nous présentons le précalcul adaptatif pour les cellules, avant de parler du cas d'objets en mouvement. Pour finir, nous décrivons comment l'information de visibilité est utilisée lors du rendu interactif.

5.1 Sélection des bloqueurs

Pour chaque cellule de vue, nous choisissons un ensemble de bloqueurs pertinents, à l'aide d'une heuristique similaire à celle de Coorg et Teller [CT97b], Zhang *et al.* [ZMHH97, Zha98b] et Hudson *et al.* [HMC+97]. L'importance d'un bloqueur est jugée d'après son angle solide approximatif. En pratique, nous le calculons au centre de la cellule.

Pour optimiser cette sélection, nous utilisons un précalcul similaire aux approches antérieures. Pour chaque bloqueur, nous précalculons la région de la scène pour laquelle il peut être pertinent. Nous stockons cette information dans un *kd-tree*. Pour chaque cellule, nous utilisons l'ensemble des bloqueurs correspondants aux nœuds du *kd-tree* qu'elle occupe.

Pour améliorer l'efficacité du test d'occultation, nous avons développé une approche adaptative qui sélectionne plus de bloqueurs dans les directions où beaucoup d'objets sont identifiés visibles. Au fur et à mesure que nous testons les objets, nous stockons avec la Carte de Profondeur une *carte d'échec* qui indique quelles régions du plan de projection ont besoin d'être remplies par des Projections de bloqueurs. Chaque fois que le test d'occultation d'un objet échoue, nous mettons à jour la carte d'échec. Pour chaque pixel de la Projection de l'objet, nous ajoutons le nombre de polygones de l'objet dans la carte d'échec. Nous utilisons ensuite cette information pour réduire le seuil d'angle solide pour le choix des bloqueurs dans cette direction.

Cette approche réclame deux passes de test d'occultation. Durant la première passe, la carte d'échec est calculée ; elle est ensuite utilisée pour améliorer la sélection des bloqueurs. La Carte de Profondeur est mise à jour avec les Projections des nouveaux bloqueurs, et seuls les objets qui n'ont pas été identifiés invisibles lors de la première passe sont à nouveau testés.

La Carte de Profondeur pourrait aussi être directement mise à contribution pour la sélection des bloqueurs, sans effectuer de test préliminaire sur les objets. Ainsi que le proposent Zhang *et al.* [ZMHH97, Zha98b], les trous de la Carte de Profondeur peuvent être détectés pour indiquer les directions dans lesquelles des bloqueurs sont requis. Cette solution n'a pas encore été implémentée.

5.2 Choix des plans de projection

Jusqu'ici, nous avons discuté notre méthode sans nous soucier de la position du plan de projection. Cependant, l'efficacité de la méthode dépend grandement du choix de ce plan. Nous commençons par essayer de comprendre les contraintes qui président au choix de ce plan, avant de présenter l'heuristique simple que nous avons choisie. Rappelons que les plans de projection sont restreints aux trois directions de la cellule de vue. Nous appuyons notre discussion sur un exemple 2D pour plus de clarté, mais les propriétés que nous décrivons sont dans ce cas similaires en 3D.

Nous devons comprendre quel est l'ensemble des Projections possibles d'un bloqueur en fonction du plan de projection. Nous définissons le *volume de Projection* comme l'ensemble des Projections sur tous les plans de projection possibles (cf. figure 5.13(a)). La partie du volume de Projection derrière le bloqueur correspond au cône d'ombre. Le volume de Projection peut être fini ou infini, tout comme le cône d'ombre. De la même façon, nous montrons dans la figure 5.13(a) le volume de Projection améliorée d'un objet. Il s'agit d'un cône qui a pour sommet le point de fuite, prolongé par une ligne droite. L'intersection du volume de Projection avec le plan de projection donne la Projection sur ce plan.

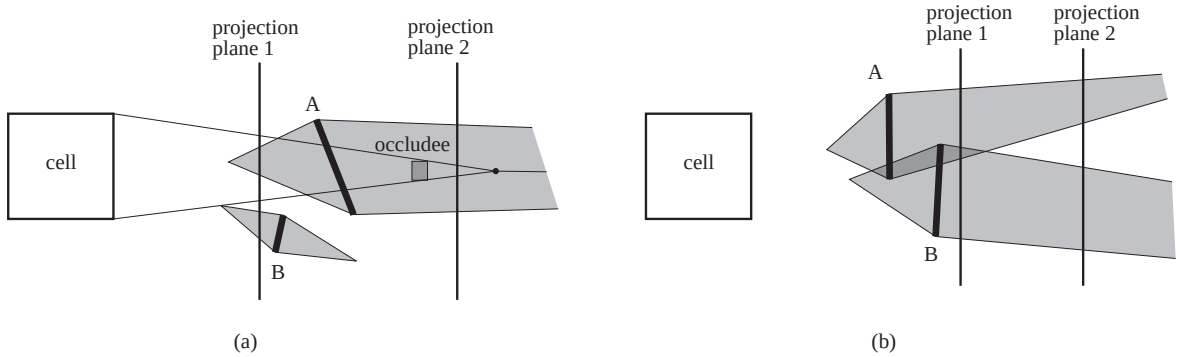


FIG. 5.13: Choix du plan de projection. Nous ne montrons pas les droites supports et séparatrices pour alléger la figure. En gris, nous montrons le volume de Projection, qui est le lieu de toutes les Projections d'un bloqueur en fonction du choix du plan de projection. (a) Nous montrons également le volume de Projection d'un objet. (b) L'intersection des volumes de Projection de *A* et *B* (en gris foncé) délimite les plans de projection qui prennent en compte la fusion de *A* et *B*.

Intéressons-nous à l'influence du choix du plan de projection sur le test entre un objet et un seul bloqueur *A*. Dans la figure 5.13, le plan de projection 1 ne permettra pas de détecter l'occultation, tandis que le plan 2 le permettra. Ceci est général : si le plan de projection est derrière le bloqueur, le test sera toujours correct en 2D. Le test est moins efficace si le plan se trouve devant le bloqueur à cause de l'utilisation des droites séparatrices (cf. figure 5.13(a)).

Nous voulons également prendre en compte la *fusion des bloqueurs*. Considérons la situation de la figure 5.13(b). *A* et *B* ne fusionneront que si le plan de projection coupe la région définie par l'intersection de leurs volumes de Projection. Le plan 1 prendra en compte la fusion des deux bloqueurs, mais pas le plan 2.

L'heuristique que nous utilisons est très simple, elle s'appuie sur l'optimisation d'un critère simple. Nous cherchons à maximiser le nombre de pixels remplis dans la Carte d'Occultation. Nous plaçons un plan candidat juste derrière chaque bloqueur. Nous évaluons la taille de la Projection de chaque bloqueur sur ce plan. Cette méthode utilise la "force brute", mais elle demeure rapide grâce au faible nombre de bloqueurs considérés. Cette heuristique ne cherche cependant pas directement à maximiser la fusion des bloqueurs. Une heuristique fondée sur les observations de la figure 5.13(b) demande à être développée.

Six plans de projection sont en pratique utilisés pour couvrir toutes les directions. Contrairement à l'hémicube [CG85] ou au "tampon de lumière" (*light buffer*) [HG86], ces plans ne définissent pas une boîte. Nous étendons les plans (par exemple d'un facteur 1,5) pour éviter les problèmes avec les Projections trop obliques (figure 5.14). Cela permet de mieux détecter les occultations dans les directions rasantes.

5.3 Précalcul adaptatif

Nous organisons les cellules de vue en une hiérarchie spatiale. Cette hiérarchie est différente de celle utilisée pour les boîtes englobantes des objets. Toute subdivision spatiale hiérarchique peut être utilisée, potentiellement guidée par les besoins spécifiques de l'application. Notre algorithme commence par une subdivision initiale de l'espace des points de vue possibles de l'observateur. Il peut s'agir de la boîte englobante de la scène si aucune connaissance n'est disponible *a priori*, mais toute subdivision naturelle peut être exploitée (les boîtes englobantes des rues par exemple pour le modèle de ville).

Pour chaque cellule de vue, nous effectuons un calcul d'occultation. Nous choisissons les bloqueurs et le plan de projection appropriés. Nous Projetons ensuite les bloqueurs pour construire la Carte de Profondeur Hiérarchique. Pour finir, les objets sont testés récursivement. Si un nœud est détecté invisible ou complètemnt visible, la récursion s'arrête. Par complètement visible, nous entendons que sa Projection n'intersecte aucune Projection de bloqueur. Dans ce cas, aucun fils de ce nœud ne pourra être déclaré invisible. Les objets déclarés visibles sont insérés dans l'ensemble d'objets potentiellement visibles (PVS) de la cellule.

Si la taille du PVS de la cellule nous satisfait, nous passons à la cellule suivante. Dans le cas contraire, la cellule est subdivisée et nous continuons le calcul de manière récursive sur les sous-cellules. Cependant, seuls les objets faisant partie du PVS de la cellule mère sont testés pour les cellules filles.

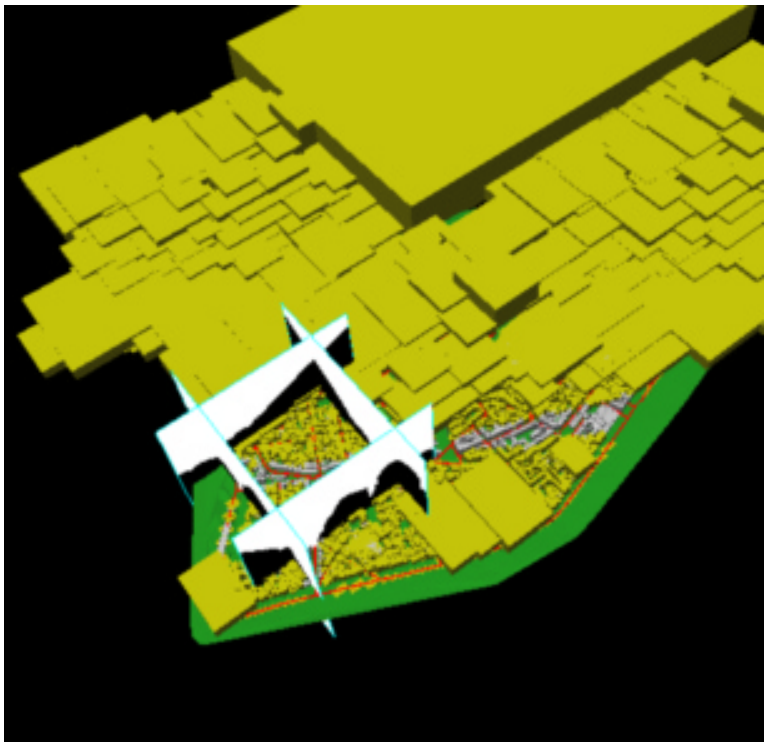


FIG. 5.14: Exemple de plans de projection pour le modèle de ville. Les boîtes jaunes correspondent aux nœuds de la hiérarchie des objets qui sont détectés invisibles depuis la cellule.

Les calculs d'occultation sont améliorés quand on rapetisse la cellule, car les points de vue deviennent plus proches les uns des autres. Les vues depuis ces points de vue sont plus semblables, ce qui rend les Projections des bloqueurs plus grandes, et celles des objets plus petites.

Le critère d'arrêt que nous avons implémenté utilise simplement un budget de polygones : si le PVS a plus qu'un certain nombre de polygones, nous subdivisons la cellule (jusqu'à un seuil de taille). Un critère plus élaboré consisterait à comparer la taille du PVS à celle d'un échantillon de vues.

L'information de PVS peut devenir très grande et requérir beaucoup de mémoire. Pour éviter ceci, nous utilisons des *delta-PVS*. Nous stockons le PVS entier pour une seule cellule initiale. Chaque cellule stocke ses cellules adjacentes et la *différence* de PVS par rapport à ses voisines.

5.4 Objets en mouvement

Notre méthode peut traiter l'occultation d'objets en mouvement par des bloqueurs fixes. Les bloqueurs dynamiques ne peuvent cependant pas être traités.

Une hiérarchie de boîtes englobantes est construite pour la région dans laquelle les objets se meuvent [SG96]. Lors du précalcul, ces boîtes englobantes subissent un test d'occultation par rapport à la Carte de Profondeur de chaque cellule. Lors de l'affichage interactif, un objet dynamique n'est affiché que s'il est contenu dans une boîte englobante visible.

Dans l'exemple du simulateur de conduite, les rues correspondent à la hiérarchie de boîtes englobantes pour les voitures en mouvement. Lors de la balade interactive, une voiture n'est affichée que si la rue dans laquelle elle se trouve est visible depuis la vue courante de l'observateur.

5.5 Rendu interactif

Une fois que le précalcul est fait et que l'information de PVS a été stockée sur disque, un programme de rendu interactif peut utiliser cette information pour afficher très rapidement des modèles très complexes.

Nous utilisons une structure de graphe de scène classique [RH94] pour représenter la scène. Un simple drapeau permet pour chaque nœud de le déclarer visible ou invisible.

Chaque fois que l'observateur entre dans une nouvelle cellule, le graphe de scène est mis à jour. Notre codage des delta-PVS permet de le faire très efficacement. Les nœuds qui étaient cachés dans la cellule précédente sont à nouveau déclarés visibles, tandis que les nœuds qui deviennent cachés sont marqués inactifs.

Le sur-coût est très faible, puisque nous ne faisons que mettre à jour des drapeaux dans le graphe de scène. Dans la section 7.3, nous esquisserons une extension pour gérer le pré-chargement depuis le réseau ou depuis le disque dur.

6 Implémentation et résultats

6.1 Implémentation

Nous avons implémenté deux systèmes indépendants, l'un pour le précalcul et l'autre pour le rendu interactif. Le précalcul utilise autant que faire se peut le matériel graphique, en particulier pour les Projections et pour la convolution. Les Cartes de Profondeur sont lues depuis la mémoire graphique, et les tests d'occultations sont effectuées en logiciel. Cette opération pourrait être optimisée en utilisant les nouvelles cartes graphiques qui permettent des test de profondeur efficaces [Sgi99]. Les PVS calculés sont stockés sur disque.

Notre implémentation du rendu interactif est basée sur la bibliothèque SGI Performer[RH94]. Performer offre un puissant graphe de scène et l'élimination des objets hors du cône de vue (*view-frustum culling*), ce qui fournit une base honnête de comparaison.

6.2 Résultats

Projections étendues sur un seul plan de projection

Les scènes de test que nous avons utilisées pour la méthode à un seul plan de projection sont les suivantes : le modèle du quartier de Montmartre, contenant 150 000 polygone ; (b) ce modèle répliqué 4 fois avec en plus 2 000 voitures en mouvement de 1 000 polygones chacune (2,6 millions de polygones au total).

Nous avons utilisé la Projection des bloqueurs avec la carte de stencil et les Projections améliorées pour les objets. Toutes les Projections sont calculées à la résolution 256x256. Nous n'avons pas observé d'artefact, mais une comparaison avec des Cartes de Profondeur de plus grande résolution est souhaitable.

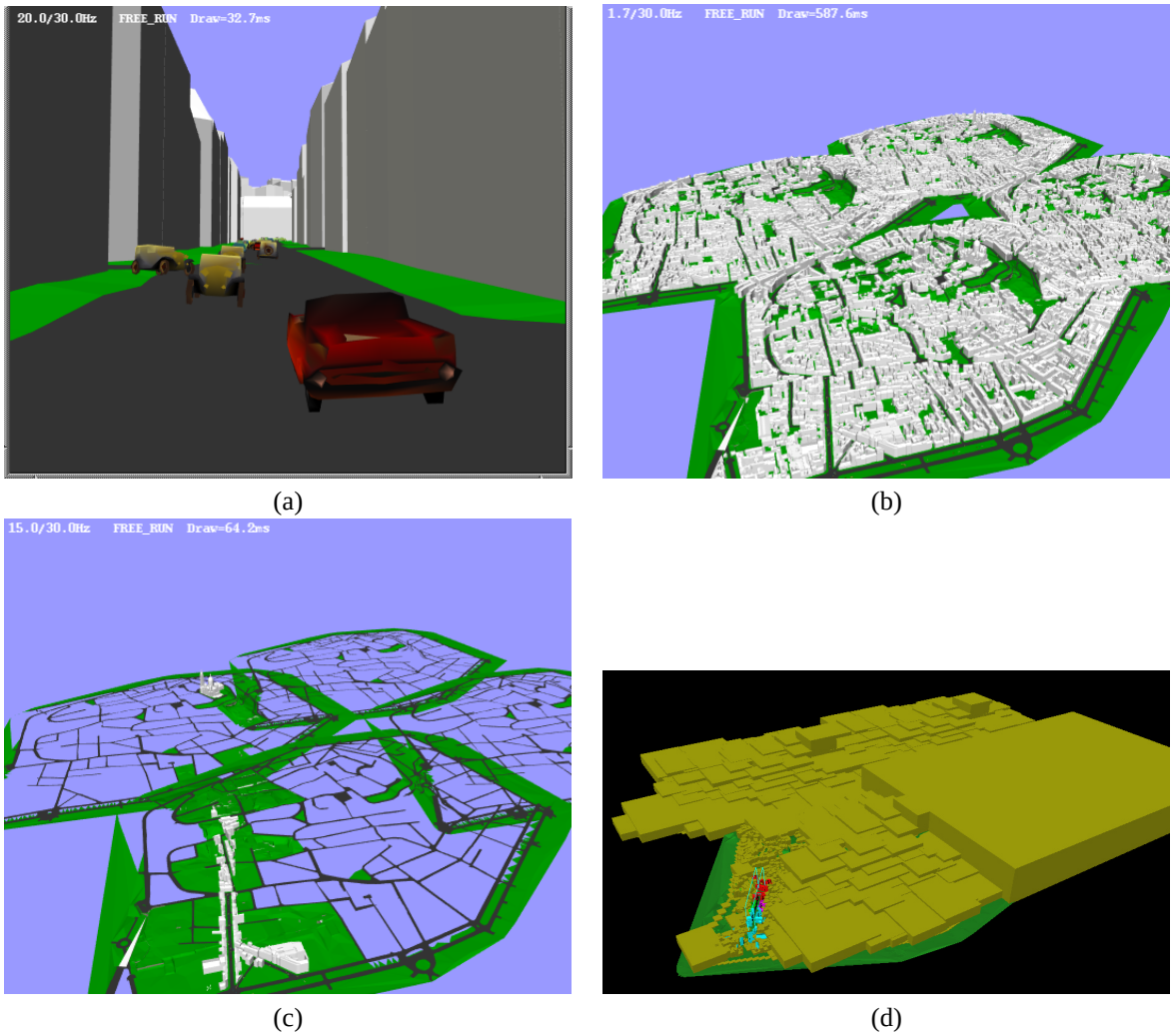
Le précalcul pour une seule instance du quartier a pris environ 1 heure sur une Onyx 2 Infinite Reality avec un processeur R10k à 196Mhz. Pour le quartier répliqué 4 fois avec les voitures, il a fallu 4 heures, pour 3479 cellules finales, soit 4,1 secondes par cellule. Dans cette dernière scène, le stockage du PVS requiert 25Mo, pour un modèle de 60Mo (sans compter les voitures). Environ 95% de la géométrie est déclarée invisible en moyenne.

Nous avons effectués nos tests de rendu interactif sur deux configurations matérielles. La première est une station de travail Silicon Graphics O^2 R5000, et la seconde une station haut de gamme Onyx 2 Infinite Reality 2xR10k. Une accélération d'environ 5 à 6 fois a été observée. Cela peut sembler faible après avoir éliminé 95% de la géométrie, mais Performer effectue une puissante élimination de la géométrie hors du cône de vue [GBW90], et environ 80% de la scène est éliminée. La figure 5.15 montre des résultats de notre méthode.

Nous avons implémenté l'algorithme de Cohen-Or *et al.* [COFHZ98, COZ98] afin d'effectuer une comparaison informelle (cf. figure 5.16). Pour le modèle de ville, cet algorithme déclare 4 fois plus d'objets potentiellement visibles que le nôtre en moyenne, pour un temps de calcul 150 fois plus élevé. Une version plus optimisée diminuerait sans aucun doute ce coût, mais les 4,1 secondes par cellule de notre algorithme semblent difficiles à battre en utilisant du lancer de rayon sur d'aussi gros modèles.

Balayage d'occultation

Nous avons effectué des test préliminaires pour le balayage d'occultation, en utilisant un modèle de forêt contenant 1 200 arbres de 1 000 feuilles triangulaires chacun. La Projection des feuilles sur le plan de projection est calculée avec l'algorithme de Projection pour bloqueurs convexes à l'aide de la carte de stencil. La taille



(a) (b)

(c) (d)

FIG. 5.15: Résultats de notre méthode. (a) Scène vue depuis la position de l'observateur. (b) Scène en vue d'oiseau sans élimination ; la scène contient 600 000 polygones pour les bâtiments et 2 000 voitures de 1 000 polygones chacune. (c) Même vue avec élimination d'occultation. (d) Les boîtes en jaune représentent les éléments de la hiérarchie d'objets qui sont éliminés.

du noyau de convolution est fixée à 5 pixels et nous utilisons 15 plans pour le balayage. Le balayage d'occultation prend environ 12 secondes par cellule, et élimine 75% des arbres. La figure 5.17 montre le balayage, et l'agrégation des feuilles dans la Carte d'Occultation.

7 Discussion

7.1 Résumé

Nous avons présenté des opérateurs de *projection étendue* qui autorisent des tests prudents d'occultation par rapport à une cellule de vue volumique. La projection étendue d'un bloqueur est l'intersection de ses vues, tandis que pour un objet, il s'agit de l'union des vues. Nous avons également défini une *profondeur étendue* qui permet la définition de *cartes de profondeur étendue* qui sont la généralisation du *z-buffer* pour des cellules volumiques.

Nos opérateurs permettent des tests d'occultation prudents et peuvent prendre en compte la *fusion de bloqueurs*, c'est-à-dire l'occultation due à la conjonction de plusieurs bloqueurs. Nous avons décrit des implémen-

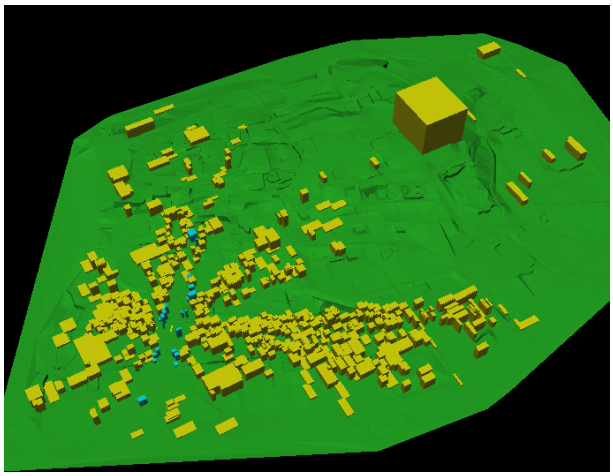


FIG. 5.16: Comparaison entre l'algorithme de Cohen-Or *et al.* [COFHZ98, COZ98] et les projections étendues. Nous représentons les bâtiments que leur algorithme déclare potentiellement visibles alors qu'ils sont éliminés par notre méthode. La cellule se trouve en bas à gauche.

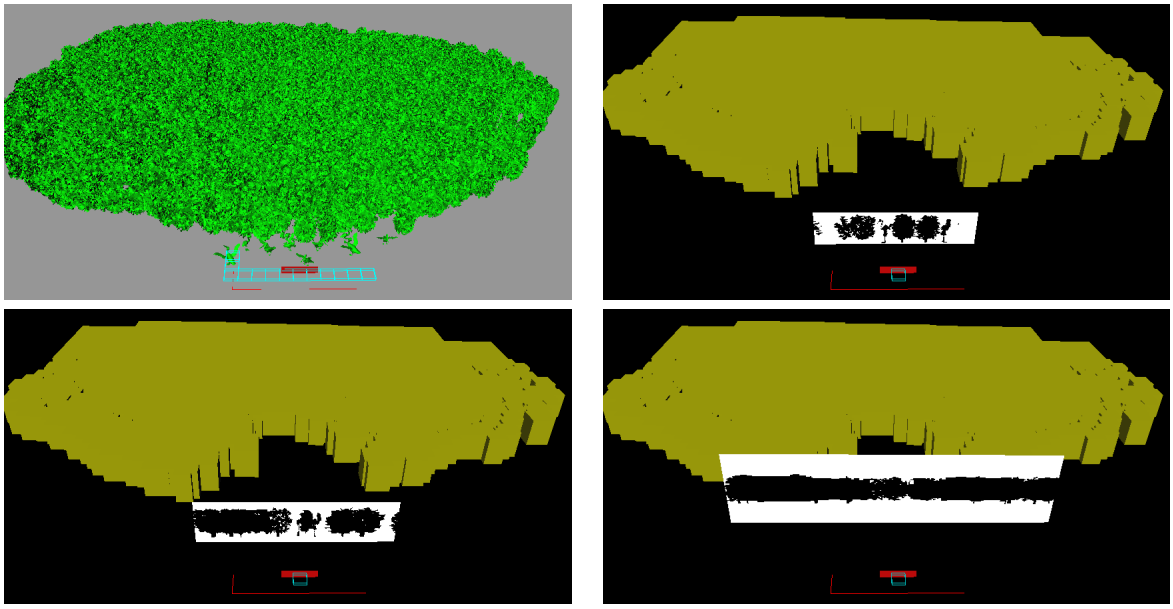


FIG. 5.17: Balayage d'occultation. (a) Modèle de forêt. (b)-(d) trois positions du plan de balayage. Les boîtes en jaunes indiquent les arbres éliminés.

tations efficaces pour ces opérateurs, ainsi qu'une amélioration pour le cas des bloqueurs convexes ou planaires.

Nous avons défini un opérateur de reprojection qui reprojette les projections étendues calculées pour un plan initial sur un nouveau plan de projection. Nous l'avons étendu en un *balayage d'occultation* qui consiste à balayer la scène par une série de plans parallèles quittant la cellule. Les occultations causées par de petits objets sont agrégées sur les plans lors du balayage.

Nous avons présenté une implémentation efficace qui bénéficie tant que faire se peut des accélérations matérielles. Notre précalcul subdivise de manière adaptative l'espace de l'observateur en cellules pour lesquelles des *ensembles d'objets potentiellement visibles* sont calculés. Nous obtenons un facteur d'accélération de 5 à 6 par rapport à un programme haut de gamme de rendu interactif, ce qui rend possible une balade dans un modèle de 2,6 millions de polygones. Des résultats préliminaires ont également été présentés qui montrent que notre balayage d'occultation permet de calculer, par rapport à une cellule volumique, les occultations dues aux feuilles des arbres dans une forêt.

7.2 Discussion

Pour commencer, il est important de souligner que notre méthode ne peut identifier qu'un sous-ensemble des objets qui peuvent être éliminés par une méthode qui effectue des calculs d'occultation par rapport à un point de vue unique, comme le z-buffer hiérarchique [GKM93] ou les cartes hiérarchiques d'occultation [ZMHH97, Zha98b]. Ces méthodes présentent également l'avantage de traiter les bloqueurs en mouvement et de ne nécessiter aucun précalcul et aucun stockage de PVS.

Cependant, notre méthode ne nécessite quasiment aucun surcoût lors de l'affichage, alors que le système de rendu de scènes complexes développé à l'Université de Caroline du Nord [ACW+99] utilise jusqu'à deux processeurs uniquement pour les calculs d'élimination d'occultation. De plus, pour certaines applications – les jeux, le rendu à travers le réseau, le tourisme virtuel, etc. – le précalcul n'est pas un véritable handicap, il peut être effectué une fois pour toute par le créateur de l'application. Les PVS que nous calculons permettent un pré-chargement prédictif qui est crucial pour transmettre des scènes à travers le réseau ou pour l'affichage de scènes qui ne peuvent être chargées entièrement en mémoire principale.

Par oppositions aux calculs exacts de visibilité (tels que ceux présentés dans les trois chapitres précédents), les opérateurs de projection étendue peuvent échouer à identifier certaines occultations dues à la conjonction de plusieurs bloqueurs. Comme nous l'avons vu à la section 5.2, la fusion des bloqueurs dépend du choix du plan de projection.

Les projections étendues réduisent un problème de visibilité qui est 4D à une représentation 2D sur un plan de projection. L'information de visibilité portée par l'ensemble des rayons qui passent par un point du plan de projection et la cellule est approximée de manière prudente par une seule valeur (binaire pour les Cartes d'Occultation, réelle pour les Cartes de Profondeur). Cela revient à projeter l'information de visibilité à l'intérieur du volume de tangence de la cellule sur une 2-variété.

L'approximation entraînée par la discrétisation sur le plan de projection est une question primordiale. Nous pensons que notre approche est moins sensible aux artefacts dus à une faible résolution que par exemple le z-buffer hiérarchique, pour lequel une plus faible résolution entraînerait des problèmes le long des silhouettes des objets. Dans ce cas, la carte de profondeur utilisée correspond directement à la vue calculée, tandis que dans notre cas nous utilisons des Projections prudentes. Notre estimation de la Projection des objets est complètement prudente, elle sur-estime légèrement le rectangle englobant de la véritable Projection (à cause de l'arrondi sur les entiers). De plus, la Projection d'un bloqueur est une sous-estimation de la vue depuis n'importe quel point de la cellule, ce qui réduit les risques de détecter de fausses occultations à cause de l'erreur de discrétisation. Cependant, la résolution de nos Cartes de Profondeur est faible (256×256) et de plus amples analyses devraient être menées.

Il existe des configurations dans lesquelles même un algorithme parfait d'élimination d'occultation (c'est-à-dire un algorithme exact d'élimination des parties cachées) ne peut éliminer suffisamment de géométrie pour parvenir à un rendu en temps réel. Par exemple, si l'observateur se trouve au sommet d'une colline ou sur le toit d'un bâtiment, la ville entière peut être visible. D'autres techniques doivent alors être utilisées avec notre élimination d'occultation, comme nous en discutons dans la section suivante.

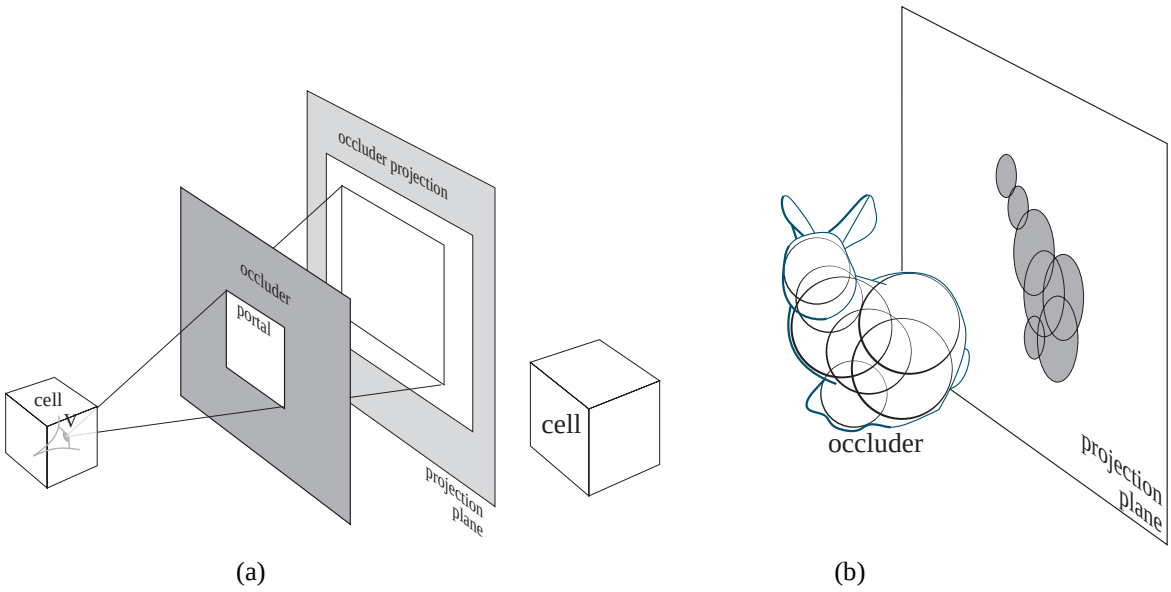
7.3 Travaux futurs

Projection étendue de bloqueurs concaves

La méthode de tranchage que nous avons présentée pour traiter les bloqueurs concaves n'est pas totalement satisfaisante. Elle est restreinte aux objets variétés qui ont effectivement une intersection avec le plan de projection. Nous proposons ici des techniques pour traiter certaines classes de bloqueurs.

Les spécificités des scènes architecturales ont été exploitées pour les calculs de visibilité (*e.g.* [ARB90, Tel92b, TS91]). Des pièces différentes ne sont visibles qu'à travers une séquence de "hublot" (*portals*) comme les portes ou les fenêtres. Les hublots convexes sont en fait les complémentaires de bloqueurs convexes. Leur Projection peut être calculée en considérant le complémentaire de l'union des vues depuis les sommets (figure 5.18(a)).

Des méthodes spécifiques peuvent être implémentées pour certaines classes de bloqueurs convexes, ce qui s'avérera utile dans le paragraphe suivant. Prenons l'exemple d'une sphère. Si l'on place une sphère englobante autour de la cellule, la Projection de la sphère est une ellipse qu'il est facile de calculer.



(a) (b)

FIG. 5.18: (a) La projection étendue d'un mur contenant un "hublot" est l'intersection de ses vues. Elle peut être calculée en prenant le complémentaire de l'union des vues du hublot. (b) Projection étendue d'un bloqueur concave en utilisant une union de bloqueurs convexes (des sphères ici).

Nous proposons d'utiliser des convexes (par exemple une union de sphères, *e.g.* [RF95], comme approximation des bloqueurs concaves (figure 5.18(b)). Le chevauchement de ces convexes est crucial, puisque la fusion de bloqueurs permet alors une estimation efficace de la Projection du concave. Cette méthode devrait permettre de s'affranchir du problèmes des interstices qui apparaissent lorsque chaque triangle d'un maillage connexe est projeté.

Nous présentons une dernière méthode pour régler les problèmes de connexité de la projection d'un maillage polyédrique concave. Il s'agit de simplement projeter le maillage depuis le centre de la cellule, avant d'effectuer un "rétrécissement". Cette idée est illustrée figure 5.19. Les arêtes de la silhouette sont translatées vers le centre de l'objet. Le calcul du vecteur de translation est similaire au calcul du noyau de convolution utilisé pour la reprojection. *A priori*, les triangles au centre de la projection du maillage n'auront pas besoin d'être translatés. Cependant, si un triangle silhouette est trop petit (comme c'est le cas du triangle T_2), la translation de leur arête doit être propagée aux triangles adjacents.

Rendu temps-réel

Notre méthode diminue de manière drastique le temps de rendu, mais elle ne permet pas pour autant de garantir une fréquence d'affichage donnée. Nous proposons ici de l'utiliser au sein des cadres de rendu temps-réel offerts par les travaux antérieurs. Ils sont tous fondés sur la notion de niveaux de détail (LOD) [Cla76] : une représentation simplifiée est utilisée pour les objets distants.

La bibliothèque *Performer* [RH94] utilise une méthode simple de régulation de l'affichage. Une valeur globale de *stress* est utilisée pour mettre à l'échelle un critère de sélection des niveaux de détail basé sur la distance. Une fréquence d'affichage cible est choisie, et si le rendu de la scène prend plus qu'une valeur donnée, disons 95% du temps dévolu à chaque image, la valeur de stress est augmentée, des niveaux de détail plus grossiers sont utilisés, et la fréquence d'affichage est régulée.

Notre précalcul peut être utilisé pour prévoir les soudaines augmentations du nombre de primitives, puisque nous connaissons le nombre d'objets potentiellement visibles depuis la cellule courante et ses voisines. Nous pouvons prédire les variations et nous y adapter de manière plus fluide.

Pour obtenir un rendu temps réel, Funkhouser *et al.* [FS93] résolvent un problème du sac à dos (*knapsack problem*) pour chaque image : un budget de polygones doit être optimalement dépensé, en choisissant le niveau de détail approprié pour chaque objet. Maciel et Shirley [MS95] ont étendu cette approche aux hiérarchies de niveaux de détail. Funkhouser *et al.* [FS93] ont montré qu'un précalcul de visibilité permet de concentrer

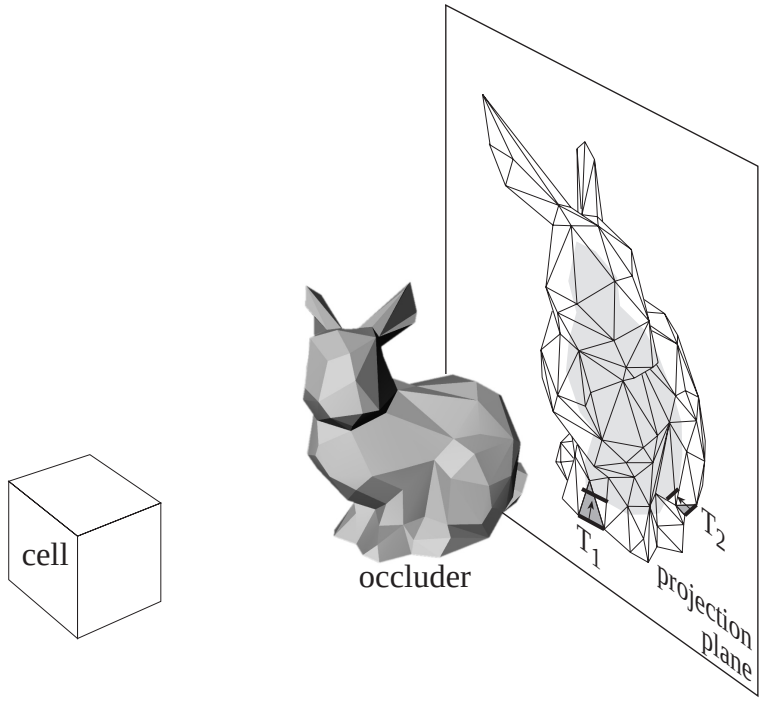


FIG. 5.19: Projection étendue d'un maillage polyédrique concave. Le maillage est projeté depuis le centre de la cellule, puis rétréci. Les arêtes de la silhouette (dans le triangle T_1 par exemple) sont translatées vers le centre du maillage. Si le triangle est trop fin (T_2), la translation doit être propagée aux triangles voisins.

l'optimisation sur les objets qui sont effectivement visibles.

Nous proposons d'utiliser une information semi-quantitative de visibilité pour une meilleure sélection des niveaux de détail. Le *pourcentage de visibilité* d'un objet par rapport à la cellule courante peut être estimé en comptant le nombre de pixels occultés dans sa Projection. Cette information peut ensuite être utilisée comme poids lors de l'optimisation des niveaux de détail, ce qui reviendrait à utiliser des versions plus grossières pour les objets les plus cachés. Cette approche peut également être utilisée avec une régulation par le stress.

Cependant, considérons le cas du même objet à moitié caché par une maison, ou par les feuilles d'un buisson. Dans le premier cas, une grande région connexe de l'objet est visible, tandis que dans le second cas, les feuilles causent de nombreuses occultations de petites régions réparties sur l'objet. Cela entraîne des hautes fréquences qui abaissent la sensibilité du système visuel humain aux artefacts dus à la simplification [FPSG97, PFFG98]. La distribution fréquentielle des pixels occultés dans la Projection doit également être prise en compte, et pas seulement leur nombre.

Pré-chargement depuis le disque ou depuis le réseau

L'un des principaux attraits de notre méthode est sa capacité à traiter les scènes qui doivent être transmises à travers le réseau ou qui ne peuvent être entièrement chargées en mémoire principale. Les techniques développées par Funkhouser *et al.* [Fun96c] peuvent facilement être adaptées. Un programme séparé se charge de la gestion de la base de données. La priorité des objets à charger est évaluée en utilisant les PVS des cellules adjacentes. L'information complète des PVS elle-même ne doit pas résider en mémoire, et doit être chargée à la demande à cause de la place qu'elle requiert. De même, un ordre de priorité doit être établi pour effacer de la mémoire les objets devenus invisibles. La capacité de prédiction offerte par notre méthode est absente des méthodes qui effectuent les calculs d'occultation pour chaque image.

Calculs d'occultation en ligne

L'approche que nous avons présentée peut être adaptée pour les calculs de visibilité en ligne ou à la demande, afin d'inclure une capacité de prédiction. La convolution (ou l'érosion [SD95]) peut être utilisée pour

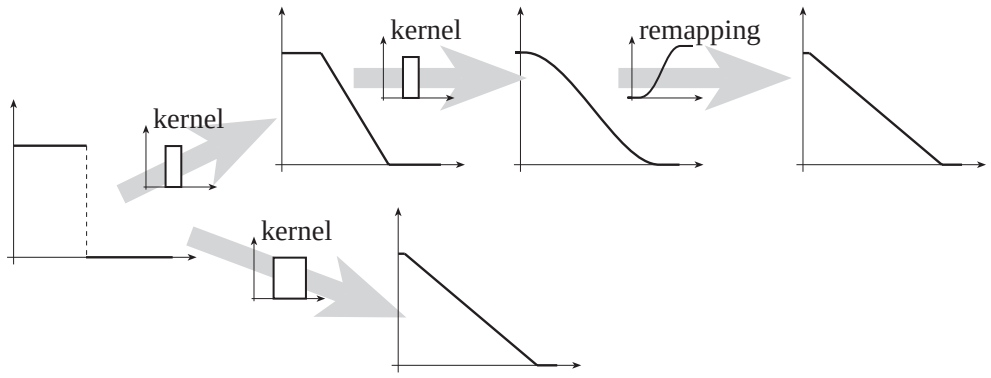


FIG. 5.20: Convolutions successives et correction. Effectuer deux convolutions n'est pas équivalent à une convolution avec un noyau de taille double.

obtenir une approximation (éventuellement non-prudente) des Projections des bloqueurs dans le voisinage du point de vue courant. De la même façon, la Projection des objets peut être estimée en grossissant le rectangle englobant leur projection depuis le point de vue courant.

Cela permettrait un calcul d'occultation en ligne qui soit valable pour plus d'une image, et qui pourrait donc être utilisé pour la gestion du stress ou pour le pré-chargement.

Notre méthode pourrait aussi être utilisée à la demande. Le temps de calcul étant de 4 secondes par cellules, si l'observateur se déplace lentement, alors l'information de visibilité peut être calculée pour les cellules voisines au fur et à mesure qu'il se balade.

Autres questions

La résolution des Cartes de Profondeur ou d'Occultation demeure un problème important. La méthode de translation des arêtes de Wonka *et al.* [WS99] devrait être implémentée pour obtenir des calculs véritablement prudents.

Le balayage d'occultation pourrait être adapté au calcul d'ombres douces. Cependant, il n'est pas équivalent d'effectuer deux convolutions et d'effectuer une convolution avec un noyau de taille double. Les tailles des zones d'ombre et de pénombre sont préservées, mais les valeurs à l'intérieur de la pénombre sont modifiées. Une correction (*remapping*) doit être appliquée après chaque convolution (figure 5.20). Après deux convolutions, on ne peut malheureusement plus garantir les valeurs.

Notre précalcul peut être utilisé pour la simulation de l'éclairage, en suivant l'exemple de Teller *et al.* [TH93, TFFH94, Fun96b].

L'information de visibilité pourrait aussi être très utile pour les niveaux de détail en animation [CH97, CF97] qui en sont toujours à leurs balbutiements. La précision d'une simulation physique ou d'une animation générative peut être abaissée dans les régions cachées.

Pour finir, le problème de la compression des PVS reste important. Le codage d'entropie, le codage hiérarchique ou la quantification de vecteur prudente sont des approches possibles.

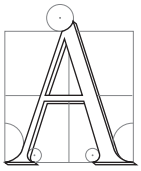
Deuxième partie

A Multidisciplinary Survey of Visibility

CHAPITRE 6

Introduction

> Il déduisit que la bibliothèque est totale, et que ses étagères consignent toutes les combinaisons possibles des vingt et quelques symboles orthographiques (nombre quoique très vaste, non infini), c'est à dire tout ce qu'il est possible d'exprimer dans toutes les langues.
>
> Jorge Luis BORGES, *La bibliothèque de Babel*



A VAST AMOUNT OF WORK has been published about visibility in many different domains. Inspiration has sometimes traveled from one community to another, but work and publications have mainly remained restricted to their specific field. The differences of terminology and interest together with the obvious difficulty of reading and remaining informed of the cumulative literature of different fields have obstructed the transmission of knowledge between communities. This is unfortunate because the different points of view adopted by different domains offer a wide range of solutions to visibility problems. Though some surveys exist about certain specific aspects of visibility, no global overview has gathered and compared the answers found in those domains. The second part of this thesis is an attempt to fill this vacuum. We hope that it will be useful to students beginning work on visibility, as well as to researchers in one field who are interested in solutions offered by other domains. We also hope that this survey will be an opportunity to consider visibility questions under a new perspective.

1 Spirit of the survey

This survey is more a "horizontal" survey than a "vertical" survey. Our purpose is not to precisely compare the methods developed in a very specific field ; our aim is to give an overview which is as wide as possible.

We also want to avoid a catalogue of visibility methods developed in each domain : Synthesis and comparison are sought. However, we believe that it is important to understand the specificities of visibility problems as encountered in each field. This is why we begin this survey with an overview of the visibility questions as they arise field by field. We will then present the solutions proposed, using a classification which is not based on the field in which they have been published.

Our classification is only an analysis and organisation tool ; as any classification, it does not offer infallible nor strict categories. A method can gather techniques from different categories, requiring the presentation of a single paper in several chapters. We however attempt to avoid this, but when necessary it will be indicated with cross-references.

We have chosen to develop certain techniques with more details not to remain too abstract. A section in general presents a paradigmatic method which illustrates a category. It is then followed by a shorter description of related methods, focusing on their differences with the first one.

We have chosen to mix low-level visibility acceleration schemes as well as high-level methods which make use of visibility. We have also chosen not to separate exact and approximate methods, because in many cases approximate methods are “degraded” or simplified versions of exact algorithms.

In the footnotes, we propose some thoughts or references which are slightly beyond the scope of this survey. They can be skipped without missing crucial information.

2 Flaws and bias

This survey is obviously far from complete. A strong bias towards computer graphics is clearly apparent, both in the terminology and number of references.

Computational geometry is insufficiently treated. In particular, the relations between visibility queries and range-searching would deserve a large exposition. 2D visibility graph construction is also treated very briefly.

Similarly, few complexity bounds are given in this survey. One reason is that theoretical bounds are not always relevant to the analysis of the practical behaviour of algorithms with “typical” scenes. Practical timings and memory storage would be an interesting information to complete theoretical bounds. This is however tedious and involved since different machines and scenes or objects are used, making the comparison intricate, and practical results are not always given. Nevertheless, this survey could undoubtedly be augmented with some theoretical bounds and statistics.

Terrain (or height field) visibility is nearly absent of our overview, even though it is an important topic, especially for Geographical Information Systems (*GIS*) where visibility is used for display, but also to optimize the placement of fire towers. We refer the interested reader to the survey by de Floriani *et al.* [FPM98].

The work in computer vision dedicated to the acquisition or recognition of shapes from shadows is also absent from this survey. See *e.g.* [Wal75, KB98].

The problem of aliasing is crucial in many computer graphics situations. It is a large subject by itself, and would deserve an entire survey. It is however not strictly a visibility problem, but we attempt to give some references.

Neither practical answers nor advice are directly provided. The reader who reads this survey with the question “what should I use to solve my problem” in mind will not find a direct answer. A practical guide to visibility calculation would unquestionably be a very valuable contribution. We nonetheless hope that the reader will find some hints and introductions to relevant techniques.

3 Structure

This survey is organised as follows. Chapter 7 introduces the problems in which visibility computations occur, field by field. In chapter 8 we introduce some preliminary notions which will we use to analyze and classify the methods in the following chapters. In chapter 9 we survey the classics of hidden-part removal. The following chapters present visibility methods according to the space in which the computations are performed : chapter 10 deals with object space, chapter 11 with image-space, chapter 12 with viewpoint-space and finally chapter 13 treats line-space methods. Chapter 14 presents advanced issues : managing precision and dealing with moving objects. Chapter 15 concludes with a discussion..

In appendix E we also give a short list of resources related to visibility which are available on the web. An index of the important terms used in this survey can be found at the end of this thesis. Finally, the references are annotated with the pages at which they are cited.

CHAPITRE 7

Visibility problems

S'il n'y a pas de solution, c'est qu'il n'y a pas de problème

LES SHADOKS

ISIBILITY PROBLEMS arise in many different contexts in various fields. In this section we review the situations in which visibility computations are involved. The algorithms and data-structures which have been developed will be surveyed later to distinguish the classification of the methods from the context in which they have been developed. We review visibility in computer graphics, then computer vision, robotics and computational geometry. We conclude this chapter with a summary of the visibility queries involved.

1 Computer Graphics

For a good introduction on standard computer graphics techniques, we refer the reader to the excellent book by Foley *et al.* [FvDFH90] or the one by Rogers [Rog97]. More advanced topics are covered in [WW92].

1.1 Hidden surface removal

View computation has been the major focus of early computer graphics research. Visibility was a synonym for the determination of the parts/polygons/lines of the scene visible from a viewpoint. It is beyond the scope of this survey to review the huge number of techniques which have been developed over the years. We however review the great classics in section 9. The interested reader will find a comprehensive introduction to most of the algorithms in [FvDFH90, Rog97]. The classical survey by Sutherland *et al.* [SSS74] still provides a good classification of the techniques of the mid seventies, a more modern version being the thesis of Grant [Gra92]. More theoretical and computational geometry methods are surveyed in [Dor94, Ber93]. Some aspects are also covered in section 4.1. For the specific topic of real time display for flight simulators, see the overview by Mueller [Mue95].

The interest in hidden-part removal algorithms has been renewed by the recent domain of *non-photorealistic rendering*, that is the generation of images which do not attempt to mimic reality, such as cartoons, technical

illustrations or paintings [MKT+97, WS94]. Some information which are more topological are required such as the visible silhouette of the objects or its connected visible areas.

View computation will be covered in chapter 9 and section 1.4 of chapter 10.

1.2 Shadow computation

The efficient and robust computation of shadows is still one of the challenges of computer graphics. Shadows are essential for any realistic rendering of a 3D scene and provide important clues about the relative positions of objects[1]. The drawings by da Vinci in his project of a *treatise on painting* or the construction by Lambert in *Freye Perspective* give evidence of the old interest in shadow computation (Fig. 7.1). See also the book by Baxandall [Bax95] which presents very interesting insights on shadows in painting, physics and computer science.

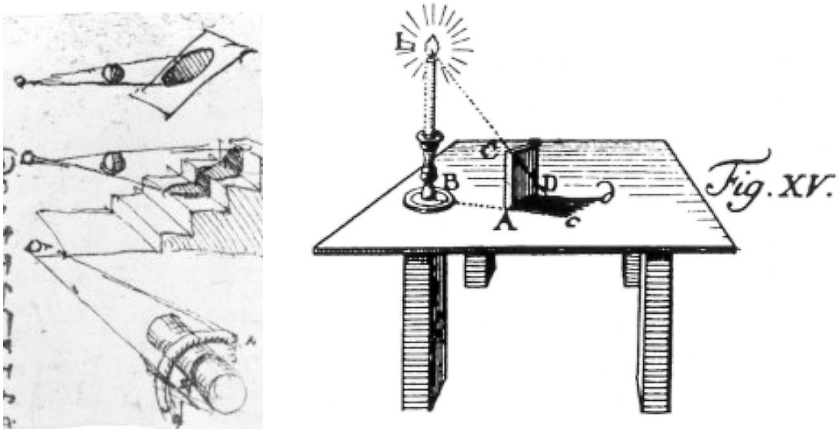


FIG. 7.1: (a) Study of shadows by Leonardo da Vinci (Manuscript *Codex Urbinas*). (a) Shadow construction by Johann Heinrich Lambert (*Freye Perspective*).

Hard shadows are caused by point or directional light sources. They are easier to compute because a point of the scene is either in full light or is completely hidden from the source. The computation of hard shadows is conceptually similar to the computation of a view from the light source, followed by a reprojection. It is however both simpler and much more involved. Simpler because a point is in shadow if it is hidden from the source by any object of the scene, no matter which is the closest. Much more involved because if reprojection is actually used, it is not trivial by itself, and intricate sampling or field of view problems appear.

Soft shadows are caused by line or area light sources. A point can see all, part, or nothing of such a source, defining the regions of total lighting, penumbra and umbra. The size of the zone of penumbra varies depending on the relative distances between the source, the blocker and the receiver (see Fig. 7.2). A single view from the light is not sufficient for their computation, explaining its difficulty.

An extensive article exists [WPF90] which surveys all the standard shadows computation techniques up to 1990.

Shadow computations will be treated in chapter 10 (section 4.1, 4.2, 4.4 and 5), chapter 11 (section 2.1 , 6 and 7) and chapter 12 (section 2.3 and 2.4).

The inverse problem has received little attention : a user imposes a shadow location, and a light position is deduced. It will be treated in section 5.6 of chapter 10. This problem can be thought as the dual of sensor placement or good viewpoint computation that we will introduce in section 2.3.

1.3 Occlusion culling

The complexity of 3D scenes to display becomes larger and larger, and can not be rendered at interactive rates, even on high-end workstations. This is particularly true for applications such as CAD/CAM where the da-

[1] The influence of the quality of shadows on the perception of the spatial relationships is however still a controversial topic. see *e.g.* [Wan92, KKMB96]

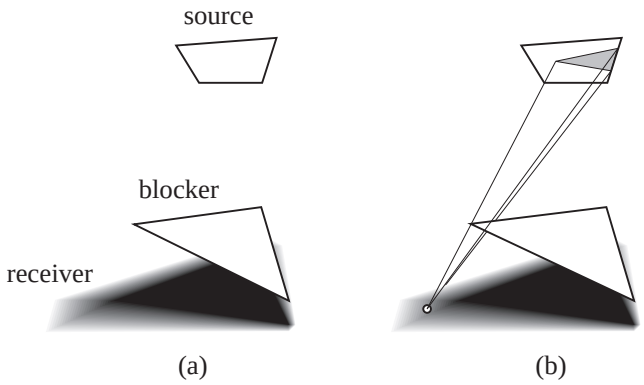


FIG. 7.2: (a) Example of a soft shadow. Notice that the size of the zone of penumbra depends on the mutual distances (the penumbra is wider on the left). (b) Part of the source seen from a point in penumbra.

tabases are often composed of millions of primitives, and also in driving/flight simulators, and in walkthroughs where a users want to walk through virtual buildings or even cities.

Occlusion culling (also called *visibility culling*) attempts to quickly discard the hidden geometry, by computing a superset of the visible geometry which will be sent to the graphics hardware. For example, in a city, the objects behind the nearby facades can be "obviously" rejected.

An occlusion culling algorithm has to be *conservative*. It may declare potentially visible an object which is in fact actually hidden, since a standard view computation method will be used to finally display the image (typically a z-buffer [FvDFH90]).

A distinction can be made between *online* and *offline* techniques. In an online occlusion culling method, for each frame the objects which are obviously hidden are rejected on the fly. While offline Occlusion culling precomputations consist in subdividing the scene into cells and computing for each cell the objects which may be visible from inside the cell. This set of visible object is often called the *potentially visible sets* of the cell. At display time, only the objects in the potentially visible set of the current cell are sent to the graphics hardware[2].

The landmark paper on the subject is by Clark in 1976 [Cla76] where he introduces most of the concepts for efficient rendering. The more recent paper by Heckbert and Garland [HG94] gives a good introduction to the different approaches for fast rendering. Occlusion culling techniques are treated in chapter 10 (section 4.4, 6.3 and 7), chapter 11 (section 3 and 4), chapter 12 (section 4) and chapter 13 (section 1.5).

1.4 Global Illumination

Global illumination deals with the simulation of light based on the laws of physics, and particularly with the interactions between objects. Light may be blocked by objects causing shadows. Mirrors reflect light along the symmetric direction with respect to the surface normal (Fig. 7.3(a)). Light arriving at a *diffuse* (or lambertian) object is reflected equally in all directions (Fig. 7.3(b)). More generally, a function called *BRDF* (Bidirectional Reflection Distribution Function) models the way light arriving at a surface is reflected (Fig. 7.3(c)). Fig 7.4 illustrates some bounces of light through a scene.

Kajiya has formalised global illumination with the *rendering equation* [Kaj86]. Light traveling through a point in a given direction depends on all the incident light, that is, it depends on the light coming from all the points which are visible. Its solution thus involves massive visibility computations which can be seen as the equivalent of computing a view from each point of the scene with respect to every other.

The interested reader will find a complete presentation in the books on the subject [CW93b, SP94, Gla95].

Global illumination method can also be applied to the simulation of sound propagation. See the book by Kutruff [Kut91] or [Dal96, FCE+98]. See section 4.3 of chapter 10. Sound however differs from light because

[2]Occlusion-culling techniques are also used to decrease the amount of communication in multi-user virtual environments : messages and updates are sent between users only if they can see each other [Fun95, Fun96a, CT97a, MGBY99]. If the scene is too big to fit in memory, or if it is downloaded from the network, occlusion culling can be used to load into memory (or from the network) only the part of the geometry which may be visible [Fun96c, COZ98].

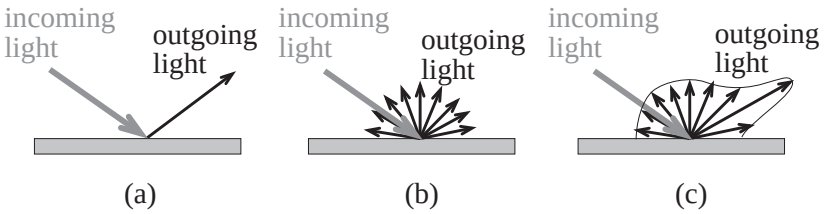


FIG. 7.3: Light reflection for a given incidence angle. (a) Perfect mirror reflection. (b) Diffuse reflection. (c) General bidirectional reflectance distribution function (BRDF).

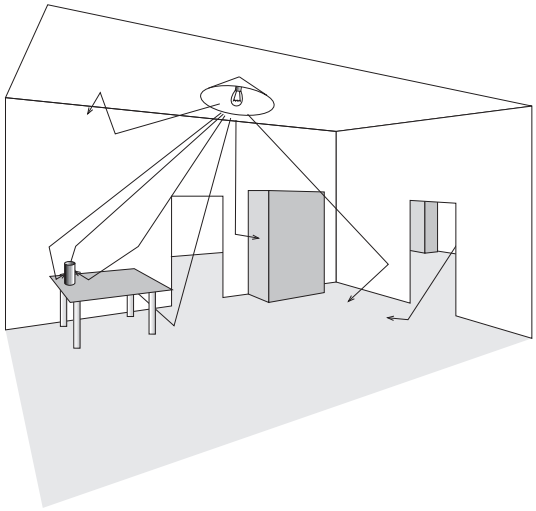


FIG. 7.4: Global illumination. We show some paths of light : light emanating from light sources bounces on the surfaces of the scene (We show only one outgoing ray at each bounce, but light is generally reflected in all direction as modeled by a BRDF).

the involved wavelength are longer. Diffraction effects have to be taken into account and binary straight-line visibility is a too simplistic model. This topic will be covered in section 2.4 of chapter 11.

In the two sections below we introduce the global illumination methods based on ray-tracing and finite elements.

1.5 Ray-tracing and Monte-Carlo techniques

Whitted [Whi80] has extended the ray-casting developed by Appel [App68] and introduced recursive *ray-tracing* to compute the effect of reflecting and refracting objects as well as shadows. A ray is simulated from the viewpoint to each of the pixels of the image. It is intersected with the objects of the scene to compute the closest point. From this point, *shadow rays* can be sent to the sources to detect shadows, and reflecting or refracting rays can be sent in the appropriate direction in a recursive manner (see Fig. 7.5). A complete presentation of ray-tracing can be found on the book by Glassner [Gla89] and an electronic publication is dedicated to the subject [Hai]. A comprehensive index of related paper has been written by Speer [Spe92a]

More complete global illumination simulations have been developed based on the Monte-Carlo integration framework and the aforementioned rendering equation. They are based on a probabilistic sampling of the illumination, requiring to send even more rays. At each intersection point some rays are stochastically sent to sample the illumination, not only in the mirror and refraction directions. The process then continues recursively. It can model any BRDF and any lighting effect, but may be noisy because of the sampling.

Those techniques are called *view dependent* because the computations are done for a unique viewpoint. Veach's thesis [Vea97] presents a very good introduction to Monte-Carlo techniques.

The atomic and most costly operation in ray-tracing and Monte-Carlo techniques consists in computing the first object hit by a ray, or in the case of rays cast for shadows, to determine if the ray intersects an object. Many

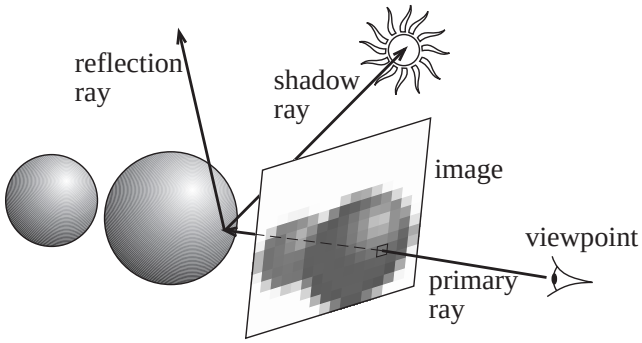


FIG. 7.5: Principle of recursive ray-tracing. Primary rays are sent from the viewpoint to detect the visible object. Shadow rays are sent to the source to detect occlusion (shadow). Reflection rays can be sent in the mirror direction.

acceleration schemes have thus been developed over the two last decades. A very good introduction to most of these techniques has been written by Arvo and Kirk [AK89].

Ray-shooting will be treated in chapter 10 (section 1 and 4.3), chapter 11 (section 2.2), chapter 13 (section 1.4 and 3) and chapter 14 (section 2.2).

1.6 Radiosity

Radiosity methods have first been developed in the heat transfer community (see *e.g.* [Bre92]) and then adapted and extended for light simulation purposes. They assume that the objects of the scene are completely diffuse (incoming light is reflected equally in all directions of the hemisphere), which may be reasonable for architectural scene. The geometry of the scene is subdivided into patches, over which radiosity is usually assumed constant (Fig. 7.6). The light exchanges between all pairs of patches are simulated. The *form factor* between patches A and B is the proportion of light leaving A which reaches B, taking occlusions into account. The radiosity problem then resumes to a huge system of linear equations, which can be solved iteratively. Formally, radiosity is a finite element method. Since lighting is assumed directionally invariant, radiosity methods provide *view independent* solutions, and a user can interactively walk through a scene with global illumination effects. A couple of books are dedicated to radiosity methods [SP94, CW93b, Ash94].

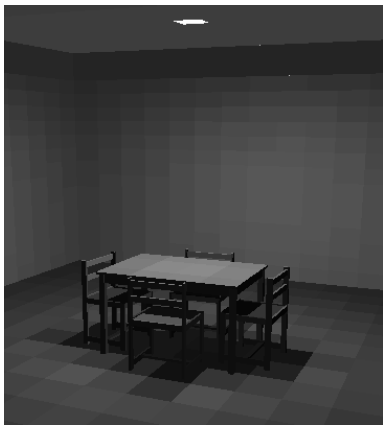


FIG. 7.6: Radiosity methods simulate diffuse interreflexions. Note how the subdivision of the geometry is apparent. Smoothing is usually used to alleviate most of these artifacts.

Form factor computation is the costliest part of radiosity methods, because of the intensive visibility computations they require [HSD94]. An intricate formula has been derived by Schroeder and Hanrahan [SH93] for the form factor between two polygons in full visibility, but no analytical solution is known for the partially occluded case.

Form factor computation will be treated in chapter 9 (section 2.2), chapter 10 (section 6.1 and 7), in chapter 11 (section 2.3), chapter 12 (section 2.3), chapter 13 (section 2.1) and chapter 14 (section 2.1).

Radiosity needs a subdivision of the scene, which is usually grid-like : a quadtree is adaptively refined in the regions where lighting varies, typically the limits of shadows. To obtain a better representation, *discontinuity meshing* has been introduced. It tries to subdivides the geometry of the scene along the discontinuities of the lighting function, that is, the limits of shadows.

Discontinuity meshing methods are presented in chapter 10 (section 5.3), chapter 12 (section 2.3 and 2.4), chapter 13 (section 2.1) and chapter 14 (section 1.3, 1.5 and 2.4) [3].

1.7 Image-based modeling and rendering

3D models are hard and slow to produce, and if realism is sought the number of required primitives is so huge that the models become very costly to render. The recent domain of *image-based rendering and modeling* copes with this through the use of image complexity which replaces geometric complexity. It uses some techniques from computer vision and computer graphics. Texture-mapping can be seen as a precursor of image-based techniques, since it improves the appearance of 3D scenes by projecting some images on the objects.

View warping [CW93a] permits the reprojection of an image with depth values from a given viewpoint to a new one. Each pixel of the image is reprojected using its depth and the two camera geometries as shown in Fig. 7.7. It permits re-rendering of images at a cost which is independent of the 3D scene complexity. However, sampling questions arise, and above all, gaps appear where objects which were hidden in the original view become visible. The use of multiple base images can help solve this problem, but imposes a decision on how to combine the images, and especially to detect where visibility problems occur.

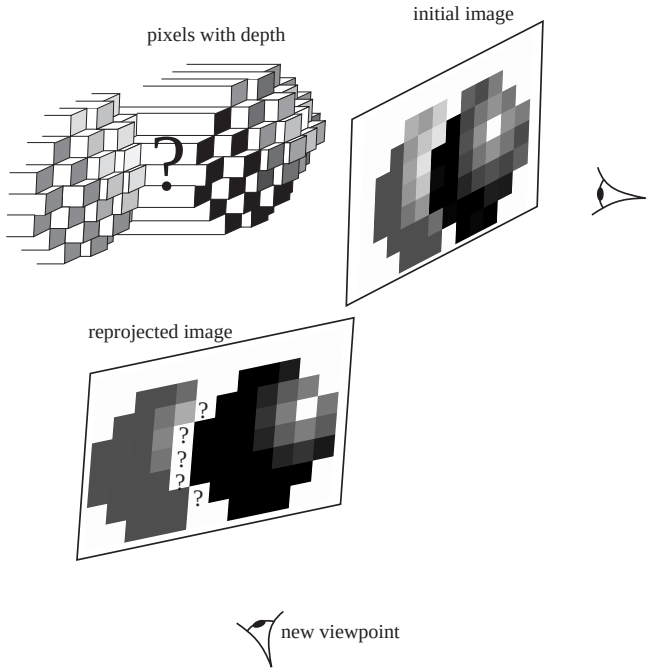


FIG. 7.7: View warping. The pixels from the initial image are reprojected using the depth information. However, some gaps due to indeterminate visibility may appear (represented as " ?" in the reprojected image)

Image-based modeling techniques take as input a set of photographs, and allow the scene to be seen from new viewpoints. Some authors use the photographs to help the construction of a textured 3D model [DTM96]. Other try to recover the depth or disparity using stereo vision [LF94, MB95]. Image warping then allows the computation of images from new viewpoints. The quality of the new images depends on the relevance of the

[3] Recent approaches have improved radiosity methods through the use of non constant bases and hierarchical representations, but the cost of form factor computation and the meshing artifact remain. Some non-diffuse radiosity computations have also been proposed at a usually very high cost. For a short discussion of the usability of radiosity, see the talk by Sillion [Sil99].

base images. A good set of cameras should be chosen to sample the scene accurately, and especially to avoid that some parts of the scene are not acquired because of occlusion.

Some image-based rendering methods have also been proposed to speedup rendering. They do not require the whole 3D scene to be redrawn for each frame. Instead, the 2D images of some parts of the scene are cached and reused for a number of frames with simple transformation (2D rotation and translation [LS97], or texture mapping on flat [SLSD96, SS96a] or simplified [SDB97] geometry). These image-caches can be organised as *layers*, and for proper occlusion and parallax effects, these layers have to be wisely organised, which has reintroduced the problem of depth ordering.

These topics will be covered in chapter 9 (section 4.3), chapter 10 (section 4.5), chapter 11 (section 5) and chapter 13 (section 1.5).

1.8 Good viewpoint selection

In production animation, the camera is placed by skilled artists. For others applications such as games, teleconference or 3D manipulation, its position is also very important to permit a good view of the scene and the understanding of the spatial positions of the objects.

This requires the development of methods which automatically optimize the viewpoint. Visibility is one of the criteria, but one can also devise other requirements to convey a particular ambiance [PBG92, DZ95, HCS96].

The visual representation of a graph (graph drawing) in 3D raises similar issues, the number of visual alignments should be minimized. See section 1.5 of chapter 12.

We will see in section 2.3 that the placement of computer vision offers similar problems. The corresponding techniques are surveyed in chapter 10 (section 4.5 and 5.5) and chapter 12 (section 3).

2 Computer Vision

An introduction and case study of many computer vision topics can be found in the book by Faugeras [Fau93] or the survey by Guerra [Gue98]. The classic by Ballard and Brown [BB82] is more oriented towards image processing techniques for vision.

2.1 Model-based object recognition

The task of object recognition assumes a database of objects is known, and given an image, it reports if the objects are present and in which position. We are interested in model-based recognition of 3D objects, where the knowledge of the object is composed of an explicit model of its shape. It first involves low-level computer vision techniques for the extraction of features such as edges. Then these features have to be compared with corresponding features of the objects. The most convenient representations of the objects for this task represent the possible views of the object (*viewer centered* representation) rather than its 3D shape (*object-centered* representation). These views can be compared with the image more easily (2D to 2D matching as opposed to 3D to 2D matching). Fig. 7.8 illustrates a model-based recognition process.

One thus needs a data-structure which is able to efficiently represent all the possible views of an object. Occlusion has to be taken into account, and views have to be grouped according to their similarities. A class of similar views is usually called an *aspect* . A good viewer-centered representation should be able to *a priori* identify all the possible different views of an object, detecting “where” the similarity between nearby views is broken.

Psychological studies have shown evidences that the human visual system possesses such a viewer-centered representation, since objects are more easily recognised when viewed under specific viewpoints [Ull89, EB92].

A recent survey exists [Pop94] which reviews results on all the aspects of object recognition. See also the book by Jain and Flynn [JF93] and the survey by Crevier and Lepage [CL97]

Object recognition has led to the development of one of the major visibility data structures, the *aspect graph*[4] which will be treated in sections 1 of chapter 12 and section 1.4 and 2.4 of chapter 14.

[4]However viewer centered representation now seem superseded by the use of geometric properties which are invariant by some geometric transformation (affine or perspective). These geometric *invariants* can be used to guide the recognition of objects [MZ92, Wei93].

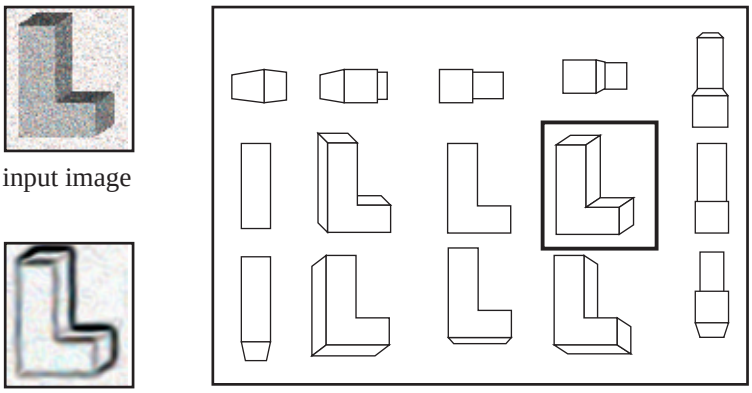


FIG. 7.8: Model-based object recognition. Features are extracted from the input image and matched against the viewer-centered representation of an L-shaped object.

2.2 Object reconstruction by contour intersection

Object reconstruction takes as input a set of images to compute a 3D model. We do not treat here the reconstruction of volumetric data from slices obtained with medical equipment since it does not involve visibility.

We are interested in the reconstruction process based on contour intersection. Consider a view, from which the contour of the object has been extracted. The object is constrained to lie inside the cone defined by the viewpoint and this contour. If many images are considered, the cones can be intersected and a model of the object is estimated [SLH89]. The process is illustrated in Fig. 7.9. This method is very robust and easy to implement especially if the intersections are computed using a volumetric model by removing voxels in an octree [Pot87].

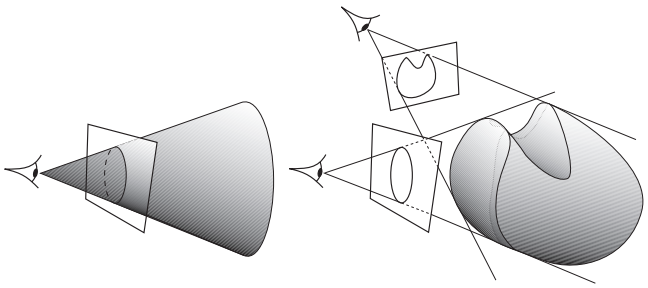


FIG. 7.9: Object reconstruction by contour intersection. The contour in each view defines a general cone in which the object is constrained. A model of the object is built using the intersection of the cones. (a) Cone resulting from one image. (b) Intersection of cones from two images.

However, how close is this model to the actual object ? Which class of objects can be reconstructed using this technique ? If an object can be reconstructed, how many views are needed ? This of course depends on self-occlusion. For example, the cavity in a bowl can never be reconstructed using this technique if the camera is constrained outside the object. The analysis of these questions imposes involved visibility considerations, as will be shown in section 3 of chapter 10.

2.3 Sensor placement for known geometry

Computer vision tasks imply the acquisition of data using any sort of sensor. The position of the sensor can have dramatic effects on the quality and efficiency of the vision task which is then processed. *Active vision* deals with the computation of efficient placement of the sensors. It is also referred to as *viewpoint planning*.

In some cases, the geometry of the environment is known and the sensor position(s) can be preprocessed. It is particularly the case for robotics applications where the same task has to be performed on many avatars of the same object for which a CAD geometrical model is known.

The sensor(s) can be mobile, for example placed on a robot arm, it is the so called “camera in hand”. One can also want to design a fixed system which will be used to inspect a lot of similar objects.

An example of sensor planning is the monitoring of a robot task like assembly. Precise absolute positioning is rarely possible, because registration can not always be performed, the controllers used drift over time and the object on which the task is performed may not be accurately modeled or may be slightly misplaced [HKL98, MI98]. Uncertainties and tolerances impose the use of sensors to monitor the robot Fig. 7.10 and 7.11 show examples of sensor controlled task. It has to be placed such that the task to be performed is visible. This principally requires the computation of the regions of space from which a particular region is not hidden. The tutorial by Hutchinson *et al.* [HH96] gives a comprehensive introduction to the visual control of robots.

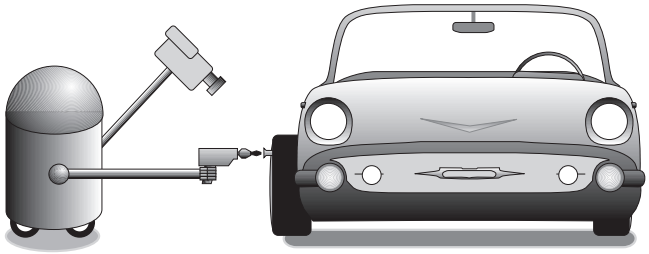


FIG. 7.10: The screwdriver must be placed very precisely in front of the screw. The task is thus controlled by a camera.

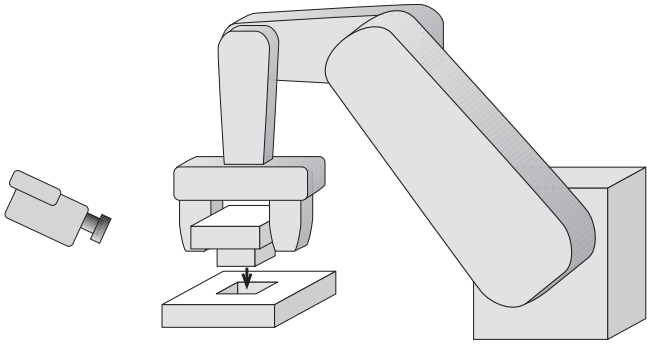


FIG. 7.11: The insertion of this peg into the hole has to be performed very precisely, under the control of a sensor which has to be carefully placed.

Another example is the inspection of a manufactured part for quality verification. Measurements can for example be performed by triangulation using multiple sensors. If the geometry of the sensors is known, the position of a feature projecting on a point in the image from a given sensor is constrained on the line going through the sensor center and the point in the image. With multiple images, the 3D position of the feature is computed by intersecting the corresponding lines. Better precision is obtained for 3 views with orthogonal directions. The sensors have to be placed such that each feature to be measured is visible in at least two images. Visibility is a crucial criterion, but surface orientation and image resolution are also very important.

The illumination of the object can also be optimized. One can require that the part to be inspected be well illuminated. One can maximize the contrast to make important features easily recognisable. The optimization of viewpoint and illumination together of course leads to the best results but has a higher complexity.

See the survey by Roberts and Marshall [RM97] and by Tarabanis *et al.* [TAT95]. Section 5.5 of chapter 10 and section 3 of chapter 12 deal with the computation of good viewpoints for known environment.

2.4 Good viewpoints for object exploration

Computer vision methods have been developed to acquire a 3D model of an unknown object. The choice of the sequence of sensing operations greatly affects the quality of the results, and active vision techniques are

required.

We have already reviewed the contour intersection method. We have evoked only the theoretical limits of the method, but an infinite number of views can not be used ! The choice of the views to be used thus has to be carefully performed as function of the already acquired data.

Another model acquisition technique uses a laser plane and a camera. The laser illuminates the object along a plane (the laser beam is quickly rotated over time to generate a plane). A camera placed at a certain distance of the laser records the image of the object, where the illumination by the laser is visible as a slice (see Fig. 7.12). If the geometry of the plane and camera is known, triangulation can be used to infer the coordinates of the illuminated slice of the object. Translating the laser plane permits the acquisition of the whole model. The data acquired with such a system are called *range images*, that is, an image from the camera location which provides the depth of the points.

Two kinds of occlusion occur with these system : some part of an illuminated slice may not be visible to the camera, and some part of the object can be hidden to the laser, as shown in Fig. 7.12.

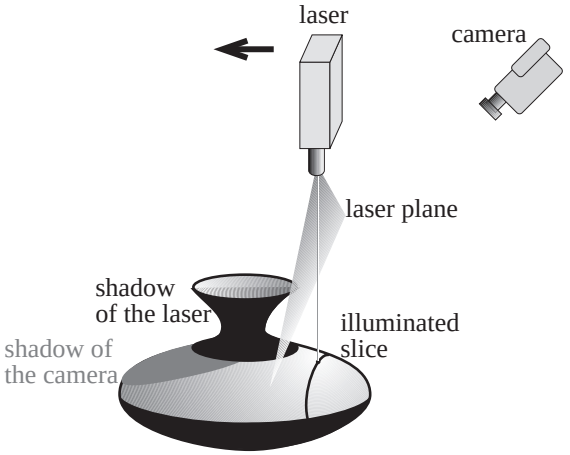


FIG. 7.12: Object acquisition using a laser plane. The laser emits a plane, and the intersection between this plane and the object is acquired by a camera. The geometry of the slice can then be easily deduced. The laser and camera translate to acquire the whole object. Occlusion with respect to the laser plane (in black) and to the camera (in grey) have to be taken into account.

These problems are referred to as *best-next-view* or *purposive viewpoint adjustment*. The next viewpoint has to be computed and optimized using the data already acquired. Previously occluded parts have to be explored.

The general problems of active vision are discussed in the report written after the *1991 Active Vision Workshop* [AAA+92]. An overview of the corresponding visibility techniques is given in [RM97, TAT95] and they will be discussed in section 4.5 of chapter 10.

3 Robotics

A comprehensive overview of the problems and specificities of robotics research can be found in [HKL98]. A more geometrical point of view is exposed in [HKL97]. The book by Latombe [Lat91] gives a complete and comprehensive presentation of motion planning techniques.

A lot of the robotics techniques that we will discuss treat only 2D scenes. This restriction is quite understandable because a lot of mobile robots are only allowed to move on a 2D floorplan.

As we have seen, robotics and computer vision share a lot of topics and our classification to one or the other specialty is sometimes arbitrary.

3.1 Motion planning

A robot has a certain number of degrees of freedom. A variable can be assigned to each degree of freedom, defining a (usually multidimensional) *configuration space*. For example a two joint robot has 4 degrees of

freedom, 2 for each joint orientation. A circular robot allowed to move on a plane has two degrees of freedom if its orientation does not have to be taken into account. Motion planning [Lat91] consists in finding a path from a start position of the robot to a goal position, while avoiding collision with obstacles and respecting some optional additional constraints. The optimality of this path can also be required.

The case of articulated robots is particularly involved because they move in high dimensional configuration spaces. We are interested here in robots allowed to translate in 2D euclidean space, for which orientation is not considered. In this case the motion planning problem resumes to the motion planning for a point, by "growing" the obstacles using the Minkovski sum between the robot shape and the obstacles, as illustrated in Fig. 7.13.

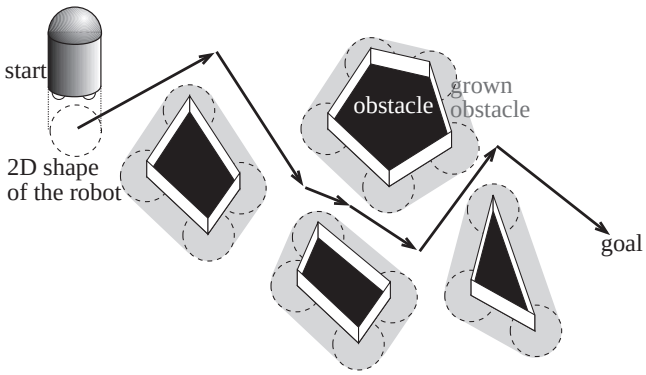


FIG. 7.13: Motion planning on a floorplan. The obstacles are grown using the Minkovski sum with the shape of the robot. The motion planning of the robot in the non-grown scene resumes to that of its centerpoint in the grown scene.

The relation between euclidean motion planning and visibility comes from this simple fact : A point robot can move in straight line only to the points of the scene which are visible from it.

We will see in Section 2 of chapter 10 that one of the first global visibility data structure, the *visibility graph* was developed for motion planning purposes. [5]

3.2 Visibility based pursuit-evasion

Recently motion planning has been extended to the case where a robot searches for an intruder with arbitrary motion in a known 2D environment. A mobile robot with 360° field of view explores the scene, "cleaning" zones. A zone is cleaned when the robot sees it and can verify that no intruder is in it. It remains clean if no intruder can go there from an uncleaned region without being seen. If all the scene is cleaned, no intruder can have been missed. Fig. 7.14 shows an example of a robot strategy to clean a simple 2D polygon.

If the environment contains a "column" (that is topologically a hole), it can not be cleaned by a single robot since the intruder can always hide behind the column.

Extensions to this problem include the optimization of the path of the robot, the coordination of multiple robots, and the treatment of sensor limitations such as limited range or field of view.

Pursuit evasion is somehow related to the art-gallery problem which we will present in section 4.3. A technique to solve this pursuit-evasion problem will be treated in section 2.2 of chapter 12.

A related problem is the tracking of a mobile target while maintaining visibility. A target is moving in a known 2D environment, and its motion can have different degrees of predictability (completely known motion, bound on the velocity). A strategy is required for a mobile tracking robot such that visibility with the target is never lost. A perfect strategy can not always be designed, and one can require that the probability to lose the target be minimal. See section 3.3 of chapter 12.

[5] Assembly planning is another thematic of robotics where the ways to assemble or de-assemble an object are searched [HKL98]. The relationship between these problems and visibility would deserve exploration, especially the relation between the possibility to translate a part and the visibility of the hole in which it has to be placed.

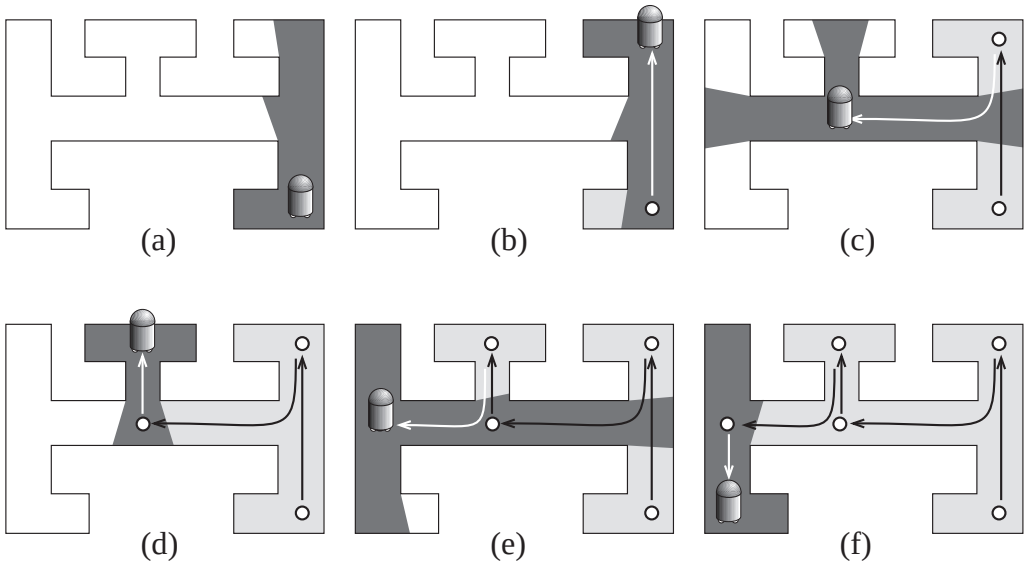


FIG. 7.14: The robot has to search for an unknown intruder. The part of the scene visible from the robot is in dark grey, while the "cleaned" zone is in light grey. At no moment can an intruder go from the unknown region to the cleaned region without being seen by the robot.

3.3 Self-localisation

A mobile robot often has to be localised in its environment. The robot can therefore be equipped with sensor to help it determine its position if the environment is known. Once data have been acquired, for example in the form of a range image, the robot has to infer its position from the view of the environment as shown in Fig. 7.15. See the work by Drumheller [Dru87] for a classic method.

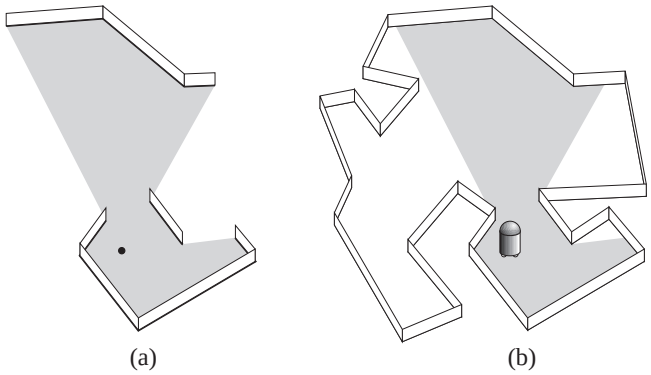


FIG. 7.15: 2D Robot localisation. (a) View from the robot. (b) Deduced location of the robot.

This problem is in fact very similar to the recognition problem studied in computer vision. The robot has to "recognise" its view of the environment. We will see in section 2.1 of chapter 12 that the approaches developed are very similar.

4 Computational Geometry

The book by de Berg *et al.* [dBvKOS97] is a very comprehensive introduction to computational geometry. The one by O'Rourke [O'R94] is more oriented towards implementation. More advanced topics are treated in various books on the subject [Ede87, BY98]. Computational geometry often borrows themes from robotics.

Traditional computational geometry deals with the theoretical complexity of problems. Implementation is

not necessarily sought. Indeed some of the algorithms proposed in the literature are not implementable because they are based on too intricate data-structures. Moreover, very good theoretical complexity sometimes hides a very high constant, which means that the algorithm is not efficient unless the size of the input is very large. However, recent reports [Cha96, TAA+96, LM98] and the CGAL project [FGK+96] (a robust computational geometry library) show that the community is moving towards more applied subjects and robust and efficient implementations.

4.1 Hidden surface removal

The problem of hidden surface removal has also been widely treated in computational geometry, for the case of *object-precision* methods and polygonal scenes. It has been shown that a view can have $O(n^2)$ complexity, where n is the number of edges (for example if the scene is composed of rectangles which project like a grid as shown in Fig. 7.16). Optimal $O(n^2)$ algorithms have been described [McK87], and research now focuses on *output-sensitive* algorithms, where the cost of the method also depends on the complexity of the view : a hidden surface algorithms should not spend $O(n^2)$ time if one object hides all the others.

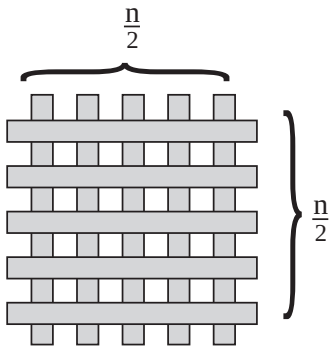


FIG. 7.16: Scene composed of n rectangles which exhibits a view with complexity $O(n^2)$: the planar map describing the view has $O(n^2)$ segments because of the $O(n^2)$ visual intersections.

The question has been studied in various context : computation of a single view, preprocessing for multiple view computation, and update of a view along a predetermined path.

Constraints are often imposed on the entry. Many papers deal with axis aligned rectangles, terrains or c-oriented polygons (the number of directions of the planes of the polygons is limited).

See the thesis by de Berg [Ber93] and the survey by Dorward [Dor94] for an overview. We will survey some computational geometry hidden-part removal methods in chapter 9 (section 2.3 and 8), chapter 10 (section 1.5) and chapter 13 (section 2.2).

4.2 Ray-shooting and lines in space

The properties and algorithms related to lines in 3D space have received a lot of attention in computational geometry.

Many algorithms have been proposed to reduced the complexity of ray-shooting (that is, the determination of the first object hit by a ray). Ray-shooting is often an atomic query used in computational geometry for hidden surface removal. Some algorithms need to compute what is the object seen behind a vertex, or behind the visual intersection of two edges.

Work somehow related to motion planning concerns the *classification* of lines in space : Given a scene composed of a set of lines, do two query lines, have the same class, *i.e.* can we continuously move the first one to the other without crossing a line of the scene ? This problem is related to the partition of rays or lines according to the object they see, as will be shown in section 2.2.

Given a set of convex objects, the *stabbing problems* searches for a line which intersects all the objects. Such a line is called a *stabbing line* or *stabber* or *transversal* (see Fig. 7.17). Stabbing is for example useful to

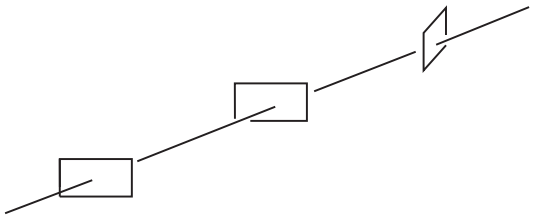


FIG. 7.17: Line stabbing a set of convex polygons in 3D space

decide if a line of sight is possible through a sequence of doors [6].

We will not survey all the results related to lines in space ; we will consider only those where the data-structures and algorithms are of a particular interest for the comprehension of visibility problems. See chapter 13. The paper by Pellegrini [Pel97b] reviews the major results about lines in space and gives the corresponding references.

4.3 Art galleries

In 1973, Klee raised this simple question : how many cameras are needed to guard an art gallery ? Assume the gallery is modeled by a 2D polygonal floorplan, and the camera have infinite range and 360° field of view. This problem is known as the *art gallery* problem. Since then, this question has received considerable attention, and many variants have been studied, as shown by the book by O'Rourke [O'R87] and the surveys on the domain [She92, Urr98]. The problem has been shown to be NP-hard.

Variation on the problem include mobile guards, limited field of view, rectilinear polygons and illumination of convex sets. The results are too numerous and most often more combinatorial than geometrical (the actual geometry of the scene is not taken into account, only its adjacencies are) so we refer the interested reader to the aforementioned references. We will just give a quick overview of the major results in section 3.1 of chapter 12.

The art gallery problem is related to many questions raised in vision and robotics as presented in section 2 and 3, and recently in computer graphics where the acquisition of models from photographs requires the choice of good viewpoints as seen in section 1.7.

4.4 2D visibility graphs

Another important visibility topic in computational geometry is the computation of *visibility graphs* which we will introduce in section 2. The characterisation of such graphs (given an abstract graph, is it the visibility graph of any scene ?) is also explored, but the subject is mainly combinatorial and will not be addressed in this survey. See *e.g.* [Gho97, Eve90, OS97].

5 Astronomy

5.1 Eclipses

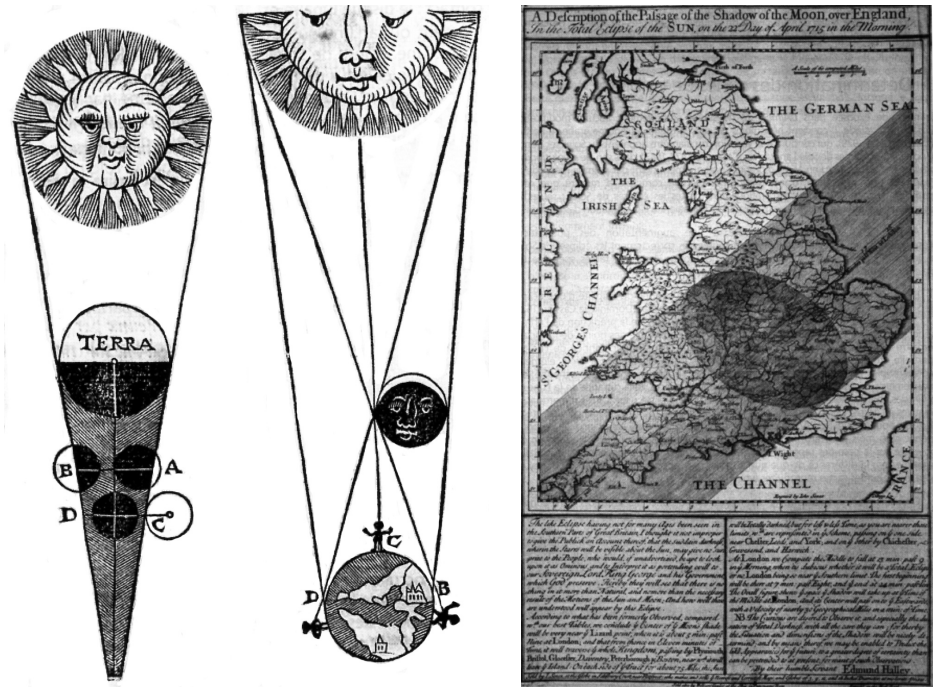
Solar and lunar eclipse prediction can be considered as the first occlusion related techniques. However, the main issue was focused on planet motion prediction rather than occlusion.

See *e.g.*
http ://sunearth.gsfc.nasa.gov/eclipse/eclipse.html
http ://www.bdl.fr/Eclipse99

5.2 Sundials

Sundials are another example of shadow related techniques.

[6] Stabbing can also have an interpretation in statistics to find a linear approximation to data with imprecisions. Each data point together with its precision interval defines a box in a multidimensional space. A stabber for these boxes is a valid linear approximation.



(a) (b)

FIG. 7.18: Eclipses. (a) Lunar and Solar eclipse by Purbach. (b) Prediction of the 1715 eclipse by Halley.

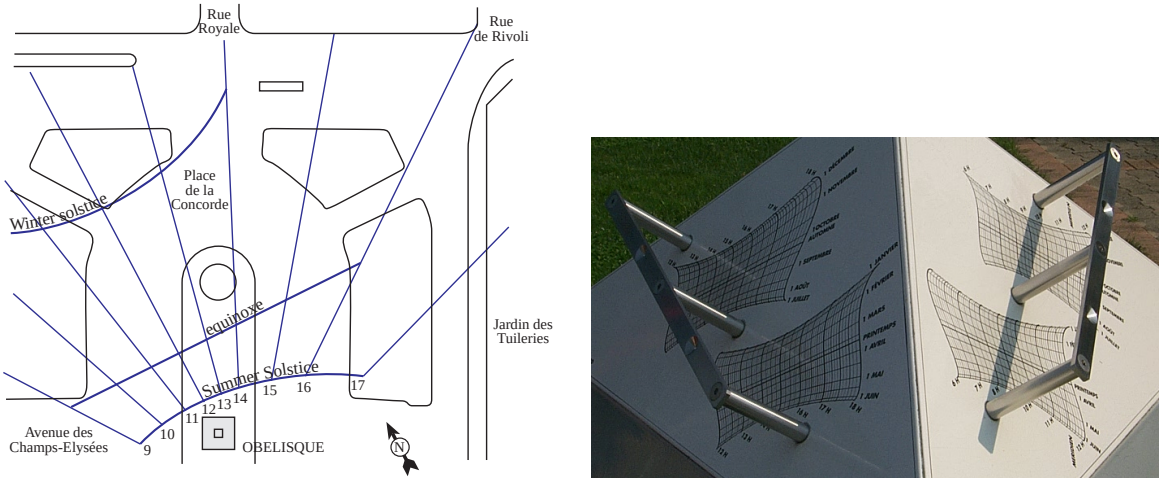
FIG. 7.19: 1994 solar eclipse and 1993 lunar eclipse. Photograph Copyright 1998 by Fred Espenak (NASA/Goddard Space Flight Center).

see *e.g.*
http ://www.astro.indiana.edu/personnel/rberring/sundial.html
http ://www.sundials.co.uk/2sundial.htm

6 Summary

Following Grant [Gra92], visibility problems can be classified according to their increasing dimensionality : The most atomic query is ray-shooting. View and hard shadow computation are two dimensional problems. Occlusion culling with respect to a point belong to the same category which we can refer to as classical visibility problems. Then comes what we call *global* visibility issues[7]. These include visibility with respect to extended regions such as extended light sources or volumes, or the computation of the region of space from which a feature is visible. The mutual visibility of objects (required for example for global illumination simulation) is a four dimensional problem defined on the pairs of points on surfaces of the scene. Finally the enumeration of all possible views of an object or the optimization of a viewpoint impose the treatment of two dimensional view computation problems for all possible viewpoints.

[7]Some author also define occlusion by other objects as global visibility effects as opposed to backface culling and silhouette computation.



(a) (b)

FIG. 7.20: (a) Project of a sundial on the Place de la Concorde in Paris. (b) Complete sundial with analemmas in front of the CICG in Grenoble.

CHAPITRE 8

Preliminaries

On apprend à reconnaître les forces sous-jacentes ; on apprends la préhistoire du visible. On apprend à fouiller les profondeurs, on apprend à mettre à nu. On apprend à démontrer, on apprend à analyser

Paul KLEE, *Théorie de l'art moderne*

EFORE presenting visibility techniques, we introduce a few notions which will be useful for the understanding and comparison of the methods we survey. We first introduce the different spaces which are related to visibility and which induce the classification that we will use. We then introduce the notion of *visual event*, which describes "where" visibility changes in a scene and which is central to many methods. Finally we discuss some of the differences which explain why 3D visibility is much more involved than its 2D counterpart.

1 Spaces and algorithm classification

In their early survey Sutherland, Sproull and Schumacker [SSS74] classified hidden-part removal algorithms into *object space* and *image-space* methods. Our terminology is however slightly different from theirs, since they designated the *precision* at which the computations are performed (at the resolution of the image or exact), while we have chosen to classify the methods we survey according to the space in which the computations are performed.

Furthermore we introduce two new spaces : the space of all viewpoints and the space of lines. We will give a few simple examples to illustrate what we mean by all these spaces.

1.1 Image-space

In what follow, we have classified as *image-space* all the methods which perform their operations in 2D projection planes (or other manifolds). As opposed to Sutherland *et al.*'s classification [SSS74], this plane is not restricted to the plane of the actual image. It can be an intermediate plane. Consider the example of hard shadow computation : an intermediate image from the point light source can be computed.

Of course if the scene is two dimensional, image space has only one dimension : the angle around the viewpoint.

Image-space methods often deal with a discrete or *rasterized* version of this plane, sometimes with a depth information for each point. Image-space methods will be treated in chapter 11.

1.2 Object-space

In contrast, object space is the 3 or 2 dimensional space in which the scene is defined. For example, some hard shadow computation methods use *shadow volumes* [FvDFH90, WPF90]. These volumes are truncated frusta defined by the point light source and the occluding objects. A portion of space is in shadow if it lies inside a shadow volume. Object-space methods will be treated in chapter 10.

1.3 Viewpoint-space

We define the *viewpoint space* as the set of all possible viewpoints. This space depends on the projection used. If perspective projection is used, the viewpoint space is equivalent to the object space. However, if orthographic (also called parallel) projection is considered, then a view is defined by a direction, and the viewpoint space is the set S^2 of directions, often called *viewing sphere* as illustrated in Fig. 8.1. Its projection on a cube is sometimes used for simpler computations.

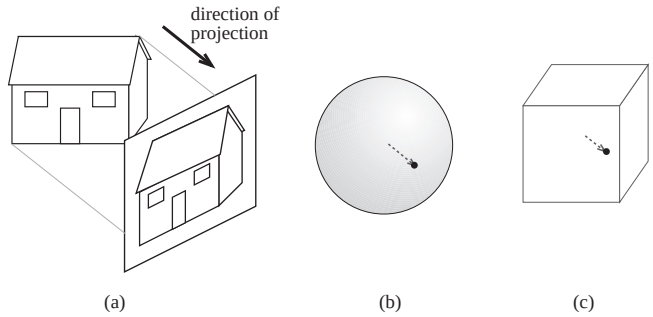


FIG. 8.1: (a) Orthographic view. (b) Corresponding point on the viewing sphere and (c) on the viewing cube.

An example of viewpoint space method would be to discretize the viewpoint space and precompute a view for each sample viewpoint. One could then render views very quickly with a simple look-up scheme. The viewer-centered representation which we have introduced in section 2.1 of the previous chapter is typically a viewpoint space approach since each possible view should be represented.

Viewpoint-space can be limited. For example, the viewer can be constrained to lie at eye level, defining a 2D viewpoint space (the plane $z = h_{eye}$) in 3D for perspective projection. Similarly, the distance to a point can be fixed, inducing a spherical viewpoint-space for perspective projection.

It is important to note that even if perspective projection is used, there is a strong difference between viewpoint space methods and object-space methods. In a viewpoint space, the properties of points are defined by their view. An orthographic viewpoint-space could be substituted in the method.

Shadow computation methods are hard to classify : the problem can be seen as the intersection of scene objects with shadow volume, but it can also be seen as the classification of viewpoint lying on the objects according to their view of the source. Some of our choices can be perceived arbitrary.

In 2D, viewpoint-space has 2 dimensions for perspective projection and has 1 dimension if orthographic projection is considered.

Viewpoint space methods will be treated in chapter 12.

1.4 Line-space

Visibility can intuitively be defined in terms of lines : two point A and B are mutually visible if no object intersects line (AB) between them. It is thus natural to describe visibility problems in line space.

For example, one can precompute the list of objects which intersect each line of a discretization of line-space to speed-up ray-casting queries.

In 2D, lines have 2 dimensions : for example its direction θ and distance to the origin ρ. In 3D however, lines have 4 dimensions. They can for example be parameterized by their direction (θ, φ) and by the intersection (u, v) on an orthogonal plane (Fig. 8.2(a)). They can also be parameterized by their intersection with two planes (Fig. 8.2(b)). These two parameterizations have some singularities (at the pole for the first one, and for lines parallel to the two planes in the second). Lines in 3D space can not be parameterized without a singularity. In section 3 of chapter 13 we will study a way to cope with this, embedding lines in a 5 dimensional space.

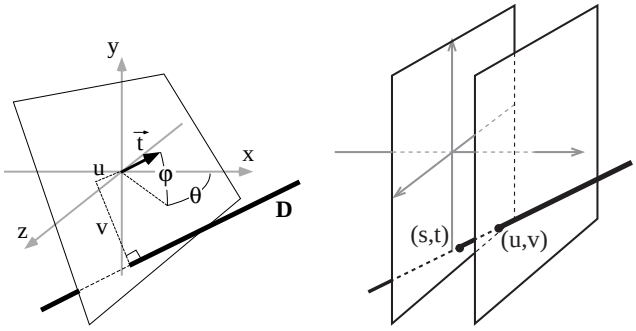


FIG. 8.2: Line parameterisation. (a) Using two angles and the intersection on an orthogonal plane. (b) Using the intersection with two planes.

The set of lines going through a point describe the view from this point, as in the ray-tracing technique (see Fig. 7.5). In 2D the set of lines going through a point has one dimension : for example their angle. In 3D, 2 parameters are necessary to describe a line going through a point, for example two angles.

Many visibility queries are expressed in terms of rays and not lines. The ray-shooting query computes the first object seen from a point in a given direction. Mathematically, a ray is a half line. Ray-space has 5 dimensions (3 for the origin and two for the direction).

The mutual visibility query can be better expressed in terms of segments. A and B are mutually visible only if segment $[AB]$ intersects no object. Segment space has 6 dimensions : 3 for each endpoint.

The information expressed in terms of rays or segments is very redundant : many colinear rays “see” the same object, many colinear segments are intersected by the same object. We will see that the notion of *maximal free segments* handles this. Maximal free segments are segments of maximal length which do not touch the objects of the scene in their interior. Intuitively these are segments which touch objects only at their extremities.

We have decided to group the methods which deal with these spaces in chapter 13. The interested reader will find some important notions about line space reviewed in appendix D.

1.5 Discussion

Some of the methods we survey do not perform all their computations in a single space. An intermediate data-structure can be used, and then projected in the space in which the final result is required.

Even though each method is easier to describe in a given space, it can often be described in a different space. Expressing a problem or a method in different spaces is particularly interesting because it allows different insights and can yield alternative methods. We particularly invite the reader to transpose visibility questions to line space or ray space. We will show throughout this survey that visibility has a very natural interpretation in line space.

However this is not an incitation to actually perform complex calculations in 4D line space. We just suggest a different way to understand problems and develop methods, even if calculations are eventually performed in image or object space.

2 Visual events, singularities

We now introduce a notion which is central to most of the algorithms, and which expresses “how” and “where” visibility changes. We then present the mathematical framework which formalizes this notion, the theory of singularities. The reader may be surprised by the space devoted in this survey to singularity theory compared to its use in the literature. We however believe that singularity theory permits a better insight on visibility problems, and allows one to generalize some results on polygonal scenes to smooth objects.

2.1 Visual events

Consider the example represented in Fig. 8.3. A polygonal scene is represented, and the views from three eyepoints are shown on the right. As the eyepoint moves downwards, pyramid P becomes completely hidden by polygon Q. The limit eyepoint is eyepoint 2, for which vertex V projects exactly on edge E. There is a topological change in visibility : it is called a *visual event* or a *visibility event*.

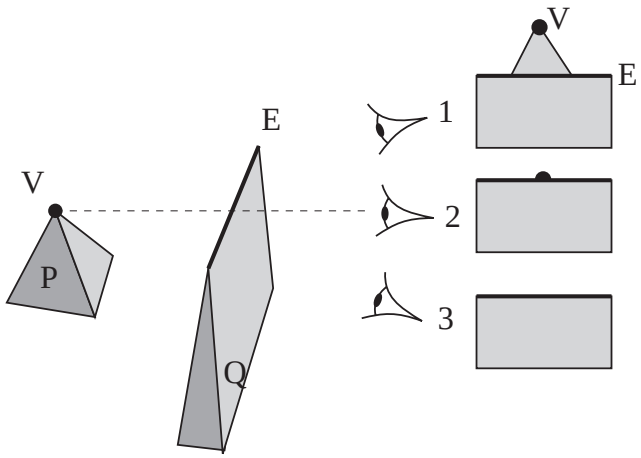
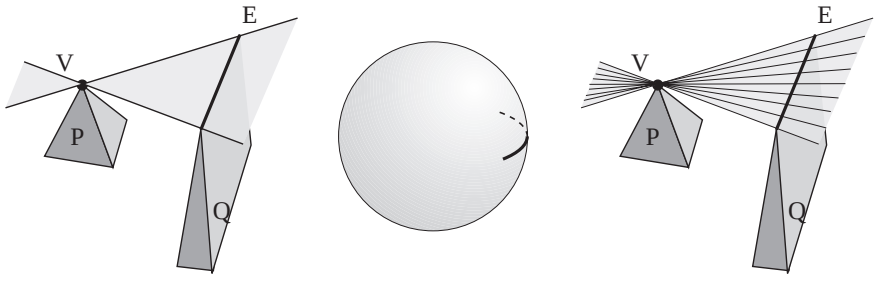


FIG. 8.3: EV visual event. The views from the three eyepoints are represented on the right. As the eyepoint moves downwards, vertex V becomes hidden. Viewpoint 2 is the limit eyepoint, it lies on a *visual event*.

Visual events are fundamental to understand many visibility problems and techniques. For example when an observer moves through a scene, objects appear and disappear at such events (Fig. 8.3). If pyramid P emits light, then eyepoint 1 is in penumbra while eyepoint 3 is in umbra : the visual event is a shadow boundary. If a viewpoint is sought from which pyramid P is visible, then the visual event is a limit of the possible solutions.



(a) (b) (c)

FIG. 8.4: Locus an EV visual event. (a) In object space or perspective viewpoint space it is a wedge. (b) In orthographic viewpoint space it is an arc of a great circle. (c) In line space it is the 1D set of lines going through V and E

Fig. 8.4 shows the locus of this visual event in the spaces we have presented in the previous section. In object space or in perspective viewpoint space, it is the wedge defined by vertex V and edge E. We say that

V and E are the *generators* of the event. In orthographic viewpoint space it is an arc of a great circle of the viewing sphere. Finally, in line-space it is the set of lines going through V and E. These *critical lines* have one degree of freedom since they can be parameterized by their intercept on E, we say that it is a 1D set of lines.

The EV events generated by a vertex V are caused by the edges which are visible from V. The set of events generated by V thus describe the view from V. Reciprocally, a line drawing of a view from an arbitrary point P can be seen as the set of EV events which would be generated if an imaginary vertex was place at P.

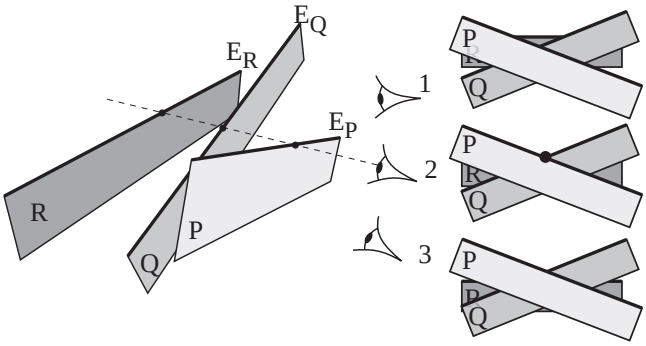


FIG. 8.5: A EEE visual event. The views from the three eyepoints are represented on the right. As the eyepoint moves downwards, polygon R becomes hidden by the conjunction of polygon P and Q. From the limit viewpoint 2, the three edges have a visual intersection.

There is also a slightly more complex kind of visual event in polygonal scenes. It involves the interaction of 3 edges which project on the same point (Fig. 8.5). When the eyepoint moves downwards, polygon P becomes hidden by the conjunction of Q and R. From the limit eyepoint 2, edges E_P, E_Q and E_R are aligned.

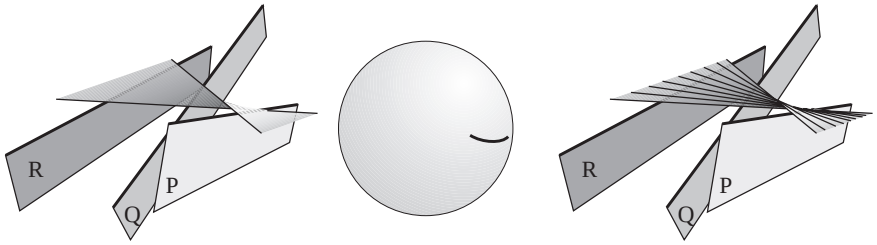


FIG. 8.6: Locus of a EEE visual event. (a) In object-space or perspective viewpoint space it is a ruled quadrics. (b) In orthographic viewpoint space it is a quadric on the viewing sphere. (c) In line space it is the set of lines stabbing the three edges.

The locus of such events in line space is the set of lines going through the three edges (we also say that they *stab* the three edges) as shown on Fig. 8.6(c). In object space or perspective viewpoint space, this defines a ruled quadric often called *swath* (Fig. 8.6(a)). (It is in fact doubly ruled : the three edges define one family of lines, the stabber defining the second.) In orthographic viewpoint space it is a quadric on the viewing sphere (see Fig. 8.6(b)).

Finally, a simpler class of visual events are caused by a viewpoint lying in the plane of faces of the scene. The face becomes visible or hidden at such an event.

Visual events are simpler in 2D : they are simply the *bitangents* and *inflexion points*of the scene.

A deeper understanding of visual events and their generalisation to smooth objects requires a strong formalism : it is provided by the singularity theory.

2.2 Singularity theory

The singularity theory studies the emergence of discrete structures from smooth continuous ones. The branch we are interested in has been developed mainly by Whitney [Whi55], Thom [Tho56, Tho72] and Arnold [Arn69]. It permits the study of sudden events (called *catastrophes*) in systems governed by smooth continuous laws. An introduction to singularity theory for visibility can be found in the masters thesis by PetitJean [Pet92] and an educational comics has been written by Ian Stewart [Ste82]. See also the book by Koenderink [Koe90] or his papers with van Doorn [Kv76, KvD82, Kø84, Koe87].

We are interested in the singularities of smooth mappings. For example a view projection is a smooth mapping which associate each point of 3D space to a point on a projection plane. First of all, singularity theory permits the description the structure of the visible parts of a smooth object.

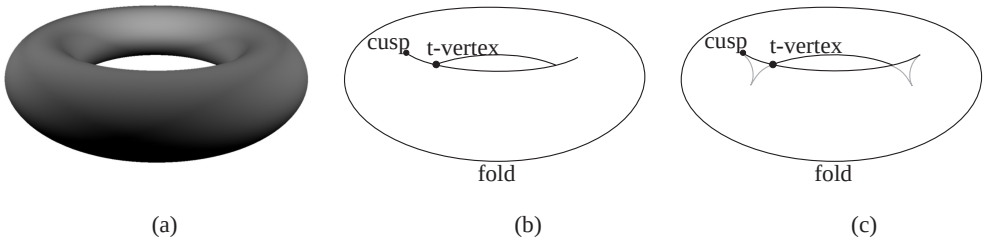


FIG. 8.7: View of a torus. (a) Shaded view. (b) Line drawing with singularities indicated (b) Opaque and transparent contour.

Consider the example of a smooth 3D object such as the torus represented in Fig. 8.7(a). Its projection on a viewing plane is continuous nearly everywhere. However, some abrupt changes appear at the so called silhouette. Consider the number of point of the surface of the object projecting on a given point on the projection plane (counting the backfacing points). On the exterior of the silhouette no point is projected. In the interior two points (or more) project on the same point. These two regions are separated by the silhouette of the object at which the number of projected point changes abruptly.

This abrupt change in the smooth mapping is called a *singularity* or *catastrophe* or *bifurcation*. The singularity corresponding to the silhouette was named *fold* (or also *occluding contour* or *limb*). The fold is usually used to make a line drawing of the object as in Fig. 8.7(b). It corresponds to the set of points which are tangent to the viewing direction[1].

The fold is the only stable curve singularity for generic surfaces : if we move the viewpoint, there will always be a similar fold.

The projection in Fig. 8.7 also exhibits two point singularities : a *t-vertex* and a *cusp*. T-vertices results from the intersection of two folds. Fig. 8.7(c) shows that a fourth fold branch is hidden behind the surface. Cusps represent the visual end of folds. In fact, a cusp corresponds to a point where the fold has an inflexion in 3D space. A second tangent fold is hidden behind the surface as illustrated in Fig. 8.7(c).

These are the only three stable singularities : all other singularities disappear after a small perturbation of the viewpoint (if the object is generic, which is not the case of polyhedral objects). These stable singularities describe the limits of the visible parts of the object. Malik [Mal87] has established a catalogue of the features of line drawings of curved objects.

Singularity theory also permits the description of how the line drawing changes as the viewpoint is moved. Consider the example represented in Fig. 8.8. As the viewpoint moves downwards, the back sphere becomes hidden by the front one. From viewpoint (b) where this visual event occurs, the folds of the two spheres are superimposed and tangent. This unstable singularity is called a *tangent crossing*. It is very similar to the *EV* visual event shown in Fig. 8.3. It is unstable in the sense that any small change in the viewpoint will make it disappear. The viewpoint is not *generic*, it is *accidental*.

[1] What is the relationship between the view of a torus and the occurrence of a sudden catastrophe ? Imagine the projection plane is the command space of a physical system with two parameters x and y. The torus is the response surface : for a pair of parameters (x,y) the depth z represents the state of the system. Note that for a pair of parameters, there may be many possible states, depending on the history of the system. When the command parameters vary smoothly, the corresponding state varies smoothly on the surface of the torus. However, when a fold is met, there is an abrupt change in the state of the system, this is a *catastrophe*. See *e.g.* [Ste82].

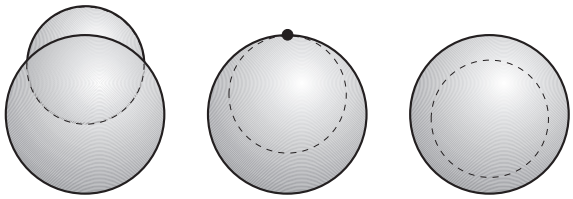


FIG. 8.8: Tangent crossing singularity. As the viewpoint moves downwards, the back sphere becomes hidden by the frontmost one. At viewpoint (b) a singularity occurs (highlighted with a point) : the two spheres are visually tangent.

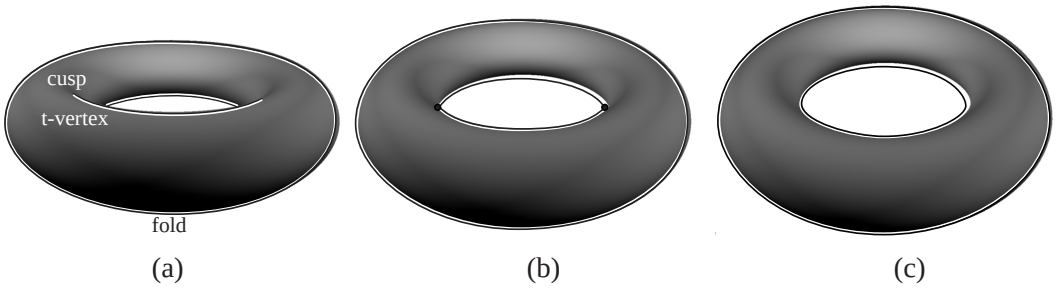


FIG. 8.9: Disappearance of a cusp at a swallowtail singularity at viewpoint (b). (in fact two swallowtails occur because of the symmetry of the torus)

Another unstable singularity is shown in Fig. 8.9. As the viewpoints moves upward, the t-vertex and the cusp disappear. In Fig. 8.9(a) the points of the plane below the cusp result from the projection of 4 points of the torus, while in Fig. 8.9(c) all points result from the projection of 2 or 0 points. This unstable singularity is called *swallowtail*.

Unstable singularities are the events at which the organisation of the view of a smooth object (or scene) is changed. These singularities are related to the differential properties of the surface. For example swallowtails occur only in hyperbolic regions of the surface, that is, regions where the surface is locally nor concave nor convex.

Singularity theory originally does not consider opaqueness. Objects are assumed transparent. As we have seen, at cusps and t-vertices, some fold branches are hidden. Moreover a singularity like a tangent crossing is considered even if some objects lie between the two sphere causing occlusion. The visible singularity are only a subset but all the changes observed in views of opaque objects can be described by singularity theory. Some catalogues now exist which describe singularities of opaque objects [2]. See Fig. 8.10.

The catalogue of singularities for views of smooth objects has been proposed by Kergosien [Ker81] and Rieger [Rie87, Rie90] who has also proposed a classification for piecewise smooth objects [Rie87] [3].

3 2D *versus* 3D Visibility

We enumerate here some points which make that the difference between 2D and 3D visibility can not be summarized by a simple increment of one to the dimension of the problem.

This can be more easily envisioned in line space. Recall that the atomic queries in visibility are expressed in line-space (first point seen along a ray, are two points mutually visible ?).

[2] Williams [WH96, Wil96] tries to fill in the gap between opaque and transparent singularities. Given the view of an object, he proposes to deduce the invisible singularities from the visible ones. For example at a t-vertex, two folds intersect but only three branches are visible ; the fourth one which is occluded can be deduced. See Fig. 8.10.

[3] Those interested in the problems of robustness and degeneracies for geometric computations may also notice that a degenerate configuration can be seen as a singularity of the space of scenes. The exploration of the relations between singularities and degeneracies could help formalize and systemize the treatment of the latter. See also section 2 of chapter 14.

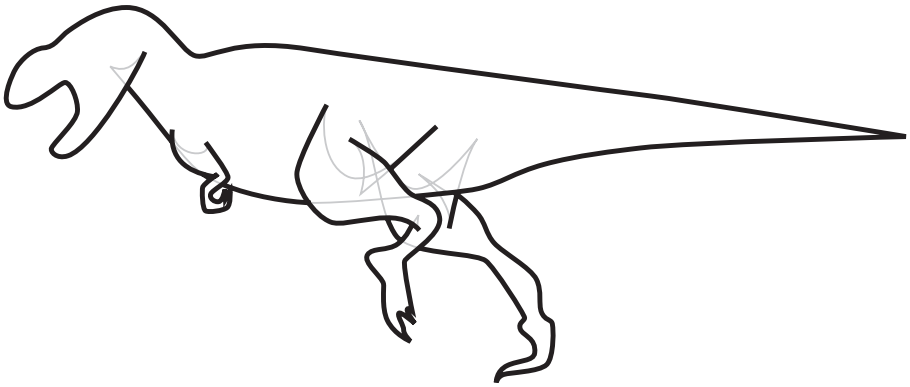


FIG. 8.10: Opaque (bold lines) and semi-transparent (grey) singularities. After [Wil96].

First of all, the increase in dimension of line-space is two, not one (in 2D line-space is 2D, while in 3D it is 4D). This makes things much more intricate and hard to apprehend.

A line is a hyperplane in 2D, which is no more the case in 3D. Thus the separability property is lost : a 3D line does not separate two half-space as in 2D.

A 4D parameterization of 3D lines is not possible without singularities (the one presented in Fig. 8.2(a) has two singularities at the pole, while the one in Fig. 8.2(b) can not represent lines parallel to the two planes). See section 3 of chapter 13 for a partial solution to this problem.

Visual events are simple in 2D : bitangent lines or tangent to inflection points. In 3D their locus are surfaces which are rarely planar (EEE or visual events for curved objects).

All these arguments make the sentence “the generalization to 3D is straightforward” a doubtful statement in any visibility paper.

CHAPITRE 9

The classics of hidden part removal

Il convient encore de noter que c'est parce que quelque chose des objets extérieurs pénétre en nous que nous voyons les formes et que nous pensons

ÉPICURE, *Doctrines et Maximes*

E FIRST BRIEFLY review the classical algorithms to solve the hidden surface removal problem. It is important to have these techniques in mind for a wider insight of visibility techniques. We will however remain brief, since it is beyond the scope of this survey to discuss all the technical details and variations of these algorithms. For a longer survey see [SSS74, Gra92], and for a longer and more educational introduction see [FvDFH90, Rog97].

The view computation problem is often reduced to the case where the viewpoint lies on the z axis at infinity, and x and y are the coordinates of the image plane ; y is the vertical axis of the image. This can be done using a perspective transform matrix (see [FvDFH90, Rog97]). The objects closer to the viewpoint can thus be said to lie "above" (because of the z axis) as well as "in front" of the others. Most of the methods treat polygonal scenes.

Two categories of approaches have been distinguished by Sutherland *et al*. *Image-precision* algorithms solve the problem for a discrete (rasterized) image, visibility being sampled only at pixels ; while *object-precision* algorithm solve the exact problem. The output of the latter category is often a *visibility map*, which is the planar map describing the view. The order in which we present the methods is not chronological and has been chosen for easier comparison.

Solutions to hidden surface removal have other applications that the strict determination of the objects visible from the viewpoint. As evoked earlier, hard shadows can be computed using a view from a point light source. Inversely, the amount of light arriving at a point in penumbra corresponds to the visible part of the source from this point as shown in Fig. 7.2(b). Interest for the application of exact view computation has thus recently been revived.

1 Hidden-Line Removal

The first visibility techniques have were developed for *hidden line removal* in the sixties. These algorithms provide information only on the visibility of edges. Nothing is known on the interior of visible faces, preventing shading of the objects.

1.1 Robert

Robert [Rob63] developed the first solution to the hidden line problem. He tests all the edges of the scene polygons for occlusion. He then computes the intersection of the wedge defined by the viewpoint and the edge and all objects in the scene using a parametric approach.

1.2 Appel

Appel [App67] has developed the notion of *quantitative invisibility* which is the number of objects which occlude a given point. This is the notion which we used to present singularity theory : the number of points of the object which project on a given point in the image. Visible points are those with 0 quantitative invisibility. The quantitative invisibility of an edge of a view changes only when it crosses the projection of another edge (it corresponds to a *t-vertex*). Appel thus computes the quantitative invisibility number of a vertex, and updates the quantitative invisibility at each visual edge-edge intersection.

Markosian *et al.* [MKT+97] have used this algorithm to render the silhouette of objects in a non-photorealistic manner. When the viewpoint is moved, they use a probabilistic approach to detect new silhouettes which could appear because an unstable singularity is crossed.

1.3 Curved objects

Curved objects are harder to handle because their silhouette (or *fold*) first has to be computed (see section 2.2 of chapter 8). Elber and Cohen [EC90] compute the silhouette using adaptive subdivision of parametric surfaces. The surface is recursively subdivided as long as it may contain parts of the silhouette. An algorithm similar to Appel's method is then used. Snyder [Sny92] proposes the use of interval arithmetic for robust silhouette computation.

2 Exact area-subdivision

2.1 Weiler-Atherton

Weiler and Atherton [WA77] developed the first object-precision method to compute a visibility map. Objects are preferably sorted according to their depth (but cycles do not have to be handled). The frontmost polygons are then used to clip the polygons behind them.

This method can also be very simply used for hard shadow generation, as shown by Atherton *et al.* [AWG78]. A view is computed from the point light source, and the clipped polygons are added to the scene database as lit polygon parts.

The problem with Weiler and Atherton's method, as for most of the object-precision methods, is that it requires robust geometric calculations. It is thus prone to numerical precision and degeneracy problems.

2.2 Application to form factors

Nishita and Nakamae [NN85] and Baum *et al.* [BRW89] compute an accurate form factor between a polygon and a point (the portion of light leaving the polygon which arrives at the point) using Weiler and Atherton's clipping. Once the source polygon is clipped, an analytical formula can be used. Using Stoke's theorem, the integral over the polygon is computed by an integration over the contour of the visible part. The jacobian of the lighting function can be computed in a similar manner [Arv94].

Vedel [Ved93] has proposed an approximation for the case of curved objects.

2.3 Mulmuley

Mulmuley [Mul89] has proposed an improvement of exact area-subdivision methods. He inserts polygons in a *randomized* order (as in quick-sort) and maintains the visibility map. Since visibility maps can have complex boundaries (concave, with holes), he uses a trapezoidal decomposition [dBvKOS97]. Each trapezoid corresponds to a part of one (possibly temporary) visible face.

Each trapezoid of the map maintains a list of *conflict* polygons, that is, polygons which have not yet been projected and which are above the face of the trapezoid. As a face is chosen for projection, all trapezoids with which it is in conflict are updated. If a face is below the temporary visible scene, no computation has to be performed.

The complexity of this algorithm is very good, since the probability of a feature (vertex, part of edge) to induce computation is inversely proportional to its quantitative invisibility (the number of objects above it). It should be easy to implement and robust due to its randomized nature. However, no implementation has been reported to our knowledge.

2.4 Curved objects

Krishnan and Manocha [KM94] propose an adaptation of Weiler and Atherton's method for curved objects modeled with NURBS surfaces. They perform their computation in the parameter space of the surface. The silhouette corresponds to the points where the normal is orthogonal to the view-line, which defines a polynomial system. They use an algebraic marching method to solve it. These silhouettes are approximated by piecewise-linear curves and then projected on the parts of the surface below, which gives a partition of the surface where the quantitative invisibility is constant.

3 Adaptive subdivision

The method developed by Warnock [War69] can be seen as an approximation of Weiler and Atherton's exact method, even though it was developed earlier. It recursively subdivides the image until each region (called a *window*) is declared homogeneous. A window is declared homogeneous if one face completely covers it and is in front of all other faces. Faces are classified against a window as intersecting or disjoint or surrounding (covering). This classification is passed to the subwindows during the recursion. The recursion is also stopped when pixel-size is reached.

The classical method considers quadtree subdivision. Variations however exist which use the vertices of the scene to guide the subdivision and which stop the recursion when only one edge covers the window.

Marks *et al.* [MWCF90] presents an analysis of the cost of adaptive subdivision and proposes a heuristic to switch between adaptive methods and brute-force z-buffer.

4 Depth order and the painter's algorithm

The painter's algorithm is a class of methods which consist in simply drawing the objects of the scene from back to front. This way, visible objects overwrite the hidden ones. This is similar to a painter who first draws a background then paints the foreground onto it. However, ordering objects according to their occlusion is not straightforward. Cycles may appear, as illustrated in Fig. 9.1(a).

The inverse order (Front to Back) can also be used, but a flag has to be indicate whether a pixel has been written or not. This order allows shading computations only for the visible pixels.

4.1 Newell Newell and Sancha

In the method by Newell, Newell and Sancha [NNS72] polygons are first sorted according to their minimum z value. However this order may not be the occlusion order. A bubble sort like scheme is thus applied. Polygons with overlapping z intervals are first compared in the image for xy overlap. If it is the case, their plane equation is used to test which occlude which. Cycles in occlusion are tested, in which case one of the polygons is split as shown in Fig. 9.1(b).

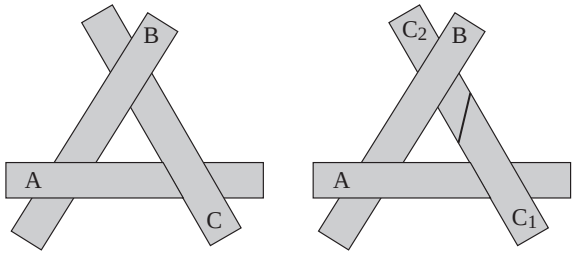


FIG. 9.1: (a) Classic example of a cycle in depth order. (b) Newell, Newell and Sancha split one of the polygons to break the cycle.

For new theoretical results on the problem of depth order, see the thesis by de Berg [Ber93].

4.2 Priority list preprocessing

Schumacker [SBGS69] developed the concept of *a priori* depth order. An object is preprocessed and an order may be found which is valid from any viewpoint (if the backfacing faces are removed). See the example of Fig. 9.2.

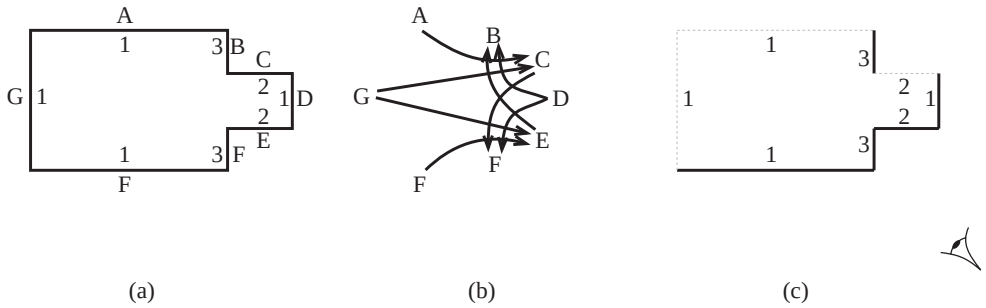


FIG. 9.2: *A priori* depth order. (a) Lower number indicate higher priorities. (b) Graph of possible occlusions from any viewpoint. An arrow means that a face can occlude another one from a viewpoint. (c) Example of a view. Backfacing polygons are eliminated and other faces are drawn in the *a priori* order (faces with higher numbers are drawn first).

These objects are then organised in clusters which are themselves depth-ordered. This technique is fundamental for flight simulators where real-time display is crucial and where cluttered scenes are rare. Moreover, antialiasing is easier with list-priority methods because the coverage of a pixel can be maintained more consistently. The survey by Yan [Yan85] states that in 1985, all simulators were using depth order. It is only very recent that z-buffer has started to be used for flight simulators (see section below).

However, few objects can be *a priori* ordered, and the design of a suitable database had to be performed mainly by hand. Nevertheless, this work has led to the development of the BSP tree which we will present in section 1.4 of chapter 10

4.3 Layer ordering for image-based rendering

Recently, the organisation of scenes into layers for image-based rendering has revived the interest in depth-ordering *à la* Newell *et al.* Snyder and Lengyel [SL98] proposed the merging of layers which form an occlusion cycle, while Decoret *al.* [DSSD99] try to group layers which cannot have occlusion relations to obtain better parallax effects.

5 The z-buffer

5.1 Z-buffer

The z-buffer was developed by Catmull [Cat74, Cat75]. It is now the most widespread view computation method.

A depth (or z-value) is stored for each pixel of the image. As each object is scan-converted (or rasterized), the depth of each pixel it covers in the image is computed and compared against the corresponding current z-value. The pixel is drawn only if it is closer to the viewpoint.

Z-buffer was developed to handle curved surfaces, which are recursively subdivided until a sub-patch covers only one pixel. See also [CLR80] for improvements.

The z-buffer is simple, general and robust. The availability of cheap and fast memory has permitted very efficient hardware implementations at low costs, allowing today's low-end computer to render thousands of shaded polygons in real-time. However, due to the rasterized nature of the produced image, aliasing artifacts occur.

5.2 A-buffer

The *A-buffer* (antialiased averaged area accumulation buffer) is a high quality antialiased version of the z-buffer. A similar rasterization scheme is used. However, if a pixel is not completely covered by an object (typically at edges) a different treatment is performed. The list of object fragments which project on these non-simple pixels is stored instead of a color value (see Fig. 9.3). A pixel can be first classified non simple because an edge projects on it, then simple because a closer object completely covers it. Once all objects have been projected, sub-pixel visibility is evaluated for non-simple pixels. 4*8 subpixels are usually used. Another advantage of the A-buffer is its treatment of transparency ; Subpixel fragments can be sorted in front-to-back order for correct transparency computations.

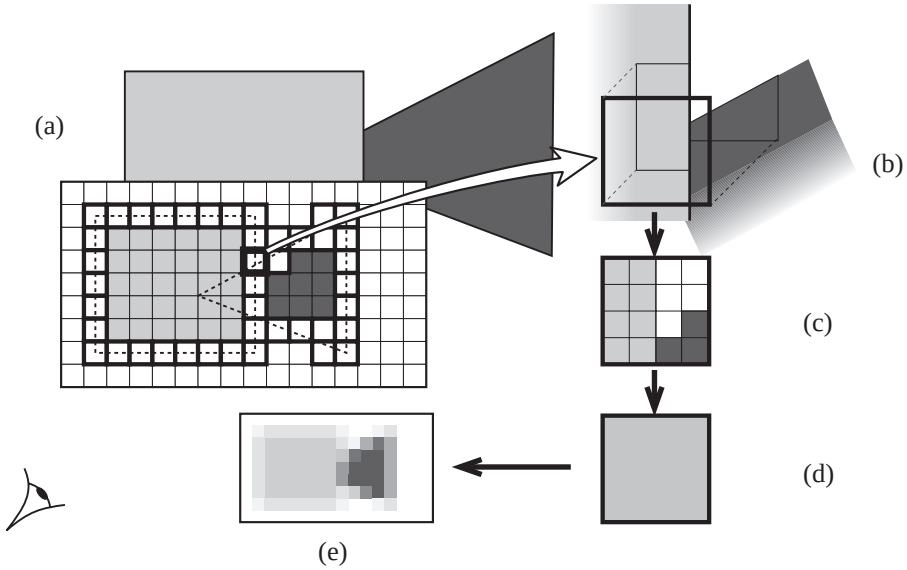


FIG. 9.3: A buffer. (a) The objects are scan-converted. The projection of the objects is dashed and non-simple pixels are represented in bold. (b) Close-up of a non-simple pixel with the depth sorted fragments (*i.e.*, the polygons clipped to the pixel boundary). (c) The pixel is subsampled. (d) The resulting color is the average of the subsamples. (e) Resulting antialiased image.

The A-buffer can be credited to Carpenter [Car84], and Fiume *et al.* [FFR83]. It is a simplification of the "ultimate" algorithm by Catmull [Cat78] which used exact sub-pixel visibility (with a Weiler-Atherton clipping) instead of sub-sampling. A comprehensive introduction to the A-buffer and a discussion of implementation is given in the book by Watt and Watt [WW92].

The A-buffer is, with ray-tracing, the most popular high-quality rendering techniques. It is for example implemented in the commercial products Alias Wavefront Maya and Pixar Renderman [CCC87]. Similar techniques are apparently present in the hardware of some recent flight simulator systems [Mue95].

Most of the image space methods we present in chapter 11 are based on the z-buffer. A-buffer-like schemes could be explored when aliasing is too undesirable.

6 Scan-line

6.1 Scan-line rendering

Scan-line approaches produce rasterized images and consider one line of the image at a time. Their memory requirements are low, which explains why they have long been very popular. Wylie and his coauthors [WREE67] proposed the first scan-line algorithms, and Bouknight [Bou70] and Watkins [Wat70] then proposed very similar methods.

The objects are sorted according to y. For each scan-line, the objects are then sorted according to x. Then for each *span* (x interval on which the same objects project) the depths of the polygons are compared. See [WC91] for a discussion of efficient implementation. Another approach is to use a z-buffer for the current scan-line. The A-buffer [Car84] was in fact originally developed in a scan-line system.

Crocker [Cro84] has improved this method to take better advantage of coherence.

Scan-line algorithms have been extended to handle curved objects. Some methods [Cla79, LC79, LCWB80] use a subdivision scheme similar to Catmull's algorithm presented in the previous section while others [Bli78, Whi78, SZ89] actually compute the intersection of the surface with the current scan-line. See also [Rog97] page 417.

Sechrest and Greenberg [SG82] have extended the scanline method to compute *object precision* (exact) views. They place scan-lines at each vertex or edge-edge intersection in the image.

Tanaka and Takahashi [TT90] have proposed an antialiased version of the scan-line method where the image is scanned both in x and y. An adaptive scan is used in-between two y scan-lines. They have applied this scheme to soft shadow computation [TT97] (see also section 1.4 of chapter 13).

6.2 Shadows

The first shadowing methods were incorporated in a scan-line process as suggested by Appel [App68]. For each span (segment where the same polygon is visible) of the scan-line, its shadowing has to be computed. The wedge defined by the span and a point light-source is intersected with the other polygons of the scene to determine the shadowed part of the span.

In section 1.1 of chapter 11 we will see an improvement to this method. Other shadowing techniques for scan-line rendering will be covered in section 4.1 of chapter 10.

7 Ray-casting

The computation of visible objects using ray-casting was pioneered by Appel [App68], the Mathematical Application Group Inc. [MAG68] and Goldstein and Nagel [GN71] in the late sixties. The object visible at one pixel is determined by casting a ray through the scene. The ray is intersected with all objects. The closest intersection gives the visible object. Shadow rays are used to shade the objects. As for the z-buffer, Sutherland *et al.* [SSS74] considered this approach brute force and thought it was not scalable. They are now the two most popular methods.

As evoked in section 1.5 of chapter 7 Whitted [Whi80] and Kay [KG79] have extended ray-casting to *ray-tracing* which treats transparency and reflection by recursively sending secondary rays from the visible points.

Ray tracing can handle any type of geometry (as soon as an intersection can be computed). Various methods have been developed to compute ray-surface intersections, *e.g.*, [Kaj82, Han89].

Ray-tracing is the most versatile rendering technique since it can also render any shading effect. Antialiasing can be performed with subsampling : many rays are sent through a pixel (see *e.g.* [DW85, Mit87]).

Ray-casting and ray-tracing send rays from the eye to the scene, which is the opposite of actual physical light propagation. However, this corresponds to the theory of scientists such as Aristote who think that "visual rays" go from the eye to the visible objects.

As observed by Hofmann [Hof92] and illustrated in Fig. 9.4 ideas similar to ray-casting were exposed by Dürer [Dür38] while he was presenting perspective.

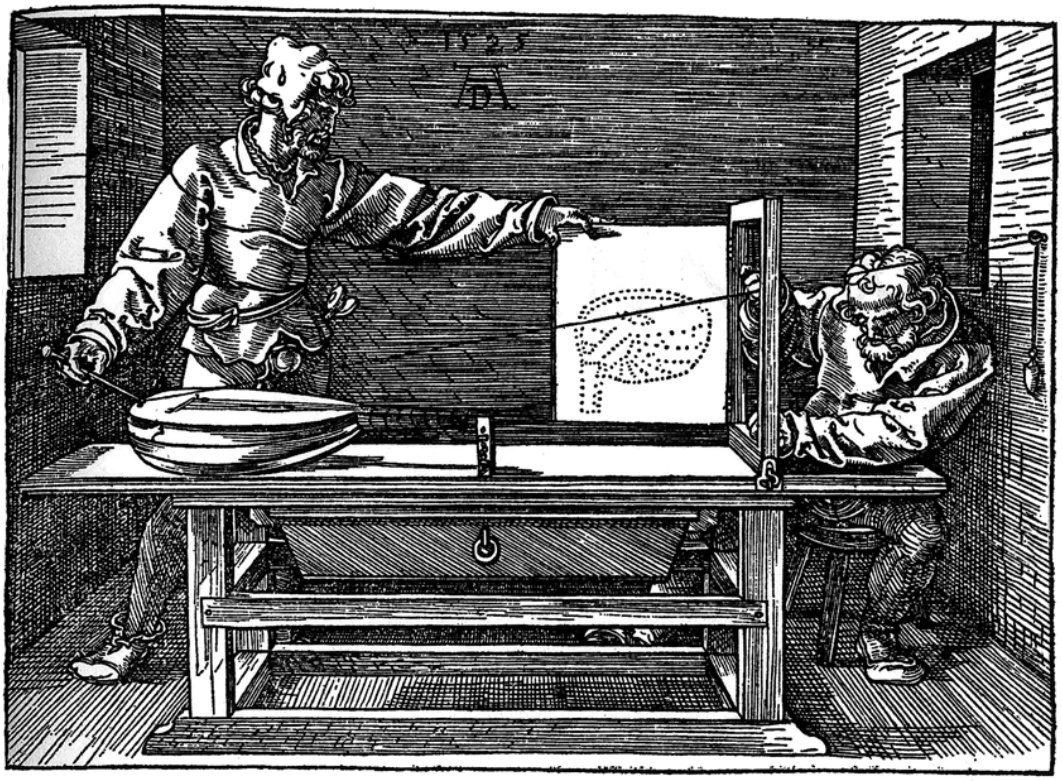


FIG. 9.4: Drawing by Dürer in 1538 to illustrate his setting to compute perspective. It can be thought of as an ancestor of ray-casting. The artist's assistant is holding a stick linked to a string fixed at an eyebolt in the wall which represents the viewpoint. He points to part of the object. The position of the string in the frames is marked by the artist using the intersection of two strings fixed to the frame. He then rotates the painting and draws the point.

8 Sweep of the visibility map

Most of the algorithms developed in computational geometry to solve the hidden part removal problem are based on a sweep of the visibility map for polygonal scenes. The idea is illustrated in Fig. 9.5. The view is swept by a vertical (not necessarily straight) line, and computations are performed only at discrete steps often called events. A list of active edges (those crossing the sweep line) is maintained and updated at each events. Possible events are the appearance the vertex of a new polygon, or a *t-vertex*, that is, the visual intersection of an active edge and another edge (possibly not active).

The problem then reduces to the efficient detection of these events and the maintenance of the active edges. As evoked in the introduction this often involves some ray shooting queries (to detect which face becomes visible at a t-vertex for example). More complex queries are required to detect some t-vertices.

The literature on this subject is vast and well surveyed in the paper by Dorward [Dor94]. See also the thesis by de Berg [Ber93]. Other recent results on the subject include [Mul91, Pel96] (see section 1.5 of chapter 10).

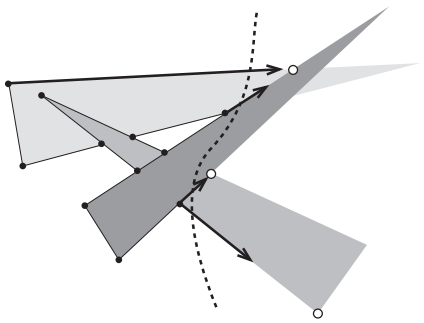


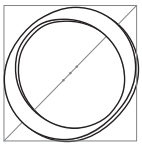
FIG. 9.5: Sweep of a visibility map. Active edges are in bold. Already processed events are black points, while white points indicate the event queue.

CHAPITRE 10

Object-Space

Ombres sans nombre
nombres sans ombre
à l'infini
au pas cadencé
Nombres des ombres
ombre des nombres
à l'infini
au pas commencé

Jacques PRÉVERT, *Fatras*



BJECT-SPACE methods exhibit the widest range of approaches. We first introduce methods which optimize visibility computation by using a well-behaved subdivision of space. We then present two important data-structures based on the object-space locus of visual events, the 2D visibility graph (section 2) and visual hull (section 3). We then survey the large class of methods which characterize visibility using pyramid-like shapes. We review methods using beams for visibility with respect to a point in section 4. We then present the extensions of these methods to compute limits of umbra and penumbra in section 5, while section 6 discusses methods using shafts with respect to volumes. Finally section 7 surveys methods developed for visibility in architectural environments where visibility information is propagated through sequences of openings.

1 Space partitioning

If all objects are convex, simple, well structured and aligned, visibility computations are much easier. This is why some methods attempt to fit the scene into simple enclosing volumes or regular spatial-subdivisions. Computations are simpler, occlusion cycles can no longer occur and depth ordering is easy.

1.1 Ray-tracing acceleration using a hierarchy of bounding volumes

Intersecting a ray with all objects is very costly. Whitted [Whi80] enclosed objects in *bounding volumes* for which the intersection can be efficiently computed (spheres in his paper). If the ray does not intersect the bounding volume, it cannot intersect the object.

Rubin and Whitted [RW80] then extended this idea with hierarchies of bounding volumes, enclosing bounding volumes in a hierarchy of successive bounding volumes. The trade-off between how the bounding volumes fits the object and the cost of the intersection has been studied by Weghorst *et al.* [WHG84] using a probabilistic approach based on surface ratios (see also section 4 of chapter 13). Kay and Kajiya [KK86] built tight-fitting bounding volumes which approximate the convex hull of the object by the intersection of parallel slabs.

The drawback of standard bounding volume methods, is that all objects intersecting the rays have to be tested. Kay and Kajiya [KK86] thus propose an efficient method for a traversal of the hierarchy which first tests the closest bounding volumes and terminates when an intersection is found which is closer than the remaining bounding volumes.

Many other methods were proposed to improve bounding volume methods for ray-tracing, see *e.g.* [Bou85, AK89, FvDFH90, Rog97, WW92]. See also [Smi99] for efficiency issues.

1.2 Ray-tracing acceleration using a spatial subdivision

The alternative to bounding volumes for ray-tracing is the use of a structured spatial subdivision. Objects of the scene are classified against *voxels* (boxes), and shooting a ray consists in traversing the voxels of the subdivision and performing intersections only for the objects inside the encountered voxels. An object can lie inside many voxels, so this has to be taken into account.

The trade-off here is between the simplicity of the subdivision traversal, the size of the structure and the number of objects per voxel.

Regular grids have been proposed by Fujimoto *et al.* [FTI86] and Amanatides and Woo [AW87]. The drawback of regular grids is that regions of high object density are “sampled” at the same rate as regions with many objects, resulting in a high cost for the latter because one voxel may contain many objects. However the traversal of the grid is very fast, similar to the rasterization of a line on a bitmap image. To avoid the time spent in traversing empty regions of the grid, the distance to the closest object can be stored at each voxel (see *e.g.* [CS94, SK97]).

Glassner [Gla84] introduced the use of octrees which result in smaller voxels in regions of high object density. Unfortunately the traversal of the structure becomes more costly because of the cost induced by the hierarchy of the octree. See [ES94] for a comparison between octrees and regular grids.

Recursive grids [JW89, KS97] are similar to octrees, except that the branching factor may be higher, which reduces the depth of the hierarchy (see Fig. 10.1(a)). The size of the voxel in a grid or sub-grid should be proportional to the cubic root of the number of objects to obtain a uniform density.

Snyder and Bar [SB87] use tight fitting regular grids for complex tessellated objects which they insert in a bounding box hierarchy.

Finally Cazals *et al.* [CDP95, CP97] propose the Hierarchy of Uniform Grids, where grids are not nested. Objects are sorted according to their size. Objects which are close and have the same size are clustered, and a grid is used for each cluster and inserted in a higher level grid (see Fig. 10.1(b)). An in-depth analysis of the performance of spatial subdivision methods is presented. Recursive grids and the hierarchy of uniform grid seem to be the best trade-off at the moment (see also [KWCH97, Woo97] for a discussion on this subject).

1.3 Volumetric visibility

The methods in the previous sections still require an intersection calculations for each object inside a voxel. In the context of radiosity lighting simulation, Sillion [Sil95] approximates visibility inside a voxel by an attenuation factor (transparency or *transmittance*) as is done for volume rendering. A multiresolution extension was presented [SD95] and will be discussed in section 1.2 of chapter 14.

The transmittance is evaluated using the area of the objects inside a voxel. These voxels (or *clusters*) are organised in a hierarchy. Choosing the level of the hierarchy used to compute the attenuation along a ray allows a trade-off between accuracy and time. The problem of refinement criteria will be discussed in section 1.1 of chapter 14.

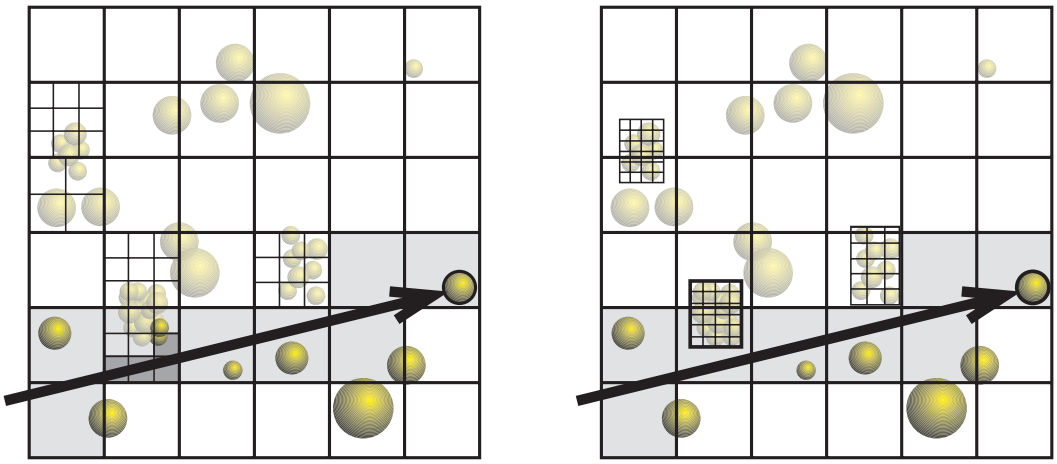


FIG. 10.1: A 2D analogy of ray-tracing acceleration. An intersection test is performed for objects which are in bold type. (a) Recursive grid. (b) Hierarchy of uniform grids. Note that fewer intersections are computed with the latter because the grids fit more tightly to the geometry.

Christensen *et al.* [CLSS97] propose another application of volumetric visibility for radiosity.

Chamberlain *et al* [CDL+96] perform real-time rendering by replacing distant geometry by semi-transparent regular voxels averaging the color and occlusion of their content. Neyret [Ney96, Ney98] presents similar ideas to model and render complex scenes with hierarchical volumetric representations called *texels*.

1.4 BSP trees

We have seen in section 4.2 of chapter 9 that an *a priori* depth order can be found for some objects. Unfortunately, this is quite rare. Fuchs and his co authors [FKN80, FAG83] have developed the *BSP* tree (Binary Space Partitioning tree) to overcome this limitation.

The principle is simple : if the scene can be separated by a plane, the objects lying on the same side of the plane as the viewer are closer than the others in a depth order. BSP trees recursively subdivide the scene along planes, resulting in a binary tree where each node corresponds to a splitting plane. The computation of a depth order is then a straightforward tree traversal : at each node the order in which the subtrees have to be drawn is determined by the side of the plane of the viewer. Unfortunately, since a scene is rarely separable by a plane, objects have to be split. Standard BSP approaches perform subdivision along the polygons of the scene. See Fig. 10.2 for an example[1].

It has been shown [PY90] that the split in BSP trees can cause the number of sub-polygons to be as high as $O(n^3)$ for a scene composed of n entry polygons. However, the choice of the order of the polygons with which subdivision is performed is very important. Paterson and Yao [PY90] give a method which builds a BSP tree with size $O(n^2)$. Unfortunately, it requires $O(n^3)$ time. However these bounds do not say much on the practical behaviour of BSPs.

See *e.g.* [NR95] for the treatment of curved objects.

Agarwal *et al.* [AGMV97, AEG98] do not perform subdivision along polygons. They build *cylindrical* BSP trees, by performing the subdivision along vertical planes going through edges of the scene (in a way similar to the method presented in the next section). They give algorithms which build a quadratic size BSP in roughly quadratic time.

Chen and Wang [CW96] have proposed the *feudal priority* algorithm which limits the number of splits compared to BSP. They first treat polygons which are back or front-facing from any other polygon, and then chose the polygons which cause the smallest number of splits.

[1] BSP trees have also been applied as a modeling representation tool and powerful *Constructive Solid Geometry* operations have been adapted by Naylor *et al.* [NAT90].

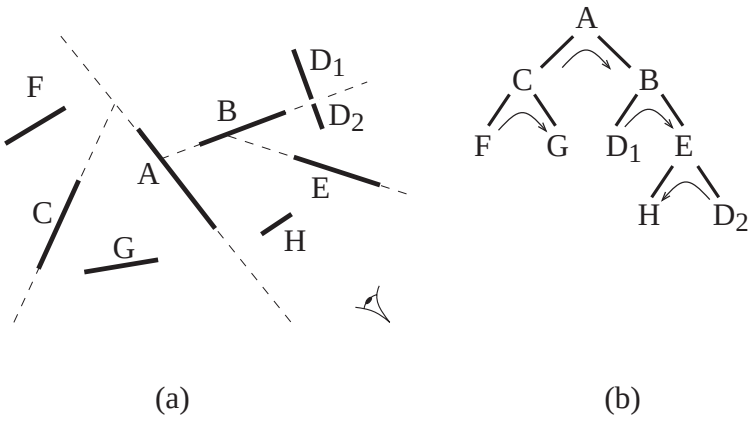


FIG. 10.2: 2D BSP tree. (a) The scene is recursively subdivided along the polygons. Note that polygon D has to be split. (b) Corresponding binary tree. The traversal order for the viewpoint in (a) is depicted using arrows. The order is thus, from back to front : $FCGAD_1BHED_2$

Naylor [Nay92] also uses a BSP tree to encode the image to perform occlusion-culling ; nodes of the object-space BSP tree projecting on a covered node of the image BSP are discarded in a manner similar to the hierarchical z-buffer which we will present in section 3 of the next chapter.

BSP trees are for example in the game *Quake* for the hidden-surface removal of the static part if the model [Abr96] (moving objects are treated using a z-buffer).

1.5 Cylindrical decomposition

Mulmuley [Mul91] has devised an efficient preprocessing algorithm to perform object-precision view computations using a sweep of the view map as presented in section 8 of chapter 9. However this work is theoretical and is unlikely to be implemented. He builds a cylindrical partition of 3D space which is similar to the BSPs that Agarwall *et al.* [AGMV97, AEG98] have later described. Nonetheless, he does not use whole planes. Each cell of his partition is bounded by parts of the input polygons and by vertical walls going through edges or vertices of the scene. His paper also contains an interesting discussion of sweep algorithms.

2 Path planning using the visibility graph

2.1 Path planning

Nilsson [Nil69] developed the first path planning algorithms. Consider a 2D polygonal scene. The *visibility graph* is defined as follows : The nodes are the vertices of the scene, and an arc joins two vertices A and B if they are mutually visible, *i.e.* if the segment $[AB]$ intersects no obstacle. As noted in the introduction, it is possible to go in straight line from A to B only if B is visible from A. The start and goal points are added to the set of initial vertices, and so are the corresponding arcs (see Fig. 10.3). Only arcs which are tangent to a pair of polygons are necessary.

It can be easily shown that the shortest path between the start point and the goal goes through arcs of the visibility graph. The rest of the method is thus a classical graph problem. See also [LPW79].

This method can be extended to non-polygonal scenes by considering bitangents and portions of curved objects.

The method unfortunately does not generalize simply to 3D where the problem has been shown to be NP-complete by Canny [Can88].

2.2 Visibility graph construction

The 2D visibility graph has size which is between linear and quadratic in the number of polygon edges. The construction of visibility graphs is a rich subject of research in computational geometry. Optimal $O(n^2)$

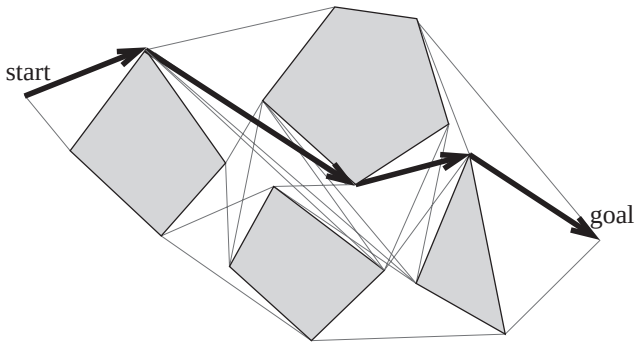


FIG. 10.3: Path planning using the visibility graph.

algorithms have been proposed [EG86] as well as *output-sensitive* approaches (their running time depends on the size of the output, *i.e.* the size of the visibility graph) [OW88, GM91].

The *2D visibility complex* which we will review in section 1.2 of chapter 13 is also a powerful tool to build visibility graphs.

In 3D, the term "visibility graph" often refers to the abstract graph where each object is a node, and where arcs join mutually visible objects. This is however not the direct equivalent of the 2D visibility graph.

2.3 Extensions to non-holonomic visibility

In this section we present some motion planning works which are hard to classify since they deal with extensions of visibility to curved lines of sight. They have been developed by Vendittelli *et al.* [VLN96] to plan the motion of a car-like robot. Car trajectories have a minimum radius of curvature, which constraints their motion. They are submitted to *non-holonomic* constraints : the tangent of the trajectory must be colinear to the velocity. Dubins [Dub57] and Reeds and Shepp [RS90] have shown that minimal-length trajectories of bounded curvature are composed of arcs of circles of minimum radius and line segments.

For example if a car lies at the origin of the plane and is oriented horizontally, the shortest path to the points of the upper quadrant are represented in Fig. 10.4(a). The rightmost paths are composed of a small arc of circle forward followed by a line segment. To go to the points on the left, a backward circle arc is first necessary, then a forward arc, then a line segment.

Now consider an obstacle such as the line segment represented in Fig. 10.4(a). It forbids certain paths. The points which cannot be reached are said to be in shadow, by analogy to the case where optimal paths are simple line segments[2].

The shape of such a shadow can be much more complex than in the line-visibility case, as illustrated in Fig. 10.4(b).

This analogy between visibility and reachability is further exploited in the paper by Nissoux *et al.* [NSL99] where they plan the motion of robots with arbitrary numbers of degrees of freedom.

3 The Visual Hull

The reconstruction of objects from silhouettes (see section 2.2 of chapter 7) is very popular because it is robust and simple. Remember that only exterior silhouettes are considered, folds caused by self occlusion of the object are not considered because they are harder to extract from images. Not all objects can be reconstructed with this method; The cavity of a bowl can not be reconstructed because it is not present on an external silhouette. The best reconstruction of a bowl one can expect is a "full" version of the initial object.

However the reconstructed object is not necessarily the convex hull of the object : the hole of a torus can be reconstructed because it is present on the exterior silhouette of some images.

[2]What we describe here are in fact shadows in a Riemannian geometry. Our curved lines of sight are in fact *geodesics*, *i.e.*c the shortest path from one point to another.

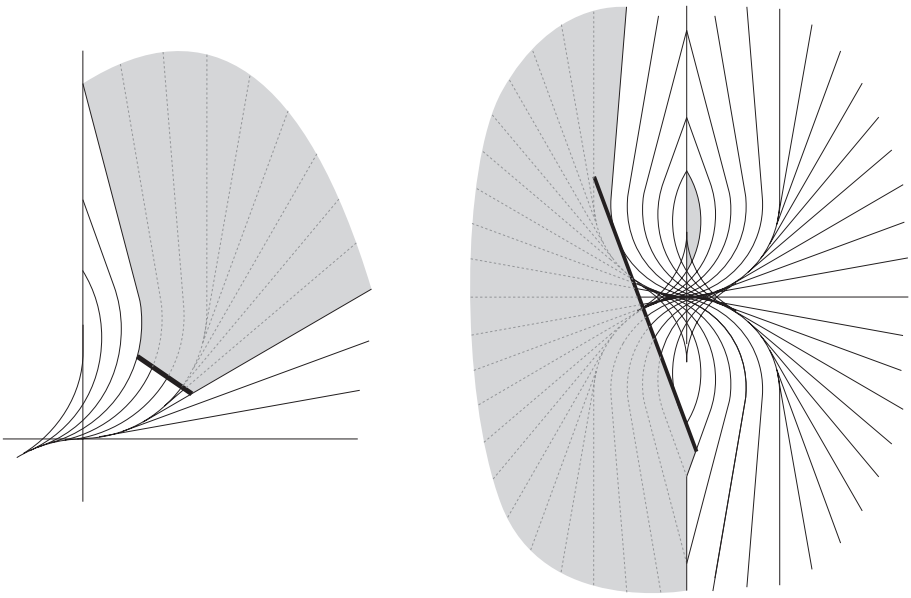


FIG. 10.4: Shadow for non-holonomic path-planning (adapted from [VLN96]). (a) Simple (yet curved) shadow. (b) Complex shadows. Some parts of the convex blocker do not lie on the shadow boundary. The small disconnected shadow is caused by the impossibility to perform an initial backward circle arc.

Laurentini [Lau94, Lau95, Lau97, Lau99] has introduced the *visual hull* concept to study this problem. A point P of space is inside the visual hull of an object A, if from any viewpoint P projects inside the projection of A. To give a line-space formulation, each line going through a point P of the visual hull intersects object A. The visual hull is the smallest object which can be reconstructed from silhouettes. See Fig. 10.5 for an example.

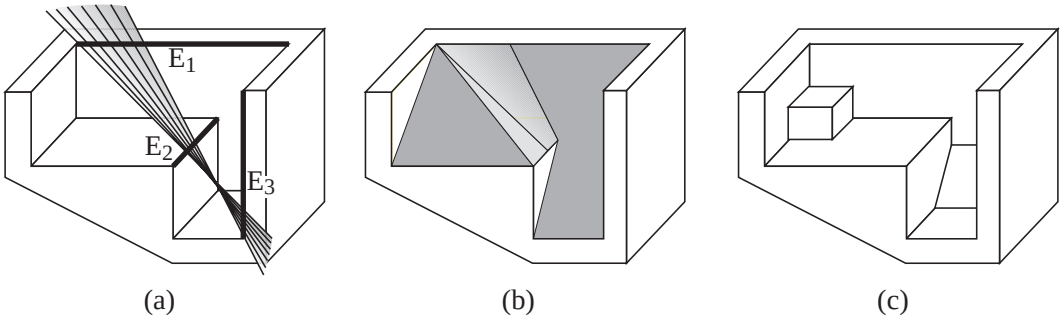


FIG. 10.5: Visual hull (adapted from [Lau94]). (a) Initial object. A EEE event is shown. (b) Visual hull of the object (the viewer is not allowed inside the convex hull of the object). It is delimited by polygons and a portion of the ruled quadric of the $E_1E_2E_3$ event. (c) A different object with the same visual hull. The two objects can not be distinguished from their exterior silhouette and have the same occlusion properties.

The exact definition of the visual hull in fact depends on the viewing region authorized. The visual hull is different if the viewer is allowed to go inside the convex hull of the object. (Half lines have to be considered instead of lines in our line-space definition)

The visual hull is delimited by visual events. The visual hull of a polyhedron is thus not necessarily a polyhedron, as shown in Fig. 10.5 where a EEE event is involved.

Laurentini has proposed a construction algorithms in 2D [Lau94] and for objects of revolution in 3D [Lau99]. Petitjean [Pet98] has developed an efficient construction algorithm for 2D visual hulls using the visibility graph.

The visual hull also represents the maximal solid with the same occlusion properties as the initial object.

This concept thus completely applies to the simplification of occluders for occlusion culling. The simplified occluder does not need to lie inside the initial occluder, but inside its visual hull. See the work by Law and Tan [LT99] on occluder simplification.

4 Shadows volumes and beams

In this section we present the rich category of methods which perform visibility computation using pyramids or cones. The apex can be defined by the viewpoint or by a point light source. It can be seen as the volume occupied by the set of rays emanating from the apex and going through a particular object. The intersection of such a volume with the scene accounts for the occlusion effects.

4.1 Shadow volumes

Shadow volumes have been developed by Crow [Cro77] to compute hard shadows. They are pyramids defined by a point light source and a blocker polygon. They are then used in a scan-line renderer as illustrated in Fig. 10.6.

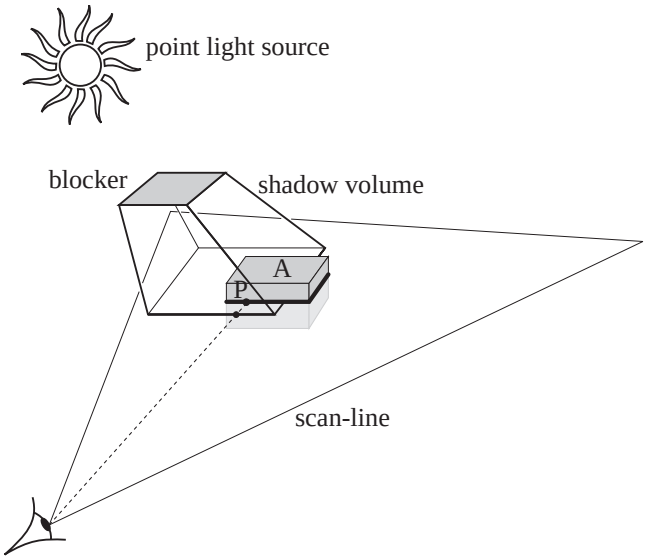


FIG. 10.6: Shadow volume. As object *A* is scan converted on the current scan-line, the shadowing of each pixel is computed by counting the number of back-facing and front-facing shadow volume polygons on the line joining it to the viewpoint. For point P, there is one front-facing intersection, it is thus in shadow.

The wedges delimiting shadow volumes are in fact visual events generated by the point light source and the edges of the blockers. In the case of a polyhedron light source, only silhouette edges (with respect to the source) need to be considered to build the shadow volume polygons.

Bergeron [Ber86] has proposed a more general version of Crow's shadow volumes. His method has long been very popular for production rendering.

Shadow volumes have also been used with ray-tracing [EK89]. Brotman and Badler [BB84] have presented a z-buffer based use of shadow volumes. They first render the scene in a z-buffer, then they build the shadow volumes and scan convert them. Instead of displaying them, for each pixel they keep the number of frontfacing and backfacing shadow volume polygons. This method is hybrid object-space and image space, the advantage over the shadow map is that only one sampling is performed. They also sample an area light source with points and add the contributions computed using their method to obtain soft shadow effects. An implementation using current graphics hardware is described in [MBGN98] section 9.4.2. A hardware implementation has also been developed on *pixel-plane* architecture [FGH+85], except that shadow volumes are simply described as plane-intersections.

Shadow volumes can also be used inversely as light-volumes to simulate the scattering of light in dusty air (*e.g.*, [NMN87, Hai91]).

Albrecht Dürer [Dür38] describes similar constructions, as shown in Fig. 10.7

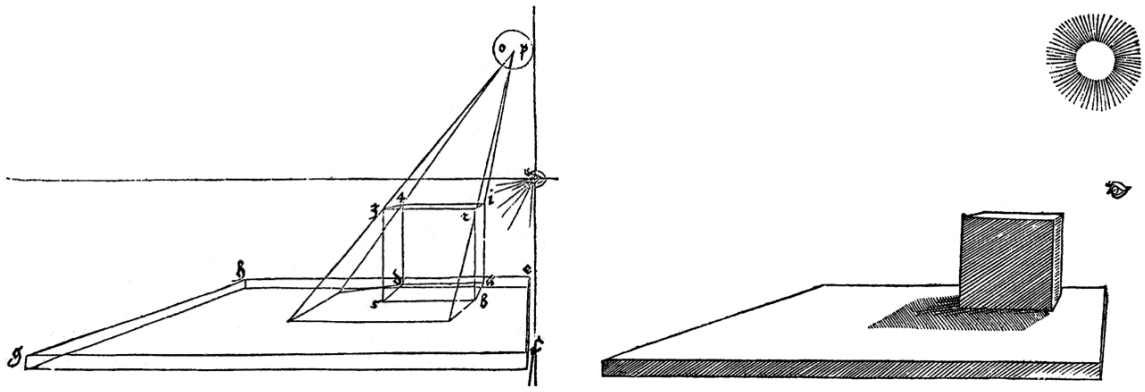


FIG. 10.7: Construction of the shadow of a cube by Dürer.

4.2 Shadow volume BSP

Chin and Feiner [CF89] compute hard shadows using BSP trees. Their method can be compared to Atherton *et al.*'s technique presented in section 2.1 of chapter 9 where the same algorithm is used to compute the view and to compute the illuminated parts of the scene. Two BSP are however used : one for depth ordering, and one called *shadow BSP tree* to classify the lit and unlit regions of space.

The polygons are traversed from front to back from the light source (using the first BSP) to build a shadow BSP tree. The shadow BSP tree is split along the planes of the shadow volumes. As a polygon is considered, it is first classified against the current shadow BSP tree (Fig. 10.8(a)). It is split into lit and unlit parts. Then the edges of the lit part are used to generate new splitting planes for the shadow BSP tree (Fig. 10.8 (b)).

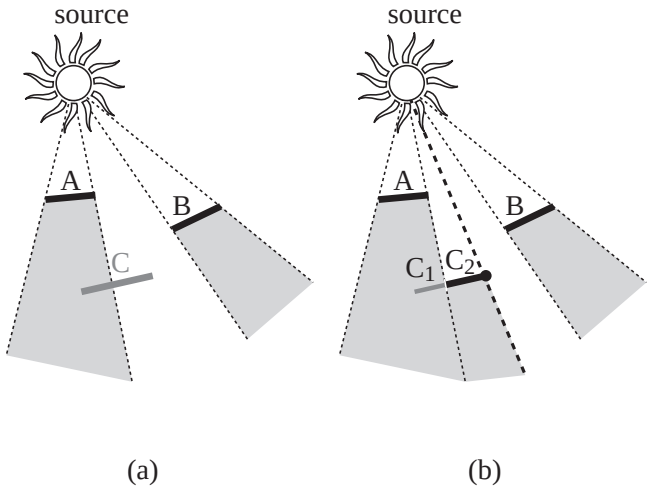


FIG. 10.8: 2D equivalent of shadow BSP. The splitting planes of the shadow BSP are represented with dashed lines. (a) Polygon C is tested against the current shadow BSP. (b) It is split into a part in shadow C_1 and a lit part C_2. The boundary of the lit part generates a new splitting plane.

The scene augmented with shadowing information can then be rendered using the standard BSP.

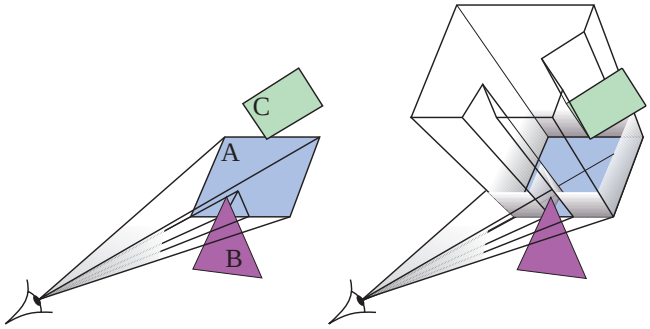
Chrysanthou and Slater [CS95] propose a method which avoids the use of the scene BSP to build the shadow BSP, resulting in fewer splits.

Campbell and Fussel [CF90] were the first to subdivide a radiosity mesh along shadow boundaries using BSPs. A good discussion and some improvements can be found in Campbell's thesis [Cam91].

4.3 Beam-tracing and bundles of rays

Heckbert and Hanrahan [HH84] developed *beam tracing*. It can be seen as a hybrid method between Weiler and Atherton's algorithm [WA77], Whitted's ray-tracing [Whi80] and shadow volumes.

Beams are traced from the viewpoint into the scene. One initial beam is cast and clipped against the scene polygons using Weiler and Atherton's exact method, thus defining smaller beams intersecting only one polygon (see Fig. 10.9(a)). If the a polygon is a mirror, a reflection beam is recursively generated. Its apex is the symmetric to the viewpoint with respect to the light source (Fig. 10.9(b)). It is clipped against the scene, and the computation proceeds.



(a) (b)

FIG. 10.9: Beam tracing. (a) A beam is traced from the eye to the scene polygons. It is clipped against the other polygons. (b) Since polygon *A* is a mirror, a reflected beam is recursively traced and clipped.

Shadow beams are sent from the light source in a preprocess step similar to Atherton *et al*'s shadowing [AWG78]. Refraction can be approximated by sending refraction beams. Unfortunately, since refraction is not linear, this computation is not exact.

Dadoon *et al.* [DKW85] propose an efficient version optimized using BSP trees.

Amanatides [Ama84] and Kirk [Kir87] use cones instead of beams. *Cone-tracing* allows antialiasing as well as depth-of-field and soft shadow effects. The practical use of this method is however questionable because secondary cones are hard to handle and because object-cone intersections are difficult to perform. Shinya *et al.* [STN87] have formalized these concepts under the name of *pencil tracing*.

Beam tracing has been used for efficient specular sound propagation by Funkhouser and his co-author. [FCE+98]. A bidirectional version has also been proposed where beams are propagated both from the sound source and from the receiver [FMC99].They moreover *amortize* the cost of beam propagation as listeners and sources move smoothly.

Speer [SDB85] has tried to take advantage of the coherence of bundles of rays by building cylinders in free-space around a ray. If subsequent rays are within the cylinder, they will intersect the same object. Unfortunately his method did not procure the expected speed-up because the construction of the cylinders was more costly than a brute-force computation.

Beams defined by rectangular windows of the image can allow high-quality antialiasing with general scenes. Ghazanfarpour and Hasenfratz [GH98, Has98] classify non-simple pixels in a manner similar to the A-buffer or to the ZZ-buffer, but they take shadows, reflection and refraction into account.

Teller and Alex [TA98] propose the use of beam-casting (without reflection) in a real-time context. Beams are adaptively subdivided according to a time budget, permitting a trade-off between time and image quality.

Finally Watt [Wat90] traces beams from the light source to simulate caustic effects which can for example be caused by the refraction of light in water.

4.4 Occlusion culling from a point

Sometimes, nearby large objects occlude most of the scene. This is the case in a city where nearby facades hide most of the buildings. Coorg and Teller [CT96, CT97b] quickly reject the objects hidden by some convex polygonal occluders. The scene is organised into an octree. A Shadow volume is generated for each occluder, and the cells of the octree are recursively classified against it as occluded, visible or partially visible, as illustrated in Fig. 10.10.

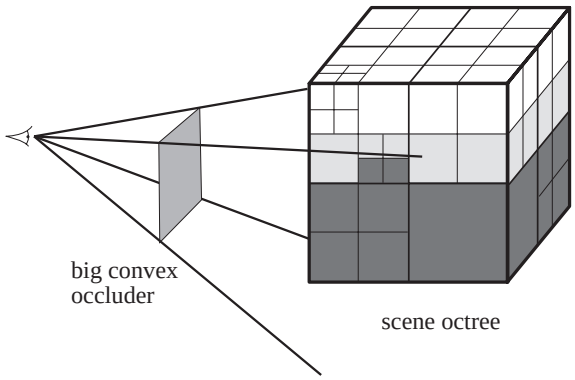


FIG. 10.10: Occlusion culling with large occluders. The cells of the scene octree are classified against the shadow volumes. In dark grey we show the hidden cells, while those partially occluded are in light grey.

The occlusion by a conjunction of occluders in not taken into account, as opposed to the hierarchical z-buffer method exposed in section 3 of chapter 11. However we will see in section 4.2 of chapter 12 that they treat frame-to-frame coherence very efficiently.

Similar approaches have been developed by Hudson *et al.* [HMC+97]. Bittner *et al.* [BHS98] use shadow volume BSP tree to take into account the occlusion caused by multiple occluders.

Woo and Amanatides [WA90] propose a similar scheme to speed-up hard shadow computation in ray-tracing. They partition the scene in a regular grid and classify each voxel against shadow volumes as completely lit, completely in umbra or complicated. Shadow rays are then sent only from complicated voxels.

Indoor architectural scenes present the dual characteristic feature to occlusion by large blockers : one can see outside a room only through doors or windows. These opening are named *portals*. Luebke and George [LG95] following ideas by Jones [Jon71] and Clark [Cla76] use the portals to reject invisible objects in adjacent rooms. The geometry of the current room is completely rendered, then the geometry of adjacent rooms is tested against the screen bounding box of the portals as shown in Fig. 10.11. They also apply their technique to the geometry reflected by mirrors.

This technique was also used for a walk through a virtual colon for the inspection of acquired medical data [HMK+97] and has been implemented in a 3D game engine [BEW+98].

4.5 Best-next-view

Best-next-view methods are used in model reconstruction to infer the position of the next view from the data already acquired. The goal is to maximize the visibility of parts of the scene which were occluded in the previous view. They are delimited by the *volume of occlusion* as represented in Fig. 10.12. These volumes are in fact the *shadow volumes* where the camera is considered as a light source.

Reed and Allen [RA96] construct a BSP model of the object as well as the boundaries of the occlusion volume. They then attempt to maximize the visibility of the latter. This usually results roughly in a 90° rotation of the camera since the new viewpoint is likely to be perpendicular to the view volume.

Similar approaches have been developed by Maver and Bajcsy [MB93] and Banta *et al.* [BZW+95].

This problem is very similar to the problem of gaps in image-based view warping (see section 1.7 of chapter 7 and Fig. 7.7 page 150). When a view is reprojected, the regions of indeterminate visibility lie on the boundary of the volumes of occlusion.

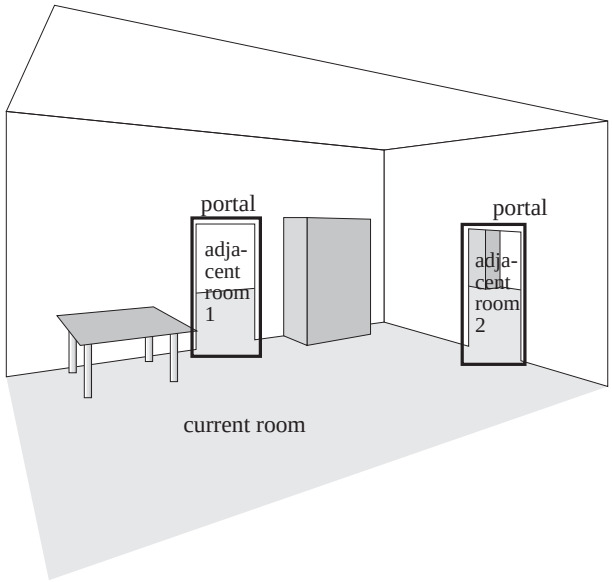


FIG. 10.11: Occlusion culling using image-space portals. The geometry of the adjacent rooms is tested against the screen bounding boxes of the portals

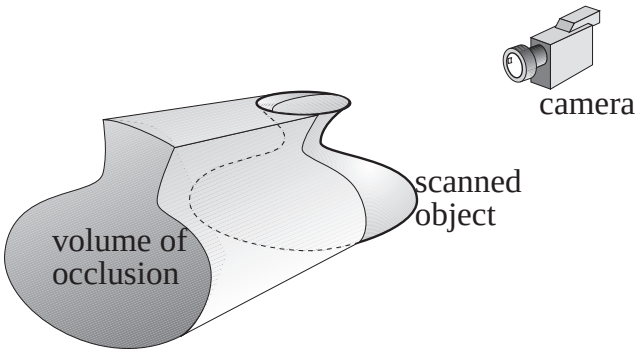


FIG. 10.12: Acquisition of the model of a 3D object using a range image. The volume of occlusion is the unknown part of space.

5 Area light sources

5.1 Limits of umbra and penumbra

Nishita and Nakamae [NN85, NON85, NN83] have computed the regions of umbra and penumbra caused by convex blockers. They show that the umbra from a polygonal light source of a convex object is the intersection of the umbra volumes from the vertices of the source (see Fig. 10.13). The penumbra is the convex hull of the union of the umbra volumes. They use Crow's shadow volumes to compute these regions.

The umbra is bounded by portions of EV events generated by one vertex of the source and one edge of the blocker, while the penumbra is bounded EV events generated by edges and vertices of both the source and the blocker.

Their method fails to compute the exact umbra caused by multiple blockers, since it is no longer the intersection of their umbras. The penumbra boundary is however valid, but some parts of the umbra are incorrectly classified as penumbra. This is not a problem in their method because a shadow computation is performed in the penumbra region (using an exact hidden line removal method). The umbra of a concave object is bounded by EV visual events and also by EEE events (for example in Fig. 8.5 page 165 if polygon R is a source, the EEE event exhibited is an umbra boundary). Penumbra regions are bounded only by EV events.

Drawings by da Vinci exhibit the first description of the limits of umbra and penumbra (Fig. 10.14).

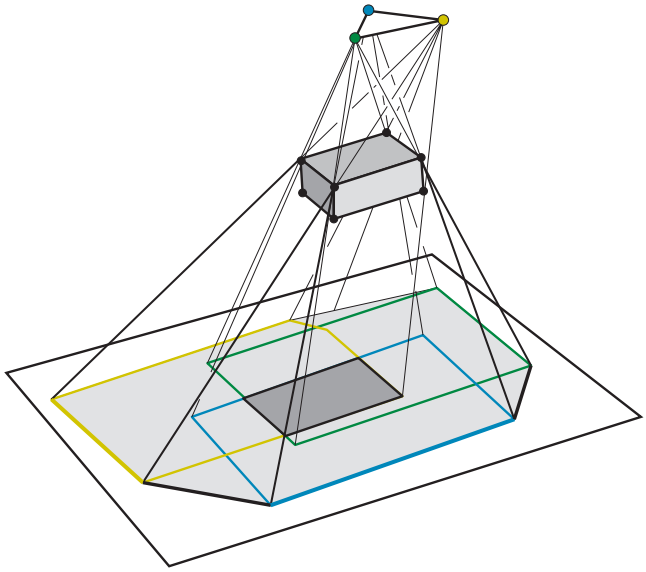


FIG. 10.13: Umbra (dark grey) and penumbra (light grey) of a convex blocker (adapted from [NN85]).

5.2 BSP shadow volumes for area light sources

Chin and Feiner [CF92] have extended their BSP method to handle area light sources. They build two shadow BSP, one for the umbra and one for the penumbra.

As in Nishita and Nakamae's case, their algorithm does not compute the exact umbra volume due to the occlusion by multiple blockers.

5.3 Discontinuity meshing

Heckbert [Hec92b, Hec92a] has introduced the notion of discontinuity meshing for radiosity computations. At a visual event, a C^2 discontinuity occurs in the illumination function (see [Arv94] for the computation of illumination gradients). Heckbert uses EV discontinuity surfaces with one generator on the source.

Other authors [LTG93, LTG92, Stu94, Cam91, CF91a, GH94] have used similar techniques. See Fig. 10.15 for an example. Hardt and Teller [HT96] also consider discontinuities which are caused by indirect lighting. Other discontinuity meshing techniques will be treated in section 2.3 of chapter 12 and 2.1 of chapter 13.

However, discontinuity meshing approaches have not yet been widely adopted because they are prone to robustness problems and also because the irregular meshes induced are hard to handle.

5.4 Linear time construction of umbra volumes

Yoo *et al.* [YKSC98] perform the same umbra/penumbra classification as Nishita and Nakamae, but they avoid the construction and intersection/union of all the shadow volumes from the vertices of the source.

They note that only EV events on separating and supporting planes have to be considered. Their algorithm walks along the chain of edges and vertices simultaneously on the source and on the blocker as illustrated in Fig. 10.16.

This can be interpreted in line space as a walk along the chain of 1 dimensional sets of lines defined by visual events.

Related methods can be found in [Cam91, TTK96].

5.5 Viewpoint constraints

As we have seen, viewpoint optimisation is often performed for the monitoring of robotics tasks. In this setting, the visibility of a particular feature of object has to be enforced. This is very similar to the computation of shadows considering that the feature is an extended light source.

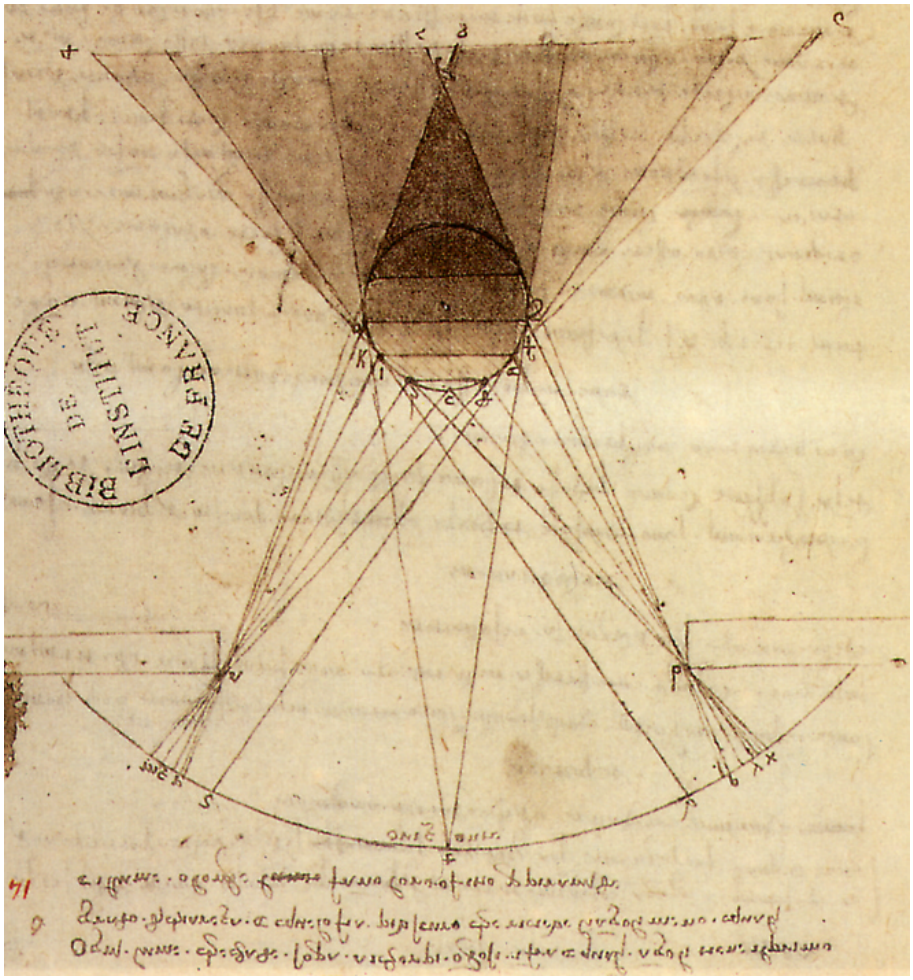


FIG. 10.14: Penumbra by Leonardo da Vinci (Manuscript). Light is coming from the lower window, and the sphere causes soft shadows.

Cowan and Kovesi [CK88] use an approach similar to Nishita and Nakamae. They compute the penumbra region caused by a convex blocker as the intersection of the half spaces defined by the separating planes of the feature and blockers (*i.e.* planes tangent to both objects such that each object lies on a different side of the plane). The union of the penumbra of all the blockers is taken and constraints related to the sensor are then included : resolution of the image, focus, depth of field and view angle. The admissible region is the intersection of these constraints.

Briggs and Donald [BD98] propose a 2D method which uses the intersection of half-planes defined by bitangents. They also reject viewpoints from which the observation can be ambiguous because of similarities in the workspace or in the object to be manipulated.

Tarabanis and Tsai [TTK96] compute occlusion free viewpoints for a general polyhedral scene and a general polygonal feature. They enumerate possible *EV* wedges and compute their intersection.

Kim *et al.* [KYCS98] also present an efficient algorithm which computes the complete visibility region of a convex object.

5.6 Light from shadows

Poulin *et al.* [PF92, PRJ97] have developed inverse techniques which allow a user to sketch the positions of shadows. The position of the light source is then automatically deduced.

The principle of shadow volumes is reversed : A point *P* lies in shadow if the point light source is in a shadow volume emanating from point *P*. The sketches of the user thus define constraints under the form of an

(a) (b)

FIG. 10.15: Global illumination simulation. (a) Without discontinuity meshing. Note the jagged shadows. (b) Using discontinuity meshing, shadows are finer (images courtesy of Dani Lischinski, Program of Computer Graphics, Cornell University).

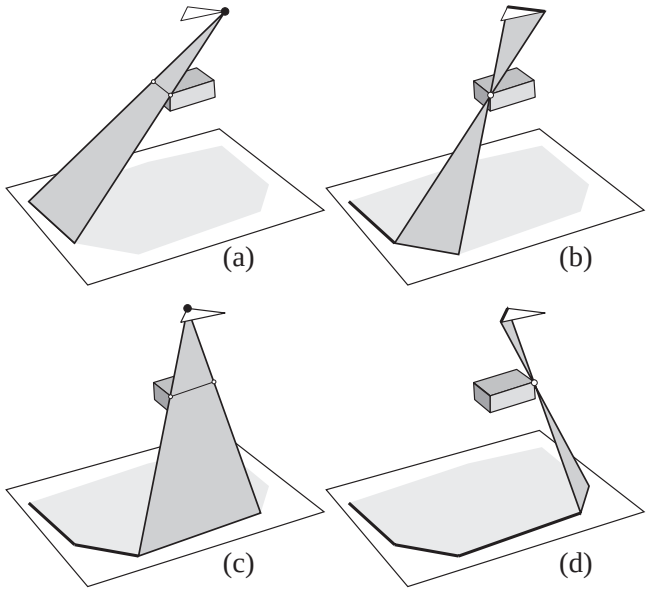


FIG. 10.16: Linear time construction of a penumbra volume.

intersection of shadow volumes (see Fig. 10.17).

Their method can also handle soft shadows, and additional constraints such as the position of highlights.

6 Shafts

Shaft method are based on the fact that occlusion between two objects can be caused only by objects inside their convex hull. Shafts can be considered as finite beams for which the apex is not a point. They can also be

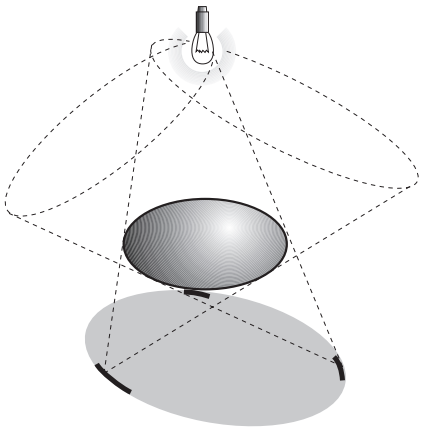


FIG. 10.17: Sketching shadows. The user specifies the shadows of the ellipsoid on the floor with the thick strokes. This generates constraint cones (dashed). The position of the light source is then deduced (adapted from [PRJ97]).

seen as the volume of space defined by the set of rays between two objects.

6.1 Shaft culling

Haines and Wallace [HW91] have developed shaft culling in a global illumination context to speed up form factor computation using ray-casting. They define a shaft between two objects (or patches of the scene) as the convex hull of their bounding box (see Fig. 10.18).

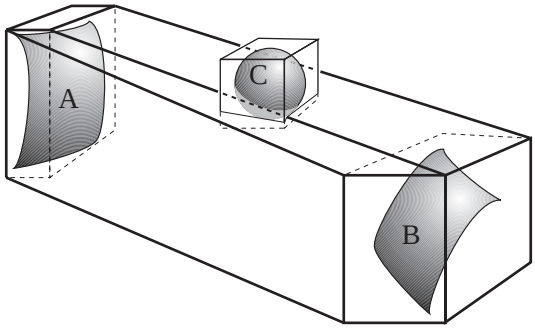


FIG. 10.18: Shaft culling. The shaft between *A* and *B* is defined as the convex hull of the union of their bounding boxes. Object *C* intersects the shaft, it may thus cause occlusion between *A* and *B*.

They have developed an efficient construction of approximate shafts which takes advantage of the axis aligned bounding boxes. The test of an object against a shaft is also optimized for bounding boxes.

Similar methods have been independently devised by Zhang [Zha91] and Campbell [Cam91].

Marks *et al* [MWCF90], Campbell [Cam91] and Drettakis and Sillion [DS97] have derived hierarchical versions of shaft culling. The hierarchy of shafts is implicitly defined by a hierarchy on the objects. This hierarchy of shaft can also be seen as a hierarchy in line-space [DS97]. Brière and Poulin [BP96] also use a hierarchy of shafts or tubes to accelerate incremental updates in ray tracing.

6.2 Use of a dual space

Zao and Dobkin [ZD93] use shaft culling between pairs of triangles. They speed up the computation by the use of a multidimensional dual space. They decompose the shaft between a pair of triangles into tetrahedra and derive the conditions for another triangle to intersect a tetrahedron. These conditions are linear inequalities depending on the coordinates of the triangle.

They use multidimensional spaces depending on the coordinates of the triangles to speed up these tests. The queries in these spaces are optimized using binary trees (kd-trees in practice).

6.3 Occlusion culling from a volume

Cohen-Or and his co-authors [COFHZ98, COZ98] compute *potentially visible sets* from viewing cells. That is, the part of the scene where the viewer is allowed (the viewing space in short) is subdivided into cells from which the set of objects which may be visible is computed. This method can thus be seen as a viewpoint space method, but the core of the computation is based on the shaft philosophy.

Their method detects if a convex occluder occludes an object from a given cell. If convex polygonal objects are considered, it is sufficient to test if all rays between pairs of vertices are blocked by the occluder. The test is early terminated as soon as a non-blocked ray is found. It is in fact sufficient to test only silhouette rays (a ray between two point is a silhouette ray if each point is on the silhouette as seen from the other).

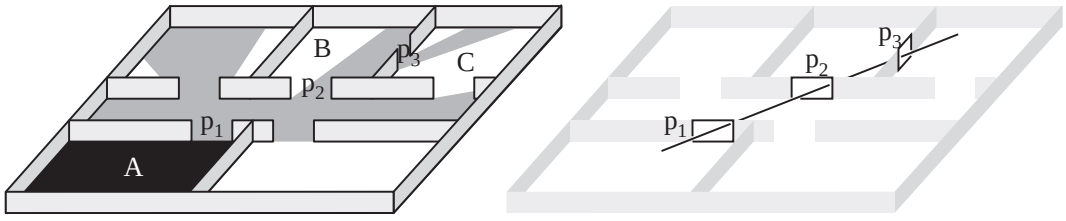
The drawback of this method is that it can not treat the occlusion caused by many blockers. The amount of storage required by the potentially visible set information is also a critical issue, as well as the cost of ray-casting.

7 Visibility propagation through portals

As already introduced, architectural scenes are organized into rooms, and inter-room visibility occurs only along openings named *portals*. This makes them particularly suitable for visibility preprocessing. Airey [Air90] and Teller [Tel92b, TS91] decompose a building into cells (roughly representing rooms) and precompute *Potentially Visible Sets* for each set. These are superset of objects visible from the cell which will then typically be sent to a z-buffer in a walkthrough application (see below).

7.1 Visibility computation

We describe here the methods proposed by Teller [Tel92b]. An adjacency graph is built connecting cells sharing a portal. Visibility is then propagated from a cell to neighbouring cells through portal sequences in a depth-first manner. Consider the situation illustrated in Fig. 10.19(a). Cell B is visible from cell A through the sequence of portals $p_1 p_2$. Cell C is neighbour of B in the adjacency graph, its visibility from A is thus tested. A sightline stabbing the portals p_1, p_2 and p_3 is searched (see Fig. 10.19(b)). A *stab-tree* is built which encodes the sequences of portals.



(a) (b)

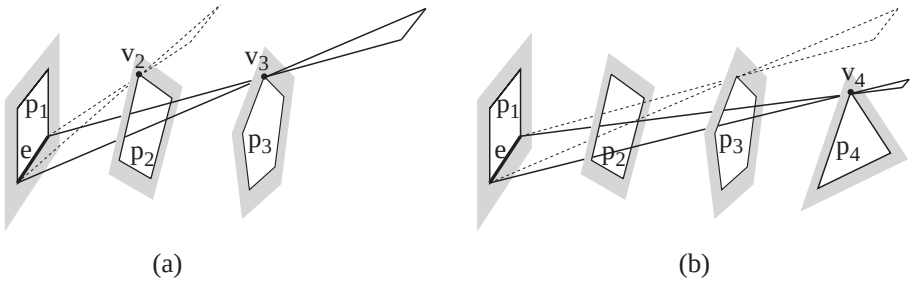
FIG. 10.19: Visibility computations in architectural environments. (a) In grey : part of the scene visible from the black cell. (b) A stabbing line (or sightline) through a sequence of portals.

If the scene is projected on a floorplan, this stabbing problem reduces to find a stabber for a set of segments and can be solved using linear programming (see [Tel92b, TS91]).

If rectangular axis-aligned portals are considered in 3D, Teller [Tel92b] shows that the problem can be solved by projecting it in 2D along the three axis directions.

If arbitrary oriented portals are computed, he proposes to compute a conservative approximation to the visible region [Tel92b, TH93]. As each portal is added to the sequence, the EV events bounding the visibility region are updated. These EV events correspond to separating planes between the portals. For each edge of

the sequence of portals, only the extremal event is considered. The process is illustrated Fig. 10.20. It is a conservative approximation because EEE boundaries are not considered.



(a) (b)

FIG. 10.20: Conservative visibility propagation through arbitrary portals. (a) The separating plane considered for e is generated by v_3 because it lies below the one generated by v_2. (b) As a new portal is added to the sequence, the separating plane is updated with the same criterion.

If the visibility region is found to be empty, the new cell is not visible from the current cell. Otherwise, objects inside the cell are tested for visibility against the boundary of the visibility region as in a shaft method.

Airey [Air90] also proposes an approximate scheme where visibility between portals is approximated by casting a certain number of rays (see section 4 of chapter 13 for the approaches involving sampling with rays). See also the work by Yagel and Ray [YR96] who describe similar ideas in 2D.

The portal sequence can be seen as a sort of infinite shaft. We will also study it as the set of lines going through the portals in section 3.3 of chapter 13.

7.2 Applications

The primary focus of these potentially visible sets methods was the use in walkthrough systems. Examples can be found in both Airey [ARB90] and Teller's thesis [TS91, Tel92b]. Teller also uses an online visibility computation which restricts the visible region to the current viewpoint. The stab-tree is used to speed up a beam-like computation.

Funkhouser *et al.* [FS93] have extended Teller's system to use other rendering acceleration techniques such as mesh simplification in a real time context to obtain a constant framerate. He and his co-authors [FST92, Fun96c] have also used the information provided by the potentially visible sets to efficiently load from the disk or from the network only the parts of the geometry which may become visible in the subsequent frames. It can also be used in a distributed virtual environment context to limit the network bandwidth to messages between clients who can see each other [Fun95].

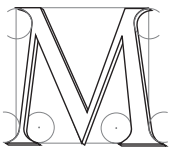
These computations have also been applied to speed-up radiosity computations by limiting the calculation of light interactions between mutually visible objects [TH93, ARB90]. It also permits lighting simulations for scenes which cannot fit into memory [TFFH94, Fun96b].

CHAPITRE 11

Image-Space

L'art de peindre n'est que l'art d'exprimer l'invisible par le visible

Eugène FROMENTIN



OST OF the image-space methods we present are based on a discretisation of an image. They often take advantage of the specialised hardware present in most of today's computers, which makes them simple to implement and very robust. Sampling rate and aliasing are however often the critical issues. We first present some methods which detect occlusions using projections on a sphere or on planes. Section 1 deals with the use of the z-buffer hardware to speed-up visibility computation. We then survey extensions of the z-buffer to perform occlusion-culling. Section 4 presents the use of a z-buffer orthogonal to the view for occlusion-culling for terrain-like scenes. Section 5 presents epipolar geometry and its use to perform view-warping without depth comparison. Section 6 discusses the computation of soft shadow using convolution, while section 7 deals with shadow-coherence in image-space.

1 Projection methods

1.1 Shadow projection on a sphere

Bouknight and Kelly [BK70] propose an optimization to compute shadows during a scan-line process as presented in section 6 of chapter 9. Their method avoids the need to intersect the wedge defined by the current span and the light source with all polygons of the scene.

As a preprocess, the polygons of the scene are projected onto a sphere centered at the point light source. A polygon can cast shadows on another polygon only if their projections overlap. They use bounding-box tests to speed-up the process.

Slater [Sla92] proposes a similar scheme to optimize the classification of polygons in shadow volume BSPs. He uses a discretized version of a cube centered on the source. Each *tile* (pixel) of the cube stores the polygon which project on it. This speeds up the determination of overlapping polygons on the cube. This shadow tiling is very similar to the light-buffer and to the hemicube which we will present in section 2.

1.2 Area light sources

Chrysanthou and Slater [CS97] have extended this technique to handle area light sources. In the methods presented above, the size of the sphere or cube does not matter. This is not the case of the extended method : a cube is taken which encloses the scene.

For each polygon, the projection used for point light sources becomes the intersection of its *penumbra volume* with the cube. The polygons with which it interacts are those which project on the same tiles.

1.3 Extended projections

The extended projection method proposed in chapter 5 of this thesis can be seen as an extension of the latter technique to perform offline occlusion culling from a volumetric cell (it can also be seen as an extension of Greene's hierarchical z-buffer surveyed in section 3). The occluders and occludees are projected onto a projection plane using *extended projection operators*. The extended projection of an occluder is the intersection of its views from all the viewpoints inside the cell. The extended projection of an occludee is the union of its views (similar to the penumbra used by Chrysanthou *et al.*).

If the extended projection of an occludee is in the cumulative extended projection of some occluders (and if it lies behind them), then it is ensured that it is hidden from any point inside the cell. This method handles *occluder fusion*.

2 Advanced z-buffer techniques

The versatility and robustness of the z-buffer together with efficient hardware implementations have inspired many visibility computation and acceleration schemes[1]. The use of the frame-buffer as a computational model has been formalized by Fournier and Fussel [FF88].

2.1 Shadow maps

As evoked in section 1.2 of chapter 7, hard shadow computation can be seen as the computation of the points which are visible from a point-light source. It is no surprise then that the z-buffer was used in this context.

FIG. 11.1: Shadow map principle. A shadow map is computed from the point of view of the light source (z-values are represented as grey levels). Then each point in the final image is tested for shadow occlusion by projecting it back in the shadow map (gallion model courtesy of Viewpoint Datalab).

A two pass method is used. An image is first computed from the source using a z-buffer. The z values of the closest points are stored in a depth map called *shadow map*. Then, as the final image is rendered, deciding if a point is in shadow or not consists in projecting it back to the shadow map and comparing its distance to

[1] Unexpected applications of the z-buffer have also been proposed such as 3D motion planning [LRDG90], Voronoi diagram computation [Hae90, ICK+99] or collision detection [MOK95].

the stored z value (similarly to shadow rays, using the depth map as a query data-structure). The shadow map process is illustrated in Fig 11.1. Shadow maps were developed by Williams [Wil78] and have the advantage of being able to treat any geometry which can be handled by a z-buffer. Discussions of improvements can be found in [Gra92, Woo92].

The main drawback of shadow masks is aliasing. Standard filtering can not be applied, because averaging depth values makes no sense in this context. This problem was addressed by Reeves *et al.* [RSC87]. Averaging the depth values of the neighbouring pixels in the shadow map before performing the depth comparison would make no sense. They thus first compare the depth value with that of the neighbouring pixels, then they compute the average of the binary results. Had-oc soft shadows are obtained with this filtering, but the size of the penumbra is arbitrary and constant. See also section 6 for soft computation using an image-space shadow-map.

Soft shadow effects can be also achieved by sampling an extended light source with point light sources and averaging the contributions [HA90, HH97, Kel97]. See also [Zat93] for a use of shadow maps for high quality shadows in radiosity lighting simulation.

Shadow maps now seem to predominate in production. Ray tracing and shadow rays are used only when the artifacts caused by shadow maps are too noticeable. A hardware implementation of shadow maps is now available on some machines which allow the comparison of a texture value with a texture coordinate [SKvW$^+$92][2].

Zhang [Zha98a] has proposed an inverse scheme in which the pixels of the shadow map are projected in the image. His approach consists in warping the view from the light source into the final view using the view warping technique presented in section 1.7 of chapter 7. This is similar in spirit to Atherton and Weiler's method presented in section 2.1 of chapter 9 : the view from the source is added to the scene database.

2.2 Ray-tracing optimization using item buffers

A z-buffer can be used to speed up ray-tracing computations. Weghorst *et al.* [WHG84] use a z-buffer from the viewpoint to speed up the computation of primary rays. An identifier of the objects is stored for each pixel (for example each object is assigned a unique color) in a so called *item buffer*. Then for each pixel, the primary ray is intersected only with the corresponding object. See also [Sun92].

Haines and Greenberg [HG86] propose a similar scheme for shadow rays. They place a *light buffer* centered on each point light source. It consists of 6 item buffers forming a cube (Fig. 11.2(a)). The objects of the scene are projected onto this buffer, but no depth test is performed, all objects projecting on a pixel are stored. Object lists are sorted according to their distance to the point light source. Shadow rays are then intersected only with the corresponding objects, starting with the closest to the source.

Poulin and Amanatides [PA91] have extended the light-buffer to linear light sources. This latter method is a first step towards line-space acceleration techniques that we present in section 1.4 of chapter 13, since it precomputes all objects intersected by the rays emanating from the light source.

Salesin and Stolfi [SS89, SS90] have extended the item buffer concept for ray-tracing acceleration. Their *ZZ-buffer* performs anti-aliasing through the use of an A-buffer like scheme. They detect completely covered pixels, avoiding the need for a subsampling of that pixel. They also sort the objects projecting on a non - simple pixel by their depth intervals. The ray-object intersection can thus be terminated earlier as soon as an intersection is found.

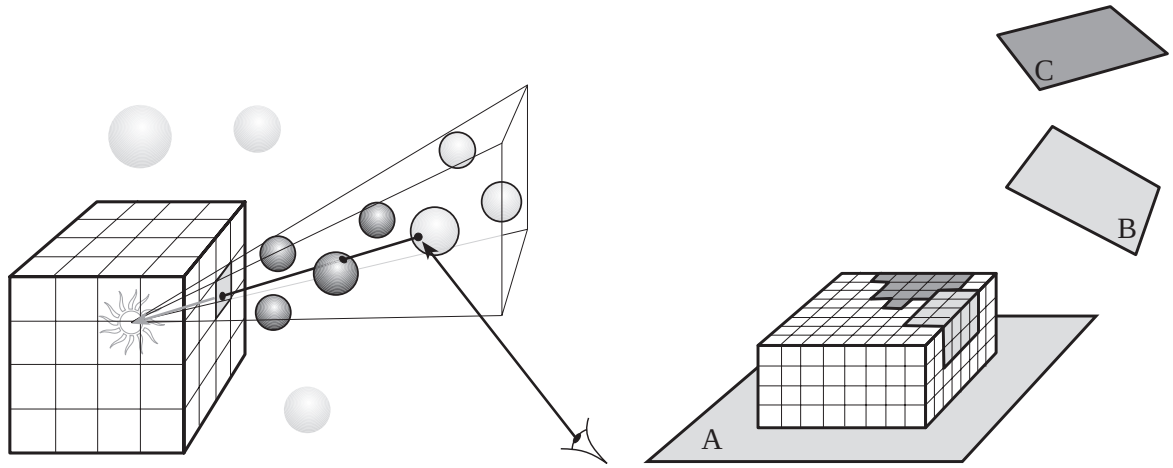
ZZ buffers can be used for primary rays and shadow rays. Depth of field and penumbra effects can also be obtained with a slightly modified ZZ-buffer.

In a commercial products such as Maya from Alias Wavefront [May99], an A-buffer and a ray-tracer are combined. The A-buffer is used to determine the visible objects, and ray-tracing is used only for pixels where high quality refraction or reflection is required, or if the shadow maps cause too many artifacts.

2.3 The hemicube

Recall that *form factors* are used in radiosity lighting simulations to model the proportion of light leaving a patch which arrives at another. The first method developed to estimate visibility for form factor computations

[2] A shadow map is computed from the point light source and copied into texture memory. The texture coordinate matrix is set to the perspective matrix from the light source. The initial u, v, w texture coordinate of a vertex are set to its 3D coordinates. After transformation, w represents the distance to the light source. It is compared against the texture value at u, v, which encodes the depth of the closest object. The key feature is the possibility to draw a pixel only if the value of w is smaller than the texture value at u, v.See [MBGN98] section 9.4.3. for implementation details.



(a) (b)

FIG. 11.2: (a) Light buffer. (b) Form factor computation using the hemicube. Five z-buffers are placed around the center of patch *A*. All form factors between *A* and the other patches are evaluated simultaneously, and occlusion of *C* by *B* is taken into account.

was the *hemicube* which uses five item-buffer images from the center of a patch as shown in Fig. 11.2(b). The form factor between one patch and all the others is evaluated simultaneously by counting the number of pixels covered by each patch.

The hemicube was introduced by Cohen *et al.* [CG85] and has long been the standard method for radiosity computations. However, as for all item buffer methods, sampling and aliasing problems are its main drawbacks. In section 2.2 of chapter 9 and section 4 of chapter 13 we present some solutions to these problems.

Sillion and Puech [SP89] have proposed an alternative to the hemicube which uses only one plane parallel the patch (the plane is however not uniformly sampled : A Warnock subdivision scheme is used.

Pietrek [Pie93] describe an anti-aliased version of the hemicube using a heuristic based on the variation between a pixel and its neighbours. See also [Mey90, BRW89]. Alonso and Holzschuch [AH97] present similar ideas as well as a deep discussion of the efficient access to the graphics hardware resources.

2.4 Sound occlusion and non-binary visibility

The wavelengths involved in sound propagation make it unrealistic to neglect diffraction phenomena. Simple binary visibility computed using ray-object intersection is far from accurate.

Tsingos and Gascuel [TG97a] use *Fresnel ellipsoids* and the graphics hardware to compute semi-quantitative visibility values between a sound source and a microphone. Sound does not propagate through lines ; Fresnel ellipsoids describe the region of space in which most of the sound propagation occurs. Their size depends on the sound frequency considered. Sound attenuation can be modeled as the amount of occluders present in the Fresnel ellipsoid. They use the graphics hardware to compute a view from the microphone in the direction of the source, and count the number of occluded pixels.

They also use such a view to compute diffraction patterns on an extended receiver such as a plane [TG97b]. One view is computed from the source, and then for each point on the receiver, and integral is computed using the z values of the view. The contribution of each pixel to diffraction is then evaluated (see Fig. 11.3 for an example).

3 Hierarchical z-buffer

The z-buffer is simple and robust, however it has linear cost in the number of objects. With the ever increasing size of scenes to display, *occlusion culling* techniques have been developed to avoid the cost incurred by objects which are not visible.

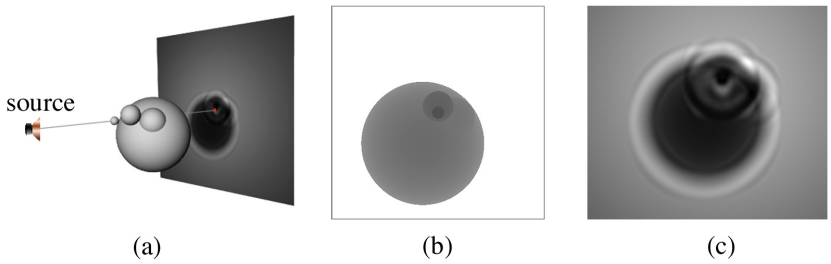


FIG. 11.3: Non binary visibility for sound propagation. The diffraction by the spheres of the sound emitted by the source causes the diffraction pattern on the plane. (a) Geometry of the scene. (b) z-buffer from the source. (c) Close up of the diffraction pattern of the plane. (Courtesy of Nicolas Tsingos, iMAGIS-GRAVIR).

Greene *et al.* [GKM93, Gre96] propose a hierarchical version of the z-buffer to quickly reject parts of the scene which are hidden. The scene is partitioned to an octree, and cells of the octree are rendered from front to back (the reverse of the original *painter algorithm*, see *e.g.* [FvDFH90, Rog97] or section 4 of chapter 9) to be able to detect the occlusion of back objects by frontmost ones. Before it is rendered, each cell of the octree is tested for occlusion against the current z values. If the cell is occluded, it is rejected, otherwise its children are treated recursively.

The z-buffer is organised in a pyramid to avoid to test all the pixels of the cell projection. Fig. 11.4 shows the principle of the hierarchical z-buffer.

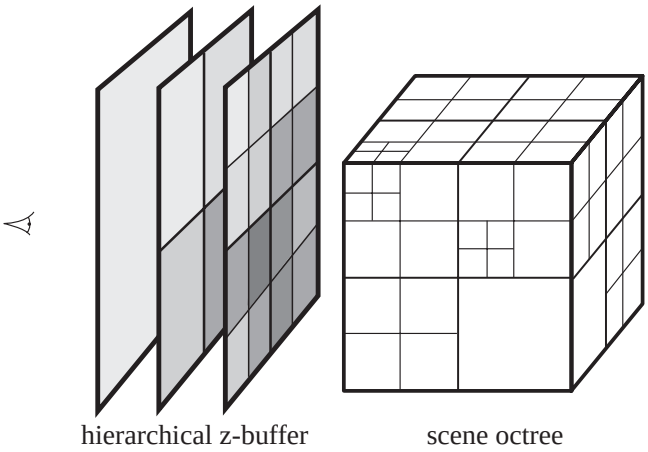


FIG. 11.4: Hierarchical z-buffer.

The hierarchical z-buffer however requires many z-value queries to test the projection of cells and the maintenance of the z-pyramid ; this can not be performed efficiently on today's graphics hardware. Zhang *et al.* [ZMHH97, Zha98b] have presented a two pass version of the hierarchical z-buffer which they have successfully implemented using available graphics hardware. They first render a subset of close and big objects called occluders, then read the frame buffer and build a so-called *hierarchical occlusion map* against which they test the bounding boxes of the objects of the scene. This method has been integrated in a massive model rendering system system [ACW+99] in combination with geometric simplification and image-based acceleration techniques.

The strength of these methods is that they consider general occluders and handle *occluder fusion*, *i.e.* the occlusion by a combination of different objects.

The library Open GL Optimizer from Silicon Graphics proposes a form of screen space occlusion culling which seems similar to that described by Zhang *et al.* Some authors [BMT98] also propose a modification to the current graphics hardware to have access to z-test information for efficient occlusion culling.

4 Occluder shadow footprints

Many 3D scenes have in fact only two and a half dimensions. Such a scene is called a *terrain*, *i.e.*, a function $z = f(x,y)$. Wonka and Schmalstieg [WS99] exploit this characteristic to compute occlusions with respect to a point using a z-buffer with a top view of a scene.

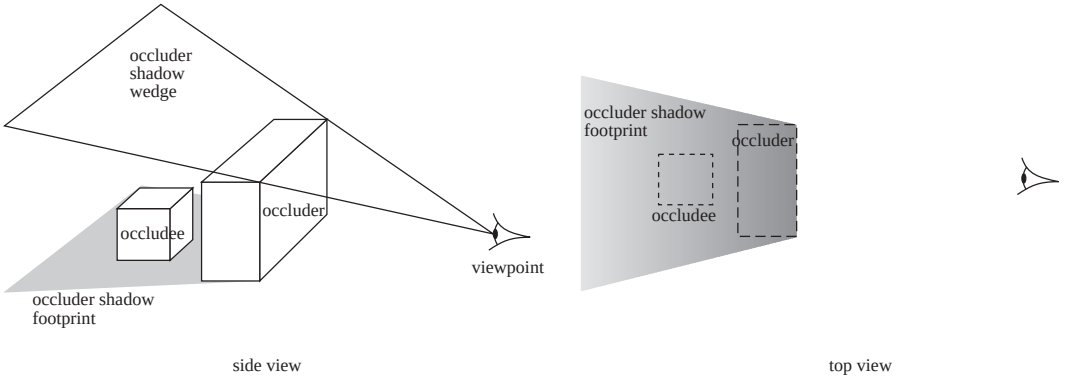


FIG. 11.5: Occluder shadow footprints. A projection from above is used to detect occlusion. Objects are hidden if they are below the occluder shadows. The footprints (with height) of the occluded regions are rasterized using a z-buffer. Depth is represented as grey levels. Note the gradient in the footprint due to the slope of the wedge.

Consider the situation depicted in Fig. 11.5 (side view). They call the part of the scene hidden by the occluder from the viewpoint the *occluder shadow* (as if the viewpoint were a light source). This occluder shadow is delimited by wedges. The projection of such a wedge on the floor is called the footprint, and an occludee is hidden by the occluder if it lies on the shadow footprint and if it is below the edge.

The z-buffer is used to scan-convert and store the height of the shadow footprints, using an orthographic top view (see Fig. 11.5). An object is hidden if its projection from above is on a shadow footprint and if it is *below* the shadow wedges *i.e*, if it is occluded by the footprints in the top view.

5 Epipolar rendering

Epipolar geometry has been developed in computer vision for stereo matching (see *e.g.* [Fau93]). Assume that the geometry of two cameras is known. Consider a point A in the first image (see Fig. 11.6). The possible point of the 3D scene must lie on the line L_A going through A and viewpoint 1. The projection of the corresponding point of the scene on the second image is constrained by the epipolar geometry : it must be on line L'_A which is the projection of L_A on image 2. The search for a correspondence can thus be restricted from a 2D search over the entire image to a 1D search on the *epipolar line*.

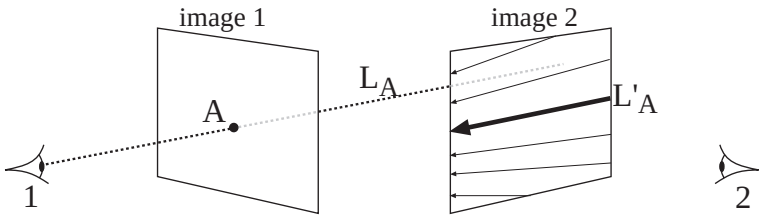


FIG. 11.6: Epipolar geometry. L_A is the set of all points of the scene possibly projecting on A. L'_A is the projection on image 2. For a warping from image 2 to image 1, points of image 2 have to be reprojected to image 1 in the order depicted by the arrows for correct occlusion.

Mc Millan and Bishop [MB95] have taken advantage of the epipolar geometry for view warping. Consider the warping from image 2 to image 1 (image 2 is the initial image, and we want to obtain image 1 by reprojecting the points of image 2). We want to decide which point(s) is reprojected on A. These are necessarily points on

the epipolar line L'_A. However, many points may project on A; only the closest has to be displayed. This can be achieved without actual depth comparison, by warping the points of the epipolar line L'_A in the order shown by the thick arrow, that is, from the farthest to the closest. If more than one point projects on A, the closest will overwrite the others. See also section 1.5 of chapter 13 for a line-space use of epipolar geometry.

6 Soft shadows using convolution

Soler and Sillion [SS98a, Sol98] have developed efficient soft shadow computations based on the use of convolutions. Some of the ideas are also present in a paper by Max [Max91]. A simplification could be to see their method as a “wise” blurring of shadow maps depending on the shape of the light source.

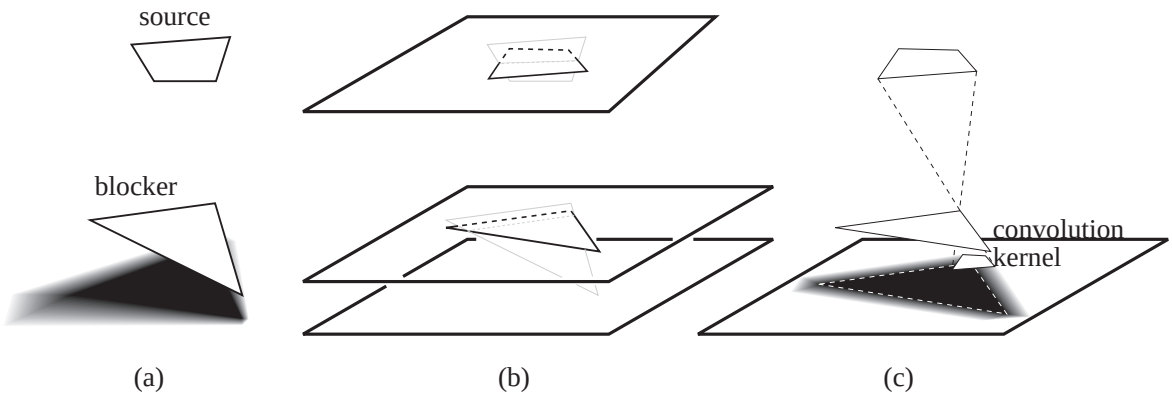


FIG. 11.7: Soft shadows computation using convolution. (a) Geometry of the scene. (b) Projection on a parallel approximate geometry. (c) The shadow is the convolution of the projection of the blockers with the inverse image of the source.

Consider an extended light source, a receiver and some blockers as shown in Fig. 11.7(a). This geometry is first projected onto three parallel planes (Fig. 11.7(b)). The shadow computation for this approximate geometry is equivalent to a convolution : the projection of the blocker(s) is convolved with the inverse projection of the light source (see Fig. 11.7(c)). The shadow map obtained is then projected onto the receiver (this is not necessary in our figures since the receiver is parallel to the approximate geometry).

In the general case, the shadows obtained are not exact : the relative sizes of umbra and penumbra are not correct. They are however not constant if the receiver is not parallel to the approximate geometry. The results are very convincing (see Fig. 11.8).

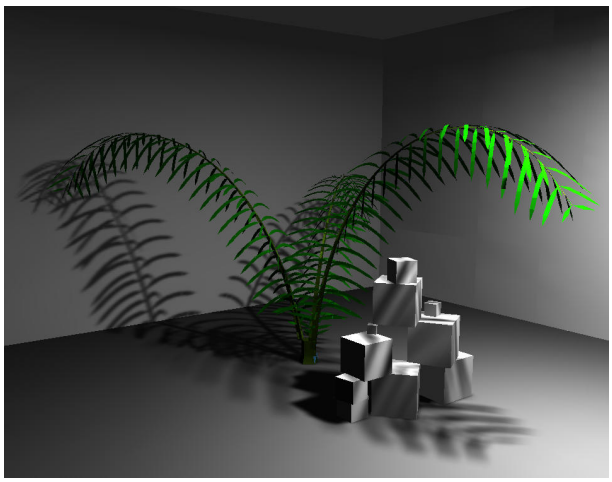


FIG. 11.8: Soft shadows computed using convolutions (image courtesy of Cyril Soler, iMAGIS-GRAVIR)

For higher quality, the blockers can be grouped according to their distance to the source. A convolution is performed for each group of blockers. The results then have to be combined ; Unfortunately the correlation between the occlusions of blockers belonging to different groups is lost (see also [Gra92] for a discussion of correlation problems for visibility and antialiasing).

This method has also been used in a global simulation system based on radiosity [SS98b].

7 Shadow coherence in image-space

Haines and Greenberg [HG86] propose a simple scheme to accelerate shadow computation in ray-tracing. Their *shadow cache* simply stores a pointer to the object which caused a shadow on the previous pixel. Because of coherence, it is very likely that this object will continue to cast a shadow on the following pixels.

Pearce and Jevans [PJ91] extend this idea to secondary shadow rays. Because of reflection and refraction, many shadow rays can be cast for each pixel. They thus store a tree of pointers to shadowing objects corresponding to the secondary ray-tree.

Worley [Wor97] pushes the idea a bit further for efficient soft shadow computation. He first computes simple hard shadows using one shadow-ray per pixel. He notes that pixels where shadow status changes are certainly in penumbra, and so are their neighbours. He thus "spreads" soft shadows, using more shadow rays for these pixels. The spreading operation stops when pixels in umbra or completely lit are encountered.

Hart *et al* [HDG99] perform a similar image-space floodfill to compute a blocker map : for each pixel, the objects casting shadows on the visible point are stored. They are determined using a low number of rays per pixel, but due to the image-space flood-fill the probability to miss blockers is very low. They then use an analytic clipping of the source by the blockers to compute the illumination of each pixel.

CHAPITRE 12

Viewpoint-Space

On ne voit bien qu'avec le cœur. L'essentiel est invisible pour les yeux.

Antoine de Saint-EXUPERY, *Le Petit Prince*

IEWPOINT-SPACE methods characterize viewpoints with respect to some visibility property. We first present the aspect graph which partitions viewpoint space according to the qualitative aspect of views. It is a fundamental visibility data-structure since it encodes all possible views of a scene. Section 2 presents some methods which are very similar to the aspect graph. Section 3 deals with the optimization of a viewpoint or set of viewpoints to satisfy some visibility criterion. Finally section 4 presents two methods which use visual events to determine the viewpoints at which visibility changes occur.

1 Aspect graph

As we have seen in section 2 of chapter 7 and Fig. 7.8 page 152, model-based object recognition requires a viewer-centered representation which encodes all the possible views of an object. This has led Koenderink and Van Doorn [Kv76, Kv79] to develop the *visual potential* of an object which is now more widely known as the *aspect graph* (other terminology are also used in the literature such as *view graph*, *characteristic views*, *principal views*, *viewing data*, *view classes* or *stable views*).

Aspect graph approaches consist in partitioning viewpoint space into cells where the view of an object are qualitatively invariant. The aspect graph is defined as follows :

– Each node represents a *general view* or *aspect* as seen form a connected cell of viewpoint space.

– Each arc represents a *visual event*, that is, a transition between two neighbouring general views.

The aspect graph is the dual graph of the partition of viewpoint space into cells of constant aspect. This partition is often named *viewing data* or *viewpoint space partition*. The terminology aspect graph and viewpoint space partition are often used interchangeably although they refer to dual concepts.

Even though all authors agree on the general definition, the actual meaning of *general view* and *visual event* varies. First approximate approaches have considered the set of visible features as defining a view. However for

exact approaches the *image structure graph* has rapidly imposed itself. It is the graph formed by the occluding contour or visible edges of the object. This graph may be labeled with the features of the object.

It is important to understand that the definition of the aspect graph is very general and that any definition of the viewing space and aspect can be exchanged. This makes the aspect graph concept a very versatile tool as we will see in section 2.

Aspect graphs have inspired a vast amount of work and it is beyond the scope of this survey to review all the literature in this field. We refer the reader to the survey by Eggert *et al.* [EBD92] or to the articles we cite and the references therein. Approaches have usually been classified according to the viewpoint space used (perspective or orthographic) and by the class of objects considered. We will follow the latter, reviewing the methods devoted to polyhedra before those related to smooth objects. But first of all, we survey the approximate method which use a discretization of viewpoint space.

1.1 Approximate aspect graph

Early aspect graph approaches have used a quasi uniform tessellation of the viewing sphere for orthographic projection. It can be obtained through the subdivision of an initial icosahedron as shown by Fig. 12.1. Sample views are computed from the vertices of this tessellation (the typical number of sample views is 2000). They are then compared, and similar views are merged. Very often, the definition of the aspect is the set of visible features (face, edge, vertex) and not their adjacencies as it is usually the case for exact aspect graphs This approach is very popular because of its simplicity and robustness, which explains that it has been followed by many researchers *e.g.* [Goa83, FD84, HK85]. We will see that most of the recognition systems using aspect graphs which have been implemented use approximate aspect graphs.

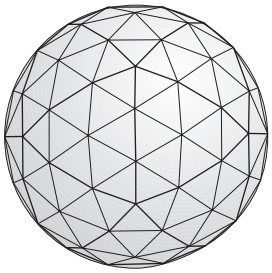


FIG. 12.1: Quasi uniform subdivision of the viewing sphere starting with an icosahedron.

We will see in section 3.2 that this quasi uniform sampling scheme has also been applied for viewpoint optimization problems.

A similar approach has been developed for perspective viewpoint space using voxels [WF90].

The drawback of approximate approaches is that the sampling density is hard to set, and approximate approach may miss some important views, which has led some researchers to develop exact methods.

1.2 Convex polyhedra

In the case of convex polyhedra, the only visual events are caused by viewpoints tangent to faces. See Fig. 12.2 where the viewpoint partition and aspect graph of a cube are represented. For orthographic projection, the directions of faces generate 8 regions on the viewing sphere, while for perspective viewpoint space, the 6 faces of the cube induce 26 regions.

The computation of the visual events only is not sufficient. Their *arrangement* must be computed, that is, the decomposition of viewpoint space into cells, which implies the computation of the intersections between the events to obtain the segments of events which form the boundaries of the cells. Recall that the arrangement of n lines (or well-behaved curves) in 2D has $O(n^2)$ cells. In 3D the arrangement of n planes has complexity $O(n^3)$ in size [dBvKOS97, O'R94, Ede87, BY98].

The first algorithms to build the aspect graph of 3D objects have dealt with convex polyhedra under orthographic [PD86] and perspective [SB90, Wat88] projection.

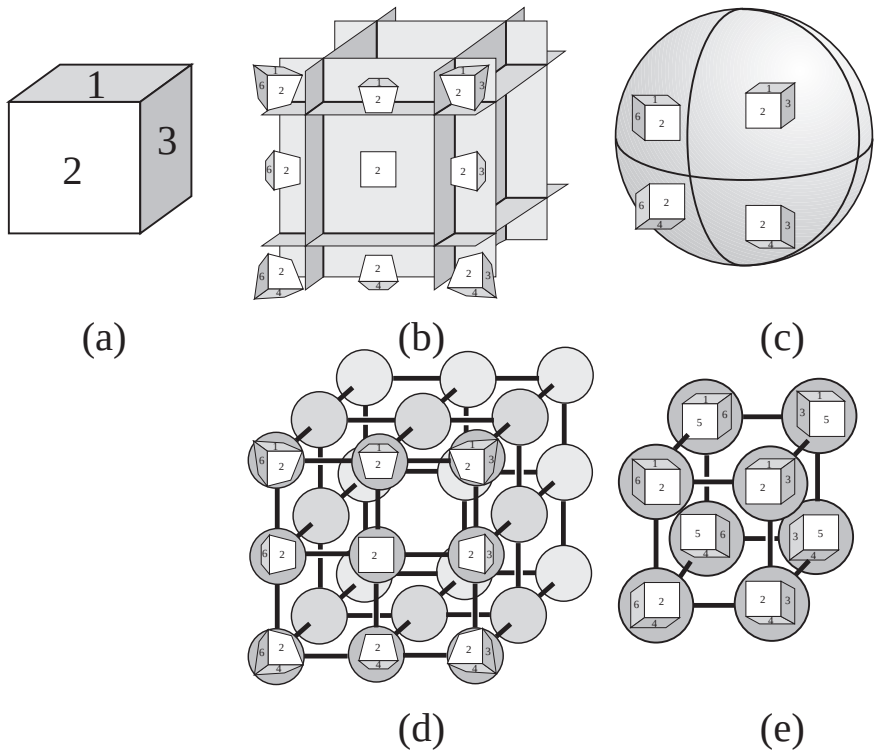


FIG. 12.2: Aspect graph of a convex cube. (a) Initial cube with numbered faces. (b) and (c) Partition of the viewpoint space for perspective and orthographic projection with some representative aspects. (d) and (e) Corresponding aspect graphs. Some aspects are present in perspective projection but not in orthographic projection, for example when only two faces are visible. Note also that the cells of the perspective viewpoint space partition have infinite extent.

1.3 General polyhedra

General polyhedra are more involved because they generate edge-vertex and triple-edge events that we have presented in chapter 8. Since the number of triple-edge events can be as high as $O(n^3)$, the size of the aspect graph of a general polygon is $O(n^6)$ for orthographic projection (since the viewing sphere is two dimensional), and $O(n^9)$ for perspective projection for which viewpoint space is three-dimensional. However these bounds may be very pessimistic. Unfortunately the lack of available data impede a realistic analysis of the actual complexity. Note also that we do not count here the size of the representative views of aspects, which can be $O(n^2)$ each, inducing a size $O(n^8)$ for the orthographic case and $O(n^{11})$ for the perspective case.

The cells of the aspect graph of a general polyhedron are not necessary convex. Partly because of the EEE events, but also because of the EV events. This is different from the 2D case where all cells are convex because in 2D visual events are line segments.

We detail here the algorithms proposed by Gigus and his co-authors [GM90, GCS91] to build the aspect graph of general polyhedra under orthographic projection.

In the first method [GM90], potential visual events are considered for each face, edge-vertex pair and triple of edges. At this step, occlusion is not taken into account : objects lying between the generators of the events are considered transparent. These potential events are projected on the viewing sphere, and the arrangement is built using a plane sweep.

However, some boundaries of the resulting partition may correspond to false visual event because of occlusion. For example, an object may lie between the edge and vertex of an EV event as shown in Fig. 12.3. Each segment of cell boundary (that is, each portion of visual event) has to be tested for occlusion. False segment are discarded, and the cells are merged.

Gigus Canny and Seidel [GCS91] propose to cope with the problem of false events before the arrangement is constructed. They compute the intersection of all the event with the object in object space as shown in Fig.

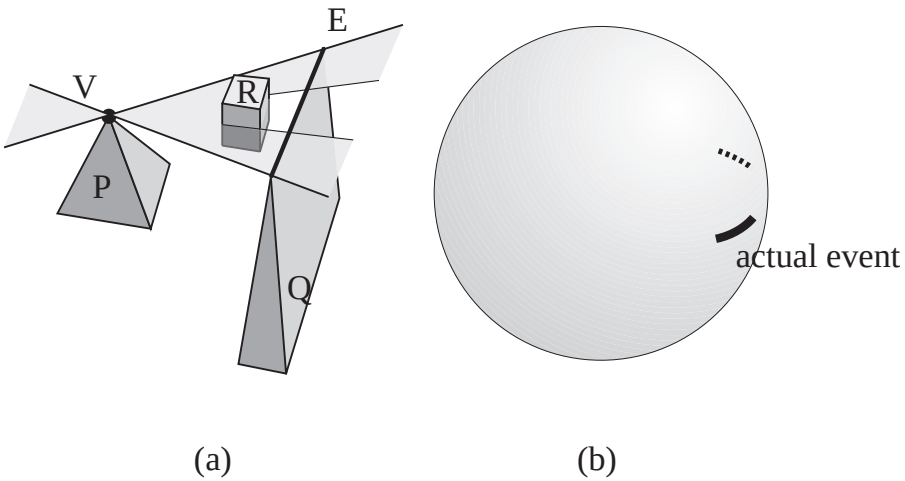


FIG. 12.3: False event ("transparent" event). Object *R* occludes vertex *V* from edge *E*, thus only a portion of the potential visual event corresponds to an actual visual event. (a) In object space. (b) In orthographic viewpoint space.

12.3(a), and only the unoccluded portion is used for the construction of the arrangement.

They also propose to store and compute the representative view efficiently. They store only one aspect for an arbitrary seed cell. Then all other views can be retrieved by walking along the aspect graph and updating this initial view at each visual event.

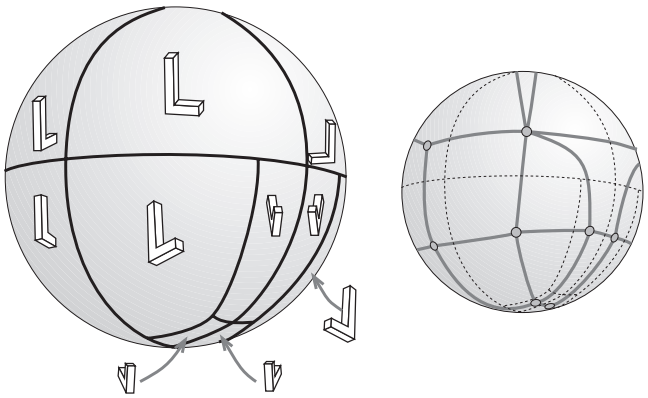


FIG. 12.4: Aspect graph of a L-shaped polyhedron under orthographic projection (adapted from [GM90]). (a) Partition of the viewing sphere and representative views. (b) Aspect graph.

These algorithms have however not been implemented to our knowledge. Fig. 12.4 shows the partition of the viewing sphere and the aspect graph of a L-shaped polyhedron under orthographic transform.

Similar construction algorithms have been proposed by Stewman and Bowyer [SB88] and Stewman [Ste91] who also deals with perspective projection.

We will see in section 1.1 of chapter 13 that Plantinga and Dyer [PD90] have proposed a method to build the aspect graph of general polyhedra which uses an intermediate line space data-structure to compute the visual events.

1.4 Curved objects

Methods to deal with curved objects were not developed till later. Seales and Dyer [SD92] have proposed the use of a polygonal approximation of curved objects with polyhedra, and have restricted the visual events

to those involving the silhouette edges. For example, an edge-vertex event EV will be considered only if E is a silhouette edge from V (as this is the case in Fig. 8.3 page 164). This is one example of the versatility of the aspect graph definition : here the definition of the aspect depends only on the silhouette.

Kriegman and Ponce [KP90] and Eggert and Bowyer [EB90] have developed methods to compute aspect graphs of solids of revolution under orthographic projection, while Eggert [Egg91] also deals with perspective viewpoint space. Objects of revolution are easier to handle because of their rotational symmetry. The problem reduces to a great circle on the viewing sphere or to one plane going through the axis of rotation in perspective viewpoint space. The rest of the viewing data can then be deduced by rotational symmetry. Eggert *et al.* [EB90, Egg91] report an implementation of their method.

The case of general curved object requires the use of the catalogue of singularities as proposed by Callahan and Weiss [CW85] ; they however developed no algorithm.

Petitjean and his co-authors [PPK92, Pet92] have presented an algorithm to compute the aspect graph of smooth objects bounded by arbitrary smooth algebraic surface under orthographic projection. They use the catalogue of singularities of Kergosien [Ker81]. There approach is similar to that of Gigus and Malik [GM90]. They first trace the visual events of the "transparent" object (occlusion is not taken into account) to build a partition of the viewing sphere. They then have to discard the false (also called occluded) events and merge the corresponding cells. Occlusion is tested using ray-casting at the center of the boundary. To trace the visual event, they derive their equation using a computer algebra system and powerful numerical techniques. The degree of the involved algebraic systems is very large, reaching millions for an object described by an equation of degree 10. This algorithm has nevertheless been implemented and an example of result is shown in Fig. 12.5.

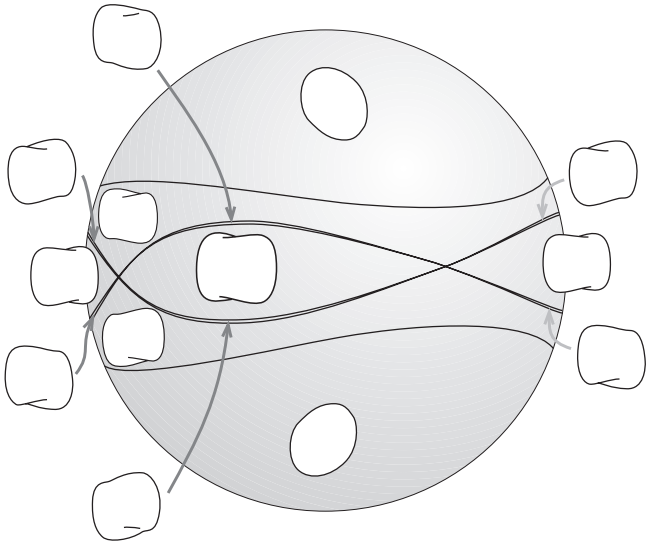


FIG. 12.5: Partition of orthographic viewpoint space for a dimple object with representative aspects. (adapted from [PPK92]).

Similar methods have been developed by Sripradisvarakul and Jain [SJ89], Ponce and Kriegman [PK90] while Rieger [Rie92, Rie93] proposes the use of symbolic computation and cylindrical algebraic decomposition [Col75] (for a good introduction to algebraic decomposition see the book by Latombe [Lat91] p. 226).

Chen and Freeman [CF91b] have proposed a method to handle quadric surfaces under perspective projection. They use a sequence of growing concentric spheres centered on the object. They trace the visual events on each sphere and compute for which radius the aspects change.

Finally PetitJean has studied the enumerative properties of aspect graphs of smooth and piecewise smooth objects [Pet95, Pet96]. In particular, he gives bounds on the number of topologically distinct views of an object using involved mathematical tools.

1.5 Use of the aspect graph

The motivation of aspect graph research was model-based object recognition. The aspect graph provides informations on all the possible views of an object. The use of this information to recognise an object and its pose are however far from straightforward, one reason being the huge number of views. Once the view of an object has been acquired from a camera and its features extracted, those features can not be compared to all possible views of all objects in a database : indexing schemes are required. A popular criterion is the number of visible features (face, edge, vertex) [ESB95].

The aspect graph is then often used to build offline a *strategy tree* [HH89]or an *interpretation tree* [Mun95]. At each node of an interpretation tree corresponds a choice of correspondence, which then recursively leads to a restricted set of possible interpretation. For example if at a node of the tree we suppose that a feature of the image corresponds to a given feature A of a model, this may exclude the possibility of another feature B to be present because feature A and B are never visible together.

The information of the viewing space partition can then be used during pose estimation to restrict the possible set of viewpoint [Ike87, ESB95]. If the observation is ambiguous, Hutchinson and Kak [HK89] and Gremban and Ikeuchi [GI87] also use the information encoded in the aspect graph to derive a new relevant viewpoint from which the object and pose can be discriminated.

Dickinson *et al.* [DPR92] have used the aspect for object composed of elementary objects which they call *geons*. They use an aspect graph for each geon and then use structural information on the assembly of geons to recognise the object.

However the aspect graph has not yet really imposed itself for object recognition. The reasons seem to be the difficulty of robust implementation of exact methods, huge size of the data-structure and the lack of obvious and efficient indexing scheme. One major drawback of the exact aspect graphs is that they capture all the possible views, whatever their likelihood or significance. The need of a notion “importance” or *scale* of the features is critical, which we will discuss in section 1 of chapter 14.

For a good discussion of the pros and cons of the aspect graph, see the report by Faugeras *et al.* [FMA+92].

Applications of the aspect graph for rapid view computation have also been proposed since all possible views have been precomputed [PDS90, Pla93]. However, the only implementation reported restricted the viewpoint movement to a rotation around one axis.

More recently Gu and his coauthors [GGH+99] have developed a data-structure which they call a *silhouette tree* which is in fact an aspect graph for which the aspect is defined only by the exterior silhouette. It is built using a sampling and merging approach on the viewing sphere. It is used to obtain images with a very fine silhouette even if a very simplified version of the object is rendered.

Pellegrini [Pel99] has also used a decomposition of the space of direction similar to the aspect graph to compute the form factor between two unoccluded triangles. The sphere S^2 is decomposed into regions where the projection of the two triangles has the same topology. This allows an efficient integration because no discontinuity of the integration kernel occur in these regions.

A somehow related issue is the choice of a good viewpoint for the view of a 3D graph. Visual intersections should be avoided. These in fact correspond to EV or EEE events. Some authors [BGRT95, HW98, EHW97] thus propose some methods which avoid points of the viewing sphere where such events project.

2 Other viewpoint-space partitioning methods

The following methods exhibit a typical aspect graph philosophy even though they use a different terminology. They subdivide the space of viewpoints into cells where a view is qualitatively invariant.

2.1 Robot Localisation

Deducing the position of a mobile robot from a view is exactly the same problem as determining the pose of an object. The differences being that a range sensor is usually used and that the problem is mostly two dimensional since mobile robots are usually naturally constrained on a plane.

Methods have thus been proposed which subdivide the plane into cells where the set of visible walls is constant [GMR95, SON96, TA96]. See Fig. 12.6. Visual events occur when the viewpoint is aligned with a

wall segments or along a line going through two vertices. Indexing is usually done using the number of visible walls.

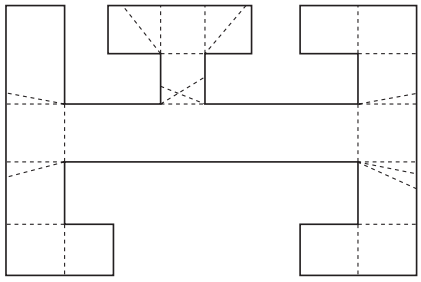


FIG. 12.6: Robot self-localization. Partition of a scene into cells of structurally invariant views by visual events (dashed).

Guibas and his co-authors [GMR95] also propose to index the aspects in a multidimensional space. To summarize, they associate to a view with m visible vertices a vector of $2m$ dimensions depending on the coordinates of the vertices. They then use standard multidimensional search methods [dBvKOS97].

2.2 Visibility based pursuit-evasion

The problem of pursuit-evasion presented in section 3 and Fig. 7.14 page 156 can also be solved using an aspect-graph-like structure. Remember that the robot has to "clean" a scene by checking if an intruder is present. "Contaminated" regions are those where the intruder can hide. We present here the solution developed by LaValle *et al.* [LLG+97, GLL+97, GLLL98].

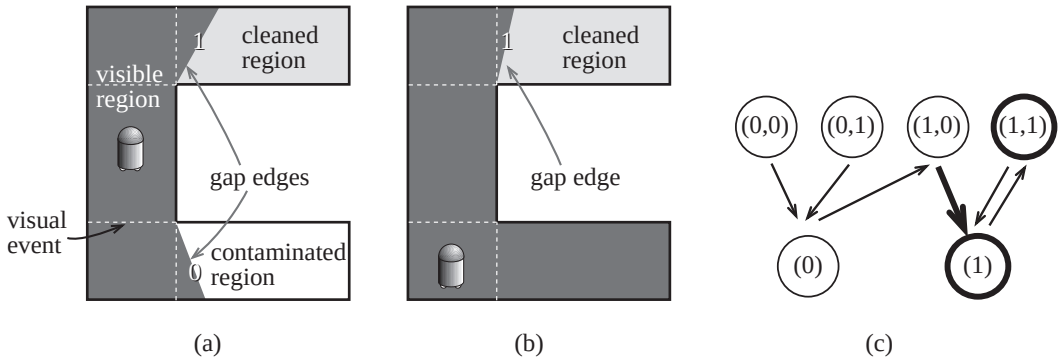


FIG. 12.7: Pursuit-Evasion strategy. (a) The contaminated region can be cleaned only if the visual event is crossed. The status of the neighbouring regions is coded on the gap edges. (b) The robot has moved to a second cell, cleaning a region. (c) Part of the graph of possible states (upper node correspond to cell in (a) while lower nodes correspond to the cell in (b)). In thick we represent the goal states and the move from (a) to (b).

Consider the situation in Fig. 12.7(a). The view from the robot is in dark grey. The contaminated region can be cleaned only when the indicated visual event is crossed as in Fig. 12.7(b).

The scene is partitioned by the visibility event with the same partition as for robot localization (see Fig. 12.6). For each cell of the partition, the structure of the view polygon is invariant, and in particular the *gap edges* (edges of the view which are not on the boundary of the scene). The status of the neighbouring regions is coded on these gap edges : 0 indicates a contaminated region while 1 indicates a cleaned one.

The state of the robot is thus coded by its current cell and the status of the corresponding gap edges. In Fig 12.7(a) the robot status is $(1,0)$, while in (b) it is (1). Solving the pursuit problem consists in finding the succession of states of the robot which end at a state where all gap edges are at 1. A graph is created with one node for each state (that means 2^m states for a cell with m edges). Edges of the graph correspond to possible transition. A transition is possible only to neighbouring cells, but not to all corresponding states. Fig. 12.7 represents a portion of this graph.

The solution is then computed using a standard Dijkstra search. See Fig. 7.14 page 156 for an example. Similar methods have also been proposed for curved environments [LH99].

2.3 Discontinuity meshing with backprojections

We now turn to the problem of soft shadow computation in polygonal environments. Recall that the penumbra region corresponds to zones where only a part of an extended light source is visible. Complete discontinuity meshing subdivides the scene polygons into regions where the topology of the visible part of the source is constant. In this regions the illumination varies smoothly, and at the region boundary there is a C^2 discontinuity.

Moreover a data-structure called *backprojection* encodes the topology of the visible part of the source as represented in Fig. 12.8(b) and 12.9(b). Discontinuity meshing is an aspect graph method where the aspect is defined by the visible part of the source, and where viewpoint space is the polygons of the scene.

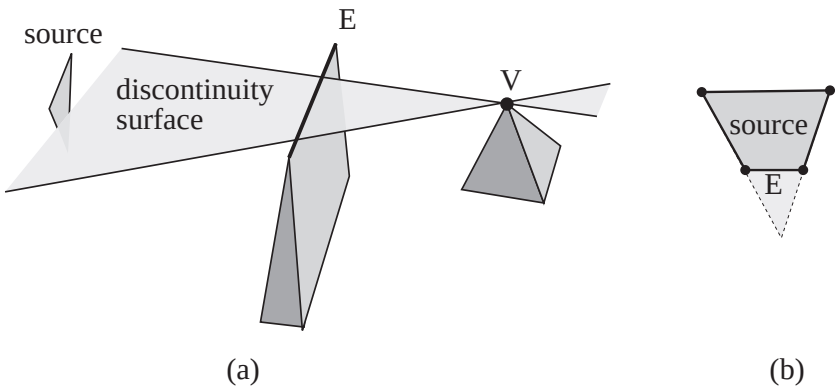


FIG. 12.8: Complete discontinuity meshing with backprojections. (a) Example of an EV event intersecting the source. (b) In thick backprojection from V (structure of the visible part of the source)

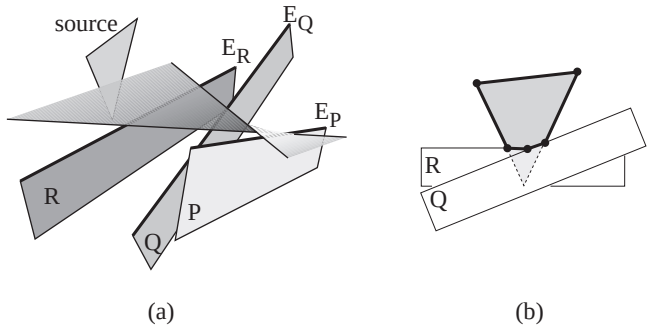


FIG. 12.9: Discontinuity meshing. (a) Example of an EEE event intersecting the source. (b) In thick backprojection from a point on E_P (structure of the visible part of the source)

Indeed the method developed and implemented by Drettakis and Fiume [DF94] is the equivalent of Gigus Canny and Seidel's algorithm [GCS91] presented in the previous section. Visual events are the EV and EEE event with one generator on the source or which intersect the source (Fig. 12.8(a) and 12.9(a)). An efficient space subdivision acceleration is used to speed up the enumeration of potential visual events. For each vertex generator V an extended pyramid is build with the light source, and only the generators lying inside this volume are considered. Space subdivision is used to accelerate this test. A similar scheme is used for edges. Space subdivision is also used to speed-up the discontinuity surface-object intersections. See Fig. 12.10 for an example of shadows and discontinuity mesh.

This method has been used for global illumination simulation using radiosity [DS96]. Both the mesh and form-factor problem are alleviated by this approach, since the backprojection allows for efficient point-to-area form factor computation (portion of the light leaving the light source arriving at a point). The experiments

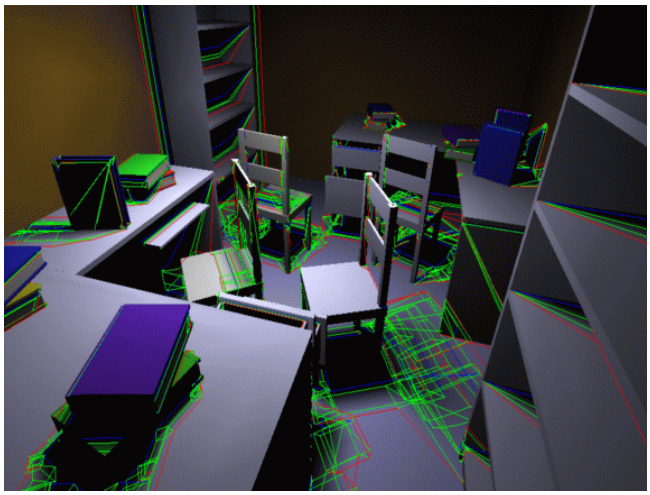


FIG. 12.10: Complete discontinuity mesh of a 1000 polygons scene computed with Drettakis and Fiume's algorithm [DF94].

exhibited show that both the quality of the induced mesh and the precision of the form-factor computation are crucial for high quality shadow rendering.

2.4 Output-sensitive discontinuity meshing

Stewart and Ghali [SG94] have proposed an output-sensitive method to build a complete discontinuity mesh. They use a similar discontinuity surface-object intersection, but their enumeration of the discontinuity surfaces is different.

It is based on the fact that a vertex V can generate a visual event with an edge E only if E lies on the boundary of the visible part of the source as seen from V (see Fig. 12.8). A similar condition arises for EEE events : the two edges closest to the source must belong to the backprojection of some part of the third edge, and must be adjacent in this backprojection as shown in Fig. 12.9.

They use an update of the backprojections at visual events. They note that a visual event has effect only on the parts of scene which are farther from the source than its generators. They thus use a sweep with planes parallel to the source. Backprojections are propagated along the edges and vertices of the scene, with an update at each edge-visual event intersection.

Backprojection have however to be computed for scratch at each *peak vertex*, that is, for each polyhedron, the vertex which is closest to the source. Standard hidden surface removal is used.

The algorithm can be summarized as follows :

- Sort the vertices of the scene according to the distance to the source.
- At peak vertices compute a backprojection and propagate it to the beginning of the edges below.
- At each edge-visual event intersection update the backprojection.
- For each new backprojection cast (intersect) the generated visual event through the scene.

This algorithm has been implemented [SG94] and extended to handle degenerate configuration [GS96] which cause some C^1 discontinuities in the illumination function.

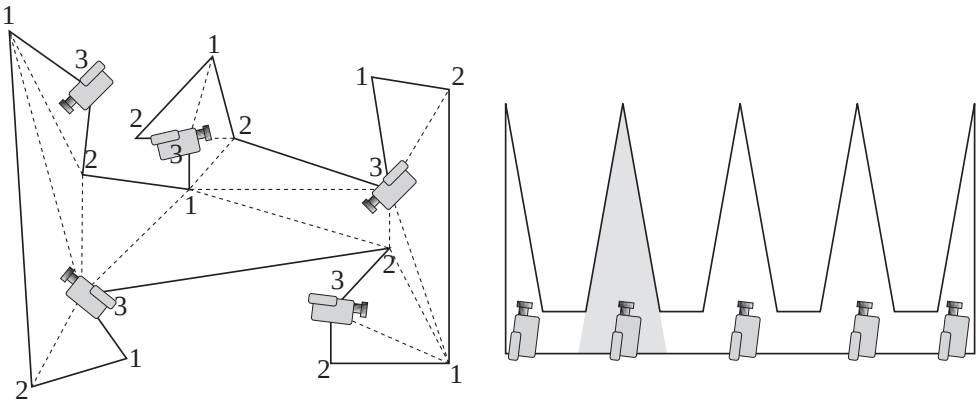
3 Viewpoint optimization

In this section we present methods which attempt to chose a viewpoint or a set of viewpoints to optimize the visibility of all or some of the features of a scene. The search is here exhaustive, all viewpoints (or a sampling) are tested. The following section will present some methods which alleviate the need to search the whole space of viewpoints. Some related results have already been presented in section 4.5 and 5.5 of chapter 10.

3.1 Art galleries

We present the most classical results on art gallery problems. The classic art gallery theorem is due to Chvátal [Chv75] but he exhibited a complex proof. We here present the proof by Fisk [Fis78] which is much simpler. We are given an art-gallery modeled by a simple (with no holes) 2D polygons.

Theorem : $\lfloor \frac{n}{3} \rfloor$ *stationary guards are always sufficient and occasionally necessary to guard a polygonal art gallery with n vertices.*



(a) (b)

FIG. 12.11: Art gallery. (a) The triangulation of a simple polygon is 3-colored with colors 1, 2 and 3. Color 3 is the less frequent color. Placing a guard at each vertex with color 3 permits to guard the polygon with less than $\lfloor \frac{n}{3} \rfloor$ guards. (b) Worst-case scene. To guard the second spike, a camera is needed in the grey region. Similar constraints for all the spikes thus impose the need of at least $\lfloor \frac{n}{3} \rfloor$ guards

The proof relies on the triangulation of the polygon with diagonals (see Fig. 12.11(a)). The vertices of such a triangulation can always be colored with 3 colors such that no two adjacent vertices share the same color (Fig. 12.11(a)). This implies that any triangle has one vertex of each color. Moreover, each vertex can guard its adjacent triangles.

Consider the color which colors the minimum number of vertices. The number of corresponding vertices is lower than $\lfloor \frac{n}{3} \rfloor$, and each triangle has such a vertex. Thus all triangles are guarded by this set of vertices. The lower bound can be shown with a scene like the one presented in Fig. 12.11(b).

Such a set of guards can be found in $O(n)$ time using a linear time triangulation algorithm by Chazelle [dBvKOS97]. The problem of finding the minimum number of guards has however been shown NP-hard by Aggarwal [Aga84] and Lee and Lin [LL86].

For other results see the surveys on the domain [O'R87, She92, Urr98].

3.2 Viewpoint optimization

The methods which have been developed to optimize the placement of sensors or lights are all based on a sampling approach similar to the approximate aspect graph.

We present here the methods developed by Tarbox and Gottschlich [TG95]. Their aim is to optimize the placement of a laser and a camera (as presented in Fig. 7.12 page 154) to be able to inspect an object whose pose and geometry are known. The distance of the camera and laser to the object is fixed, viewpoint space is thus a viewing sphere even if perspective projection is used. The viewing sphere is tessellated starting with an icosahedron (Fig. 12.1 page 204). Sample points are distributed over the object. For each viewpoint, the visibility of each sample point is tested using ray-casting. It is recorded in a two dimensional array called the *viewability matrix* indexed by the viewpoint and sample point. (In fact two matrices are used since the visibility constraints are not the same for the camera and for the laser.)

The viewability matrix can be seen as a structure in segment space : each entry encodes if the segment joining a given viewpoint and a given sample point intersects the object.

The set of viewpoints which can see a given feature is called the *viewpoint set*. For more robustness, especially in case of uncertainties in the pose of the object, the viewpoints of the boundary of a viewpoint set are discarded, that is, the corresponding entry in the viewability matrix is set to 0. For each sample point, a difficulty-to-view is computed which depends on the number of viewpoints from which it is visible.

A set of pairs of positions for the laser and the camera are then searched which resumes to a set-cover problem. The first strategy they propose is greedy. The objective to maximize is the number of visible sample points weighted by their difficulty-to-view. Then each new viewpoint tries to optimize the same function without considering the already seen points until all points are visible from at least one viewpoint.

The second method uses simulated annealing (which is similar to a gradient descend which can "jump" over local minima). An arbitrary number of viewpoints are randomly placed on the viewing sphere, and their positions are then perturbated to maximize the number of visible sample points. If no solution is found for n, a new viewpoint is added and the optimization proceeds. This method provides results with fewer viewpoints.

Similar methods have been proposed for sensor placement [MG95, TUWR97], data acquisition for mobile robot on a 2D floorplan [GL99] and image-based representation [HLW96]. See Fig. 12.12 for an example of sensor planning.

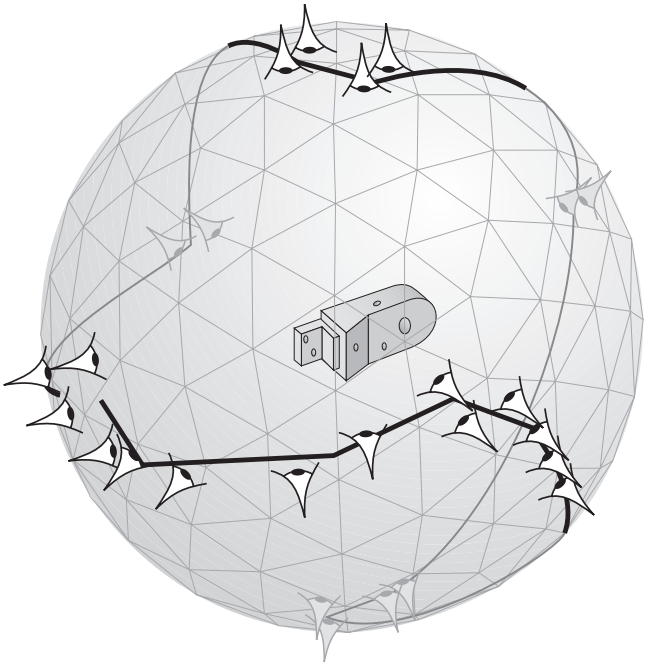


FIG. 12.12: Planning of a stereo-sensor to inspect an object (adapted from [TUWR97])

Stuerzlinger [Stu99] also proposes a similar method for the image-based representation of scenes. His viewpoint space is a horizontal plane at human height. Both objects and viewpoint space are adaptively subdivided for more efficient results. He then uses simulated annealing to optimize the set of viewpoints.

3.3 Local optimization and target tracking

Yi, Haralick and Shapiro [YHS95] optimize the position of both a camera and a light source. The position of the light should be such that features have maximal contrast in the image observed by the camera. Occlusion is not really handled in their approach since they performed their experiments only on a convex box. However their problem is in spirit very similar to that of viewpoint optimization for visibility constraints, so we include it in this survey because occlusion could be very easily included in their optimization metric.

They use no initial global computation such as the viewability matrix studied in the previous paragraph, but instead perform a local search. They perform a gradient descent successively on the light and camera positions. This method does not necessarily converge to a global maximum for both positions, but they claim that in their experiments the function to optimize is well behaved and convex and that satisfactory results are obtained.

Local optimization has also been proposed [LGBL97, FL98] for the computation of the motion of a mobile robot which has to keep a moving target in view. Assume the motion of the target is only partially predictable (by bound on the velocity for example). A local optimization is performed in the neighbourhood of the pursuer position in a game theoretic fashion : the pursuer has to take into account all the possible movements of the target to decide its position at the next timestep. For a possible pursuer position in free space, all the possible movements of the target are enumerated and the probability of its being visible is computed. The pursuer position with the maximum probability of future visibility is chosen. See Fig. 12.13 for an example of pursuit. The range of the sensor is taken into account.

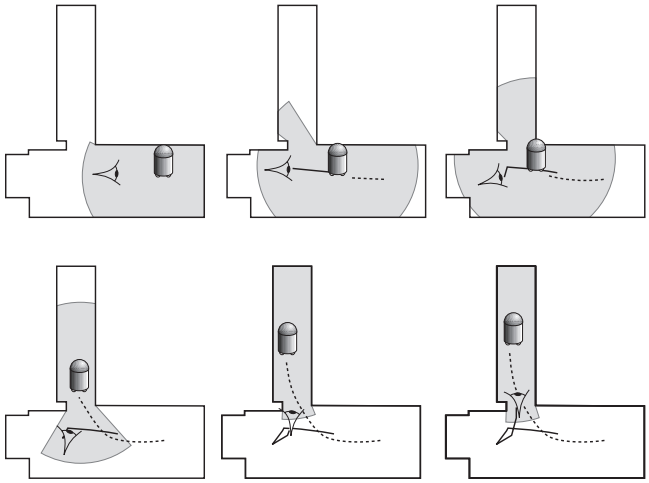


FIG. 12.13: Tracking of a mobile target by an observer. The region in which the target is visible is in light grey (adapted from [LGBL97]).

They also propose another strategy for a better prediction [LGBL97]. The aim is here to maximize the escape time of the target. For each possible position of the pursuer, its visibility region is computed (the inverse of a shadow volume). The distance of the target to the boundary of this visibility region defines the minimum distance it has to cover to escape the pursuer (see Fig. 12.14).

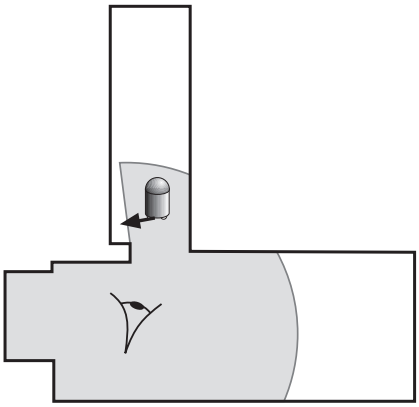


FIG. 12.14: Tracking of a mobile target by an observer. The region in light grey is the region in which the target is visible from the observer. The thick arrow is the shortest path for the target to escape.

The extension of these methods to the prediction of many timesteps is unfortunately exponential.

4 Frame-to-frame coherence

In section 1.5 we have presented applications of the aspect graph to updating a view as the observer continuously moves. The cost induced by the aspect graph has prevented the use of these methods. We now present methods which use the information encoded by visual events to update views, but which consider only a subset of them.

4.1 Coherence constraints

Hubschman and Zucker [HZ81, HZ82] have studied the so-called *frame-to-frame coherence* for static scenes. This approach is based on the fact that if the viewpoint moves continuously, two successive images are usually very similar. They study the occlusions between pairs of convex polyhedra.

They note that a polyhedron will start (or stop) occluding another one only if the viewpoint crosses one of their separating planes. This corresponds to EV visual events. Moreover this can happen only for silhouette edges.

Each edge stores all the separating planes with all other polyhedra. These planes become active only when the edge is on the silhouette in the current view. As the viewpoint crosses one of the active planes, the occlusion between the two corresponding polyhedra is updated.

This approach however fails to detect occlusions caused by multiple polyhedra (EEE events are not considered). Furthermore, a plane is active even if both polyhedra are hidden by a closer one, in which case the new occlusion has no actual effect on the visibility of the scene ; Transparent as well as opaque events are considered. These limitations however simplify the approach and make it tractable. Unfortunately, no implementation is reported.

4.2 Occlusion culling with visual events

Coorg and Teller [CT96] have extended their shadow-volume based occlusion culling presented in section 4.4 of chapter 10 to take advantage of frame-to-frame coherence.

The visibility of a cell of the scene subdivision can change only when a visual event is crossed. For each large occluder visibility changes can occur only for the neighbourhood of partially visible parts of the scene (see Fig. 12.15). They thus dynamically maintain the visual events of each occluders and test the viewpoint against them.

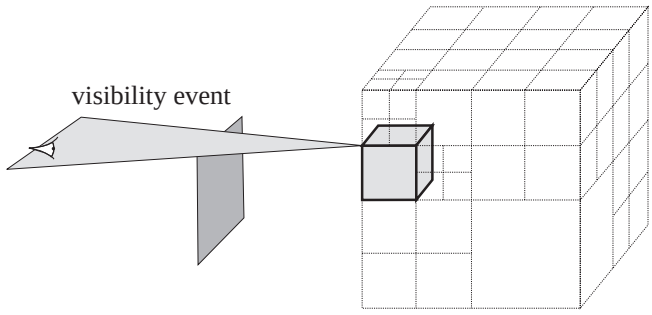


FIG. 12.15: Occlusion culling and visual events

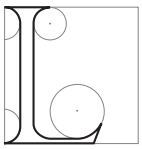
They explain that this can be seen as a local linearized version of the aspect graph. Indeed they maintain a superset of the EV boundaries of the current cell of the perspective aspect graph of the scene.

CHAPITRE 13

Line-Space

Car il ne sera fait que de pure lumière
Puisée au foyer saint des rayons primitifs

Charles BAUDELAIRE, *Les Fleurs du Mal*



INE-SPACE methods characterize visibility with respect to line-object intersections. The methods we present in section 1 partition lines according to the objects they intersect. Section 2 introduces graphs in line-space, while section 3 discusses Plücker coordinates, a powerful parameterization which allows the characterization of visibility using hyperplanes in 5D. Finally section 4 presents stochastic and probabilistic approaches in line-space.

1 Line-space partition

1.1 The Asp

Plantinga and Dyer [PD87, PD90, Pla88] devised the *asp* as an auxiliary data-structure to compute the aspect graph of polygonal objects. The definition of the *asp* depends on the viewing space considered. We present the *asp* for orthographic projection.

A duality is used which maps oriented lines into a 4 dimensional space. Lines are parameterized as presented in section 1.4 of chapter 8 and Fig. 8.2(a) (page 163) by their direction, denoted by two angles (θ, φ) and the coordinates (u, v) on an orthogonal plane. The *asp* for θ and φ constant is thus an orthographic view of the scene from direction (θ, φ). The *asp* of an object corresponds to the set of lines intersecting this object. See Fig. 13.1(a) and (b).

Occlusion in a view corresponds to subtraction in the *asp* : if object *A* is occluded by object *B*, then the *asp* of *B* has to be subtracted from the *asp* of *A* as shown in Fig. 13.1(c). In fact the intersection of the *asp* of two objects is the set of lines going through them. Thus if object *B* is in front of object *A*, and these lines no longer "see" *A*, they have to be removed from the *asp* of *A*.

The 1 dimensional boundaries of the *asp* correspond to the visual events necessary to build the aspect graph. See Fig. 13.1(c) where an *EV* event is represented. Since it is only a slice of the *asp*, only one line of the event is present under the form of a point. Since occlusion has been taken into account with subtraction, the *asp*

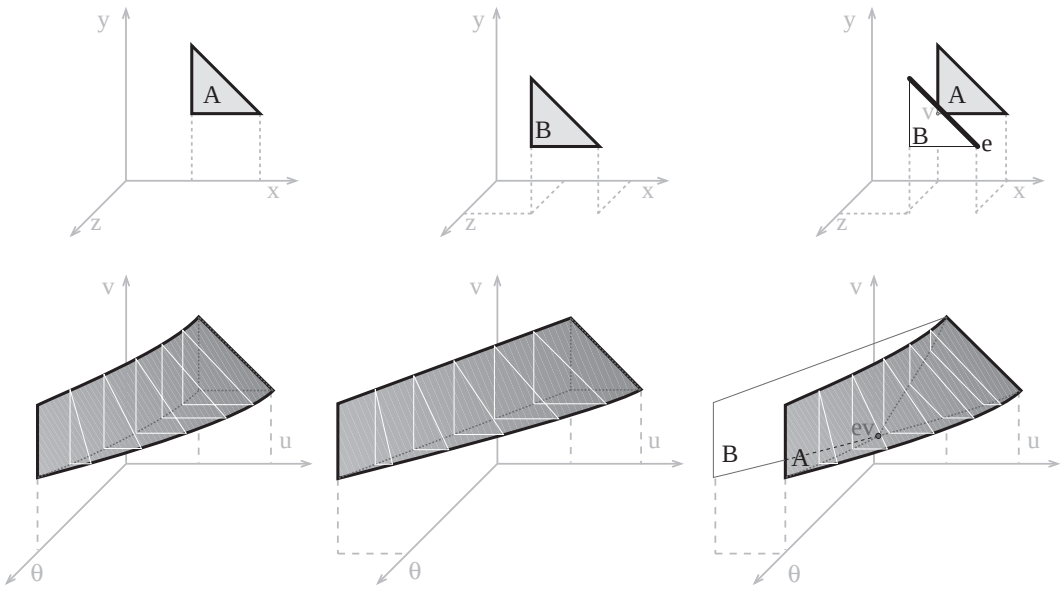


FIG. 13.1: Slice of the *asp* for $\varphi = 0$ (adapted from [PD90]). (a) and (b) *Asp* for one triangle. The θ slices in white correspond to orthographic views of the triangle. When $\theta = 90°$ the view of the triangle is a segment. (c) Occlusion corresponds to subtraction in *asp* space. We show the *asp* of triangle *A* which is occluded by *B*. Note the occlusion in the θ slices in white. We also show the outline of the *asp* of *B*. The visual event *EV* is a point in *asp* asp space.

contains only the opaque events, transparent events do not have to be detected and discarded as in Gigus and Malik's method [GM90] presented in section 1.3. Unfortunately no full implementation is reported. The size of the *asp* can be as high as $O(n^4)$, but as already noted, this does not give useful information about its practical behaviour with standard scenes.

In the case of perspective projection, the *asp* is defined in the 5 dimensional space of rays. Occlusion is also handled with subtractions. Visual events are thus the 2 dimensional boundaries of the *asp*.

1.2 The 2D Visibility Complex

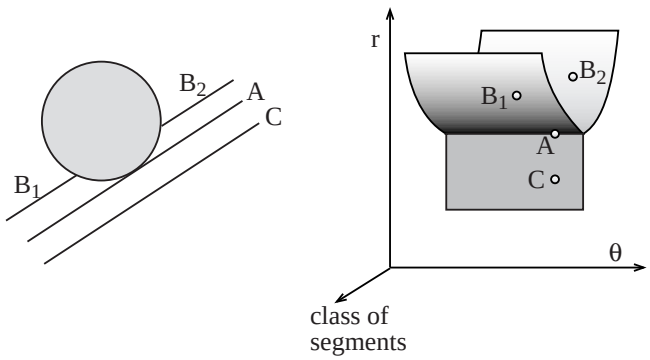
Pocchiola and Vegter [PV96b, PV96a] have developed the 2D *visibility complex* which is a topological structure encoding the visibility of a 2D scene. The idea is in a way similar to the *asp* to group rays which "see" the same objects. See [DP95b] for a simple video presentation.

The central concept is that of *maximal free segments*. These are segments of maximal length that do not intersect the interior of the objects of the scene. More intuitively, a maximal free segment has its extremities on the boundary of objects, it may be tangent to objects but does not cross them. A line is divided in many maximal free segment by the objects it intersects. A maximal free segment represents a group of colinear rays which see the same objects. The manifold of 2D maximal free segments is two-dimensional nearly everywhere, except at certain branchings corresponding to tangents of the scene. A tangent segment has neighbours on both sides of the object and below the object (see Fig. 13.2).

The visibility complex is the partition of maximal free segments according to the objects at their extremities. A face of the visibility complex is bounded by chains of segments tangent to one object (see Fig. 13.3).

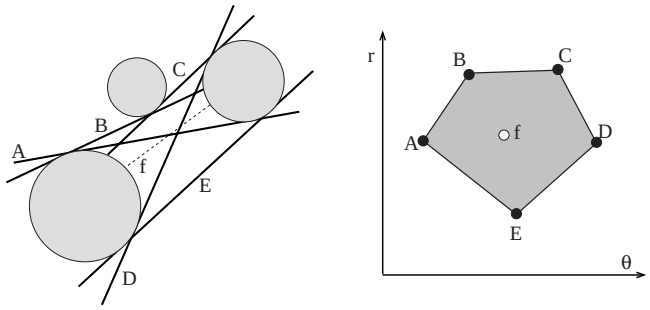
Pocchiola and Vegter [PV96b, PV96a] propose optimal output sensitive construction algorithms for the visibility complex of scenes of smooth objects. Rivière [Riv95, Riv97a] has developed an optimal construction algorithm for polygonal scenes.

The visibility complex implicitly encodes the visibility graph (see section 2 of chapter 10) of the scene : its vertices are the bitangents forming the visibility graph.



(a) (b)

FIG. 13.2: Topology of maximal free segments. (a) In the scene. (b) In a dual space where lines are mapped into points (the polar parameterization of line is used).



(a) (b)

FIG. 13.3: A face of the visibility complex. (a) In the scene. (b) In a dual space.

The 2D visibility complex has been applied to the 2D equivalent of lighting simulation by Orti *et al.* [ORDP96, DORP96]. The form factor between two objects corresponds to the face of the complex grouping the segments between these two objects. The limits of umbra and penumbra are the vertices (bitangents) of the visibility complex.

1.3 The 3D Visibility Complex

Durand *et al.* [DDP96, DDP97b] have proposed a generalization of the visibility complex for 3D scenes of smooth objects and polygons. The space of maximal free segments is then a 4D manifold embedded in 5D because of the branchings. Faces of the complex are bounded by tangent segments (which have 3 dimensions), bitangent segments (2 dimension), tritangent segments (1D) and finally vertices are segments tangent to four objects. If polygons are considered, the 1-faces are the EV and EEE critical lines.

The visibility complex is similar to the *asp*, but the same structure encodes the information for both perspective and orthographic projection. It moreover provides adjacencies between sets of segments.

Langer and Zucker [LZ97] have developed similar topological concepts (particularly the branchings) to describe the manifold of rays of a 3D scene in a shape-from-shading context.

See also section 4 where the difference between lines and maximal free segments is exploited.

1.4 Ray-classification

Ray classification is due to Arvo and Kirk [AK87]. The 5 dimensional space of rays is subdivided to accelerate ray-tracing computation. A ray is parameterized by its 3D origin and its direction which is encoded

on a cube for simpler calculations. Beams in ray-space are defined by an XYZ interval (an axis aligned box) and an interval on the cube of directions (see Fig. 13.4).

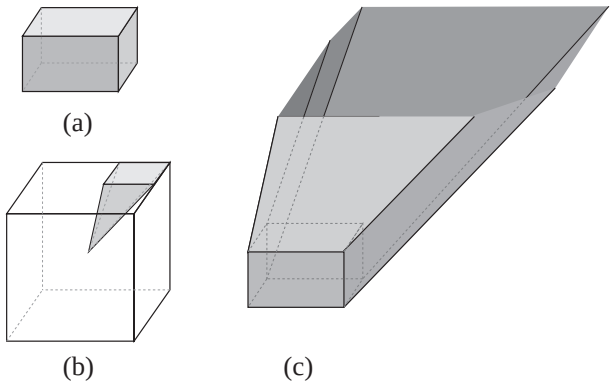


FIG. 13.4: Ray classification. (a) interval in origin space. (b) interval in direction space. (c) Corresponding beam of rays.

The objects lying in the beam are computed using a cone approximation of the beam. They are also sorted by depth to the origin box. Each ray belonging to the beam then needs only be intersected with the objects inside the beam. The ray-intervals are lazily and recursively constructed. See Fig. 13.5 for an example of result.

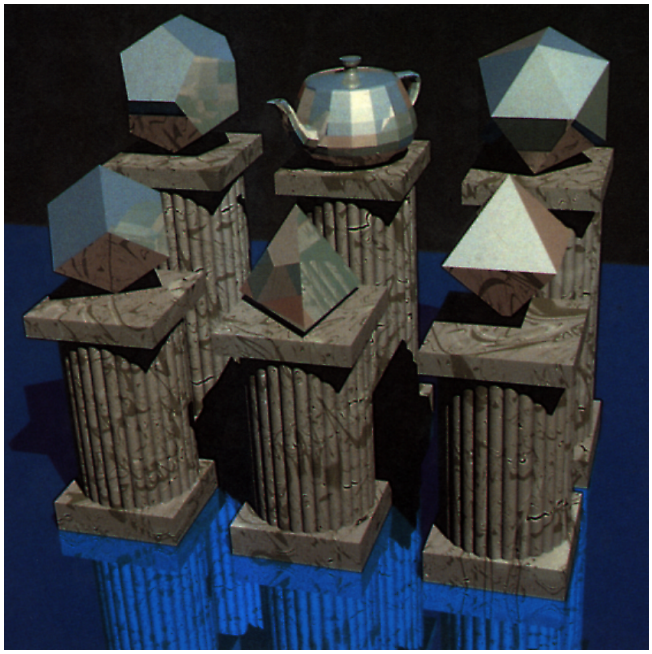


FIG. 13.5: Image computed using ray classification (courtesy of Jim Arvo and David Kirk, Apollo Computer Inc.)

Speer [Spe92b] describes similar ideas and Kwon *et al* [KKCS98] improve the memory requirements of ray-classification, basically by using 4D line space instead of 5D ray-space. This method is however still memory intensive, and it is not clear that it is much more efficient that 3D regular grids.

The concept of the light buffer presented in section 2.2 of chapter 11 has been adapted for linear and area light source by Poulin and Amanatides [PA91] and by Tanaka and Takahashi [TT95, TT97]. The rays going through the source are also classified into beams. The latter paper uses an analytical computation of the visible part of the light source using the cross-scanline method reviewed in section 6 of chapter 9.

Lamparter *et al.* [LMW90] discretize the space of rays (using adaptive quadtrees) and rasterize the objects of the scene using a z-buffer like method. Hinkenjann and Müller [HM96] propose a similar scheme to classify segments using a 6 dimensional space (3 for each extremity of a segment).

1.5 Multidimensional image-based approaches

Recently there has been great interest in both computer vision and computer graphics for the study of the description of a scene through the use of a multidimensional function in ray-space. A 3D scene can be completely described by the light traveling through each point of 3D space in each direction. This defines a 5D function named the *plenoptic function* by Adelson and Bergen [AB91].

The plenoptic function describes light transport in a scene, similar data-structures have thus been applied for global illumination simulation [LF96, LW95, GSHG98].

Gortler *et al.* [GGSC96] and Levoy and Hanrahan [LH96] have simplified the plenoptic function by assuming that the viewer is outside the convex hull of the scene and that light is not modified while traveling in free-space. This defines a function in the 4 dimensional space of lines called *lumigraph* or *light-field*. This space is discretized, and a color is kept for each ray. A view can then be extracted very efficiently from any viewpoint by querying rays in the data structure. This data structure is more compact than the storage of one view for each 3D point (which defines a 5D function) for the same reason exposed before : a ray is relevant for all the viewpoints lying on it. There is thus redundancy if light does not vary in free-space.

A two plane parameterization is used both in the light-field [LH96] and lumigraph [GGSC96] approaches (see Fig 8.2(b) page 163). Xu *et al.* [GGC97] have studied the form of some image features in this dual space, obtaining results similar to those obtained in the aspect graph literature [PD90, GCS91]. Camahort *et al.* [CLF98] have studied the (non) uniformity of this parameterization and proposed alternatives based on tessellations of the direction sphere. Their first parameterization is similar to the one depicted in Fig. 8.2(a) using a direction and an orthogonal plane, while the second uses parameterization line using two points on a sphere bounding the scene. See section 4 and the book by Santalo [San76] for the problems of measure and probability on sets of lines. See also the paper by Halle [Hal98] where images from multiple viewpoints (organised on a grid) are rendered simultaneously using epipolar geometry.

Chrysanthou *et al.* [CCOL98] have adapted the lumigraph methods to handle ray occlusion query. They re-introduce a fifth dimension to handle colinear rays, and their scheme can be seen as a discretization of the 3D visibility complex.

Wang *et al.* [WBP98] perform an occlusion culling preprocessing which uses concepts from shaft culling, ray classification and lumigraph. Using a two-plane parameterization of rays leaving a given cell of space, they recursively subdivide the set of rays until each beam can be classified as blocked by a single object or too small to be subdivided.

2 Graphs in line-space

In this section we present some methods which build a graph in line space which encodes the visual events of a scene. As opposed to the previous section, only one and zero dimensional sets of lines are considered.

2.1 The Visibility Skeleton

Durand *et al* [DDP97c, DDP97a] have defined the *visibility skeleton* which can be seen either as a simplification of the 3D visibility complex or as a graph in line space defined by the visual events.

Consider the situation represented in Fig. 13.6(a). A visual event V_1V_2 and the corresponding critical line set are represented. Recall that it is a one dimensional set of lines. It is bounded by two *extremal stabbing lines* V_1V_2 and V_1V_3. Fig. 13.6(b) shows another visual event V_2E_2 which is adjacent to the same extremal stabbing line. This defines a graph structure in line space represented in Fig. 13.6(c). The arcs are the 1D critical line sets and the nodes are the extremal stabbing lines. Other extremal stabbing lines include lines going through one vertex and two edges and lines going through four edges (see Fig. 13.7).

Efficient access to the arcs of this graph is achieved through a two dimensional array indexed by the polygons at the extremity of each visual event. The visibility skeleton is built by detecting the extremal stabbing lines. The adjacent arcs are topologically deduced thanks to a catalogue of adjacencies. This avoids explicit geometric calculations on the visual events.

The visibility skeleton has been implemented and used to perform global illumination simulation [DDP99]. Point-to-area form factors can be evaluated analytically, and the limits of umbra and penumbra can be quickly

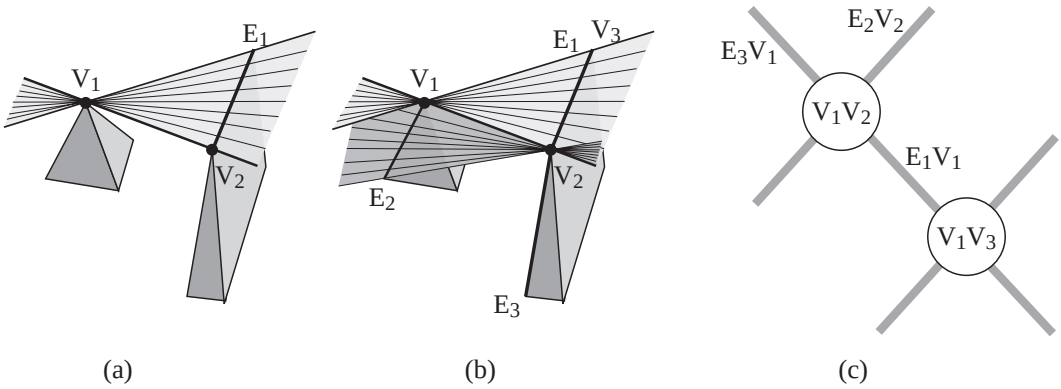


FIG. 13.6: (a) An EV critical line set. It is bounded by two extremal stabbing lines V_1V_2 and V_1V_3. (b) Other EV critical line sets are adjacent to V_1V_2. (c) Corresponding graph structure in line space.

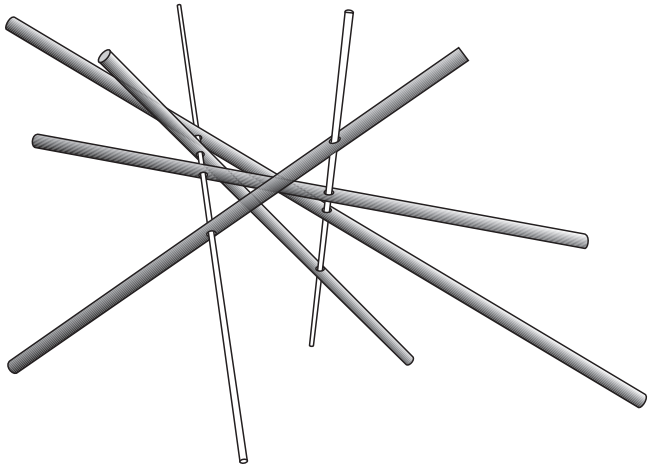


FIG. 13.7: Four lines in general position are stabbed by two lines (adapted from [Tel92b])

computed considering any polygon as a light source (as opposed to standard discontinuity meshing where only a small number of primary light sources are considered).

2.2 Skewed projection

McKenna et O'Rourke [MO88] consider a scene which is composed of lines in 3D space. Their aim is to study the class of another line in a sense similar to the previous section if the original lines are the edges of polyhedron, or to compute the mutually visible faces of polyhedra.

They use a *skewed projection* to reduce the problem to 2D computations. Consider a pair of lines L_1 and L_2 as depicted in Fig. 13.8. Consider the segment joining the two closest points of the lines (shown dashed) and the plane P orthogonal to this segment and going through its mid-point. Each point on P defines a unique line going through L_1 and L_2. Consider a third line L_3. It generates EEE critical lines. The intersections of these critical lines with plane P lie on an hyperbola H.

The intersections of the hyperbolae defined by all other lines of the scene allow the computation of the extremal stabbing lines stabbing L_1 and L_2. The computation of course has to be performed in the $O(n^2)$ planes defined by all pairs of lines. A graph similar to the visibility skeleton is proposed (but for sets of lines). No implementation is reported.

The skewed projection duality has also been used by Jaromczyk and Kowaluk [JK88] in a stabbing context and by Bern *et al.* [BDEG90] to update a view along a linear path (the projection is used to compute the visual events at which the view has to be updated).

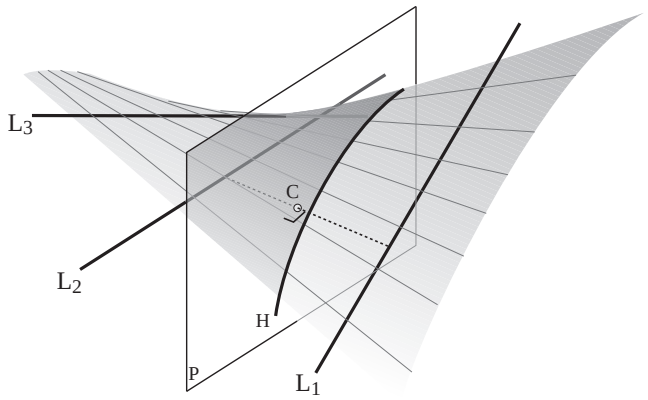


FIG. 13.8: Skewed projection.

3 Plücker coordinates

3.1 Introduction to Plücker coordinates

Lines in 3D space can not be parameterized continuously. The parameterizations which we have introduced in section 1.4 of chapter 8 both have singularities. In fact there cannot be a smooth parameterization of lines in 4D without singularity. One intuitive way to see this is to note that it is not possible to parameterize the S^2 sphere of directions with two parameters without a singularity. Nevertheless, if S^2 is embedded in 3D, there is a trivial parameterization, *i.e.* x,y,z. However not all triples of coordinates correspond to a point on S^2.

Similarly, oriented lines in space can be parameterized in a 5D space with the so-called *Plücker coordinates* [Plü65]. The equations are given in appendix D, here we just outline the principles. One nice property of Plücker coordinates is that the set of lines which intersect a given line a is a hyperplane in Plücker space (its dual Π_a ; The same notation is usually used for the dual of a line and the corresponding hyperplane). It separates Plücker space into oriented lines which turn around ℓ clockwise or counterclockwise (see Fig. 13.9).

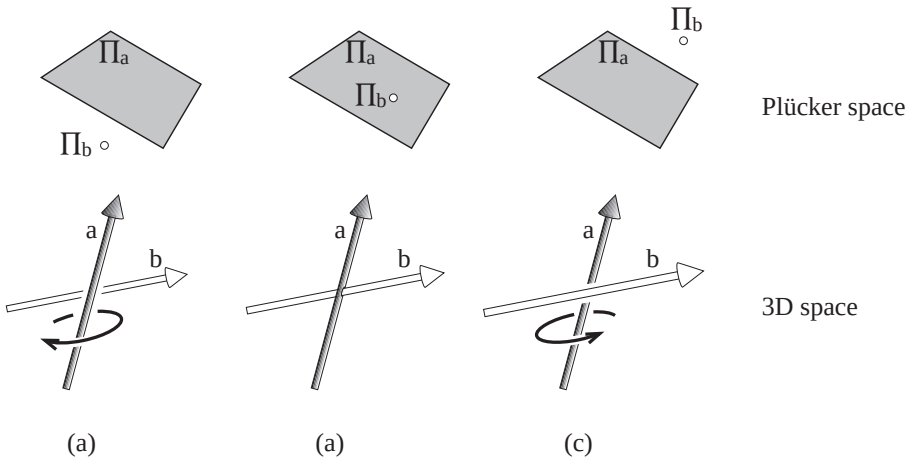


FIG. 13.9: In Plücker space the hyperplane corresponding to a line a separates lines which turn clockwise and counterclockwise around a. (The hyperplane is represented as a plane because a five-dimensional space is hard to illustrate, but note that the hyperplane is actually 4D).

As for the embedding of S^2 which we have presented, not all 5-uples of coordinates in Plücker space correspond to a real line. The set of lines in this parameterization lie on a quadric called the *Plücker hypersurface* or *Grassman manifold* or *Klein quadric*.

Now consider a triangle in 3D space. All the lines intersecting it have the same orientation with respect to the three lines going through its edges (see Fig. 13.10). This makes stabbing computations very elegant in

Plücker space. Linear calculations are performed using the hyperplanes corresponding to the edges of the scene, and the intersection of the result with the Plücker hypersurface is then computed to obtain real lines.

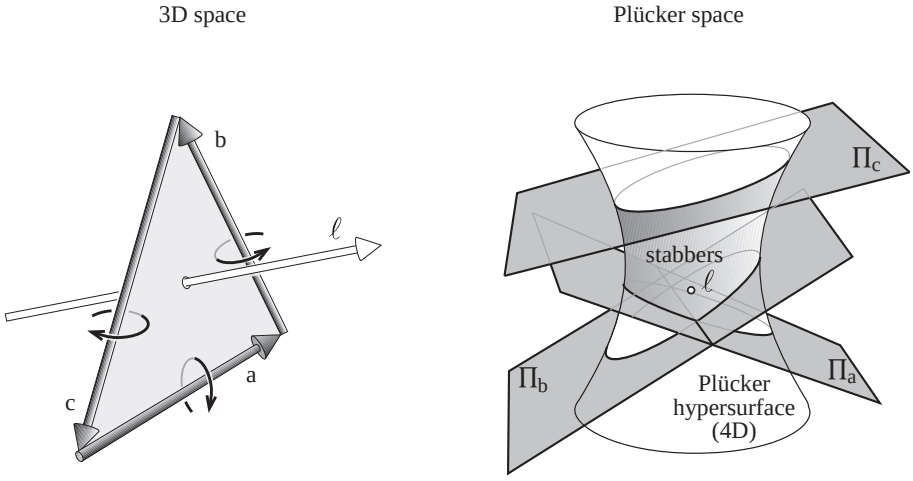


FIG. 13.10: Lines stabbing a triangle. In 3D space, if the edges are well oriented, all stabbers rotate around the edges counterclockwise. In Plücker space this corresponds to the intersection of half spaces. To obtain real lines, the intersection with the Plücker hypersurface must be considered. (In fact the hyperplanes are tangent to the Plücker hypersurface)

Let us give a last example of the power of Plücker duality. Consider three lines in 3D space. The lines stabbing each line lie on its (4D) hyperplanes in Plücker space. The intersection of the three hyperplane is a 2D plane in Plücker space which can be computed easily. Once intersected with the Plücker hypersurface, we obtain the EEE critical line set as illustrated Fig. 13.11.

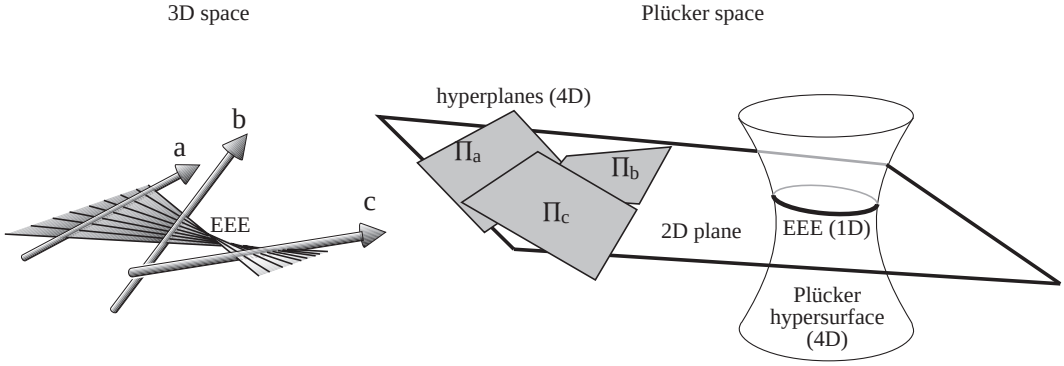


FIG. 13.11: Three lines define a EEE critical line set in 3D space. This corresponds to the intersection of hyperplanes (not halfspaces) in Plücker space. Note that hyperplanes are 4D while their intersection is 2D. Unfortunately they are represented similarly because of the lack of dimensions of this sheet of paper.(adapted from [Tel92b]).

More detailed introductions to Plücker coordinates can be found in the books by Sommerville [Som51] or Stolfi [Sto91] and in the thesis by Teller [Tel92b] [1]. See also Appendix D.

3.2 Use in computational geometry

Plücker coordinates have been used in computational geometry mainly to find stabbers of sets of polygons, for ray-shooting and to classify lines with respect to sets of lines (given a set of lines composing the scene and two query lines, can we continuously move the first to the second without intersecting the lines of the scene).

[1] Plücker coordinates can also be extended to use the 6 coordinates to describe forces and motion. See *e.g.* [MS85, PPR99]

We give an overview of a paper by Pellegrini [Pel93] which deals with ray-shooting in a scene composed of triangles. He builds the arrangement of hyperplanes in Plücker space corresponding to the scene edges. He shows that each cell of the arrangement corresponds to lines which intersect the same set of triangles. The whole 5D arrangement has to be constructed, but then only cells intersecting the Plücker hypersurface are considered. He uses results by Clarkson [Cla87] on point location using random sampling to build a point-location data-structure on this arrangement. Shooting a ray then consists in locating the corresponding line in Plücker space. Other results on ray shooting can be found in [Pel90, PS92, Pel94].

This method is different in spirit from ray-classification where the object-beam classification is calculated in object space. Here the edges of the scene are transformed into hyperplanes in Plücker space.

The first use of Plücker space in computational geometry can be found. in a paper by Chazelle *et al.* [CEG+96]. The orientation of lines in space also has implications on the study of cycles in depth order as studied by Chazelle *et al.* [CEG+92] who estimate the possible number of cycles in a scene . Other references on lines in space and the use of Plücker coordinates can be found in the survey by Pellegrini [Pel97b].

3.3 Implementations in computer graphics

Teller [Tel92a] has implemented the computation of the *antipenumbra* cast by a polygonal source through a sequence of polygonal openings *portals* (*i.e.* the part of space which may be visible from the source). He computes the polytope defined by the edges of all the openings, then intersects this polytope with the Plücker hypersurface, obtaining the critical line sets and extremal stabbing lines bounding the antipenumbra (see Fig. 13.12 for an example).

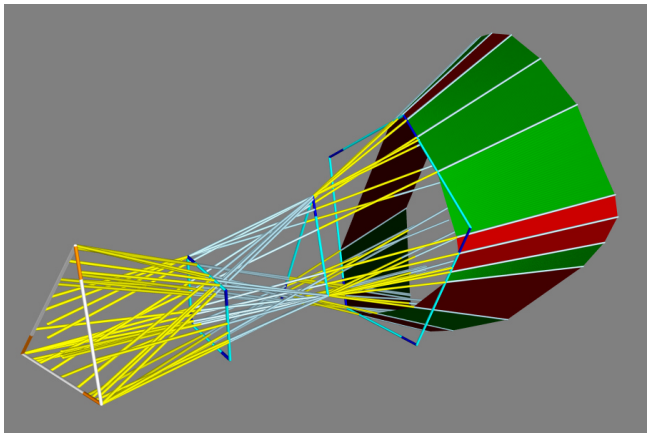


FIG. 13.12: Antipenumbra cast by a triangular light source through a sequence of three polygonal openings. EEE boundaries are in red (image courtesy of Seth J. Teller, University of Berkeley).

He however later noted [TH93] that this algorithm is not robust enough for practical use.

Nevertheless, in this same paper he and Hanrahan [TH93] actually used Plücker coordinates to classify the visibility of objects with respect to parts of the scene in a global illumination context for architectural scenes (see section 7 of chapter 10). They avoid robustness issues because no *geometric construction* is performed in 5D space (like computing the intersection between two hyperplanes), only *predicates* are evaluated (“is this point above this hyperplane ?”).

4 Stochastic approaches

This section surveys methods which perform visibility calculation using a probabilistic sampling in line-space.

4.1 Integral geometry

The most relevant tool to study probability over sets of lines is *integral geometry* introduced by Santalo [San76]. Defining probabilities and measure in line-space is not straightforward. The most natural constraint is to impose that this measure be invariant under rigid motion. This defines a unique measure in line-space, up to a scaling factor.

Probabilities can then be computed on lines, which is a valuable tool to understand ray-casting. For example, the probability that a line intersects a convex object is proportional to its surface.

An unexpected result of integral geometry is that a uniform sampling of the lines intersecting a sphere can be obtained by joining pairs of points uniformly distributed on the surface of the sphere (note that this is not true in 2D).

The classic parameterization of lines $x = az + p$, $y = bz + q$ (similar to the two plane parameterization of Fig. 8.2(b) page 163) has density $\frac{dadbdpdq}{(1+a^2+b^2)^2}$. If a, b, p, q are uniformly and randomly sampled, this formula expresses the probability that a line is picked. It also expresses the variation of sampling density for light-field approaches described in section 1.5. Regions of line space with large values of a, b will be more finely sampled. Intuitively, sampling is higher for lines that have a gazing angle with the two planes used for the parameterization.

Geometric probability is also covered in the book by Solomon [Sol78].

4.2 Computation of form factors using ray-casting

Most radiosity implementations now use ray-casting to estimate the visibility between two patches, as introduced by Wallace *et al.* [WEH89]. A number of rays (typically 4 to 16) are cast between a pair of patches. The number of rays can vary, depending on the *importance* of the given light transfer. Such issues will be treated in section 1.1 of chapter 14.

The integral geometry interpretation of form factors has been studied by Sbert [Sbe93] and Pellegrini [Pel97a]. They show that the form factor between two patches is proportional the probability that a line intersecting the first one intersects the second. This is the measure of lines intersecting the two patches divided by the measure of lines intersecting the first one. Sbert [Sbe93] proposes some estimators and derives expressions for the variance depending on the number of rays used.

4.3 Global Monte-Carlo radiosity

Buckalew and Fussel [BF89] optimize the intersection calculation performed on each ray. Indeed, in global illumination computation, all intersections of a line with the scene are relevant for light transfer. As shown in Fig. 13.13, the intersections can be sorted and the contribution computed for the interaction between each consecutive pair of objects. They however used a fixed number of directions and a deterministic approach.

Sbert [Sbe93] introduced *global Monte-Carlo radiosity*. As in the previous approach all intersections of a line are taken into account, but a uniform random sampling of lines is used, using pairs of points on a sphere.

Related results can be found in [Neu95, SPP95, NNB97]. Efficient hierarchical approaches have also been proposed [TWFP97, BNN+98].

4.4 Transillumination plane

Lines sharing the same direction can be treated simultaneously in the previous methods. This results in a sort of orthographic view where light transfers are computed between consecutive pairs of objects overlapping in the view, as shown in Fig. 13.14.

The plane orthogonal to the projection direction is called the *transillumination plane*. An adapted hidden-surface removal method has to be used. The z-buffer can be extended to record the z values of all objects projecting on a pixel [SKFNC97], or an analytical method can be used [Pel99, Pel97a].

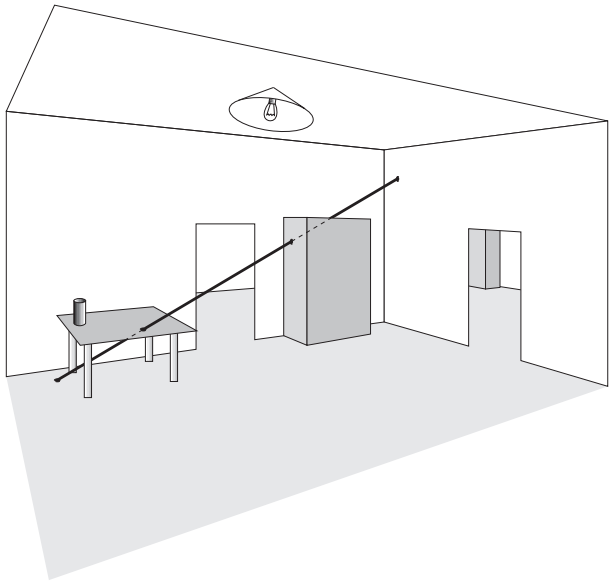


FIG. 13.13: Global Monte-Carlo radiosity. The intersection of the line in bold with the scene allows the simulation of light exchanges between the floor and the table, between the table and the cupboard and between the cupboard and the ceiling.

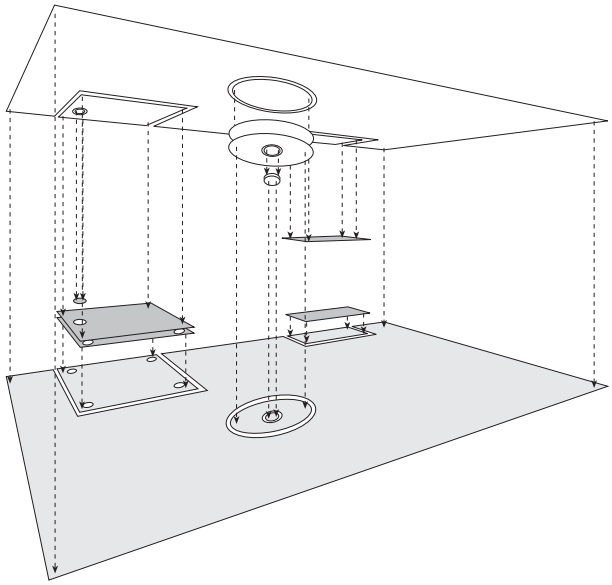


FIG. 13.14: Transillumination plane. The exchanges for one direction (here vertical) are all evaluated simultaneously using an extended hidden surface removal algorithm.

CHAPITRE 14

Advanced issues

Au reste, il n'est pas inutile de remarquer que tout ce qu'on démontre, soit dans l'optique, soit dans la perspective sur les ombres des corps, est exact à la vérité du côté mathématique, mais que si on traite cette matière physiquement, elle devient alors fort différente. L'explication des effets de la nature dépend presque toujours d'une géométrie si compliquée qu'il est rare que ces effets s'accordent avec ce que nous en aurions attendu par nos calculs.

FORMEY, article sur l'ombre de l'*Encyclopédie*.

E NOW TREAT two issues which we believe crucial for visibility computations and which unfortunately have not received much attention. Section 1 deals with the control of the precision of computations either to ensure that a required precision is satisfied, or to simplify visibility information to make it manageable. Section 2 treats methods which attempt to take advantage of temporal coherence in scenes with moving objects.

1 Scale and precision

Visibility computations are often involved and costly. We have surveyed some approximate methods which may induce artifacts, and some exact methods which are usually resource-intensive. It is thus desirable to control the error in the former, and trade-off time versus accuracy in the latter. Moreover, all visibility information is not always relevant, and it can be necessary to extract what is useful.

1.1 Hierarchical radiosity : a paradigm for refinement

Hierarchical radiosity [HSA91] is an excellent paradigm of refinement approaches. Computational resources are spent for "important" light exchanges. We briefly review the method and focus on the visibility problems involved.

In hierarchical radiosity the scene polygons are adaptively subdivided into patches organised in a pyramid. The radiosity is stored using Haar wavelets [SDS96] : each quadtree node stores the average of its children. The light exchanges are simulated at different levels of precision : exchanges will be simulated between smaller elements of the quadtree to increase precision as shown in Fig. 14.1. *Clustering* improves hierarchical radiosity by using a full hierarchy which groups *clusters* of objects [SAG94, Sil95].

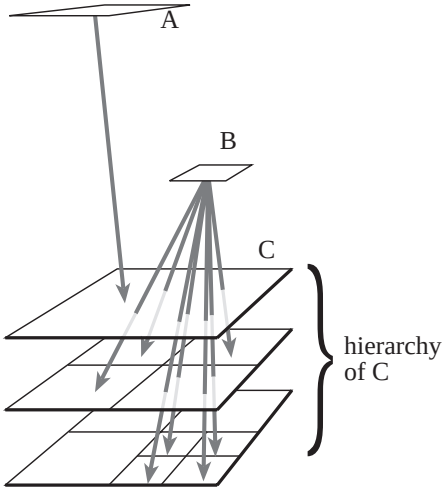


FIG. 14.1: Hierarchical radiosity. The hierarchy and the exchanges arriving at *C* are represented. Exchanges with *A* are simulated at a coarser level, while those with *B* are refined.

The crucial component of a hierarchical radiosity system is the *refinement criterion* (or *oracle*) which decides at which level a light transfer will be simulated. Originally, Hanrahan *et al.* [HSA91] used a radiometric criterion (amount of energy exchanged) and a visibility criterion (transfers with partial visibility are refined more). This results in devoting more computational resources for light transfers which are important and in shadow boundary regions. See also [GH96].

For a deeper analysis and treatment of the error in hierarchical radiosity, see *e.g.*, [ATS94, LSG94, GH96, Sol98, HS99].

1.2 Other shadow refinement approaches

The volumetric visibility method presented in section 1.3 of chapter 10 is also well suited for a progressive refinement scheme. An oracle has to decide at which level of the volumetric hierarchy the transmittance has to be considered. Sillion and Drettakis [SD95] use the size of the *features* of the shadows.

The key observation is that larger object which are closer to the receiver cast more significant shadows, as illustrated by Fig. 14.2. They moreover take the *correlation* of multiple blockers into account using an image-based approach. The objects inside a cluster are projected in a given direction onto a plane. Bitmap erosion operators are then used to estimate the size of the connected portions of the blocker projection. This can be seen as a first approximation of the convolution method covered in section 6 of chapter 11 [SS98a].

Soler and Sillion [SS96b, Sol98] propose a more complete treatment of this refinement with accurate error bounds. Unfortunately, the bounds are harder to derive in 3D and provide looser estimates.

The refinement of shadow computation depending on the relative distances of blockers and source has also been studied by Asensio [Ase92] in a ray-tracing context.

Telea and van Overveld [Tv97] efficiently improve shadows in radiosity methods by performing costly visibility computations only for blockers which are close to the receiver.

1.3 Perception

The goal of most image synthesis methods is to produce images which will be seen by human observers. Gibson and Hubbold [GH97b] thus perform additional computation in a radiosity method only if they may induce a change which will be noticeable. Related approaches can be found in [Mys98, BM98, DDP99, RPG99].

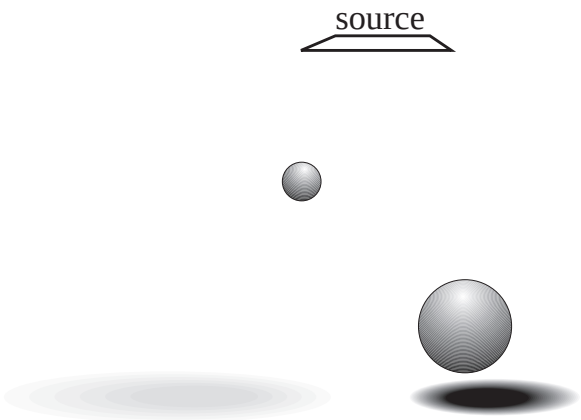


FIG. 14.2: Objects which are larger and closer to the receiver cast more significant shadows. Note that the left hand sphere casts no umbra, only penumbra.

Perceptual metrics have also been applied to the selection of discontinuities in the illumination function [HWP97, DDP99].

1.4 Explicitly modeling scale

One of the major drawbacks of aspect graphs [FMA+92] is that they have been defined for perfect views : all features are taken into account, no matter the size of their projection.

The *Scale-space aspect graph* has been developed by Eggert *et al.* [EBD+93] to cope with this. They discuss different possible definitions of the concept of "scale". They consider that two features are not distinguishable when their subtended angle is less than a given threshold. This defines a new sort of visual event, which corresponds to the visual merging of two features. These are circles in 2D (the set of points which form a given angle with a segment is a circle). See Fig. 14.3.

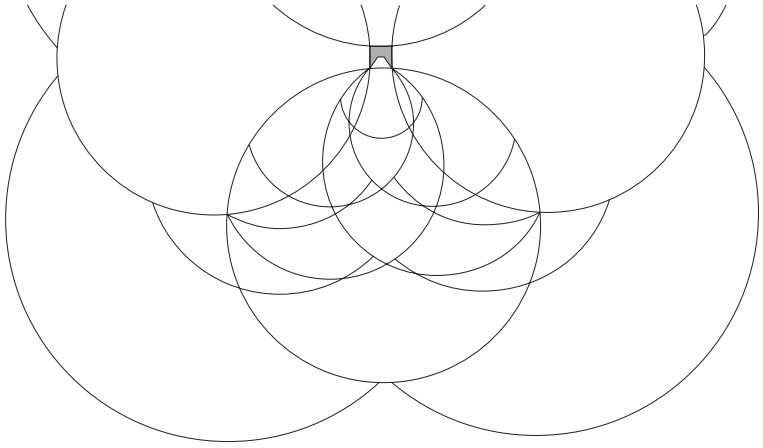
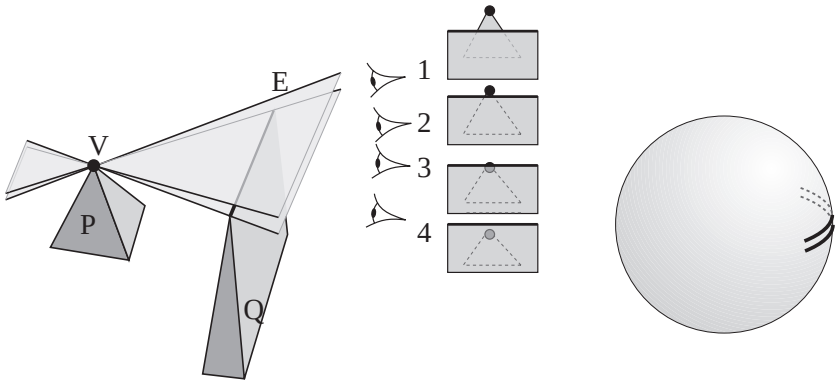


FIG. 14.3: Scale-space aspect graph in 2D using perspective projection for the small object in grey. Features which subtend an angle of less than 4° are considered indistinguishable. The circles which subdivide the plane are the visual events where features of the object visually merge.

Scale (the angle threshold) defines a new dimension of the viewpoint space. Fig. 14.3 in fact represents a slice $scale = 4°$ of the scale-space aspect graph. Cells of this aspect graph have a scale extent, and their boundaries change with the scale parameter. This approach allows an explicit model of the resolution of features, at the cost of an increases complexity.

Shimshoni and Ponce [SP97] developed the *finite resolution aspect graph* in 3D. They consider orthographic projection and a single threshold. When resolution is taken into account, some *accidental* views are likely

to be observed : An edge and a vertex seem superimposed in the neighbourhood of the exact visual event. Visual events are thus doubled as illustrated in Fig. 14.4.



(a) (b)

FIG. 14.4: Finite resolution aspect graph. (a) The *EV* event is doubled. Between the two events (viewpoint 2 and 3), *E* and *V* are visually superimposed. (b) The doubled event on the viewing sphere.

For the objects they test, the resulting finite resolution aspect graph is larger. The number events discarded because the generators are merged does not compensate the doubling of the other events. However, tests on larger objects could exhibit different results.

See also the work by Weinshall and Werman on the likelihood and stability of views [WW97].

1.5 Image-space simplification for discontinuity meshing

Stewart and Karkanis [SK98] propose a finite resolution construction of discontinuity meshes using an image-space approach. They compute views from the vertices of the polygonal source using a z-buffer. The image is segmented to obtain a visibility map. The features present in the images are used as visual event generators.

This naturally eliminates small objects or features since they aggregate in the image. Robustness problems are also avoided because of the image-space computations. Unfortunately, only partial approximate discontinuity meshes are obtained, no backprojection computation is proposed yet.

2 Dynamic scenes

We have already evoked *temporal coherence* in the case of a moving viewpoint in a static scene (section 4.2 of chapter 12). In this section we treat the more general case of a scene where objects move. If the motions are continuous, and especially if few objects move, there is evidence that computation time can be saved by exploiting the similarity between consecutive timesteps.

In most cases, the majority of the objects are assumed static while a subset of objects actually move. We can distinguish cases where the motion of the objects is known in advance, and those where no *a priori* information is known, and thus updates must be computed on a per frame basis.

Different approaches can be chosen to take advantage of coherence :

- The computation is completely re-performed for a sub-region of space ;
- The dynamic objects are deleted (and the visibility information related to them is discarded) then re-inserted at their new position ;
- A validity time-interval is computed for each piece of information ;
- The visibility information is “smoothly updated”.

2.1 Swept and motion volumes

A *swept volume* is the volume swept by an object during a time interval. Swept volumes can also be used to bound the possible motion of an object, especially in robotics where the degrees of freedom are well defined [AA95]. These swept volumes are used as static blockers.

A *motion volume* is a simplified version of swept volumes similar to the shafts defined in section 6.1 of chapter 10. They are simple volume which enclose the motion of an object. Motion volumes were first used in radiosity by Baum *et al.* [BWCG86] to handle the motion of one object. A hemicube is used for form-factor computation. Pixels where the motion volume project are those which need recomputation.

Shaw [Sha97] and Drettakis and Sillion [DS97] determine form factors which require recomputation using a motion volume-shaft intersection technique.

Sudarsky and Gotsman [SG96] use motion volumes (which they call *temporal bounding volumes*) to perform occlusion culling with moving objects. They alleviate the need to update the spatial data-structure (BSP or octree) for each frame, because these volumes are used in place of the objects, making computations valid for more than one frame.

2.2 4D methods

Some methods have been proposed to speed-up ray-tracing animations using a four dimensional space-time framework developed by Glassner [Gla88]. The temporal extent of ray-object intersections is determined, which avoids recomputation when a ray does not intersect a moving object. See also [MDC93, CCD91] for similar approaches.

Ray-classification has also been extended to 6D (3 for the origin of a ray, 2for its direction, and 1 for time) [Qua96, GP91].

Global Monte-Carlo radiosity presented in section 4.3 of chapter 13 naturally extends to 4D as demonstrated by Besuievsky *et al* [BS96]. Each ray-static object intersection is used for the whole length of the animation. Only intersections with moving objects require recomputation.

2.3 BSP

BSP trees have been developed for rapid view computation in static scenes. Unfortunately, their construction is a preprocessing which cannot be performed for each frame.

Fuchs *et al.* [FAG83] consider pre-determined paths and place bounding planes around the paths. Torres [Tor90] builds a multi-level BSP tree, trying to separate objects with different motion without splitting them.

Chrysanthou and Slater [CS92, CS95, CS97] remove the moving objects from the database, update the BSP tree, and then re-introduce the object at its new location. The most difficult part of this method is the update of the BSP tree when removing the object, especially when the polygons of the object are used at a high level of the tree as splitting planes. In this case, all polygons which are below it in the BSP-tree have to be updated in the tree. This approach was also used to update limits of umbra and penumbra [CS97].

Agarwal *et al.* [AEG98] propose an algorithm to maintain the cylindrical BSP tree which we have presented in section 1.4 of chapter 10. They compute the events at which their BSP actually needs a structural change. This happens when a triangle becomes vertical, when an edge becomes parallel to the *yz* plane, or when a triangle enters or leaves a cell defined by the BSP tree.

2.4 Aspect graph for objects with moving parts

Bowyer *et al.* [EB93] discuss the extension of aspect graphs for articulated assemblies. The degrees of freedom of the assembly increase the dimensionality of viewpoint space (which they call aspect space). For example, if the assembly has only one translational degree of freedom and if 3D perspective is used, the aspect graph has to be computed in 4D, 3 dimensions for the viewpoint and one for translation. This is similar to the scale-space aspect graph presented in section 1.4 where scale increases dimensionality.

Accidental configurations correspond to values of the parameters of the assembly where the aspect graph changes. They occur at a generalization of visual events in the higher dimensional aspect space. For example when two faces become parallel.

Two extensions of the aspect graph are proposed, depending on the way accidental configurations are handled. They can be used to partition aspect space like in the standard aspect graph definition. They can also be used to partition first the configuration space (in our example, it would result in intervals of the translational parameter), then a different aspect graph is computed for each cell of the configuration space partition. This latter approach is more memory demanding since cells of different aspect graphs are shared in the first approach. Construction algorithms are just sketched, and no implementation is reported.

2.5 Discontinuity mesh update

Loscos and Drettakis [LD97] and Worall *et al*. [WWP95, WHP98] maintain a discontinuity mesh while one of the blockers moves. Limits of umbra and penumbra move smoothly except when an object starts or stops casting shadows on another one. Detecting when a shadow limit goes off an object is easy.

To detect when a new discontinuity appears on one object, the discontinuities cast on other objects can be used as illustrated in Fig. 14.5.

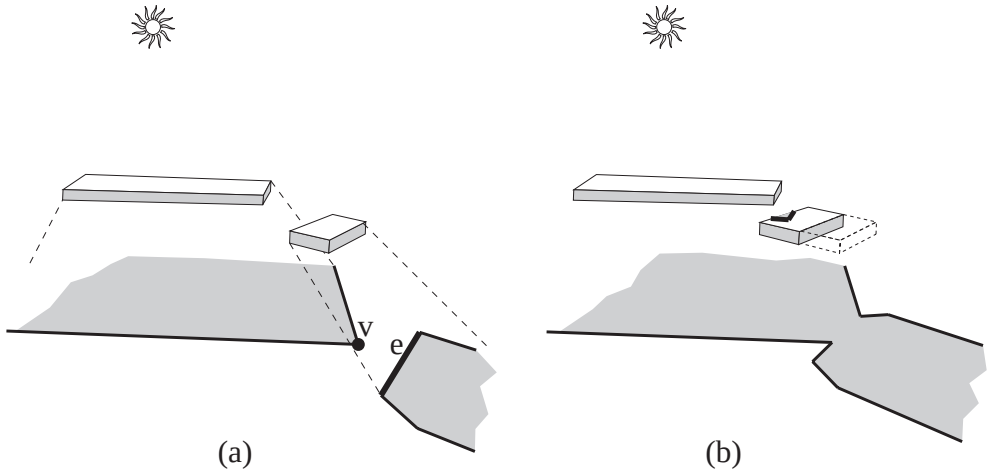


FIG. 14.5: Dynamic update of limits of shadow. The situation where shadows appear on the moving object can be determined by checking the shadows on the floor. This can be generalized to discontinuity meshes (after [LD97]).

2.6 Temporal visual events and the visibility skeleton

In chapter 2 and 3 of this thesis, we have presented the notion of a *temporal visual event*. Temporal visual events permit the generalization of the results presented in the previous section. They correspond to the accidental configurations studied for the aspect graph of an assembly.

Temporal visual events permit the update of the visibility skeleton while objects move in the scene. This is very similar to the static visibility skeleton, since temporal visual events describe adjacencies which determine which nodes and arcs of the skeleton should be modified.

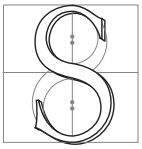
Similarly, a catalogue of singularities has been developed for moving objects, defining a *temporal visibility complex*.

CHAPITRE 15

Conclusions of the survey

Ils ont tous gagné !

Jacques MARTIN



URVEYING work related to visibility reveals a great wealth of solutions and techniques. The organisation of the second part of this thesis has attempted to structure this vast field. We hope that this survey will be an opportunity to derive new methods or improvements from techniques developed in other fields. Considering a problem under different angles is a powerful way to open one's mind and find creative solutions. We again invite the reader not to consider our classification as restrictive ; on the contrary, we suggest that methods which have been presented in one space be interpreted in another space. In what follows, we give a summary of the methods which we have surveyed, before presenting a short discussion.

1 Summary

In chapter 7 we have presented visibility problems in various domains : computer graphics, computer vision, robotics and computational geometry.

In chapter 8 we have propose a classification of these methods according to the space in which the computations are performed : object space, image space, viewpoint space and line-space. We have described the *visual events* and the *singularities of smooth mappings* which explain "how" visibility changes in a scene : the appearance or disappearance of objects when an observer moves, the limits of shadows, etc.

We have briefly surveyed the classic hidden-part removal methods in chapter 9.

In chapter 10 we have dealt with object-space methods. The two main categories of methods are those which use a "regular" spatial decomposition (grid, hierarchy of bounding volumes, BSP trees), and those which use frusta or shafts to characterize visibility. Among the latter class of methods, the main distinction is between those which are interested in determining if a point (or an object) lies inside the frustum or shaft, and those which compute the boundaries of the frustum (*e.g.*, shadow boundaries). Fundamental data-structures have also been presented : The 2D visibility graph used in motion planning links all pairs of mutually visible vertices of a planar polygonal scene, and the visual hull of an object A represents the largest object with the same occlusion properties as A.

Images-space methods, surveyed in chapter 11 perform computation directly in the plane of the final image, or use an intermediate plane. Most of them are based on the z-buffer algorithm.

Chapter 12 has presented methods which consider viewpoints and the the visibility properties of the corresponding views. The aspect graph encodes all the possible views of an object. The viewpoints are partitioned into cells where a view is qualitatively invariant, that is, the set of visible features remains constant. The boundaries of such cells are the visual events. This structure has important implications and applications in computer vision, robotics, and computer graphics. We have also presented methods which optimize the viewpoint according to the visibility of a feature, as well as methods based on visual events which take advantage of *temporal coherence* by predicting when a view changes.

In chapter 13 we have surveyed work in line or ray space. We have presented methods which partition the rays according to the object they see. We have seen that visual events can be encoded by lines in line-space. A powerful dualisation has been studied which maps lines into five dimensional points, allowing for efficient and elegant visibility characterization. We have presented some elements of probability over sets of lines, and their applications to lighting simulation.

Finally, in the previous chapter we have discussed two important issues : precision and moving objects. We have studied techniques which refine their computations where appropriate, as well as techniques which attempt to cope with intensive and intricate visibility information by culling too fine and unnecessary information. Techniques developed to deal with dynamic scenes include swept or motion volumes, 4D method (where time is the fourth dimension), and smooth updates of BSP trees or shadow boundaries.

Table 15.1 summarizes the techniques which we have presented, by domain and space.

2 Discussion

A large gap exists between exact and approximate methods. Exact methods are often costly and prone to robustness problems, while approximate methods suffer from aliasing artifacts. Smooth trade-off and efficient adaptive approximate solutions should be developed. This requires both to be able to refine a computation and to efficiently determine the required accuracy.

Visibility with moving objects and temporal coherence have received little attention. Dynamic scenes are mostly treated as successions of static timesteps for which everything is recomputed from scratch. Solutions should be found to efficiently identify the calculations which actually need to be performed after the movement of objects.

As evoked in the introduction of this survey, no practical guide to visibility techniques really exists. Some libraries or programs are available (see for example appendix E) but the implementation of reusable visibility code in the spirit of *C-GAL* [FGK+96] would be a major contribution, especially in the case of 3D visibility.

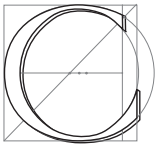
	Object space	Image space	Viewpoint space	Line space
view computation	BSP			
Hard shadow	shadow volume shadow BSP	shadow map shadow cache		
Soft shadows	limits of penumbra	sampling convolution	complete discontinuity mesh	antipenumbra
Occlusion culling	use of frusta architectural scenes from a volume	hierarchical z-buffer shadow footprints	temporal coherence use of the aspect graph	line-space subdivision
Ray-tracing	bounding volumes space subdivision beam-tracing	item and light buffer ZZ-buffer		ray classification
Radiosity	discontinuity mesh shaft culling volumetric visibility architectural scenes	hemicube convolution	complete discontinuity mesh	visibility skeleton global Monte Carlo Plücker for blockers
Image-based		epipolar rendering	silhouette tree	lumigraph, light-field
Object recognition			aspect graph	asp
Contour intersection	visual hull			
Sensor placement	viewpoint constraint best-next view		viewpoint optimization art gallery	
Motion planning	visibility graph	use of the z-buffer	pursuit-evasion self localisation target tracking	

TAB. 15.1: Recapitulation of the techniques presented by field and by space.

Conclusion

Étudie d'abord la science, puis apprends la pratique née de cette science.

Léonard DE VINCI, *Codex Urbinas*



E MÉMOIRE décrit des travaux à la fois théoriques et pratiques sur la visibilité. Grâce à une étude dans l'espace des droites, nous avons pu mieux comprendre les propriétés de visibilité d'une scène tridimensionnelle ainsi que leur cohérence. Cela nous a permis de développer une structure efficace et flexible, qui a démontré son utilité dans une application particulièrement exigeante en calculs de visibilité : la simulation de l'éclairage. Nous avons présenté une méthode efficace de précalcul pour l'affichage de scènes très complexes. Elle permet de ramener des propriétés tridimensionnelles d'occultation à des tests simples sur un plan. Enfin, nous avons proposé un vaste tour d'horizon des travaux sur la visibilité. Dans ce qui suit, nous résumons tout d'abord les contributions de cette thèse, puis nous proposons des pistes de travaux futurs, avant de conclure.

Résumé des contributions

Le complexe de visibilité tridimensionnelle

Le *complexe de visibilité* propose un nouveau regard sur les questions de visibilité. Puisqu'il est fondé sur la notion atomique de rayon, tout problème de visibilité s'y exprime naturellement. La notion de segment libre maximal permet de simplifier l'espace des rayons en une variété quadridimensionnelle, ce qui facilite la description des propriétés et rend mieux compte de la véritable dimensionnalité des problèmes. Le complexe de visibilité permet d'expliciter la notion de *cohérence* ; chaque cellule contient un ensemble connexe de segments qui voient les mêmes objets, et les adjacences entre les faces de différentes dimensions représentent la structure des limites de visibilité d'une scène (limites d'ombre et de pénombre, apparition ou disparition d'objets lors du déplacement du point de vue, etc.). Le complexe de visibilité décrit en un même cadre toutes les propriétés de visibilité d'une scène composée de polygones ou d'objets courbes, ainsi que leurs changements en cas de mouvement.

La complexité théorique de la taille du complexe est vague, entre linéaire et $O(n^4)$ en fonction du nombre d'objets, ce qui donne peu d'information sur son comportement en pratique. Nous avons décrit un modèle théorique simplifié de scène "normale", et un calcul probabiliste a permis de montrer que la taille du complexe de visibilité est dans ce cas $O(n^{2.67})$.

Le squelette de visibilité

Pour éviter le traitement d'un complexe cellulaire quadridimensionnel, nous avons simplifié le complexe de visibilité en un graphe dans l'espace des droites, le *squelette de visibilité*. L'algorithme de construction que nous avons développé est topologique et local : calculer les nœuds du graphe et en déduire les arcs grâce à un catalogue d'adjacences. Toutes les méthodes à base d'événements visuels imposaient le traitement complexe et délicat de surfaces (souvent non planes) dans l'espace tridimensionnel, ou de courbes sur la sphère des directions. Nous ne considérons que de simples droites, ce qui rend notre construction robuste et flexible. Les événements visuels géométriquement complexes sont déduit par des adjacences topologiques simples.

Grâce à cet algorithme, nous avons montré qu'il est possible d'implémenter une structure de visibilité tridimensionnelle globale et générique. Une fois la structure construite, elle permet des requêtes très efficaces (quelques millisecondes pour un calcul de limites de pénombre par exemple). Le squelette de visibilité est plus simple et plus robuste que les approches antérieures comme les maillages de discontinuité qui sont pourtant plus restreintes, puisqu'elles ne traitent la visibilité que depuis les sources de lumière.

Les applications potentielles du squelette dépassent la simulation de l'éclairage. La flexibilité de sa construction "à la demande" permettrait l'adaptation à bien des problèmes. Le calcul de graphes d'aspect ou de zones de visibilité d'un objet comptent parmi les nombreuses possibilités.

Radiosité hiérarchique guidée par la visibilité

L'utilisation du squelette de visibilité pour la simulation de l'éclairage par la méthode de radiosité hiérarchique permet de gagner sur le double tableau du temps de calcul et de la précision. Les méthodes antérieures utilisaient un échantillonnage pour le calcul de la visibilité qui s'avérait imprécis et coûteux, alors que le squelette permet des calculs efficaces et exacts des coefficients de transfert lumineux, les facteurs de forme.

Il permet aussi la subdivision du maillage utilisé pour représenter l'éclairement le long des limites d'ombre et de pénombre, et ce en considérant n'importe quel polygone comme source de lumière là où les méthodes antérieures se limitaient à un faible nombre de sources primaires. En particulier, nous traitons les discontinuités de la fonction d'éclairement dues aux limites de visibilité de l'éclairage indirect. Grâce à cette technique et à l'introduction d'ondelettes paresseuses définies sur des triangulations hiérarchiques, nous obtenons une représentation de l'éclairage de haute qualité, directement utilisable pour des balades virtuelles.

Nous utilisons de plus des critères de raffinement basés sur l'information de visibilité fournie par le squelette et sur une métrique perceptuelle. Cela nous permet de raffiner un calcul uniquement lorsque la précision supplémentaire ainsi obtenue est potentiellement visible pour un observateur humain. Nous évitons le réglage fastidieux de seuils de raffinements arbitraires et imprévisibles qui constituaient jusqu'alors un frein majeur à l'utilisabilité des simulations hiérarchiques.

Les tests que nous avons conduits ont démontré l'efficacité de notre approche, y compris pour des scènes traditionnellement difficiles présentant de nombreuses sources de lumière ou un éclairage principalement indirect. Les comparaisons avec un des algorithmes les plus récents ont montré un net avantage pour notre méthode, que ce soit en terme de qualité ou de temps de calcul.

Calculs généraux d'occultations à l'aide de projections étendues

Notre précalcul d'occultation est le premier à permettre de calculer la visibilité depuis une cellule volumique en prenant en compte l'occultation due à de multiples bloqueurs. Les tests de visibilité sont effectués dans des plans, où nos opérateurs de *projection étendue* sous-estiment la projection d'un bloqueur par rapport à n'importe quel point de vue de la cellule, tandis qu'ils surestiment celle d'un objet. La technique est prudente (aucun objet visible n'est déclaré invisible), simple à mettre en œuvre et efficace grâce à l'utilisation des cartes graphiques spécialisées.

Nous avons de plus développé un *balayage d'occultation* qui permet de traiter des configurations particulièrement difficiles, comme la visibilité depuis une cellule volumique dans une forêt. Lors du balayage par des plans s'éloignant de la cellule, les occultations dues aux feuilles des arbres sont accumulées grâce à un opérateur de reprojection.

La validité de ces méthodes a donné lieu à la démonstration de propriétés originales des ombres, en particulier sur les conclusions que l'on peut tirer de l'inclusion de l'ombre d'un objet dans celle d'un autre.

Un tour d'horizon multidisciplinaire de la visibilité

Dans la seconde partie du mémoire, nous avons effectué un tour d'horizon des techniques de visibilité dans différents domaines. Nous avons proposé une classification selon l'espace dans lequel sont effectués les calculs. Cela nous a permis de passer en revue un grand nombre de méthodes sans nous en remettre à un catalogue par domaine. Les approches ont pu être abordées et comparées avec un regard synthétique.

Cette partie peut être utile à toute personne qui débute des travaux sur la visibilité, tout comme à ceux qui souhaitent trouver dans d'autres communautés inspiration et solutions. Elle peut aussi servir à qui souhaite retrouver rapidement une référence à une technique. Bien qu'elle n'ait pas été organisée dans ce but, nous espérons qu'elle pourra aussi répondre aux attentes de ceux qui cherchent des solutions à des problèmes concrets.

Travaux futurs

Les extensions directes de nos travaux ont été proposées au fur et à mesure du mémoire, aussi invitons-nous le lecteur à s'y référer. Nous développons dans ce qui suit des thèmes plus généraux de recherche future.

Le complexe de visibilité, comme le squelette de visibilité, ne présentent malheureusement pas un bon passage à l'échelle, contrairement aux approches approximatives. Oublions la complexité théorique en $O(n^4)$ qui ne reflète en rien le comportement pratique. Cependant, le coût quadratique prédit par notre étude probabiliste et observé dans nos tests n'est pas acceptable pour une utilisation pratique sur la totalité d'une grosse scène. L'utilité d'une telle information est d'ailleurs discutable. Est-il utile de savoir que l'arête d'une tuile et le creux entre deux briques d'une autre maison séparées de plusieurs kilomètres engendrent un événement visuel ? La quantité d'information est non seulement coûteuse à calculer et à stocker, elle est de plus tout à fait inexploitable.

Des approches multi-échelles doivent être explorées. Un premier pas dans cette direction est la simplification de l'information du squelette ou du complexe de visibilité. Deux obstacles principaux doivent être franchis. L'essence globale des propriétés de visibilité rend difficile une simplification locale dans l'espace des droites, puisque l'influence spatiale peut être infinie. On peut attaquer le problème en étudiant la simplification de l'information de visibilité pour une utilisation en dehors d'un groupe d'objets.

Le second obstacle est la définition même d'une information approximative ou simplifiée de la visibilité. Des pas dans ce sens ont été franchi pour le calcul de maillage de discontinuité [SK98] ainsi que pour les graphes d'aspect [EBD+93, SP97]. Ces dernières approches, où la taille d'un segment visible permet de définir l'échelle, nous semblent particulièrement intéressantes, même si leur mise en œuvre n'a pour l'heure abouti qu'à une complexification de l'information de visibilité.

Les problèmes de robustesse demandent également un traitement efficace qui, comme nous l'avons vu, peut aller de pair avec le passage à l'échelle grâce à la définition de seuils pertinents. Nous répétons que nous pensons que les configurations dégénérées ne doivent pas être ramenées à des situations générales à l'aide de calculs exacts. Nous voulons éviter le calcul d'événements visuels infinitésimaux susceptibles de créer des problèmes en cascade. Les scènes rencontrées en pratiques présentent énormément de dégénérescence ; celles-ci doivent êtres exploitées et non évitées.

Une autre voie pour s'attaquer à la fois au passage à l'échelle et aux problèmes de robustesse consiste à utiliser des calculs purement approximatifs, en particulier à l'aide d'approches discrètes [SK98] ou volumiques [SD95]. Les points difficiles sont le contrôle de l'approximation et la possibilité de raffiner un calcul [SS96b, Sol98].

Nous l'avons déjà souligné, les bornes théoriques ne donnent que très peu d'indications sur la complexité effective pour des scènes "normales". Des approches probabilistes et un modèle "raisonnable" de scènes (dans l'esprit de de Berg *et al.* [dBKvdSV97] ou de notre étude sur les événements triple-tangentes) devraient permettre une meilleure compréhension de la complexité pratique des propriétés de visibilité, ainsi que du comportement des algorithmes. Une telle approche doit s'accompagner de validations statistiques expérimentales. Cette question ne se pose pas seulement dans le cadre de la visibilité, elle est pertinente pour l'analyse de tout calcul géométrique.

Pour finir, le traitement des scènes en mouvement reste le parent pauvre des travaux en visibilité. Les événements visuels temporels sont un pas vers la compréhension des phénomènes entraînés. Des algorithmes utilisables et efficace doivent néanmoins être développés pour éviter la redondance des calculs, qui ramène à

l'heure actuelle les scènes dynamiques à une succession de scènes statiques pour lesquelles quasiment tout est recalculé.

Pour conclure

Cette thèse a abordé les questions de visibilité sous le double point de vue de la théorie et de la pratique. Elle s'expose sans doute à la critique de théoriciens qui la trouveront trop appliquée et pas suffisamment formelle, tandis que des praticiens réprouveront les développements abstraits et multidimensionnels.

Nous espérons cependant avoir montré que des problèmes tout à fait pratiques soulèvent des questions théoriques difficiles et fascinantes. La visibilité tridimensionnelle offre un cadre captivant tant d'un point de vue algorithmique que géométrique. Au delà de la simple visibilité, la prise en compte des propriétés et des spécificités des scènes et des applications "normales" demande une analyse appropriée qui offre un champ théorique très intéressant.

Nous espérons également avoir montré qu'une approche analytique peut avoir des apports concrets sur la résolution efficace de problèmes pratiques. Notre travail analytique nous a permis de développer des solutions efficaces et plutôt plus simples que les méthodes antérieures, bien que nous nous soyons au départ appuyés sur un cadre multidimensionnel complexe. Nous sommes persuadés que la compréhension acquise grâce à une étude théorique peut permettre de prendre du recul et d'aborder des questions sous un angle original.

ANNEXE A

Le complexe de visibilité 3D

1 Catalogue of adjacencies

Table A.1 presents the adjacencies between all types of faces of the 3D Visibility Complex and their higher dimensional faces. Recall that two faces can have the same generators but different extremities.

We explain here how the number of adjacent 4-faces is obtained. Consider a face with k generators (lying on objects O_i, $i \in \{1 \dots k\}$) and with extremities (O_0, O_{k+1}). Its adjacent 4 faces are the faces with extremities $(O_i O_j)$ with $i > j$. The number of adjacent 4-faces with right extremity O_i is i. The total number of adjacent 4-faces is thus :

$$\sum_{i=1}^{k+1} i = \frac{(k+1)(k+2)}{2}$$

2 Concave objects

Critical line set and visual events for concave smooth objects are described by the singularities of smooth mappings as introduced in chapter 8. The catalogues have been established by Rieger [Rie87, Rie90] and Kergosien [Ker81]. We give here a short overview inspired by [Pet92, Ker81, Rie90]. We refer the interested reader to these references where he will find a more comprehensive presentation.

Local events are related to the differential properties of the surface, in particular to the *order* of its tangent vectors (intuitively, the codimension of a segment depends on its "contact" with the surface).

Consider a surface F defined by the implicit equation $F(x) = 0$ (such a representation always exist), and a point x on the surface. Denote $\vec{t}$ a vector. $\vec{t}$ has order n at x if the $(n-1)$ first derivatives of F are null in direction $\vec{t}$. Tangent vectors are for example of order 2.

The points of a generic smooth surface can be separated into the following 8 classes according to the order of their tangents (see also Fig. A.1 and Fig. A.2) :

1. In the **elliptic domain** all tangents have order 2 ;
2. In the **hyperbolic domain** points where each point has two tangents of order higher than 2 (called the asymptotic tangents) ;

k	k-face	(k+1)-faces	(k+2)-faces	(k+3)-faces	(k+4)-faces
3	T	3 4-faces			
2	$T_1 + T_2$	2 T_1 2 T_2	6 4-faces		
	V	at least 2 T^i	3 4-faces		
1	$T_1 + T_2 + T_3$	2 $T_1 + T_2$ 1 $T_1 + T_3$ 2 $T_2 + T_3$	3 T_1 4 T_2 3 T_3	10-faces	
	$T_1 + +T_2$	1 $T_1 + T_2$	2 T_1 2 T_2	6 4-faces	
	$T_1 + V_2$	2 V_2 2 $T_1 + T_2^i$	2 T_1 2 T_2^i per edge	6 4-faces	
0	$T_1 + T_2 + T_3 + T_4$	2 $T_1 + T_2 + T_3$ 1 $T_1 + T_2 + T_4$ 1 $T_1 + T_3 + T_4$ 2 $T_2 + T_3 + T_4$ 2 $T_2 + T_3 + T_4$ 2 $T_2 + T_3 + T_4$	3 $T_1 + T_2$ 4 $T_2 + T_3$ 3 $T_3 + T_4$ 2 $T_1 + T_3$ 2 $T_2 + T_4$ 1 $T_1 + T_4$	4 T_1 6 T_2 6 T_3 4 T_4	15 4-faces
	$T_1 + +T_2 + T_3$	2 $T_1 + +T_2$ 1 $T_1 + T_2 + T_3$	2 $T_1 + T_2$ 2 $T_2 + T_3$	3 T_1 4 T_2 3 T_3	10 4-faces
	$T_1 + V_2 + T_3$	2 $T_1 + V_2$ 2 $T_3 + V_2$ 2 $T_1 + T_2^i + T_3$	2 $T_1 + T_2^i$ per edge 1 $T_1 + T_3$ 2 $T_2^i + T_3$ per edge 4 V_2	3 T_1 4 T_2^i per edge 3 T_3	10 4-faces
	$T_1 + T_2 + V_3$	1 $T_1 + V_3$ 2 $T_2 + V_3$ 1 $T_1 + T_2 + T_3^i$ per edge	2 $T_1 + T_2$ 1 $T_1 + T_3^i$ per edge 2 $T_2 + T_3^i$ per edge	3 T_1 4 T_2 3 T_3^i per edge	10 4-faces
	$V_1 + V_2$	1 $T_1^i + V_2$ per edge 1 $T_2^j + V_1$ per edge	1 $T_1^i + T_2^j$ per edge pair	2 T_1^i per edge 2 T_2^j per edge	6 4-faces

TAB. A.1: Adjacencies of the faces of the visibility complex of polygons and smooth objects. A vertex V_k is adjacent to edges T_k^i.

3. The **curve of parabolic points** separates the two previous domains. Each point has one tangent of order higher than 2 (called the asymptotic tangent) ;
4. The **flecnodal curve** lies inside the hyperbolic domain and corresponds to the inflection of one asymptotic tangent, which has order at least 4 ;
5. The **biflecnodes** have a tangent of order 5.
6. The points of **inflexion of both asymptotic lines** which correspond to the self-crossing of the flecnodal curve : both asymptotic tangents have order 4.
7. The **godrons** are the points of tangency of the flecnodal curve and the parabolic curve.
8. The **gutterpoints** are stationary points of the asymptotic tangents on the parabolic curve.

Cusp singularities occur for segments colinear to an asymptotic tangent in the hyperbolic domain (Fig. A.1). They have codimension 2, they thus generate 2-faces of the visibility complex.

Codimension 3 singularities correspond to the appearance or disappearance of a pair of cusps. They are thus adjacent to two cusp 2-faces of the complex. **Swallowtail** singularities occur for segments colinear to the asymptotic tangent at a point of the flecnodal curve. **lip** and **beak-to-beak** occur on the parabolic curve.

Codimension 4 singularities are more involved. They are also adjacent to multilocal events such as **tangent crossings** or **cusp crossings** (where a segment of a cusp is also tangent to an object). **Gulls** occur at godrons, **geese** at gutterpoints, and **butterflies** at biflecnodes. The adjacencies of the corresponding 0-faces of the visibility complex with 1-faces are represented in Fig. A.2.

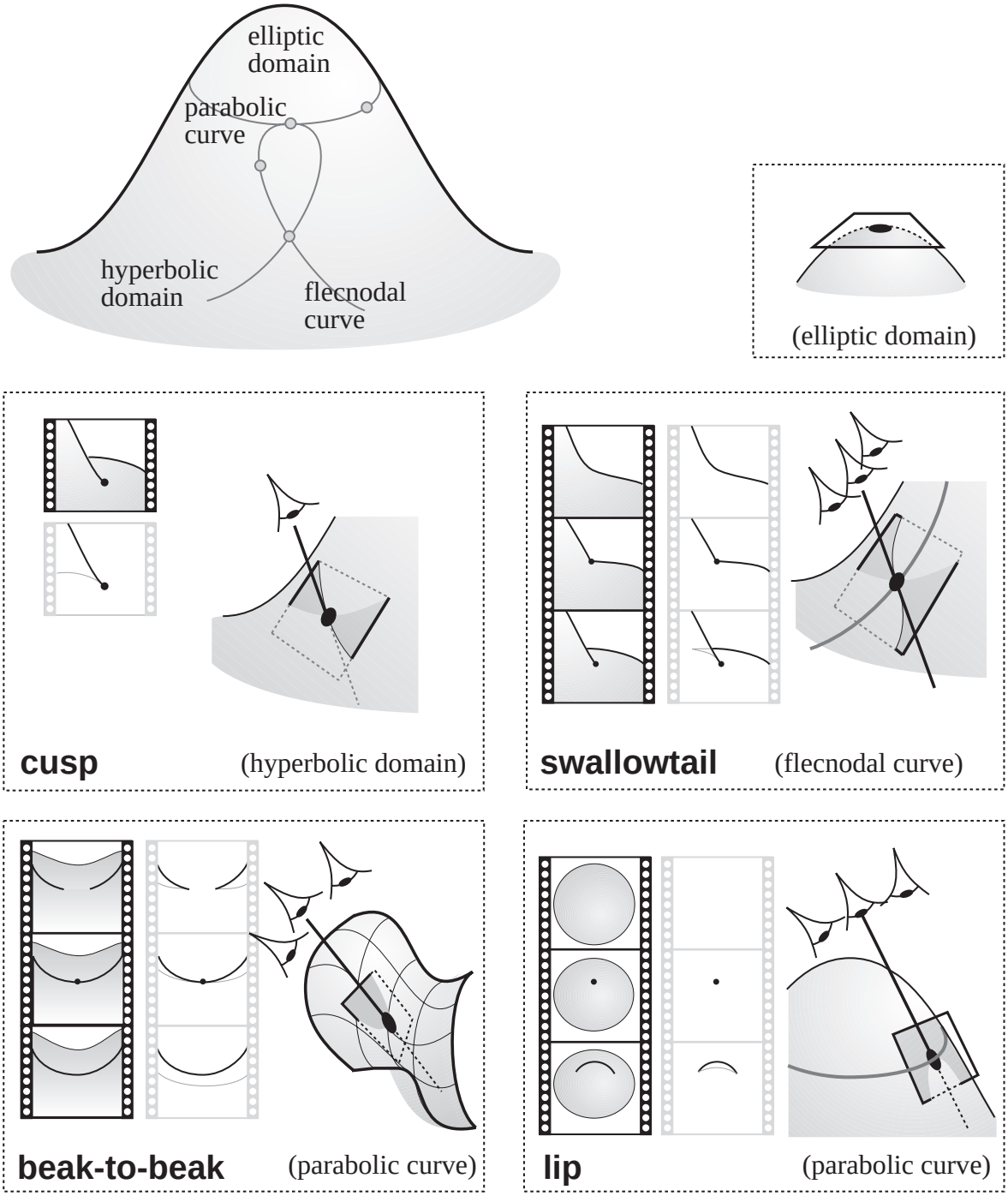


FIG. A.1: Singularities of a general smooth surface (adapted from [Pet92]). We represent the views both for segment-visibility (black frames) and line visibility (grey-frames, hidden-lines are in light-grey).

3 Discontinuity meshing for curved objects

A direct application of the concepts developed for the visibility complex is the implementation of the computation of the limits of umbra and penumbra for smooth curved objects illuminated by an area polygonal light source. Fig. A.3 illustrates the case of a single sphere and a triangle source.

The limits of umbra and penumbra are :

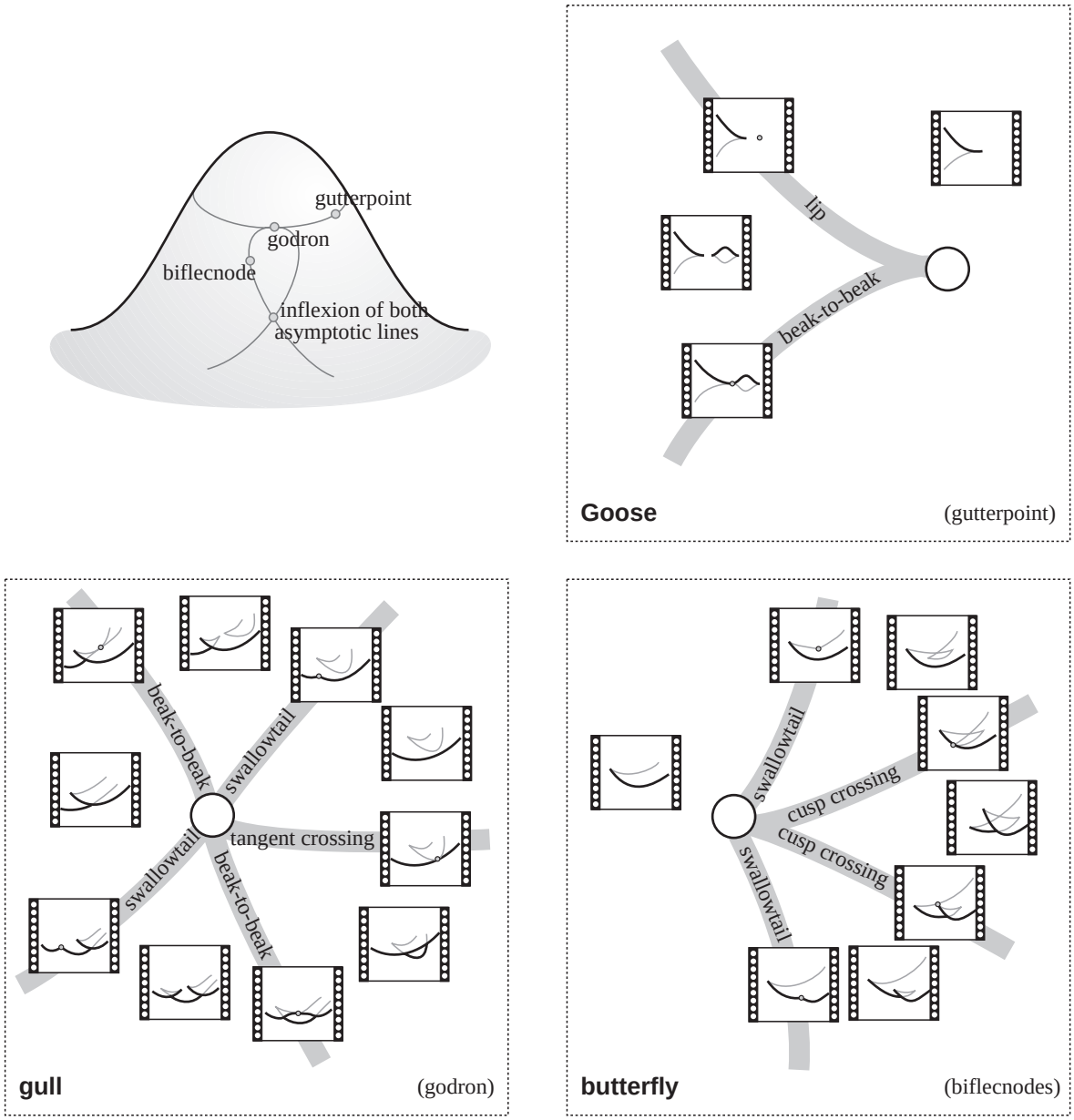


FIG. A.2: Singularities of concave surfaces. Codimension 4 singularities (adapted from [Pet92]).

– the $T+V$ events, which correspond to the views of the object from the vertices of the source (Fig. A.3(a)) ;

– the $T++T$ events corresponding to planes going through an edge of the scheme

These kind of events can both be computed easily. If multiple objects are considered, these limits have to be intersected with the other objects, in a way similar to discontinuity meshing for polygonal scenes [DF94]. However, if the limits of umbra are sought, complex $T+T+T$ events should be handled.

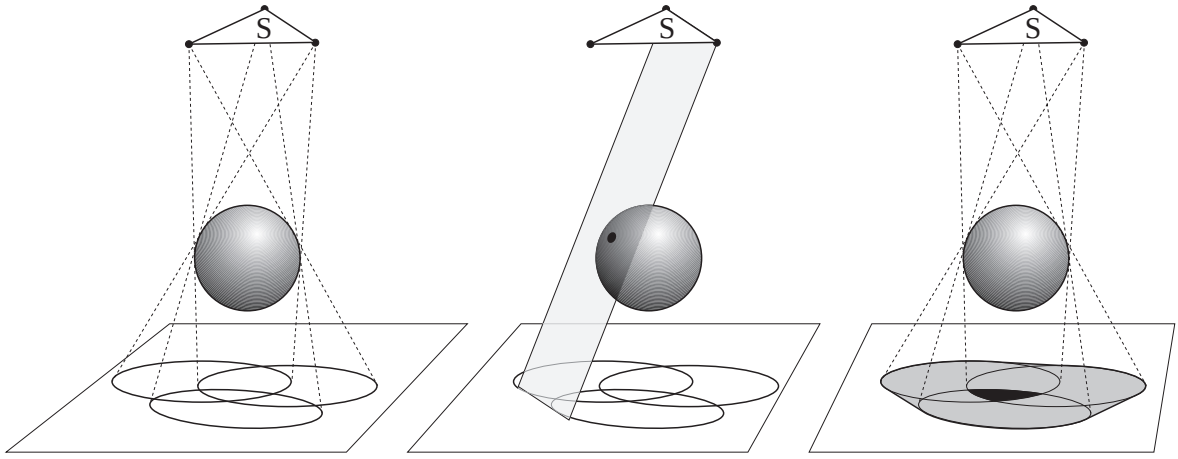


FIG. A.3: Discontinuity mesh of a curved object. (a) $T+V$ events. (b) A $T++T$ event. (c) Complete discontinuity mesh.

ANNEXE B

Le squelette de visibilité

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. B.1, the interaction of a face a vertex and an edge is shown in Fig. B.3 and finally the interaction of two faces is shown in Fig. B.2.

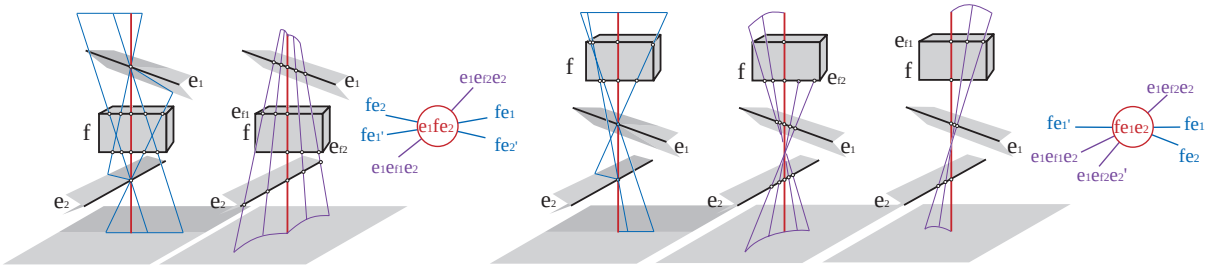


FIG. B.1: An EFE node.

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. B.4, and the adjacent arc $e_1e_2e_3$. The question that needs to be answered is whether the 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.

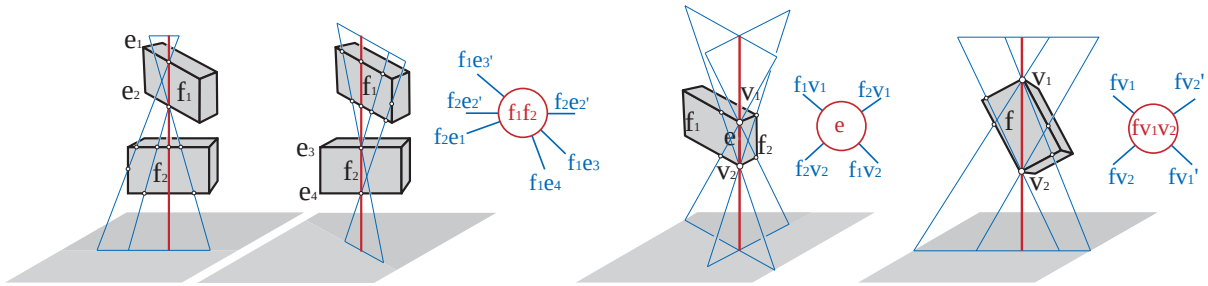


FIG. B.2: (a) A FF node, (b) an Fe node and (c) and Fvv node.

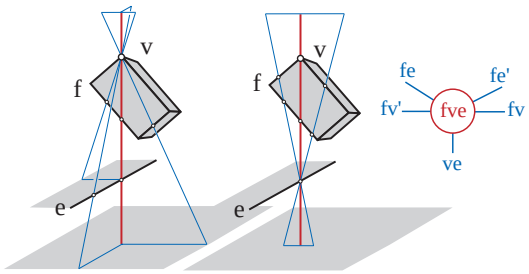


FIG. B.3: A FvE node.

Consider the infinitesimal motion $d\vec{\varepsilon_1}$ 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 $d\vec{\varepsilon_1}$ on e_1 corresponds to a rotation of $\alpha = \frac{\vec{\varepsilon_1}.\vec{n}}{d_1}$ of the plane around e_2. Symmetrically, this rotation corresponds to the motion $d\vec{\varepsilon_3}$ on e_3 and we have $\alpha = \frac{d\vec{\varepsilon_3}.\vec{n}}{d_3}$, by angle equality. Thus, $d\vec{\varepsilon_3} = \vec{e_3}\frac{d_3 d\vec{\varepsilon_1}.\vec{n}}{d_1 \vec{e_3}.\vec{n}}$.

Now we want to obtain $d\vec{\varepsilon_4}$, 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 $d\vec{\varepsilon_1}$ of the origin, the direction vector of moves by $d\vec{\varepsilon_3} - d\vec{\varepsilon_1}$, and thus $d\vec{\varepsilon_4} = d\vec{\varepsilon_1} + \frac{d_4}{d_3 - d_1}(d\vec{\varepsilon_3} - \vec{e_1})$.

The sign of $(\vec{\varepsilon_4} \times \vec{e_4}).\vec{node}$ 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 (Some errors present in the Siggraph version [DDP97c] for the criteria for the $e_1 v_e 2$ node were corrected).

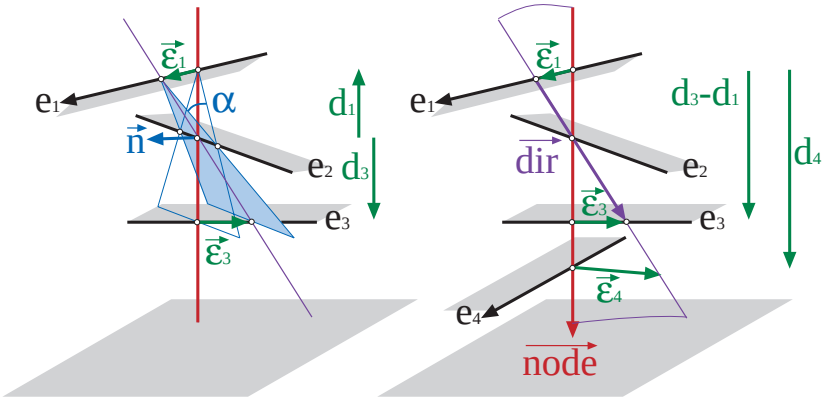


FIG. B.4: Determining the direction of an E4 node insertion.

Node	**Adjacent arc**	**Start or End Criterion**
v_1v_2	v_1e_3	$v_1 == startV(e)$
ve_1e_2	ve_2 $e_3e_1e_2$ $e_2e_1e_3$	$(\vec{e_2} \times \vec{e_1}).\vec{node} > 0$ $v == startV(e_3)$ $\vec{n} = \vec{normal}(v, \vec{e_2})$ $\vec{e_3}.\vec{n} * \vec{e_1}.\vec{n} > 0$
e_1ve_2	ve_2 $e_1e_3e_2$	$(\vec{e_1} \times \vec{e_2}).\vec{node} > 0$ $\vec{n} = \vec{normal}(v, \vec{e_2})$ $\vec{e_3}.\vec{n} * \vec{e_1}.\vec{n} > 0$
$e_1e_2e_3e_4$	$e_1e_2e_3$	$\vec{n} = \vec{normal}(\vec{e_2}, \vec{node})$; $\vec{\varepsilon_3} = \vec{e_3}\frac{d_3\vec{e_1}.\vec{n}}{d_1\vec{e_3}.\vec{n}}$ $\vec{\varepsilon_4} = \vec{e_1} + \frac{d_4}{d_3-d_1}(\vec{\varepsilon_3} - \vec{e_1})$ $(\vec{\varepsilon_4} \times \vec{e_4}).\vec{node} > 0$
e_1fe_2	fe_1 $e_1e_{f1}e_2$	$\vec{e_2}.\vec{normal}(f) > 0$ $\vec{e_1}.\vec{normal}(f) > 0$
fe_1e_2	fe_2 $e_2e_1e_{f1}$	$\vec{e_1}.\vec{normal}(f) > 0$ $\vec{n} = \vec{normal}(\vec{node}, \vec{e_1})$ $\vec{n}.\vec{e_2} * \vec{n}.\vec{e_{f1}} > 0$
fve	fv ve	$\vec{e}.\vec{normal}(f) > 0$ $\vec{e}.\vec{normal}(f) > 0$

TAB. B.1: For each arc adjacent to a created node, there is a criterion that tells if it is a start node or an ending node.

ANNEXE C

Projections étendues efficaces avec Open GL

Recall that in Section 2.2 of chapter 5, we described how to project convex occluders onto a projection plane. The general ideas were presented there : the Projection is the intersection of the views from the vertices of the viewing cell. Here we present the details of an efficient OpenGL implementation.

One of the problems is that during the projection of convex occluders we need to write consistent z-values and also treat the case of multiple blockers. An efficient way to do this in OpenGL is to use the stencil buffer, and a slightly involved z-buffer.

Recall that depth is defined orthogonal to the projection plane, and that the *Depth* of a point in the Projection of an occluder is the maximum of the depth of the corresponding projected points. To efficiently handle multiple occluders, the *minimum* of their Depths must be used.

For a perspective projection, depth is considered from the viewpoint. Mapping the z value to our definition of depth requires an addition to set the zero on the projection plane. Unfortunatetely, *OpenGL* stores $\frac{1}{z}$ in the z-buffer, preventing a simple addition.

For a given occluder and a given cell, we project (in software) the blocker onto the projection plane, including the calculation of z values. The resulting 2D polygons are then rendered orthographically using a stencil buffer. Z-testing is performed with respect to z-values potentially written by a previously projected blocker, but depth values are not written. The stencil buffer is incremented by one. After all the polygons corresponding to each cell vertex have been rendered, the umbra region is defined by the region of the stencil buffer with the value 8 (i.e., blocked with respect to all cell vertices).

The eight 2D polygons are rendered again, using the stencil buffer to restrict writing to the umbra region only. The first polygon is rendered and z-values are written to the z-buffer. The 7 other polygons are then rendered but the z-test is inverted. This results in the *maximum* z-value being written to the z-buffer.

The result of these operations is the creation of the required umbra zone, including appropriately consistent z-values.

This process is summarized in the pseudo-code of Figure C.1.

```
ProjectBlocker ( occluder A, cell C, projection plane P )
  for each vertex v_i of C
    project A onto P in software
    create 2D polygons p_i, i = 1..8
  endfor
  enable stencil buffer, increment by one
  // do z-test in case previous blocker
  // mapped to same pixels
  enable z-test
  disable z-write
  for each 2D polygon p_i, i = 1..8
    render p_i orthographically
  endfor
  // initialize for max calculation
  enable z-write
  render polygon p_1
  // use inverted z for max calculation
  enable invert z-mode
  for each 2D polygon p_i, i = 2..8
    render polygon p_i
  endfor
```

FIG. C.1: Efficient OpenGL implementation of blocker projection.

ANNEXE D

Some Notions in Line Space

Plücker coordinates

Consider a *directed* line ℓ in 3D defined by two points $P(x_P, y_P, z_P)$ and $Q(x_Q, y_Q, z_Q)$. The *Plücker coordinates* [Plü65] of ℓ are :

$$\begin{pmatrix} \pi_{\ell 0} \\ \pi_{\ell 1} \\ \pi_{\ell 2} \\ \pi_{\ell 3} \\ \pi_{\ell 4} \\ \pi_{\ell 5} \end{pmatrix} = \begin{pmatrix} x_P y_Q - y_P x_Q \\ x_P z_Q - z_P x_Q \\ x_P - x_Q \\ y_P z_Q - z_P y_Q \\ z_P - z_Q \\ y_Q - y_P \end{pmatrix}$$

(The signs and order may vary with the authors). These coordinates are homogenous, any choice of P and Q will give the same Plücker coordinates up to a scaling factor (Plücker space is thus a 5D projective space).

The dot product between two lines a and b with Plücker duals Π_a and Π_b is defined by

$$\Pi_a \odot \Pi_b = \pi_{a0}\pi_{b4} + \pi_{a1}\pi_{b5} + \pi_{a2}\pi_{b3} + \pi_{a4}\pi_{b0} + \pi_{a5}\pi_{b1} + \pi_{a3}\pi_{b2}$$

The sign of the dot products indicates the relative orientation of the two lines. If the dot product is null, the two lines intersect. The equation $\Pi_a \odot \Pi_\ell = 0$ defines the hyperplane associated with a.

The *Plücker hypersurface* or *Grassman manifold* or *Klein quadric* is defined by

$$\Pi_\ell \odot \Pi_\ell = 0$$

ANNEXE E

Online Ressources

1 General ressources

An index of computer graphics web pages can be found at
http ://www-imagis.imag.fr/˜Fredo.Durand/book.html

A lot of computer vision ressources are listed at
http ://www.cs.cmu.edu/ cil/vision.html
A commented and sorted vision bibliography :
http ://iris.usc.edu/Vision-Notes/bibliography/contents.html
An excellent Compendium of Computer Vision :
http ://www.dai.ed.ac.uk/CVonline/

For robotics related pages, see
http ://www-robotics.cs.umass.edu/robotics.html
http ://www.robohoo.com/

Many sites are dedicated to computational geometry, *e.g.* :
http ://www.ics.uci.edu/˜eppstein/geom.html
http ://compgeom.cs.uiuc.edu/˜jeffe/compgeom/

Those interested in human and animal vision will find several links at :
http ://www.visionscience.com/.

An introduction to perception is provided under the form of an excellent web book at :
http ://www.yorku.ca/eye/

2 Available code.

CGAL is a robust and flexible computational geometry librairy
http ://www.cs.ruu.nl/CGAL

Nina Amenta maintains some links to geometrical softwares :
http ://www.geom.umn.edu/software/cglist/welcome.html

The implementation of Luebke and George's online portal occlusion-culling technique is available at :
http ://www.cs.virginia.edu/~luebke/visibility.html

Electronic articles on shadows, portals, etc. :
http ://www.flipcode.com/features.htm

Information on Open GL, including shadow computation :
http ://reality.sgi.com/opengl/

Visibility graph programs can be found at :
http ://www.cs.uleth.ca/~wismath/vis.html
http ://cs.smith.edu/~halef/research.html
http ://willkuere.informatik.uni-wuerzburg.de/ lupinho/java.html

Many ray-tracer are available *e.g.* :
http ://www.povray.org/
http ://www-graphics.stanford.edu/- cek/rayshade/rayshade.html
http ://www.rz.tu-ilmenau.de/~juhu/GX/intro.html (with different acceleration schemes, including ray-classification)

A radiosity implementation :
http ://www.ledalite.com/software/software.htm

RenderPark provides many global illumination methods, such as radiosity or Monte-Carlo path-tracing :
http ://www.cs.kuleuven.ac.be/cwis/research-/graphics/RENDERPARK/

Aspect graphs :
http ://www.dai.ed.ac.uk/staff/-personal_pages/eggertd/software.html

BSP trees :
http ://www.cs.utexas.edu/users/atc/

A list of info and links about BSP :
http ://www.ce.unipr.it/ marchini/jaluit.html

Mel Slater's shadow volume BSP :
ftp ://ftp.dcs.qmw.ac.uk/people/mel/BSP/

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

INDEX

D

E

F

G

H

I

K

L

M

N

O

P

T

U

V

W

Z

BIBLIOGRAPHIE

[AA95] Steven Abrams and Peter K. Allen. Swept volumes and their use in viewpoint computation in robot work-cells. In *Proceedings IEEE International Symposium on Assembly and Task Planning*, August 1995. http ://www.cs.columbia.edu/˜abrams/.

[AAA+92] A. L. Abbott, N. Ahuja, P.K. Allen, J. Aloimonos, M. Asada, R. Bajcsy, D.H. Ballard, B. Bederson, R. M. Bolle, P. Bonasso, C. M. Brown, P. J. Burt, D. Chapman, J. J. Clark, J. H. Connell, P. R. Cooper, J. D. Crisman, J. L. Crowley, M. J. Daily, J.O. Eklundh, F. P. Firby, M. Herman, P. Kahn, E. Krotkov, N. da Vitoria Lobo, H. Moraff, R. C. Nelson, H. K. Nishihara, T. J. Olson, D. Raviv, G. Sandini, and E. L. Schwartz. Promising directions in active vision. *International Journal on Computer Vision*, 1992.

[AB91] E. H. Adelson and J. R. Bergen. The plenoptic function and the elements of early vision. *M. Landy and J. A. Movshon, (eds) Computational Models of Visual Processing*, 1991.

[Abr96] Michael Abrash. Inside quake : Visible-surface determination. *Dr. Dobb's Journal of Software Tools*, 21(??) :41–??, January/February 1996.

[ACW+99] D. Aliaga, J. Cohen, A. Wilson, Eric Baker, H. Zhang, C. Erikson, K. Hoff, T. Hudson, W. Stuerzlinger, R. Bastos, M. Whitton, F. Brooks, and D. Manocha. Mmr : An interactive massive model rendering system using geometric and image-based acceleration. In *1999 Symposium on Interactive 3D Graphics*. ACM SIGGRAPH, April 1999. http ://www.cs.unc.edu :80/˜walk/research/index.html.

[AEG98] Pankaj K. Agarwal, Jeff Erickson, and Leonidas J. Guibas. Kinetic binary space partitions for intersecting segments and disjoint triangles. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA-98)*, pages 107–116, New York, January 25–27 1998. ACM Press.

[Aga84] P. K. Agarwal. *The art gallery problem : Its Variations, applications, and algorithmic aspects*. PhD thesis, Johns Hopkins University, Baltimore, 1984.

[AGMV97] P. Agarwal, L. Guibas, T. Murali, and J. Vitter. Cylindrical static and kinetic binary space partitions. In *Proceedings of the 13th International Annual Symposium on Computational Geometry (SCG-97)*, pages 39–48, New York, June 4–6 1997. ACM Press.

[AH97] Laurent Alonso and Nicolas Holzschuch. Using graphics hardware to speed up your visibility queries. Technical Report 99-R-030, LORIA, March 1997. http ://www.loria.fr.

[Air90] John M. Airey. *Increasing Update Rates in the Building Walkthrough System with Automatic Model-Space Subdivision and Potentially Visible Set Calculations*. PhD thesis, Dept. of CS, U. of North Carolina, July 1990. TR90-027.

[AK87] James Arvo and David Kirk. Fast ray tracing by ray classification. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21-4, pages 55–64, July 1987.

[AK89] James Arvo and David Kirk. A survey of ray tracing acceleration techniques. In Andrew S. Glassner, editor, *An introduction to ray tracing*, pages 201–262. Academic Press, 1989.

[Ama84] John Amanatides. Ray tracing with cones. *Computer Graphics*, 18(3) :129–135, July 1984.

[App67] Arthur Appel. The notion of quantitative invisibility and the machine rendering of solids. *Proc. ACM Natl. Mtg.*, page 387, 1967.

[App68] Arthur Appel. Some techniques for shading machine renderings of solids. In *AFIPS 1968 Spring Joint Computer Conf.*, volume 32, pages 37–45, 1968.

[ARB90] John M. Airey, John H. Rohlf, and Frederick P. Brooks, Jr. Towards image realism with interactive update rates in complex virtual building environments. In Rich Riesenfeld and Carlo Sequin, editors, *Computer Graphics (1990 Symposium on Interactive 3D Graphics)*, volume 24-2, pages 41–50, March 1990.

[Arn69] V.I. Arnold. Singularities of smooth mappings. *Russian Mathematical Surveys*, pages 3–44, 1969.

[Arv94] James Arvo. The irradiance Jacobian for partially occluded polyhedral sources. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 343–350. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.

[Ase92] Frederic Asensio. A hierarchical ray-casting algorithm for radiosity shadows. *Third Eurographics Workshop on Rendering*, pages 179–188, May 1992.

[Ash94] Ian Ashdown. *Radiosity : A Programmer's Perspective*. John Wiley & Sons, New York, NY, 1994.

[ATS94] 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, 1994 (ACM SIGGRAPH '94 Proceedings)*, pages 75–84, 1994.

[AW87] John Amanatides and Andrew Woo. A fast voxel traversal algorithm for ray tracing. In G. Marechal, editor, *Eurographics '87*, pages 3–10. North-Holland, August 1987.

[AWG78] P. Atherton, K. Weiler, and D. Greenberg. Polygon shadow generation. In *Computer Graphics (SIGGRAPH '78 Proceedings)*, volume 12(3), pages 275–281, August 1978.

[Bax95] Michael Baxandall. *Shadows and enlightenment*. Yale University Press, London, UK, 1995. (Ombres et Lumières, Gallimard).

[BB82] D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice Hall, Englewood Cliffs, 2 edition, 1982.

[BB84] Lynne Shapiro Brotman and Norman I. Badler. Generating soft shadows with a depth buffer algorithm. *IEEE Computer Graphics and Applications*, 4(10) :5–12, October 1984.

[BD98] Amy J. Briggs and Bruce R. Donald. Visibility-based planning of sensor control strategies. *Algorithmica*, 1998. accepted for publication, http ://www.middlebury.edu/˜briggs/Research/alg.html.

[BDEG90] Marshall Bern, David Dobkin, David Eppstein, and Robert Grossman. Visibility with a moving point of view. In David Johnson, editor, *Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '90)*, pages 107–117, San Francisco, CA, USA, January 1990. SIAM.

[Ber86] P. Bergeron. A general version of Crow's shadow volumes. *IEEE Computer Graphics and Applications*, 6(9) :17–28, 1986.

[Ber93] M. De Berg. Ray shooting, depth orders and hidden surface removal. In *Lecture Notes in Computer Science*, volume 703. Springer Verlag, New York, 1993.

[BEW+98] Lars Bishop, Dave Eberly, Turner Whitted, Mark Finch, and Michael Shantz. Designing a PC game engine. *IEEE Computer Graphics and Applications*, 18(1) :46–53, January/February 1998.

[BF89] Chris Buckalew and Donald Fussell. Illumination networks : Fast realistic rendering with general reflectance functions. In Jeffrey Lane, editor, *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23(3), pages 89–98, July 1989.

[BGRT95] P. Bose, F. Gomez, P. Ramos, and G. Toussaint. Drawing nice projections of objects in space. *Lecture Notes in Computer Science*, 1027 :52– ? ?, 1995.

[BHS98] J. Bittner, V. Havran, and P. Slavik. Hierarchical visibility culling with occlusion trees. In IEEE Computer Society Press, editor, *Proceedings of Computer Graphics International '98 (CGI'98)*, pages 327–335, June 1998.

[BK70] W. J. Bouknight and K. C. Kelly. An algorithm for producing half-tone computer graphics presentations with shadows and movable light sources. In *Proc. AFIPS JSCC*, volume 36, pages 1–10, 1970.

[Bli78] J. F. Blinn. A scan line algorithm for displaying parametrically defines surfaces. *Computer Graphics (Special SIGGRAPH '78 Issue, preliminary papers)*, pages 1–7, August 1978.

[BM98] Mark R. Bolin and Gary W. Meyer. A perceptually based adaptive sampling algorithm. In Michael F. Cohen, editor, *Computer graphics : proceedings : SIGGRAPH 98 Conference proceedings, July 19–24, 1998*, Computer Graphics -proceedings- 1998, pages 299–310, New York, NY 10036, USA and Reading, MA, USA, 1998. ACM Press and Addison-Wesley.

[BMT98] D. Bartz, M. Meissner, and T.Huettner. Extending graphics hardware for occlusion queries in opengl. In *Eurographics/Siggraph Workshop on Graphics Hardware Lisbon, Portugal August 31 and September 1*, 1998.

[BNN+98] Ph. Bekaert, L. Neumann, A. Neumann, M. Sbert, and Y.D. Willems. Hierarchical monte carlo radiosity. In George Drettakis and Nelson Max, editors, *Eurographics Rendering Workshop 1998*, New York City, NY, June 1998. Eurographics, Springer Wein.

[Bou70] W. Jack Bouknight. A procedure for generation of three-dimensional half-toned computer graphics presentations. *Communications of the ACM*, 13(9) :527–536, September 1970.

[Bou85] C. Bouville. Bounding ellipsoids for ray-fractal intersection. In B. A. Barsky, editor, *Computer Graphics (SIGGRAPH '85 Proceedings)*, volume 19(3), pages 45–52, July 1985.

[BP95] Kadi Bouatouch and Sumanta N. Pattanaik. Discontinuity Meshing and Hierarchical Multiwavelet Radiosity. In W. A. Davis and P. Prusinkiewicz, editors, *Proceedings of Graphics Interface '95*, pages 109–115, San Francisco, CA, May 1995. Morgan Kaufmann.

[BP96] Normand Brière and Pierre Poulin. Hierarchical view-dependent structures for interactive scene manipulation. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 83–90. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[Bre92] M. Quinn Brewster. *Thermal Radiative Transfer & Properties*. John Wiley & Sons, New York, 1992.

[BRW89] Daniel R. Baum, Holly E. Rushmeier, and James M. Winget. Improving radiosity solutions through the use of analytically determined form-factors. In Jeffrey Lane, editor, *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23(3), pages 325–334, July 1989.

[BS96] Gonzalo Besuievsky and Mateu Sbert. The multi-frame lighting method : A monte carlo based solution for radiosity in dynamic environments. In Xavier Pueyo and Peter Schröder, editors, *Eurographics Rendering Workshop 1996*, pages 185–194, New York City, NY, June 1996. Eurographics, Springer Wien. ISBN 3-211-82883-4.

[BWCG86] 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(5) :298–306, September 1986.

[BY98] Jean-Daniel Boissonnat and Mariette Yvinec. *Algorithmic geometry*. Cambridge University Press, 1998.

[BZW+95] J.E. Banta, Y. Zhien, X.Z. Wang, G. Zhang, M.T. Smith, and M.A. Abidi. A "best-next-view" algorithm for three dimensional scene reconstruction using range images. In *Computer Graphics (SIGGRAPH '87 Proceedings)SPI Intelligent Robots and Computer Vision XIV. Algorithms, Techniques, Active Vision and Material Handling, Philadelphia, PA*, 1995.

[Cam91] A. T. Campbell, III. *Modeling Global Diffuse Illumination for Image Synthesis*. PhD thesis, CS Dept, University of Texas at Austin, December 1991. Tech. Report TR-91-39, http ://www.cs.utexas.edu/users/atc/.

[Can88] John F. Canny. *The Complexity of Robot Motion Planning*. MIT Press, Cambridge, Massachusetts, 1988.

[Car84] Loren Carpenter. The A-buffer, an antialiased hidden surface method. In Hank Christiansen, editor, *Computer Graphics (SIGGRAPH '84 Proceedings)*, volume 18, pages 103–108, July 1984.

[Cat74] Edwin E. Catmull. *A Subdivision Algorithm for Computer Display of Curved Surfaces*. Ph.d. thesis, University of Utah, December 1974.

[Cat75] Edwin E. Catmull. Computer display of curved surfaces. In *Proceedings of the IEEE Conference on Computer Graphics, Pattern Recognition, and Data Structure*, pages 11–17, May 1975.

[Cat78] E. Catmull. A hidden-surface algorithm with anti-aliasing. In *Computer Graphics (SIGGRAPH '78 Proceedings)*, volume 12(3), pages 6–11, August 1978.

[CCC87] Robert L. Cook, Loren Carpenter, and Edwin Catmull. The reyes image rendering architecture. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21(4), pages 95–102, July 1987.

[CCD91] J. Chapman, T. W. Calvert, and J. Dill. Spatio-temporal coherence in ray tracing. In *Proceedings of Graphics Interface '91*, pages 101–108, June 1991.

[CCOL98] Yiorgos Chrysanthou, Daniel Cohen-Or, and Dani Lischinski. Fast approximate quantitative visibility for complex scenes. In *Proceedings of Computer Graphics International*, June 1998. http ://www.math.tau.ac.il/˜daniel/.

[CDL+96] Bradford Chamberlain, Tony DeRose, Dani Lischinski, David Salesin, and John Snyder. Fast rendering of complex environments using a spatial hierarchy. In Wayne A. Davis and Richard Bartels, editors, *Graphics Interface '96*, pages 132–141. Canadian Information Processing Society, Canadian Human-Computer Communications Society, May 1996. ISBN 0-9695338-5-3.

[CDP95] Frédéric Cazals, George Drettakis, and Claude Puech. Filtering, clustering and hierarchy construction : a new solution for ray-tracing complex scenes. *Computer Graphics Forum*, 14(3) :371–382, August 1995. Proceedings of Eurographics '95. ISSN 1067-7055.

[CEG+92] B. Chazelle, H. Edelsbrunner, L. J. Guibas, R. Pollack, R. Seidel, M. Sharir, and J. Snoeyink. Counting and cutting cycles of lines and rods in space. *Comput. Geom. Theory Appl.*, 1 :305–323, 1992.

[CEG+96] B. Chazelle, H. Edelsbrunner, L. J. Guibas, M. Sharir, and J. Stolfi. Lines in space : Combinatorics and algorithms. *Algorithmica*, 15(5) :428–447, May 1996.

[CF89] Norman Chin and Steven Feiner. Near real-time shadow generation using BSP trees. In Jeffrey Lane, editor, *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23(3), pages 99–106, July 1989.

[CF90] A. T. Campbell, III and Donald S. Fussell. Adaptive Mesh Generation for Global Diffuse Illumination. In *Computer Graphics (ACM SIGGRAPH '90 Proceedings)*, volume 24-4, pages 155–164, August 1990. http ://www.cs.utexas.edu/users/atc/.

[CF91a] A. T. Campbell III and Donald S. Fussell. An analytic approach to illumination with area light sources. Technical Report CS-TR-91-25, University of Texas, Austin, August 1, 1991. http ://www.cs.utexas.edu/users/atc/.

[CF91b] S. Chen and H. Freeman. On the characteristic views of quadric-surfaced solids. In *IEEE Workshop on Directions in Automated CAD-Based Vision*, pages 34–43, June 1991.

[CF92] Norman Chin and Steven Feiner. Fast object-precision shadow generation for area light source using BSP trees. In Marc Levoy and Edwin E. Catmull, editors, *Proceedings of the 1992 Symposium on Interactive 3D Graphics*, pages 21–30, Cambridge, MA, March–April 1992. ACM Press.

[CF97] Stephen Chenney and David Forsyth. View-dependent culling of dynamic systems in virtual environments. In *Proceedings of the Symposium on Interactive 3D Graphics*, pages 55–58, New York, April 27–30 1997. ACM Press.

[CG85] M. F. Cohen and D. P. Greenberg. The hemicube : A radiosity solution for complex environments. In *Proceedings of SIGGRAPH '85*, volume 19(3), pages 31–40, San Francisco, CA, 22–26 July 1985. ACM SIGGRAPH.

[CH97] Deborah A. Carlson and Jessica K. Hodgins. Simulation levels of detail for real-time animation. In *Graphics Interface*, pages 1–8, May 1997.

[Cha96] B. Chazelle. The computational geometry impact task force report : An executive summary. *Lecture Notes in Computer Science*, 1148 :59– ? ?, 1996. http ://www.cs.princeton.edu/˜chazelle/.

[Chv75] V. Chvátal. A combinatorial theorem in plane geometry. *J. Combin. Theory Ser. B*, 18 :39–41, 1975.

[CK88] Cregg K. Cowan and Peter D. Kovesi. Automatic sensor placement from vision task requirements. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-10(3) :407–416, May 1988.

[CL97] D. Crevier and R. Lepage. Knowledge-based image understanding systems : A survey. *Computer Vision and Image Understanding : CVIU*, 67(2) :161– ? ?, ? ? ? ? 1997.

[Cla76] James H. Clark. Hierarchical geometric models for visible surface algorithms. *Communications of the ACM*, 19(10) :547–554, October 1976.

[Cla79] James H. Clark. A fast scan-line algorithm for rendering parametric surfaces. *Computer Graphics, Special SIGGRAPH '79 Issue*, 13(2) :7–11, August 1979.

[Cla87] K. L. Clarkson. New applications of random sampling to computational geometry. *Discrete and Computational Geometry*, 2 :195–222, 1987.

[CLF98] Emilio Camahort, Apostolos Lerios, and Don Fussell. Uniformly sampled light fields. In George Drettakis and Nelson Max, editors, *Eurographics Rendering Workshop 1998*, New York City, NY, June 1998. Eurographics, Springer Wein.

[CLR80] Elaine Cohen, Tom Lyche, and Richard F. Riesenfeld. Discrete B-spline subdivision techniques in computer-aided geometric design and computer graphics. *Computer Graphics and Image Processing*, 14 :87–111, October 1980.

[CLSS97] Per H. 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. ISSN 0730-0301.

[COFHZ98] Daniel Cohen-Or, Gadi Fibich, Dan Halperin, and Eyal Zadicario. Conservative visibility and strong occlusion for visibility partitionning of densely occluded scenes. In *Proceedings of Eurographics*, September 1998. http ://www.math.tau.ac.il/˜daniel/.

[Col75] George Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Proceedings Second GI Conference on Automata Theory and Formal Languages*, volume 33 of *Lecture Notes in Computer Science*, pages 134–183, Berlin, 1975. Springer-Verlag.

[COZ98] Daniel Cohen-Or and Eyal Zadicario. Visibility streaming for network-based walkthroughs. In *Proceedings of Graphics Interface*, June 1998. http ://www.math.tau.ac.il/˜daniel/.

[CP97] 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, June 4–6 1997. ACM Press.

[Cro77] Franklin C. Crow. Shadow algorithms for computer graphics. In James George, editor, *Computer Graphics (SIGGRAPH '77 Proceedings)*, volume 11(2), pages 242–248, July 1977.

[Cro84] Gary A. Crocker. Invisibility coherence for faster scan-line hidden surface algorithms. In Hank Christiansen, editor, *Computer Graphics (SIGGRAPH '84 Proceedings)*, volume 18, pages 95–102, July 1984.

[CS92] Y. Chrysanthou and M. Slater. Computing dynamic changes to BSP trees. In A. Kilgour and L. Kjelldahl, editors, *Computer Graphics Forum (EUROGRAPHICS '92 Proceedings)*, volume 11-3, pages 321–332, September 1992.

[CS94] D. Cohen and Z. Sheffer. Proximity clouds : An acceleration technique for 3D grid traversal. *The Visual Computer*, 11 ? :27–38, 1994 ?

[CS95] Yiorgos Chrysanthou and Mel Slater. Shadow volume BSP trees for computation of shadows in dynamic scenes. In Pat Hanrahan and Jim Winget, editors, *1995 Symposium on Interactive 3D Graphics*, pages 45–50. ACM SIGGRAPH, April 1995. ISBN 0-89791-736-7.

[CS97] Yiorgos Chrysanthou and Mel Slater. Incremental updates to scenes illuminated by area light sources. In Julie Dorsey and Philipp Slusallek, editors, *Eurographics Rendering Workshop 1997*, pages 103–114, New York City, NY, June 1997. Eurographics, Springer Wein. ISBN 3-211-83001-4.

[CSSD96] Per H. Christensen, Eric J. Stollnitz, David H. Salesin, and Tony D. DeRose. Global illumination of glossy environments using wavelets and importance. *ACM Transactions on Graphics*, 15(1) :37–71, January 1996. ISSN 0730-0301.

[CT96] Satyan Coorg and Seth Teller. Temporally coherent conservative visibility. In *Proceedings of the Twelfth Annual Symposium On Computational Geometry (ISG '96)*, pages 78–87, New York, May 1996. ACM Press. http ://graphics.lcs.mit.edu/˜satyan/pubs.html.

[CT97a] Michael Capps and Seth Teller. Communications visibility in shared virtual worlds. In *Sixth Workshops on Enabling Technologies : Infrastructure for Collaborative Enterprises*, pages 187–192. ACM Press, June18–2- 1997. http ://graphics.lcs.mit.edu/˜capps/.

[CT97b] Satyan Coorg and Seth Teller. Real-time occlusion culling for models with large occluders (color plate S. 189). In *Proceedings of the Symposium on Interactive 3D Graphics*, pages 83–90, New York, April27–30 1997. ACM Press. http ://graphics.lcs.mit.edu/˜satyan/pubs.html.

[CW85] J. Callahan and R. Weiss. A model for describing surface shape. In *Proceedings, CVPR '85 (IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, June 10–13, 1985)*, IEEE Publ. 85CH2145-1., pages 240–245. IEEE, IEEE, 1985.

[CW93a] 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.

[CW93b] Michael F. Cohen and John R. Wallace. *Radiosity and Realistic Image Synthesis*. Academic Press Professional, Boston, MA, 1993.

[CW96] H.-M. Chen and W.-T. Wang. The feudal priority algorithm on hidden-surface removal. *Computer Graphics*, 30(Annual Conference Series) :55–64, 1996.

[Dal96] Bengt-Inge L. Dalenbäck. Room acoustic predictiom based on a unified treatment of diffuse and specular reflection. *Journal of the Acoustical Society of America*, 100(2) :899–909, August 1996.

[dBKvdSV97] M. de Berg, M. Katz, F. van der Stappen, and J. Vleugels. Realistic input models for geometric algorithms. In *Proceedings of the 13th International Annual Symposium on Computational Geometry (SCG-97)*, pages 294–303, New York, June 4–6 1997. ACM Press.

[dBvKOS97] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry : Algorithms and Applications.* Springer-Verlag, 1997.

[DDP96] Frédo Durand, George Drettakis, and Claude Puech. The 3D visibility complex : A new approach to the problems of accurate visibility. In Xavier Pueyo and Peter Schröder, editors, *Eurographics Rendering Workshop 1996*, pages 245–256, New York City, NY, June 1996. Eurographics, Springer Wein. ISBN 3-211-82883-4, http ://www-imagis.imag.fr/Publications/.

[DDP97a] F. Durand, G. Drettakis, and C. Puech. 3d visibility made visibly simple. In *video 13th Annu. ACM Sympos. Comput. Geom.*, 1997.

[DDP97b] 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 *Canadian Conference on Computational Geometry*, August 1997. http ://www-imagis.imag.fr/~Fredo.Durand.

[DDP97c] Fredo Durand, George Drettakis, and Claude Puech. The visibility skeleton : a powerful and efficient multi-purpose global visibility tool. *Computer Graphics*, 31(3A) :89–100, August 1997. http ://www-imagis.imag.fr/Publications/.

[DDP99] Frédo Durand, George Drettakis, and Claude Puech. Fast and accurate hierarchical radiosity using global visibility. *ACM Transactions on Graphics*, April 1999. to appear. http ://www-imagis.imag.fr/~Fredo.Durand.

[DF93] G. Drettakis and E. Fiume. Accurate and consistent reconstruction of illumination functions using structured sampling. *Computer Graphics Forum (Eurographics '93)*, 13(3) :273–284, September 1993.

[DF94] G. Drettakis and E. Fiume. A fast shadow algorithm for area light sources using backprojection. *Computer Graphics*, 28(Annual Conference Series) :223–230, July 1994. http ://www-imagis.imag.fr/~George.Drettakis/pub.html.

[DKW85] Nou Dadoun, David G. Kirkpatrick, and John P. Walsh. The geometry of beam tracing. In *Proceedings of the ACM Symposium on Computational Geometry*, pages 55–61, June 1985.

[Dor94] Susan Dorward. A survey of object-space hidden surface removal. *International Journal of Computational Geometry and Applications*, 4(3) :325–362, 1994.

[DORP96] Frédo Durand, Rachel Orti, Stéphane Rivière, and Claude Puech. Radiosity in flatland made visibly simple. In *video 12th Annu. ACM Sympos. Comput. Geom.*, 1996.

[DP95a] L. De Floriani and E. Puppo. Hierarchical triangulation for multiresolution surface description geometric design. *ACM Transactions on Graphics*, 14(4) :363–411, October 1995. ISSN 0730-0301.

[DP95b] Frédo Durand and Claude Puech. The visibility complex made visibly simple. In *video 11th Annu. ACM Sympos. Comput. Geom.*, 1995.

[DPR92] S. J. Dickinson, A. P. Pentland, and A. Rosenfeld. 3-D shape recovery using distributed aspect matching. *IEEE Transactions on Pattern Analysis and machine Intelligence*, 14(2) :174–198, February 1992.

[Dru87] M. Drumheller. Mobile robot localization using sonar. *IEEE Trans. Pattern Analysis and Machine Intelligence*, PAMI-9(2) :325–332, 1987. See also : S. B. Thesis, Dept. of Mechanical Engineering, MIT, 1984 and MIT AI Lab Memo BZ6, Mobile Robot Localization Osing Sonar.

[DS96] George Drettakis and François Sillion. Accurate visibility and meshing calculations for hierarchical radiosity. In Xavier Pueyo and Peter Schröder, editors, *Eurographics Rendering Workshop 1996*, pages 269–278, New York City, NY, June 1996. Eurographics, Springer Wein. ISBN 3-211-82883-4, http ://www-imagis.imag.fr/~George.Drettakis/pub.html.

[DS97] George Drettakis and François Sillion. Interactive update of global illumination using A line-space hierarchy. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 57–64. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7, http ://www-imagis.imag.fr/~George.Drettakis/pub.html.

[DSSD99] Xavier Decoret, Gernot Schaufler, Francois Sillion, and Julie Dorsey. Improved image-based impostors foraccelerated rendering. In *Eurographics'99*, August 1999.

[DTM96] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs : A hybrid geometry- and image-based approach. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 11–20. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[Dub57] L. E. Dubins. On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents. *Amer. J. Math.*, 79 :497–516, 1957.

[Dür38] Albrecht Dürer. *Underweysung der Messung mit dem Zirckel und richtscheyd*. Nuremberg, 1538.

[Dur95] Frédo Durand. étude du complexe de visibilité. Master's thesis, Institut National Polytechnique de Grenoble, June 1995.

[DW85] Mark A. Z. Dippé and Erling Henry Wold. Antialiasing through stochastic sampling. In B. A. Barsky, editor, *Computer Graphics (SIGGRAPH '85 Proceedings)*, volume 19, pages 69–78, July 1985.

[DZ95] Steven M. Drucker and David Zeltzer. CamDroid : A system for implementing intelligent camera control. In Pat Hanrahan and Jim Winget, editors, *1995 Symposium on Interactive 3D Graphics*, pages 139–144. ACM SIGGRAPH, April 1995. ISBN 0-89791-736-7.

[EB90] David W. Eggert and Kevin W. Bowyer. Computing the orthographic projection aspect graph of solids of revolution. *Pattern Recognition Letters*, 11(11) :751–763, November 1990.

[EB92] Shimon Edelman and Heinrich H. Bulthoff. Orientation dependence in the recognition of familiar and novel views of 3D objects. *Vision Research*, 32 :2385–2400, 1992. http ://eris.wisdom.weizmann.ac.il/-˜edelman/abstracts.html#visres.

[EB93] D.W. Eggert and K.W. Bowyer. Computing the generalized aspect graph for objects with moving parts. *PAMI*, 15(6) :605–610, June 1993.

[EBD92] David Eggert, Kevin Bowyer, and Chuck Dyer. Aspect graphs : State-of-the-art and applications in digital photogrammetry. In *Proceedings of the 17th Congress of the International Society for Photogrammetry and Remote Sensing, Part B5*, pages 633–645, 1992.

[EBD+93] David W. Eggert, Kevin W. Bowyer, Charles R. Dyer, Henrik I. Christensen, and Dmitry B. Goldgof. The scale space aspect graph. *Pattern Analysis and Machine Intelligence*, 15(11) :1114–1130, November 1993.

[EC90] Gershon Elber and Elaine Cohen. Hidden curve removal for free form surfaces. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24-4, pages 95–104, August 1990.

[Ede87] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer Verlag, 1987.

[EG86] Herbert Edelsbrunner and Leonidas J. Guibas. Topologically sweeping an arrangement. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, pages 389–403, Berkeley, California, 28–30 May 1986.

[Egg91] D.W. Eggert. Aspect graphs of solids of revolution. In *Ph. D.*, 1991.

[EHW97] P. Eades, M. E. Houle, and R. Webber. Finding the best viewpoints for three-dimensional graph drawings. *Lecture Notes in Computer Science*, 1353 :87–? ?, 1997.

[EK89] K. S. Eo and C. M. Kyung. Hybrid shadow testing scheme for ray tracing. *Computer-Aided Design*, 21(1) :38–48, January/February 1989.

[ES94] R. Endl and M. Sommer. Classification of ray-generators in uniform subdivisions and octrees for ray tracing. *Computer Graphics Forum*, 13(1) :3–19, March 1994.

[ESB95] D.W. Eggert, L. Stark, and K.W. Bowyer. Aspect graphs and their use in object recognition. *Annals of Mathematics and Artificial Intelligence*, 13(3-4) :347–375, 1995.

[Eve90] H. Everett. *Visibility Computations in Densely Occluded Polyhedral EnvironmentsVisibility Graph Recognition*. PhD thesis, Department of. Computer Science, University of Toronto, 1990. Tech. Report 231/90.

[FAG83] H. Fuchs, G. D. Abram, and E. D. Grant. Near real-time shaded display of rigid objects. In *Computer Graphics (SIGGRAPH '83 Proceedings)*, volume 17(3), pages 65–72, July 1983.

[Fau93] O. Faugeras. *Three-dimensional computer vision.* MIT Press, Cambridge, MA, 1993.

[FCE+98] Thomas Funkhouser, Ingrid Carlbom, Gary Elko, Gopal Pingali, Mohan Sondhi, and Jim West. A beam tracing approach to acoustic modeling for interactive virtual environments. In Michael F. Cohen, editor, *Computer graphics : proceedings : SIGGRAPH 98 Conference proceedings, July 19–24, 1998*, Computer Graphics -proceedings- 1998, pages 21–32, New York, NY 10036, USA and Reading, MA, USA, 1998. ACM Press and Addison-Wesley.

[FD84] G. Fekete and L. S. Davis. Property spheres : a new representation for 3-D object recognition. In *Proceedings of the Workshop on Computer Vision : Representation and Control (Annapolis, MD, April 30-May 2, 1984)*, IEEE Publ. 84CH2014-9., pages 192–201. IEEE, IEEE, 1984.

[FF88] Alain Fournier and Donald Fussell. On the power of the frame buffer. *ACM Transactions on Graphics*, 7(2) :103–128, 1988.

[FFR83] E. Fiume, A. Fournier, and L. Rudolph. A parallel scan conversion algorithm with anti-aliasing for a general purpose ultracomputer. In *Computer Graphics (SIGGRAPH '83 Proceedings)*, volume 17(3), pages 141–150, July 1983.

[FGH+85] Henry Fuchs, Jack Goldfeather, Jeff P. Hultquist, Susan Spach, John D. Austin, Frederick P. Brooks, Jr., John G. Eyles, and John Poulton. Fast spheres, shadows, textures, transparencies, and image enhancements in Pixel-Planes. In B. A. Barsky, editor, *Computer Graphics (SIGGRAPH '85 Proceedings)*, volume 19, pages 111–120, July 1985.

[FGK+96] A. Fabri, G.-J. Giezeman, L. Kettner, S. Schirra, and S. Schoenherr. The cgal kernel : A basis for geometric computation. in proceedings workshop on applied computational geometry. In *roceedings of the Workshop on Applied Computational Geometry, May 27-28, 1996, Philadelphia, Pennsylvania.*, 1996. http ://www.cs.ruu.nl/CGAL.

[Fis78] S. Fisk. Short proof of chvatal's watchman theorem. *Journal of Combinatorial Theory*, 24 :374, 1978.

[FKN80] H. Fuchs, Z. M. Kedem, and B. F. Naylor. On visible surface generation by a priori tree structures. In *Computer Graphics (SIGGRAPH '80 Proceedings)*, volume 14(3), pages 124–133, July 1980.

[FL98] Patrick Fabiani and Jean-Claude Latombe. Tracking a partially predictable object with uncertainty and visibility constraints : a game-theoretic approach. Technical report, Univeristy of Stanford, December 1998. http ://underdog.stanford.edu/.

[FMA+92] Olivier Faugeras, Joe Mundy, Narendra Ahuja, Charles Dyer, Alex Pentland, Ramesh Jain, and Katsushi Ikeuchi. Why aspect graphs are not (yet) practical for computer vision. *Computer Vision and Image Understanding : CVIU*, 55(2) :322–335, March 1992.

[FMC99] Thomas A. Funkhouser, Patrick Min, and Ingrid Carlbom. Real-time acoustic modeling for distributed virtual environments. In *Computer Graphics (SIGGRAPH '99 Proceedings)*, August 1999.

[FPM98] L. De Floriani, E. Puppo, and P. Magillo. Geometric structures and algorithms for geographical information systems. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook on Computational Geometry*. Elsevier Science, 1998. Preliminary version available as : Technical Report DISI-TR-97-08, Department of Computer and Information Sciences, University of Genova, http ://www.disi.unige.it/person/DeflorianiL/.

[FPSG96] James A. Ferwerda, Sumant Pattanaik, Peter Shirley, and Donald P. Greenberg. A model of visual adaptation for realistic image synthesis. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 249–258. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[FPSG97] James A. Ferwerda, Sumanta N. Pattanaik, Peter Shirley, and Donald P. Greenberg. A model of visual masking for computer graphics. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 143–152. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.

[FS93] Thomas A. Funkhouser and Carlo H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 247–254, August 1993. http ://www.bell-labs.com/user/funk/.

[FST92] Thomas A. Funkhouser, Carlo H. Sequin, and Seth J. Teller. Management of large amounts of data in interactive building walkthroughs. In David Zeltzer, editor, *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, volume 25-2, pages 11–20, March 1992. http ://www.bell-labs.com/user/funk/.

[FTI86] Akira Fujimoto, Takayuki Tanaka, and Kansei Iwata. ARTS : Accelerated ray tracing system. *IEEE Computer Graphics and Applications*, 6(4) :16–26, 1986.

[Fun95] T. Funkhouser. RING - A client-server system for multi-user virtual environments. *SIGGRAPH Symposium on Interactive 3D Graphics*, pages 85–92, 1995.

[Fun96a] T. Funkhouser. Network topologies for scalable multi-user virtual environments. *Proceedings of VRAIS'96, Santa Clara CA*, pages 222–229, 1996.

[Fun96b] Thomas A. Funkhouser. Coarse-grained parallelism for hierarchical radiosity using group iterative methods. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 343–352. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[Fun96c] Thomas A. Funkhouser. Database management for interactive display of large architectural models. In Wayne A. Davis and Richard Bartels, editors, *Graphics Interface '96*, pages 1–8. Canadian Information Processing Society, Canadian Human-Computer Communications Society, May 1996. ISBN 0-9695338-5-3, http ://www.bell-labs.com/user/funk.

[FvDFH90] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics : Principles and Practice*. Addison-Wesley Publishing Co., Reading, MA, 2nd edition, 1990. T 385.C587.

[GBW90] B. Garlick, D. Baum, and J. Winget. Interactive viewing of large geometric data bases using multiprocessor graphics workstations. In *Parallel Algorithms and Architectures for 3D Image Generation*, pages 239–245. ACM SIGGRAPH, 1990. Siggraph '90 Course Notes, Vol. 28.

[GCS91] 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.

[GGC97] X. Gu, S. J. Gortler, and M. Cohen. Polyhedral geometry and the two-plane parameterization. In Julie Dorsey and Philipp Slusallek, editors, *Eurographics Rendering Workshop 1997*, New York City, NY, June 1997. Eurographics, Springer Wein. ISBN 3-211-83001-4, http ://hillbilly.deas.harvard.edu/~sjg/.

[GGH+99] X. Gu, S.J. Gortler, H. Hoppe, L. Mcmillan, B. Brown, and A. Stone. Silhouette mapping. Technical Report TR-1-99, Harvard Computer Science, March 1999. http ://cs.harvard.edu/~xgu/paper/Silhouette_Map/.

[GGSC96] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 43–54. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996, http ://hillbilly.deas.harvard.edu/~sjg/.

[GH94] Neil Gatenby and W. T. Hewitt. Optimizing Discontinuity Meshing Radiosity. In *Fifth Eurographics Workshop on Rendering*, pages 249–258, Darmstadt, Germany, June 1994.

[GH96] S. Gibson and R. J. Hubbold. Efficient hierarchical refinement and clustering for radiosity in complex environments. *Computer Graphics Forum*, 15(5) :297–310, 1996. ISSN 0167-7055.

[GH97a] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 209–216. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7, http ://www.cs.cmu.edu/~garland/multires/my-work.html.

[GH97b] S. Gibson and R. J. Hubbold. Perceptually-driven radiosity. *Computer Graphics Forum*, 16(2) :129–141, 1997. ISSN 0167-7055.

[GH98] Djamchid Ghazanfarpour and Jean-Marc Hasenfratz. A beam tracing with precise antialiasing for polyhedral scenes. *Computer & Graphics*, 22(1) :103–115, 1998.

[Gho97] Ghosh. On recognizing and characterizing visibility graphs of simple polygons. *GEOMETRY : Discrete & Computational Geometry*, 17, 1997.

[GI87] Keith D. Gremban and Katsushi Ikeuchi. Planning multiple observation for object recognition. *International Journal of Computer Vision*, 1(2) :145–65, 1987.

[GKM93] Ned Greene, Michael Kass, and G.avin Miller. Hierarchical Z-buffer visibility. In *Computer Graphics Proceedings, Annual Conference Series, 1993*, pages 231–240, 1993.

[GL99] Héctor González-Baños and Jean-Claude Latombe. Planning robot motions for range-image acquisition and automatic 3d model construction. In *AAAI Spring Symposium*, 1999.

[Gla84] Andrew S. Glassner. Space sub-division for fast ray tracing. *IEEE Computer Graphics and Applications, October 1984*, 4 :15–22, 1984.

[Gla88] Andrew S. Glassner. Spacetime ray tracing for animation. *IEEE Computer Graphics and Applications*, 8(2) :60–70, March 1988.

[Gla89] Andrew Glassner. *An introduction to raytracing*. Academic Press, Reading, MA, 1989.

[Gla95] Andrew S. Glassner. *Principles of Digital Image Synthesis*. Morgan Kaufmann, San Francisco, CA, 1995.

[GLL+97] Leonidas J. Guibas, Jean-Claude Latombe, Steven M. LaValle, David Lin, and Rajeev Motwani. Visibility-based pursuit-evasion in a polygonal environment. In *Algorithms and Data Structures, 5th International Workshop*, Lecture Notes in Computer Science, Halifax, Nova Scotia, Canada, 6–8 August 1997. Springer. http ://underdog.stanford.edu/.

[GLLL98] L. J. Guibas, J.-C. Latombe, S. M. LaValle, and D. Lin. Visibility-based pursuit-evasion in a polygonal environment. *Lecture Notes in Computer Science*, 1272 :17– ? ?, 1998.

[GM90] 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.

[GM91] Subir Kumar Ghosh and David M. Mount. An output-sensitive algorithm for computing visibility graphs. *SIAM Journal on Computing*, 20(5) :888–910, October 1991.

[GMR95] Guibas, Motwani, and Raghavan. The robot localization problem. In Goldberg, Halperin, Latombe, and Wilson, editors, *Algorithmic Foundations of Robotics, The 1994 Workshop on the Algorithmic Foundations of Robotics, A. K. Peters*, 1995.

[GN71] Robert A. Goldstein and Roger Nagel. 3-D visual simulation. *Simulation*, 16(1) :25–31, January 1971.

[Goa83] Chris Goad. Special purpose automatic programming for 3D model-based vision. In Martin A Fischler Oscar Firschein, editor, *Readings in Computer Vision*, pages 371–381. Morgan Kaufman Publishers, August 1983.

[GP91] E. Gröller and W. Purgathofer. Using temporal and spatial coherence for accelerating the calculation of animation sequences. In Werner Purgathofer, editor, *Eurographics '91*, pages 103–113. North-Holland, September 1991.

[Gra92] Charles W. Grant. *Visibility Algorithms in Image Synthesis*. PhD thesis, U. of California, Davis, 1992. http ://www.hooked.net/˜grant/.

[Gre96] Ned Greene. Hierarchical polygon tiling with coverage masks. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 65–74. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[GS85] Leonidas Guibas and Jorge Stolfi. Primitives for the manipulation of general subdivisions and computation of voronoi diagrams. *ACM Transactions on Graphics*, 4(2) :74–123, April 1985.

[GS96] Sherif Ghali and A. James Stewart. A complete treatment of D1 discontinuities in a discontinuity mesh. In Wayne A. Davis and Richard Bartels, editors, *Graphics Interface '96*, pages 122–131. Canadian Information Processing Society, Canadian Human-Computer Communications Society, May 1996. ISBN 0-9695338-5-3, http ://www.dgp.toronto.edu/people/ghali/papers/.

[GSCH93] Steven J. Gortler, Peter Schroder, Michael F. Cohen, and Pat Hanrahan. Wavelet radiosity. In *Computer Graphics Proceedings, Annual Conference Series, 1993*, pages 221–230, 1993.

[GSHG98] Gene Greger, Peter Shirley, Philip M. Hubbard, and Donald P. Greenberg. The irradiance volume. *IEEE Computer Graphics & Applications*, 18(2) :32–43, 1998.

[Gue98] Concettina Guerra. Vision and image processing algorithms. In M. Atallah, editor, *CRC Handbook of Algorithms and Theory of Computation*. CRC Press, 1998.

[HA90] Paul Haeberli and Kurt Akeley. The accumulation buffer : Hardware support for high-quality rendering. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24(4), pages 309–318, August 1990.

[Hae90] Paul Haeberli. Paint by numbers : Abstract image representations. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24(4), pages 207–214, August 1990.

[Hai] Eric Haines. Ray tracing news. http ://www.povray.org/rtn/.

[Hai91] Eric A. Haines. Beams O' Light : Confessions of a Hacker. In *SIGGRAPH '91 Course Notes - Frontiers in Rendering*. ACM, July 1991.

[Hal98] Michael Halle. Multiple viewpoint rendering. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 243–254. ACM SIGGRAPH, Addison Wesley, July 1998. ISBN 0-89791-999-8.

[Han89] Pat Hanrahan. A survey of ray-surface intersection algorithms. In Andrew S. Glassner, editor, *An introduction to ray tracing*, pages 79–119. Academic Press, 1989.

[Has98] Jean-Marc Hasenfratz. *Lancer de Faisceaux en Synthèse d'Image*. PhD thesis, Université de Limoges, 1998.

[HCS96] Li-wei He, Michael F. Cohen, and David H. Salesin. The virtual cinematographer : A paradigm for automatic real-time camera control and directing. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 217–224. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[HDG99] David Hart, Philip Dutré, and Donald P. Greenberg. Direct illumination with lazy visibility evaluation. In *Computer Graphics (SIGGRAPH '99 Proceedings)*, August 1999.

[Hec87] Paul S. Heckbert. Ten unsolved problems in rendering. In *Workshop on Rendering Algorithms and Systems, CHI+GI '87*, Toronto, April 1987.

[Hec92a] Paul Heckbert. Discontinuity meshing for radiosity. *Third Eurographics Workshop on Rendering*, pages 203–226, May 1992. http ://www.cs.cmu.edu/ ph.

[Hec92b] Paul S. Heckbert. Radiosity in flatland. In A. Kilgour and L. Kjelldahl, editors, *Computer Graphics Forum (EUROGRAPHICS '92 Proceedings)*, volume 11(3), pages 181–192, September 1992.

[HG86] Eric A. Haines and Donald P. Greenberg. The light buffer : A ray tracer shadow testing accelerator. *IEEE Computer Graphics and Applications*, 6(9) :6–16, September 1986.

[HG94] P. Heckbert and M. Garland. Multiresolution modelling for fast rendering. *Proceedings of Graphics Interface'94*, pages 43–50, 1994.

[HGar] Paul S. Heckbert and Michael Garland. Survey of polygonal surface simplification algorithms. Technical report, CS Dept., Carnegie Mellon U., to appear. http ://www.cs.cmu.edu/ ph.

[HH84] Paul S. Heckbert and Pat Hanrahan. Beam tracing polygonal objects. In *Computer Graphics (SIGGRAPH '84 Proceedings)*, volume 18(3), pages 119–127, July 1984.

[HH89] Charles Hansen and Thomas C. Henderson. CAGD-based computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-11(11) :1181–1193, November 1989.

[HH96] Seth Hutchinson and Gregory D. Hager. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5) :651–670, October 1996.

[HH97] Paul S. Heckbert and Michael Herf. Simulating soft shadows with graphics hardware. Technical Report CMU-CS-97-104, CS Dept., Carnegie Mellon U., January 1997. http ://www.cs.cmu.edu/~ph.

[HK85] M. Hebert and T. Kanade. The 3-D profile method for object recognition. In *Proceedings, CVPR '85 (IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, June 10–13, 1985)*, IEEE Publ. 85CH2145-1., pages 458–463. IEEE, IEEE, 1985.

[HK89] Seth A. Hutchinson and Avinash C. Kak. Planning sensing strategies in a robot work cell with multi-sensor capabilities. *IEEE Journal of Robotics and Automation*, 5(6) :765–783, December 1989.

[HKL97] Dan Halperin, Lydia Kavraki, and Jean-Claude Latombe. Robotics. In J.E. Goodman and J. O'Rourke, editors, *CRC Handbook of Discrete and Computational Geometry*. CRC Press, 1997. http ://robotics.stanford.edu/˜latombe/pub.html.

[HKL98] D. Halperin, L.E. Kavraki, and J.C. Latombe. Robot algorithms. In M. Atallah, editor, *CRC Handbook of Algorithms and Theory of Computation*. CRC Press, 1998. http ://robotics.stanford.edu/˜latombe/pub.html.

[HLW96] V. Hlavac, A. Leonardis, and T. Werner. Automatic selection of reference views for image-based scene representations. *Lecture Notes in Computer Science*, 1064 :526– ? ?, 1996.

[HM96] André Hinkenjann and Heinrich Müller. Hierarchical blocker trees for global visibility calculation. Research Report 621/1996, University of Dortmund, Universität Dortmund, 44221 Dortmund, Germany, August 1996. http ://ls7-www.cs.uni-dortmund.de/˜hinkenja/.

[HMC+97] T. Hudson, D. Manocha, J. Cohen, M. Lin, K. Hoff, and H. Zhang. Accelerated occlusion culling using shadow frusta. In *Proceedings of the 13th International Annual Symposium on Computational Geometry (SCG-97)*, pages 1–10, New York, June4–6 1997. ACM Press. http ://www.cs.unc.edu/ hudson/projects/occlusion/scg97.ps.

[HMK+97] Lichan Hong, Shigeru Muraki, Arie Kaufman, Dirk Bartz, and Taosong He. Virtual voyage : Interactive navigation in the human colon. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 27–34. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.

[Hof92] Georg Rainer Hofmann. Who invented ray tracing ? a historical remark. *The Visual Computer*, 9(1) :120–125, 1992.

[Hop96] H. Hoppe. Progressive meshes. *Computer Graphics*, 30(Annual Conference Series) :99–108, 1996.

[HS95] Nicolas Holzschuch and Francois Sillion. Accurate Computation of the Radiosity Gradient for Constant and Linear Emitters. In P. M. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, pages 186–195, New York, NY, 1995. Springer-Verlag.

[HS99] Nicolas Holzschuch and François X. Sillion. An exhaustive error-bounding algorithm for hierarchical radiosity. *Computer Graphics Forum*, 1999. to appear, http ://www.loria.fr/˜holzschu.

[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, Francois Sillion, and George Drettakis. An Efficient Progressive Refinement Strategy for Hierarchical Radiosity. In *Fifth Eurographics Workshop on Rendering*, pages 343–357, Darmstadt, Germany, June 1994. http ://www.loria.fr/˜holzschu.

[HT96] Stephen Hardt and Seth Teller. High-fidelity radiosity rendering at interactive rates. In Xavier Pueyo and Peter Schröder, editors, *Eurographics Rendering Workshop 1996*, pages 71–80, New York City, NY, June 1996. Eurographics, Springer Wein. ISBN 3-211-82883-4.

[HW91] Eric Haines and John Wallace. Shaft culling for efficient ray-traced radiosity. In *Eurographics Workshop on Rendering*, 1991.

[HW98] Michael E. Houle and Richard Webber. Approximation algorithms for finding best viewpoints. In Sue H. Whitesides, editor, *Proceedings of the 6th International Symposium on Graph Drawing*, number vol. 1547 in Lecture Notes in Computer Science, pages 210–223. Springer, Heidelberg, Germany, 1998.

[HWP97] David Hedley, Adam Worrall, and Derek Paddon. Selective culling of discontinuity lines. In J. Dorsey and P. Slusallek, editors, *Rendering Techniques '97*, pages 69–81, 8th EG workshop on Rendering, Saint Etienne, France, June 1997. Springer Verlag.

[HZ81] H. Hubschman and S. W. Zucker. Frame-to-frame coherence and the hidden surface computation : constraints for a convex world. *Computer Graphics*, 15(3) :45–54, August 1981.

[HZ82] H. Hubschman and S. W. Zucker. Frame-to-frame coherence and the hidden surface computation : constraints for a convex world. *ACM Transactions on Graphics*, 1(2) :129–162, April 1982.

[ICK+99] Kenneth E. Hoff III, Tim Culver, John Keyser, Ming Lin, and Dinesh Manocha. Fast computation of generalized voronoi diagrams using graphics hardware. In *Computer Graphics (SIGGRAPH '99 Proceedings)*, August 1999.

[Ike87] Katsushi Ikeuchi. Generating an interpretation tree from a CAD model for 3-D object recognition in bin-picking tasks. *International Journal of Computer Vision*, 1(2) :145–65, 1987.

[JF93] A. K. Jain and P. J. Flynn. *Three Dimensional Object Recognition Systems*. Elsevier, Amsterdam, 1993.

[JK88] J. W. Jaromczyk and M. Kowaluk. Skewed projections with an application to line stabbing in R^3. In *Proceedings of the Fourth Annual Symposium on Computational Geometry (Urbana-Champaign, IL, June 6–8, 1988)*, pages 362–370, New York, 1988. ACM, ACM Press.

[Jon71] C. B. Jones. A new approach to the 'hidden line' problem. *The Computer Journal*, 14(3) :232–237, August 1971.

[JW89] David Jevans and Brian Wyvill. Adaptive voxel subdivision for ray tracing. In *Proceedings of Graphics Interface '89*, pages 164–172, June 1989.

[Kaj82] James T. Kajiya. Ray tracing parametric patches. In *Computer Graphics (SIGGRAPH '82 Proceedings)*, volume 16(3), pages 245–254, July 1982.

[Kaj86] J. T. Kajiya. The rendering equation. In David C. Evans and Rusell J. Athay, editors, *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20(4), pages 143–150, August 1986.

[KB98] D. Kriegman and P. Belhumeur. What shadows reveal about object structure. In *Proceedings European Conference on Computer Vision*, pages 399–414, 19998. http ://www-cvr.ai.uiuc.edu/~kriegman/.

[Kel97] Alexander Keller. Instant radiosity. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 49–56. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.

[Ker81] Y.L. Kergosien. La famille des projections orthogonales d'une surface et ses singularités. *C.R. Acad. Sc. Paris*, 292 :929–932, 1981.

[KG79] Douglas S. Kay and Donald P. Greenberg. Transparency for computer synthesized images. In *Computer Graphics (SIGGRAPH '79 Proceedings)*, volume 13(3), pages 158–164, August 1979.

[Kir87] D. B. Kirk. The simulation of natural features using cone tracing. *The Visual Computer*, 3(2) :63–71, August 1987.

[KK86] Timothy L. Kay and James T. Kajiya. Ray tracing complex scenes. In David C. Evans and Russell J. Athay, editors, *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20, pages 269–278, August 1986.

[KKCS98] Bomjun Kwon, Dae Seoung Kim, Kyung-Yong Chwa, and Sung Yong Shin. Memory-efficient ray classification for visibility operations. *IEEE Transactions on Visualization and Computer Graphics*, 4(3), July – September 1998. ISSN 1077-2626.

[KKMB96] D. Kersten, D.C. Knill, P. Mamassian, and I. Bülthoff. Illusory motion from shadow. *Nature*, 379(31), 1996.

[KM94] S. Krishnan and D. Manocha. Global visibility and hidden surface algorithms for free form surfaces. Technical Report TR94-063, UNC Chapel Hill, February 1994. http ://www.cs.unc.edu/Research/tech-report.html.

[Kø84] J. J. Kœnderink. What does the occluding contour tell us about solid shape ? *Perception*, 13 :321–330, 1984.

[Koe87] J. J. Koenderink. An internal representation for solid shape based on the topological properties of the apparent contour. In W. Richards and S. Ullman, editors, *Image Understanding 1985–86*, pages 257–285, Norwood, NJ, 1987. Ablex.

[Koe90] Jan J. Koenderink. *Solid Shape*. MIT Press, Cambridge, Massachusetts, 1990.

[KP90] D. Kriegman and J. Ponce. Computing exact aspect graphs of curved objects : Solids of revolution. *International Journal of Computer Vision*, 5(2) :119–135, 1990.

[KS97] Krzysztof S. Klimaszewski and Thomas W. Sederberg. Faster ray tracing using adaptive grids. *IEEE Computer Graphics and Applications*, 17(1) :42–51, January 1997. bounding volume hierarchy with grids at each level & more.

[Kut91] H. Kutruff. *Room Acoustics (3rd edition)*. Elseiver Applied Science, 1991.

[Kv76] J.J. Koenderink and A.J. vanDoorn. The singularities of the visual mapping. *BioCyber*, 24(1) :51–59, 1976.

[Kv79] J.J. Koenderink and A.J. vanDoorn. The internal representation of solid shape with respect to vision. *BioCyber*, 32 :211–216, 1979.

[KvD82] Jan J. Kœnderink and Andrea J van Doorn. What does the occluding contour tell us about solid shape ? *Perception*, 11 :129–137, 1982.

[KWCH97] Kris Klimaszewski, Andrew Woo, Frederic Cazals, and Eric Haines. Additional notes on nested grids. *Ray Tracing News*, 10(3), December 1997. http ://www.povray.org/rtn/.

[KYCS98] Kim, Yoo, Chwa, and Shin. Efficient algorithms for computing a complete visibility region in three-dimensional space. *Algorithmica*, 20, 1998.

[Lat91] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer, Dordrecht, The Netherlands, 1991.

[Lau94] A. Laurentini. The visual hull concept for silhouette-based image understanding. *T-PAMI*, 16 :150–162, 1994.

[Lau95] A. Laurentini. How far 3d shapes can be understood from 2d silhouettes. *T-PAMI*, 17 :188–195, 1995.

[Lau97] A. Laurentini. How many 2D silhouettes does it take to reconstruct a 3D object ? *Computer Vision and Image Understanding : CVIU*, 67(1) :81– ? ?, ? ? ? ? 1997.

[Lau99] A. Laurentini. Computing the visual hull of solids of revolution. *Pattern Recognition*, 32(3), 1999.

[LC79] Jeff Lane and Loren Carpenter. A generalized scan line algorithm for the computer display of parametrically defined surfaces. *Computer Graphics and Image Processing*, 11(3) :290–297, November 1979.

[LCWB80] Jeffrey M. Lane, Loren C. Carpenter, J. Turner Whitted, and James F. Blinn. Scan line methods for displaying parametrically defined surfaces. *Communications of the ACM*, 23(1) :23–34, January 1980.

[LD97] Celine Loscos and George Drettakis. Interactive high-quality soft shadows in scenes with moving objects. *Computer Graphics Forum*, 16(3) :C219–C230, September 4–8 1997. http ://www-imagis.imag.fr/Publications/.

[LF94] Stephane Laveau and Olivier Faugeras. 3-D scene representation as a collection of images and fundamental matrices. Technical Report RR-2205, Inria, Institut National de Recherche en Informatique et en Automatique, 1994.

[LF96] Robert R. Lewis and Alain Fournier. Light-driven global illumination with a wavelet representation of light transport. In Xavier Pueyo and Peter Schröder, editors, *Eurographics Rendering Workshop 1996*, pages 11–20, New York City, NY, June 1996. Eurographics, Springer Wien. ISBN 3-211-82883-4.

[LG95] David Luebke and Chris Georges. Portals and mirrors : Simple, fast evaluation of potentially visible sets. In Pat Hanrahan and Jim Winget, editors, *1995 Symposium on Interactive 3D Graphics*, pages 105–106. ACM SIGGRAPH, April 1995. ISBN 0-89791-736-7, http ://www.cs.unc.edu/˜luebke/publications/portals.html.

[LGBL97] Steven LaValle, Hector H. González-Baños, Craig Becker, and Jean-Claude Latombe. Motion strategies for maintaining visibility of a moving target. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1997.

[LH96] Marc Levoy and Pat Hanrahan. Light field rendering. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 31–42. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996, http ://www-graphics.stanford.edu/papers/light/.

[LH99] Steven M. LaValle and John E. Hinrichsen. Visibility-based pursuit-evasion : The case of curved environments. In *IEEE International Conference on Robotics and Automation*, August 1999. http ://janowiec.cs.iastate.edu/˜lavalle.

[Lis94] Dani Lischinski. Incremental delaunay triangulation. In Paul S. Heckbert, editor, *Graphics Gems IV*, pages 47–59. Academic Press Professional, San Diego, CA, 1994.

[LL86] D. T. Lee and A. K. Lin. Computational complexity of art gallery problems. *IEEE Transactions on Information Theory*, 32 :276–282, 1986.

[LLG+97] Steven M. LaValle, David Lin, Leonidas J. Guibas, Jean-Claude Latombe, and Rajev Motwani. Finding an unpredictable target in a workspace with obstacles. In *IEEE International Conference on Robotics and Automation*, August 1997. http ://janowiec.cs.iastate.edu/˜lavalle.

[LM98] M. Lin and D. Manocha. Applied computational geometry. In *Encyclopedia on Computer Science and Technology*. Marcel Dekker, 1998. http ://www.cs.unc.edu/˜lin.

[LMW90] Bernd Lamparter, Heinrich Müller, and Jorg Winckler. The ray-z-buffer - an approach for ray-tracing arbitrarily large scenes. Technical Report 675/1998, University of Freiburg, Institut fur Mathematik, April 1990.

[LPW79] Tomás Lozano-Pérez and Michael A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10) :560–570, October 1979.

[LRDG90] Jed Lengyel, Mark Reichert, Bruce R. Donald, and Donald P. Greenberg. Real-time robot motion planning using rasterizing computer graphics hardware. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24(4), pages 327–335, August 1990.

[LS97] Jed Lengyel and John Snyder. Rendering with coherent layers. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 233–242. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.

[LSG94] 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.

[LT99] Fei-Ah Law and Tiow-Seng Tan. Preprocessing occlusion for real-time selective refinement. In *1999 Symposium on Interactive 3D Graphics*. ACM SIGGRAPH, April 1999.

[LTG92] Dani Lischinski, Filippo Tampieri, and Donald P. Greenberg. Discontinuity meshing for accurate radiosity. *IEEE Computer Graphics and Applications*, 12(6) :25–39, November 1992. http ://www.cs.huji.ac.il/%7Edanix/publications.html.

[LTG93] D. Lischinski, F. Tampieri, and D. P. Greenberg. Combining hierarchical radiosity and discontinuity meshing. *Computer Graphics*, 27(Annual Conference Series) :199–208, 1993. http ://www.cs.huji.ac.il/%7Edanix/publications.html.

[LW95] Eric P. Lafortune and Yves D. Willems. A 5D Tree to Reduce the Variance of Monte Carlo Ray Tracing. In P. M. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, pages 11–20, New York, NY, 1995. Springer-Verlag.

[LZ97] M. S. Langer and S. W. Zucker. Casting light on illumination : A computational model and dimensional analysis of sources. *Computer Vision and Image Understanding : CVIU*, 65(2) :322–335, February 1997.

[MAG68] MAGI. 3-D simulated graphics offered by service bureau. *Datamation*, 14 :69, February 1968.

[Mal87] Jitendra Malik. Interpreting line drawings of curved objects. *International Journal of Computer Vision*, 1 :73–103, 1987.

[Max91] Nelson L. Max. Unified sun and sky illumination for shadows under trees. *Computer Vision, Graphics, and Image Processing. Graphical Models and Image Processing*, 53(3) :223–230, May 1991.

[May99] Using Maya, rendering. Alias Wavefront, (Maya user manual), 1999.

[MB93] J. Maver and R. Bajcsy. Occlusions as a guide for planning the next view. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(5) :417–433, May 1993.

[MB95] L. McMillan and G. Bishop. Plenoptic modeling : An image-based rendering system. *Computer Graphics*, 29(Annual Conference Series) :39–46, 1995.

[MBGN98] Tom McReynolds, David Blythe, Brad Grantham, and Scott Nelson. Advanced graphics programming techniques using Open GL. Siggraph'1998 course notes, 1998.

[McK87] Michael McKenna. Worst-case optimal hidden-surface removal. *ACM Transactions on Graphics*, 6(1) :19–28, January 1987.

[MDC93] Herve Maurel, Ives Duthen, and Rene Caubet. A 4D ray tracing. In R. J. Hubbold and R. Juan, editors, *Eurographics '93*, pages 285–294, Oxford, UK, 1993. Eurographics, Blackwell Publishers.

[Mey90] Urs Meyer. Hemi-cube ray-tracing : A method for generating soft shadows. In C. E. Vandoni and D. A. Duce, editors, *Eurographics '90*, pages 365–376. North-Holland, September 1990.

[MG95] Scott O. Mason and Armin Grün. Automatic sensor placement for accurate dimensional inspection. *Computer Vision and Image Understanding : CVIU*, 61(3) :454–467, May 1995.

[MGBY99] Yohai Makbily, Craig Gotsman, and Reuven Bar-Yehuda. Geometric algorithms for message filtering in decentralized virtual environments. In *1999 Symposium on Interactive 3D Graphics*. ACM SIGGRAPH, April 1999.

[MI98] Jun Miura and Katsushi Ikeuchi. Task-oriented generation of visual sensing strategies in assembly tasks. *IEEE Transactions on Pattern Analysis and machine Intelligence*, 20(2) :126–138, February 1998. also available as Carnegie Mellon University Technical Report CS-95-116.

[Mit87] Don P. Mitchell. Generating antialiased images at low sampling densities. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 65–72, July 1987.

[Mit96] Don P. Mitchell. Consequences of stratified sampling in graphics. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 277–280. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[MKT+97] Lee Markosian, Michael A. Kowalski, Samuel J. Trychin, Lubomir D. Bourdev, Daniel Goldstein, and John F. Hughes. Real-time nonphotorealistic rendering. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 415–420. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.

[MO88] M. McKenna and J. O'Rourke. Arrangements of lines in 3-space : a data structure with applications. In *Proceedings of the Fourth Annual Symposium on Computational Geometry (Urbana-Champaign, IL, June 6–8, 1988)*, pages 371–380, New York, 1988. ACM, ACM Press.

[MOK95] Karol Myszkowski, Oleg G. Okunev, and Tosiyasu L. Kunii. Fast collision detection between complex solids using rasterizing graphics hardware. *The Visual Computer*, 11(9) :497–512, 1995. ISSN 0178-2789.

[MPT97] Ignacio Martin, Xavier Pueyo, and Dani Tost. An Image Space Refinement Criterion for Linear Hierarchical Radiosity. In *Proceedings of Graphics Interface '97*, San Francisco, CA, May 1997. Morgan Kaufmann.

[MS85] Matthew T. Mason and J. Kenneth Salisbury Jr. *Robot hands and the mechanics of manipulation*. Cambridge, Mass. : MIT Press, c1985, 298 p. (The MIT Press series in artificial intelligence) CALL NUMBER : TJ211 .M366 1985, 1985.

[MS95] Paulo W. C. Maciel and Peter Shirley. Visual navigation of large environments using textured clusters. In Pat Hanrahan and Jim Winget, editors, *1995 Symposium on Interactive 3D Graphics*, pages 95–102. ACM SIGGRAPH, April 1995. ISBN 0-89791-736-7.

[Mue95] Carl Mueller. Architecture of image generators for flight simulators. Technical Report TR-95-015, UNC Chapel Hill, February 1995. http ://www.cs.unc.edu/Research/tech-report.html.

[Mul89] Ketan Mulmuley. An efficient algorithm for hidden surface removal. In Jeffrey Lane, editor, *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics (SIGGRAPH '89)*, pages 379–388, Boston, MA, USA, July 1989. ACM Press.

[Mul91] Mulmuley. Hidden surface removal with respect to a moving view point (extended abstract). In *STOC : ACM Symposium on Theory of Computing (STOC)*, 1991.

[Mun95] Olaf Munkelt. Aspect-trees : Generation and interpretation. *Computer Vision and Image Understanding : CVIU*, 61(3) :365–386, May 1995.

[Mur87] G. Murch. olor displays and color science. In H. John Durret, editor, *Color and the Computer*, pages 1–25. Academic Press, Boston, 1987.

[MWCF90] Joseph Marks, Robert Walsh, Jon Christensen, and Mark Friedell. Image and intervisibility coherence in rendering. In *Proceedings of Graphics Interface '90*, pages 17–30, May 1990.

[Mys98] Karol Myszkowski. The visible differences predictor : applications to global illumination problems. In *Eurographics Workshop on Rendering*, Vienna, Austria, June 1998.

[MZ92] J.L. Mundy and A. Zisserman. *Geometric Invariance in Computer Vision*. MIT press, Cambridge, MA, 1992.

[NAT90] Bruce Naylor, John Amanatides, and William Thibault. Merging BSP trees yields polyhedral set operations. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24(4), pages 115–124, August 1990.

[Nay92] Bruce F. Naylor. Partitioning tree image representation and generation from 3D geometric models. In *Proceedings of Graphics Interface '92*, pages 201–212, May 1992.

[Neu95] Laszlo Neumann. Monte Carlo Radiosity. *Computing*, 55(1) :23–42, 1995.

[Ney96] Fabrice Neyret. Synthesizing verdant landscapes using volumetric textures. In Xavier Pueyo and Peter Schröder, editors, *Eurographics Rendering Workshop 1996*, pages 215–224, New York City, NY, June 1996. Eurographics, Springer Wien. ISBN 3-211-82883-4.

[Ney98] Fabrice Neyret. Modeling, Animating, and Rendering Complex Scenes Using Volumetric Textures. *IEEE Transactions on Visualization and Computer Graphics*, 4(1) :55–70, January 1998.

[Nil69] Nils J. Nilsson. A mobile automaton : An application of artificial intelligence techniques. In Donald E. Walker and Lewis M. Norton, editors, *Proceedings of the 1st International Joint Conference on Artificial Intelligence*, pages 509–520, Washington, D. C., May 1969. William Kaufmann.

[NMN87] Tomoyuki Nishita, Yasuhiro Miyawaki, and Eihachiro Nakamae. A shading model for atmospheric scattering considering luminous intensity distribution of light sources. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21(4), pages 303–310, July 1987.

[NN83] Tomoyuki Nishita and Eihachiro Nakamae. Half-tone representation of 3-D objects illuminated by area or polyhedron sources. In *Proc. of IEEE Computer Society's Seventh International Computer Software and Applications Conference (COMPSAC83)*, pages 237–242, November 1983.

[NN85] T. Nishita and E. Nakamae. Continuous tone representation of three-dimensional objects taking account of shadows and interreflection. In B. A. Barsky, editor, *Computer Graphics (SIGGRAPH '85 Proceedings)*, volume 19(3), pages 23–30, July 1985.

[NNB97] L. Neumann, A. Neumann, and P. Bekaert. Radiosity with well distributed ray sets. *Computer Graphics Forum*, 16(3) :261–270, August 1997. Proceedings of Eurographics '97. ISSN 1067-7055.

[NNS72] M. E. Newell, R. G. Newell, and T. L. Sancha. A solution to the hidden surface problem. In *Proceedings of the ACM Annual Conference*, volume I, pages 443–450, Boston, Massachusetts, August 1972.

[NON85] T. Nishita, I. Okamura, and E. Nakamae. Shading models for point and linear sources. *ACM Transactions on Graphics*, 4(2) :124–146, April 1985.

[NR95] Bruce Naylor and Lois Rogers. Constructing partitioning trees from bezier-curves for efficient intersections and visibility. In Wayne A. Davis and Przemyslaw Prusinkiewicz, editors, *Graphics Interface '95*, pages 44–55. Canadian Information Processing Society, Canadian Human-Computer Communications Society, May 1995. ISBN 0-9695338-4-5.

[NSL99] C. Nissoux, T. Simeon, and J.P. Laumond. Visibility based probabilistic roadmaps. Technical Report 99057, LAAS, February 1999. doc@laas.fr.

[O'R87] J. O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, New York, NY, 1987.

[O'R94] Joseph O'Rourke. *Computational geometry in C*. Cambridge University Press, 1994.

[ORDP96] 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(3), pages 237–249, Poitiers, France, September 1996.

[Ort97] Rachel Orti. *Radiosité dynamique 2D et complexe de visibilité*. PhD thesis, Université Joseph Fourier (Grenoble), 1997. PhD Thesis.

[OS97] J. O'Rourke and I. Streinu. Vertex-edge pseudo-visibility graphs : Characterization and recognition. In *Proceedings of the 13th International Annual Symposium on Computational Geometry (SCG-97)*, pages 119–128, New York, June 4–6 1997. ACM Press.

[OW88] M. H. Overmars and E. Welzl. New methods for computing visibility graphs. In *Proceedings of the Fourth Annual Symposium on Computational Geometry (Urbana-Champaign, IL, June 6–8, 1988)*, pages 164–171, New York, 1988. ACM, ACM Press.

[PA91] Pierre Poulin and John Amanatides. Shading and shadowing with linear light sources. *Computers and Graphics*, 15(2) :259–265, 1991.

[PBG92] Cary B. Phillips, Norman I. Badler, and John Granieri. Automatic viewing control for 3D direct manipulation. In Marc Levoy and Edwin E. Catmull, editors, *Proceedings of the 1992 Symposium on Interactive 3D Graphics*, pages 71–74, Cambridge, MA, March–April 1992. ACM Press.

[PD86] W. H. Plantinga and C. R. Dyer. An algorithm for constructing the aspect graph. In *27th Annual Symposium on Foundations of Computer Science*, pages 123–131, Los Angeles, Ca., USA, October 1986. IEEE Computer Society Press.

[PD87] H. Plantinga and C. R. Dyer. The asp : a continuous viewer-centered representation for 3D object recognition. In *First International Conference on Computer Vision, (London, England, June 8–11, 1987)*, pages 626–630, Washington, DC., 1987. IEEE Computer Society Press.

[PD90] H. Plantinga and C. R. Dyer. Visibility, occlusion, and the aspect graph. *International Journal of Computer Vision*, 5(2) :137–160, 1990.

[PDS90] Harry Plantinga, Charles R. Dyer, and W. Brent Seales. Real-time hidden-line elimination for a rotating polyhedral scene using the aspect representation. In *Proceedings of Graphics Interface '90*, pages 9–16, May 1990.

[Pel90] M. Pellegrini. Stabbing and ray shooting in 3-dimensional space. In *Proc. 6th Annu. ACM Sympos. Comput. Geom.*, pages 177–186, 1990.

[Pel93] Marco Pellegrini. Ray shooting on triangles in 3-space. *Algorithmica*, 9 :471–494, 1993.

[Pel94] Pellegrini. On lines missing polyhedral sets in 3-space. *GEOMETRY : Discrete & Computational Geometry*, 12, 1994.

[Pel96] Marco Pellegrini. Repetitive hidden surface removal for polyhedra. *Journal of Algorithms*, 21(1) :80–101, July 1996. http ://www.imc.pi.cnr.it/˜marcop.

[Pel97a] Pellegrini. Monte carlo approximation of form factors with error bounded a priori. *Discrete & Computational Geometry*, 17, 1997.

[Pel97b] Marco Pellegrini. Ray-shooting and lines in space. In J.E. Goodman and J. O'Rourke, editors, *CRC Handbook of Discrete and Computational Geometry*, pages 599–614. CRC Press, 1997. http ://www.imc.pi.cnr.it/˜marcop/.

[Pel99] Marco Pellegrini. A geometric approach to computing higher-order form factors. In *Proceedings of the 15th International Annual Symposium on Computational Geometry (SCG-99)*, New York, June 4–6 1999. ACM Press.

[Pet92] Sylvain PetitJean. Computing exact aspect graphs of smooth objects bounded by smooth algebraic surfaces. Master's thesis, University of Illinois, Urbana-Champaign, IL, June 1992. availabel as technical report UIUC-BI-AI-RCV-92-04.

[Pet95] Sylvain Petitjean. The number of views of piecewise-smooth algebraic objects. *Proceedings of Proceedings of the Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science*, 900 :571– ? ?, 1995.

[Pet96] Sylvain Petitjean. The enumerative geometry of projective algebraic surfaces and the complexity of aspect graphs. *International Journal of Computer Vision*, 19(3) :1–27, 1996.

[Pet98] Sylvain Petitjean. A computational geometric approach to visual hulls. *International Journal of Computational Geometry and Applications*, 8(4) :407–436, 1998. Special issue on applied computational geometry, edited by Ming Lin and Dinesh Manocha. http ://www.loria.fr/ petitjea/.

[PF92] Pierre Poulin and Alain Fournier. Lights from highlights and shadows. In David Zeltzer, editor, *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, volume 25(2), pages 31–38, March 1992. http ://www.iro.umontreal.ca/labs/infographie/papers/.

[PFFG98] Sumanta N. Pattanaik, James A. Ferwerda, Mark D. Fairchild, and Donald P. Greenberg. A multiscale model of adaptation and spatial vision for realistic image display. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 287–298. ACM SIGGRAPH, Addison Wesley, July 1998. ISBN 0-89791-999-8.

[Pie93] Georg Pietrek. Fast Calculation of Accurate Formfactors. In *Fourth Eurographics Workshop on Rendering*, Series EG 93 RW, pages 201–220, Paris, France, June 1993.

[PJ91] Andrew Pearce and David Jevans. Exploiting shadow coherence in ray-tracing. In *Proceedings of Graphics Interface '91*, pages 109–116, June 1991.

[PK90] Jean Ponce and David J. Kriegman. Computing exact aspect graphs of curved objects : parametric surfaces. In William Dietterich, Tom ; Swartout, editor, *Proceedings of the 8th National Conference on Artificial Intelligence*, pages 1074–1081, Hynes Convention Centre ?, July 29–August 3 1990. MIT Press.

[Pla88] William Harry Plantinga. *The asp : a continuous, viewer-centered object representation for computer vision*. PhD thesis, The University of Wisconsin, Madison, December 1988.

[Pla93] Harry Plantinga. Conservative visibility preprocessing for efficient walkthroughs of 3D scenes. In *Proceedings of Graphics Interface '93*, pages 166–173, Toronto, Ontario, Canada, May 1993. Canadian Information Processing Society.

[Plü65] J. Plücker. On a new geometry of space. *Phil. Trans. Royal Soc. London*, 155 :725–791, 1865.

[Pop94] Arthur R. Pope. Model-based object recognition : A survey of recent research. Technical Report TR-94-04, University of British Columbia, Computer Science Department, January 1994. http ://www.cs.ubc.ca/tr/1994/TR-94-04.

[Pot87] Michael Potmesil. Generating octree models of 3D objects from their silhouettes in a sequence of images. *Computer Vision, Graphics, and Image Processing*, 40(1) :1–29, October 1987.

[PPK92] S. Petitjean, J. Ponce, and D.J. Kriegman. Computing exact aspect graphs of curved objects : Algebraic surfaces. *IJCV*, 1992.

[PPR99] Helmut Pottmann, Martin Peternell, and Bahram Ravani. An introduction to line geometry with application. *Computer-Aided Design*, 31 :3–16, 1999.

[PRJ97] Pierre Poulin, Karim Ratib, and Marco Jacques. Sketching shadows and highlights to position lights. In *Proceedings of Computer Graphics International 97*, pages 56–63. IEEE Computer Society, June 1997.

[PS92] Pellegrini and Shor. Finding stabbing lines in 3-space. *GEOMETRY : Discrete & Computational Geometry*, 8, 1992.

[PV96a] Pocchiola and Vegter. Topologically sweeping visibility complexes via pseudotriangulations. *GEOMETRY : Discrete & Computational Geometry*, 16, 1996.

[PV96b] M. Pocchiola and G. Vegter. The visibility complex. *Internat. J. Comput. Geom. Appl.*, 1996. special issue devoted to ACM-SoCG'93.

[PY90] Paterson and Yao. Efficient binary space partitions for hidden-surface removal and solid modeling. *GEOMETRY : Discrete & Computational Geometry*, 5, 1990.

[Qua96] Matthew Quail. Space time ray-tracing using ray classification. Master's thesis, Macquarie University, November 1996.

[RA96] Michael Reed and Peter K. Allen. Automated model acquisition using volumes of occlusion. In *IEEE International Conference on Robotics and Automation*, April 1996. http ://www.cs.columbia.edu/robotics/.

[RB98] Paul Rademacher and Gary Bishop. Multiple-center-of-projection images. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 199–206. ACM SIGGRAPH, Addison Wesley, July 1998. ISBN 0-89791-999-8.

[Rei92] Mark C. Reichert. A two-pass radiosity method driven by lights and viewer position. Master's thesis, Program of Computer Graphics, Cornell University, January 1992.

[RF95] V. Ranjan and A. Fournier. Union of spheres (UoS) model for volumetric data. In *Proceedings of the 11th Annual Symposium on Computational Geometry*, pages C2–C3, New York, NY, USA, June 1995. ACM Press.

[RH94] John Rohlf and James Helman. IRIS performer : A high performance multiprocessing toolkit for real–Time 3D graphics. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 381–395. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.

[Rie87] J.H. Rieger. On the classification of views of piecewise smooth objects. *Image and Vision Computing*, 1987.

[Rie90] J. H. Rieger. The geometry of view space of opaque objects bounded by smooth surfaces. *Artificial Intelligence(1-2)*, 44 :1–40, 1990.

[Rie92] J.H. Rieger. Global bifurcation sets and stable projections of non-singular algebraic surface. *IJCV*, 1992.

[Rie93] J. H. Rieger. Computing view graphs of algebraic surfaces. *Journal of Symbolic Computation*, 16(3) :259–272, September 1993.

[Riv95] S. Rivière. Topologically sweeping the visibility complex of polygonal scenes. In *Comm. 11th Annu. ACM Sympos. Computat. Geom.*, 1995.

[Riv97a] S. Rivière. *Calculs de visibilité dans un environnement polygonal 2D.* PhD thesis, Université Joseph Fourier (Grenoble), 1997. PhD Thesis.

[Riv97b] S. Rivière. Dynamic visibility in polygonal scenes with the visibility complex. In *Comm. 13th Annu. ACM Sympos. Computat. Geom.*, 1997.

[Riv97c] S. Rivière. Walking in the visibility complex with applications to visibility polygons and dynamic visibility. In *9th Canadian Conference on Computational Geometry*, August 1997.

[RM97] D.R. Roberts and A.D. Marshall. A review of viewpoint planning. Technical Report 97007, Univeristy of Wales, Cardiff, August 1997. http ://www.cs.cf.ac.uk/Research/Rrs/1997/detail007.html.

[Rob63] L.G. Roberts. Machine perception of three dimensional solids. Technical Report TR-315, Lincoln Laboratory, MIT, Cambridge, MA, May 1963. Also in Tippet, J.T *et al.*, eds., *Optical and Electro Optical Information Processing*, MIT Press, Cambridge, MA, 1964.

[Rog97] David F. Rogers. *Procedural Elements for Computer Graphics*. Mc Graw-Hill, 2 edition, 1997.

[RPG99] Mahesh Ramasubramanian, Sumanta N. Pattanaik, and Donald P. Greenberg. A perceptually based physical error metric for realistic image synthesis. In *Computer Graphics (SIGGRAPH '99 Proceedings)*, August 1999.

[RS90] J. A. Reeds and L. A. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2), 1990.

[RSC87] William T. Reeves, David H. Salesin, and Robert L. Cook. Rendering antialiased shadows with depth maps. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21(4), pages 283–291, July 1987.

[RW80] Steven M. Rubin and Turner Whitted. A 3-dimensional representation for fast rendering of complex scenes. In *Computer Graphics (SIGGRAPH '80 Proceedings)*, volume 14(3), pages 110–116, July 1980.

[SAG94] Brian Smits, James Arvo, and Donald Greenberg. A clustering algorithm for radiosity in complex environments. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 435–442. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.

[San76] L. A. Santaló. *Integral Geometry and Geometric Probability*. Addison-Wesley, Reading, MA, 1976.

[SB87] John M. Snyder and Alan H. Barr. Ray tracing complex models containing surface tessellations. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21(4), pages 119–128, July 1987.

[SB88] J. Stewman and K. Bowyer. Creating the perspective projection aspect graph of polyhedral objects. In *Second International Conference on Computer Vision (Tampa,, FL, December 5–8, 1988)*, pages 494–500, Washington, DC,, 1988. Computer Society Press.

[SB90] John H. Stewman and Kevin W. Bowyer. Direct construction of the perspective projection aspect graph of convex polyhedra. *Computer Vision, Graphics, and Image Processing*, 51(1) :20–37, July 1990.

[Sbe93] M. Sbert. An integral geometry based method for fast form-factor computation. *Computer Graphics Forum*, 12(3) :C409–C420, 1993.

[SBGS69] R. A. Schumacker, R. Brand, M. Gilliland, and W. Sharp. Study for applying computer-generated images to visual simulation. Technical Report AFHRL–TR–69–14, U.S. Air Force Human Resources Laboratory, 1969.

[Sch93] L. L. Schumaker. Computing optimal triangulations using simulated annealing. *Computer Aided Geometric Design*, 10(3) :329–346, August 1993.

[SD92] W. Brent Seales and Charles R. Dyer. Viewpoint from occluding contour. *Computer Vision, Graphics and Image Processing : Image Understanding*, 55 :198–211, 1992.

[SD95] 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. held in Los Angeles, California, 06-11 August 1995, http ://www-imagis.imag.fr/˜Francois.Sillion/.

[SDB85] L. R. Speer, T. D. Derose, and B. A. Barsky. A theoretical and empirical analysis of coherent ray-tracing. In M. Wein and E. M. Kidd, editors, *Graphics Interface '85 Proceedings*, pages 1–8. Canadian Inf. Process. Soc., 1985.

[SDB97] François Sillion, George Drettakis, and Benoit Bodelet. Efficient impostor manipulation for real-time visualization of urban scenery. In D. Fellner and L. Szirmay-Kalos, editors, *Computer Graphics Forum (Proc. of Eurographics '97)*, volume 16-3, pages 207–218, Budapest, Hungary, September 1997. http ://www-imagis.imag.fr/~Francois.Sillion/Papers/Index.html.

[SDS96] Eric J. Stollnitz, Tony D. DeRose, and David H. Salesin. *Wavelets for Computer Graphics : Theory and Applications*. Morgann Kaufmann, San Francisco, CA, 1996.

[SG82] S. Sechrest and D. P. Greenberg. A visible polygon reconstruction algorithm. *ACM Transactions on Graphics*, 1(1) :25–42, January 1982.

[SG94] 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, http ://www.dgp.toronto.edu/people/JamesStewart/papers/.

[SG96] Oded Sudarsky and Craig Gotsman. Output-sensitive visibility algorithms for dynamic scenes with applications to virtual reality. *Computer Graphics Forum*, 15(3) :C249–C258, September 1996. http ://www.cs.technion.ac.il/~sudar/cv_eng.html.

[Sgi99] Silicon Graphics 320 datasheet. http ://visual.sgi.com/research/data/whitepapers.html, 1999.

[SH93] Peter Schröder and Pat Hanrahan. On the form factor between two polygons. In *Computer Graphics Proceedings, Annual Conference Series, 1993*, pages 163–164, 1993.

[Sha97] Erin Shaw. Hierarchical radiosity for dynamic environments. *Computer Graphics Forum*, 16(2) :107–118, 1997. ISSN 0167-7055.

[She92] Thomas Shermer. Recent results in art galleries. In *Proc. IEEE*, pages 80 :1384–1399, September 1992.

[She96] Shewchuk. Triangle : Engineering a 2D quality mesh generator and delaunay triangulator. In *WACG : 1st Workshop on Applied Computational Geometry : Towards Geometric Engineering, WACG*. LNCS, 1996.

[Sil95] Francois 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.

[Sil99] François Sillion. Will anyone really use radiosity ? In *Invited talk at Graphics Interface, Kingston, Ontario*, 1999. http ://www.dgp.toronto.edu/gi99/papers/programme.html.

[SJ89] T. Sripradisvarakul and R. Jain. Generating aspect graphs for curved objects. In *Proceedings, Workshop on Interpretation of 3D Scenes (Austin, TX, November 27–29, 1989)*, pages 109–115, Washington, DC., 1989. Computer Society Press, Computer Society Press.

[SK97] Sudhanshu K. Semwal and Hakan Kvarnstrom. Directed safe zones and the dual extend algorithms for efficient grid tracing during ray tracing. In Wayne A. Davis, Marilyn Mantei, and R. Victor Klassen, editors, *Graphics Interface '97*, pages 76–87. Canadian Information Processing Society, Canadian Human-Computer Communications Society, May 1997. ISBN 0-9695338-6-1 ISSN 0713-5424.

[SK98] A. James Stewart and Tasso Karkanis. Computing the approximate visibility map, with applications to form factors and discontinuity meshing. *Eurographics Workshop on Rendering*, June 1998. http ://www.dgp.toronto.edu/people/JamesStewart/.

[SKFNC97] L. Szirmay-Kalos, T. Fóris, L. Neumann, and B. Csébfalvi. An analysis of quasi-monte carlo integration applied to the transillumination radiosity method. *Computer Graphics Forum*, 16(3) :271–282, August 1997. Proceedings of Eurographics '97. ISSN 1067-7055.

[SKvW+92] Mark Segal, Carl Korobkin, Rolf van Widenfelt, Jim Foran, and Paul Haeberli. Fast shadows and lighting effects using texture mapping. In Edwin E. Catmull, editor, *Proceedings of the 19th Annual ACM Conference on Computer Graphics and Interactive Techniques*, pages 249–252, New York, NY, USA, July 1992. ACM Press.

[SL98] John Snyder and Jed Lengyel. Visibility sorting and compositing without splitting for image layer decomposition. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 219–230. ACM SIGGRAPH, Addison Wesley, July 1998. ISBN 0-89791-999-8.

[Sla92] M. Slater. A comparison of three shadow volume algorithms. *The Visual Computer*, 9(1) :25–38, 1992.

[SLH89] Partha Srinivasan, Ping Liang, and Susan Hackwood. Computational Geometric Methods in Volumetric Intersection for 3D Reconstruction. In *Proc. 1989 IEEE Int. Conf. Robotics and Automation*, pages 190–195, 1989.

[SLSD96] J. Shade, D. Lischinski, D. H. Salesin, and T. DeRose. Hierarchical image caching for accelerated walkthroughs of complex environments. *Computer Graphics*, 30(Annual Conference Series) :75–82, 1996.

[Smi99] Brian Smits. Efficiency issues for ray tracing. *Journal of Graphics Tools*, 1999. to appear, http ://www2.cs.utah.edu/˜bes/.

[Sny92] John M. Snyder. Interval analysis for computer graphics. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26(2), pages 121–130, July 1992.

[Sol78] Herbert Solomon. *Geometruc Probability*. SIAM CBMS-NSF Regional Conference Series in Applied Mathematics, Philadelphia, PA, 1978.

[Sol98] Cyril Soler. *Représentation hiérarchique de la visibilité pour le contrôle de l'erreur en simulation de l'éclairage*. PhD thesis, Université Joseph Fourier, Grenoble I, December 1998. http ://www-imagis.imag.fr/˜Cyril.Soler.

[Som51] D. M. Y. Sommerville. *Analytical Geometry in Three Dimensions*. Cambridge University Press, Cambridge, 1951.

[SON96] Kristian T. Simsarian, Thomas J. Olson, and N. Nandhakumar. View-invariant regions and mobile robot self-localization. *IEEE Transactions on Robotics and Automation*, 12(5) :810–816, October 1996.

[SP89] Francois Sillion and Claude Puech. A general two-pass method integrating specular and diffuse reflection. In Jeffrey Lane, editor, *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23(3), pages 335–344, July 1989.

[SP94] Francois Sillion and Claude Puech. *Radiosity and Global Illumination*. Morgan Kaufmann, San Francisco, CA, 1994.

[SP97] I. Shimshoni and J. Ponce. Finite-resolution aspect graphs of polyhedral objects. *PAMI*, 19(4) :315–327, April 1997. http ://pete.cs.uiuc.edu/˜trpethe/ponce_publ.html.

[Spe92a] L. Richard Speer. An updated cross-indexed guide to the ray-tracing literature. *Computer Graphics*, 26(1) :41–72, January 1992.

[Spe92b] R. Speer. A new data structure for high-speed, memory efficient ray shooting. In *Eurographics Workshop on Rendering*, 1992.

[SPP95] Mateu Sbert, Frederic Perez, and Xavier Pueyo. Global Monte Carlo : A Progressive Solution. In P. M. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, pages 231–239, New York, NY, 1995. Springer-Verlag.

[SS89] David Salesin and Jorge Stolfi. The ZZ-buffer : a simple and efficient rendering algorithm with reliable antialiasing. In *Proceedings of the PIXIM '89*, pages 451–466, 1989.

[SS90] David Salesin and Jorge Stolfi. Rendering CSG models with a ZZ-buffer. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24(4), pages 67–76, August 1990.

[SS96a] G. Schaufler and W. Stürzlinger. A three-dimensional image cache for virtual reality. In *Proceedings of EUROGRAPHICS'96*, 1996.

[SS96b] Cyril Soler and François Sillion. Accurate error bounds for multi-resolution visibility. In Xavier Pueyo and Peter Schröder, editors, *Eurographics Rendering Workshop 1996*, pages 133–142, New York City, NY, June 1996. Eurographics, Springer Wein. ISBN 3-211-82883-4, http ://www-imagis.imag.fr/~Cyril.Soler/csoler.gb.html.

[SS98a] Cyril Soler and François Sillion. Fast calculation of soft shadow textures using convolution. In *Computer Graphics Proceedings, Annual Conference Series :* SIGGRAPH '98 (Orlando, FL), page ? ? ACM SIGGRAPH, New York, July 1998. http ://www-imagis.imag.fr/~Cyril.Soler/csoler.gb.html.

[SS98b] Cyril Soler and François Sillion. Automatic calculation of soft shadow textures for fast, high-quality radiosity. In Nelson Max and George Drettakis, editors, *Eurographics Rendering Workshop 1998*, New York City, NY, June 1998. Eurographics, Springer Wein. http ://www-imagis.imag.fr/~Cyril.Soler/csoler.gb.html.

[SSS74] Ivan E. Sutherland, Robert F. Sproull, and Robert A. Schumacker. A characterization of ten hidden-surface algorithms. *ACM Computing Surveys*, 6(1) :1–55, March 1974.

[Ste82] Ian Stewart. *Oh Catastrophe !* Belin, 1982.

[Ste91] J.H. Stewman. *Viewer-centered Representation for Polyhedral Objects*. PhD thesis, Department of Computer Science and Engineering, University of South Florida, Tampa, 1991. PhD Dissertation.

[STN87] Mikio Shinya, Tokiichiro Takahashi, and Seiichiro Naito. Principles and applications of pencil tracing. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21(4), pages 45–54, July 1987.

[Sto91] J. Stolfi. *Oriented Projective Geometry : A Framework for Geometric Computations*. Academic Press, 1991.

[Stu94] W. Sturzlinger. Adaptive Mesh Refinement with Discontinuities for the Radiosity Method. In *Fifth Eurographics Workshop on Rendering*, pages 239–248, Darmstadt, Germany, June 1994. http ://prometheus.gup.uni-linz.ac.at :8001/papers/.

[Stu99] Wolfgang Stuerzlinger. Imaging all visible surfaces. In *Proceedings of Graphics Interface, Kingston, Ontario*, 1999. http ://www.dgp.toronto.edu/gi99/papers/programme.html.

[Sun92] Kelvin Sung. Area sampling buffer : Tracing rays with Z-buffer hardware. In A. Kilgour and L. Kjelldahl, editors, *Computer Graphics Forum (EUROGRAPHICS '92 Proceedings)*, volume 11(3), pages 299–310, September 1992.

[SZ89] Thomas W. Sederberg and Alan K. Zundel. Scan line display of algebraic surfaces. In Jeffrey Lane, editor, *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23(3), pages 147–156, July 1989.

[TA96] Raj Talluri and J.K. Aggarwal. Mobile robot self-location using model-image feature correspondence. *IEEE Transactions on Robotics and Automation*, 12(1) :63–77, February 1996.

[TA98] Seth Teller and John Alex. Frustum casting for progressive interactive rendering. Technical Report TR-740, MIT Laboratory for Computer Science, January 1998.

[TAA+96] Roberto Tamassia, Pankaj K. Agarwal, Nancy Amato, Danny Z. Chen, David Dobkin, Scot Drysdale, Steven Fortune, Michael T. Goodrich, John Hershberger, Joseph O'Rourke, Franco P. Preparata, Joerg-Rudiger Sack, Subhash Suri, Ioannis Tollis, Jeffrey S. Vitter, and Sue Whitesides. Strategic directions in computational geometry. *ACM Computing Surveys*, 28(4) :591–606, December 1996.

[Tam93] Filippo Tampieri. *Discontinuity Meshing for Radiosity Image Synthesis*. Ph.D. thesis, Cornell University, Ithaca, NY, 1993.

[TAT95] K.A. Tarabanis, P.K. Allen, and R.Y. Tsai. A survey of sensor planning in computer vision. *IEEE Transactions on robotics and automation*, 11(1) :86–104, February 1995.

[Tel92a] Seth J. Teller. Computing the antipenumbra of an area light source. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26-2, pages 139–148, July 1992. http ://graphics.lcs.mit.edu/~seth/pubs/pubs.html.

[Tel92b] Seth J. Teller. *Visibility Computations in Densely Occluded Polyhedral Environments*. PhD thesis, CS Division, UC Berkeley, October 1992. Tech. Report UCB/CSD-92-708, http ://graphics.lcs.mit.edu/˜seth/pubs/pubs.html.

[TFFH94] Seth Teller, Celeste Fowler, Thomas Funkhouser, and Pat Hanrahan. Partitioning and ordering large radiosity computations. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 443–450. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0, http ://graphics.lcs.mit.edu/˜seth/pubs/pubs.html.

[TG95] G. H. Tarbox and S. N. Gottschlich. Planning for complete sensor coverage in inspection. *Computer Vision and Image Understanding : CVIU*, 61(1) :84–111, January 1995.

[TG97a] Nicholas Tsingos and Jean-Dominique Gascuel. Sound rendering in dynamic environments with occlusions. In Wayne A. Davis, Marilyn Mantei, and R. Victor Klassen, editors, *Graphics Interface '97*, pages 9–16. Canadian Information Processing Society, Canadian Human-Computer Communications Society, May 1997. ISBN 0-9695338-6-1 ISSN 0713-5424, http ://www.bell-labs.com/user/tsingos/.

[TG97b] Nicolas Tsingos and Jean-Dominique Gascuel. Fast rendering of sound occlusion and diffraction effects for virtual acoustic environments. In *104th AES convention, Amsterdam, The Netherlands, preprint n. 4699 (P4-7)*, May 1997. http ://www.bell-labs.com/user/tsingos/.

[TH91] Seth J. Teller and Michael E. Hohmeyer. Computing the lines piercing four lines. Technical report, CS Dpt. UC Berkeley, 1991.

[TH93] Seth Teller and Pat Hanrahan. Global visibility algorithms for illumination computations. In *Computer Graphics Proceedings, Annual Conference Series, 1993*, pages 239–246, 1993. http ://graphics.lcs.mit.edu/˜seth/pubs/pubs.html.

[Tho56] R. Thom. Les singularités des applications différentiables. *Annales Institut Fourier*, 6 :43–87, 1956.

[Tho72] R. Thom. *Structural Stability and Morphogenesis*. Benjamin, New-York, 1972.

[Tor90] Enric Torres. Optimization of the binary space partition algorithm (BSP) for the visualization of dynamic scenes. In C. E. Vandoni and D. A. Duce, editors, *Eurographics '90*, pages 507–518. North-Holland, September 1990.

[TR93] Jack Tumblin and Holly E. Rushmeier. Tone reproduction for realistic images. *IEEE Computer Graphics and Applications*, 13(6) :42–48, November 1993. also appeared as Tech. Report GIT-GVU-91-13, Graphics, Visualization & Usability Center, Coll. of Computing, Georgia Institute of Tech.

[TS91] Seth J. Teller and Carlo H. Séquin. Visibility preprocessing for interactive walkthroughs. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 61–69, July 1991. http ://graphics.lcs.mit.edu/˜seth/pubs/pubs.html.

[TT90] Toshimitsu Tanaka and Tokiichiro Takahashi. Cross scanline algorithm. In C. E. Vandoni and D. A. Duce, editors, *Eurographics '90*, pages 63–74. North-Holland, September 1990.

[TT95] Toshimitsu Tanaka and Tokiichiro Takahashi. Fast shadowing algorithm for linear light sources. *Computer Graphics Forum*, 14(3) :205–216, August 1995. Proceedings of Eurographics '95. ISSN 1067-7055.

[TT97] T. Tanaka and T. Takahashi. Fast analytic shading and shadowing for area light sources. *Computer Graphics Forum*, 16(3) :231–240, August 1997. Proceedings of Eurographics '97. ISSN 1067-7055.

[TTK96] Konstantinos Tarabanis, Roger Y. Tsai, and Anil Kaul. Computing occlusion-free viewpoints. *PAMI*, 18(3) :279–292, March 1996.

[TUWR97] Emmanuelle Trucco, Manickam Umasuthan, Andrew M. Wallace, and Vito Roberto. Model-based planning of optimal sensor placement for inspection. *IEEE Transactions on Robotics and Automation*, 13(2) :182–194, April 1997.

[Tv97] A. C. Telea and C. W. A. M. van Overveld. The close objects buffer : a sharp shadow detection technique for radiosity methods. *Journal of Graphics Tools*, 2(2) :1–8, 1997. ISSN 1086-7651.

[TWFP97] Robert F. Tobler, Alexander Wilkie, Martin Feda, and Werner Purgathofer. A hierarchical subdivision algorithm for stochastic radiosity methods. In Julie Dorsey and Philipp Slusallek, editors, *Eurographics Rendering Workshop 1997*, pages 193–204, New York City, NY, June 1997. Eurographics, Springer Wien. ISBN 3-211-83001-4.

[Ull89] Shimon Ullman. Aligning pictorial descriptions : An approach to object recognition. *Cognition*, 32(3) :193–254, August 1989.

[Urr98] Jorge Urrutia. Art gallery and illumination problems. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook on Computational Geometry*. Elsevier Science, 1998. http ://www.csi.uottawa.ca :80/˜jorge/online_papers.

[UT97] Carlos Ureña and Juan C. Torres. Improved irradiance computation by importance sampling. In Julie Dorsey and Philipp Slusallek, editors, *Eurographics Rendering Workshop 1997*, pages 275–284, New York City, NY, June 1997. Eurographics, Springer Wien. ISBN 3-211-83001-4.

[Vea97] Eric Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. Ph.d. thesis, Stanford University, December 1997. http ://www-graphics.stanford.edu/˜ericv/.

[Ved93] Christophe Vedel. Computing Illumination from Area Light Sources by Approximate Contour Integration. In *Proceedings of Graphics Interface '93*, pages 237–244, San Francisco, CA, May 1993. Morgan Kaufmann.

[VG97] Eric Veach and Leonidas J. Guibas. Metropolis light transport. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 65–76. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.

[VLN96] M. Venditelli, J.P. Laumond, and C. Nissoux. Obstacle distances and visibility in the car-like robot metrics. Technical Report 96437, LAAS, November 1996. doc@laas.fr.

[WA77] K. Weiler and K. Atherton. Hidden surface removal using polygon area sorting. *Computer Graphics*, 11(2) :214–222, July 1977. Proceedings of SIGGRAPH'77, held in San Jose, California ; 20–22 July 1977.

[WA90] Andrew Woo and John Amanatides. Voxel occlusion testing : a shadow accelerator for ray tracing. In *Proceedings of Graphics Interface '90*, pages 213–220, June 1990.

[Wal75] David Waltz. Understanding lines drawings of scenes with shadows. In Patrick H. Winston, editor, *The Psychology of Computer Vision*, Computer Science Series, pages 19–91. McGraw-Hill, 1975.

[Wan92] Leonard Wanger. The effect of shadow quality on the perception of spatial relationships in computer generated imagery. In David Zeltzer, editor, *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, volume 25(2), pages 39–42, March 1992.

[War69] J. Warnock. A hidden-surface algorithm for computer generated half-tone pictures. Technical Report TR 4–15, NTIS AD-733 671, University of Utah, Computer Science Department, 1969.

[War94] Greg Ward. A contrast-based scalefactor for luminance display. In Paul Heckbert, editor, *Graphics Gems IV*, pages 415–421. Academic Press, Boston, 1994.

[Wat70] G.S. Watkins. *A Real Time Visible Surface Algorithm*. Ph.d. thesis, University of Utah, June 1970.

[Wat88] N. A. Watts. Calculating the principal views of a polyhedron. In *Ninth International Conference on Pattern Recognition (Rome, Italy, November 14–17, 1988)*, pages 316–322, Washington, DC, 1988. Computer Society Press.

[Wat90] Mark Watt. Light-water interaction using backward beam tracing. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24(4), pages 377–385, August 1990.

[WBP98] Y. Wang, H Bao, and Q. Peng. Accelerated walkthroughs of virtual environments based on visibility processing and simplification. In *Computer Graphics Forum (Proc. of Eurographics '98)*, volume 17-3, pages C–187–C195, Lisbon, Portugal, September 1998.

[WC91] Andrew Woo and Steve Chall. An efficient scanline visibility implementation. In *Proceedings of Graphics Interface '91*, pages 135–142, June 1991.

[WEH89] John R. Wallace, Kells A. Elmquist, and Eric A. Haines. A ray tracing algorithm for progressive radiosity. In Jeffrey Lane, editor, *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23(3), pages 315–324, July 1989.

[Wei93] I. Weiss. Geometric invariant and object recognition. *Internat. J. Comput. Vision*, 10(3) :207–231, 1993.

[WF90] R. Wang and H. Freeman. Object recognition based on characteristic view classes. In *Proceedings* 10*th International Conference on Pattern Recognition*, Atlantic City, NJ, 17-21 June 1990.

[WH96] L. R. Williams and A. R. Hanson. Perceptual completion of occluded surfaces. *Computer Vision and Image Understanding : CVIU*, 64(1), 1996.

[WHG84] Hank Weghorst, Gary Hooper, and Donald P. Greenberg. Improved computational methods for ray tracing. *ACM Transactions on Graphics*, 3(1) :52–69, January 1984.

[Whi55] H. Whitney. On singularities of mappings of euclidean spaces. i. mappings of the plane into the plane. *Annals of Mathematica*, 62(3) :374–410, 1955.

[Whi78] T. Whitted. A scan line algorithm for computer display of curved surfaces. In *Computer Graphics (Special SIGGRAPH '78 Issue, preliminary papers)*, pages 8–13, August 1978.

[Whi80] Turner Whitted. An improved illumination model for shaded display. *CACM, 1980*, 23(6) :343–349, 1980.

[WHP98] Adam Worrall, David Hedley, and Derek Paddon. Interactive animation of soft shadows. In *Proceedings of Computer Animation 1998*, pages 88–94. IEEE Computer Society, June 1998. http ://www.cs.bris.ac.uk/˜worrall/scope/port95.html.

[Wil78] Lance Williams. Casting curved shadows on curved surfaces. In *Computer Graphics (SIGGRAPH '78 Proceedings)*, volume 12(3), pages 270–274, August 1978.

[Wil96] L. R. Williams. Topological reconstruction of a smooth manifold-solid from its oclluding contour. *Computer Vision and Image Understanding : CVIU*, 64(2), 1996.

[Woo92] Andrew Woo. The shadow depth map revisited. In David Kirk, editor, *Graphics Gems III*, pages 338–342, 582. Academic Press, Boston, MA, 1992.

[Woo97] Andrew Woo. Recursive grids and ray bounding box comments and timings. *Ray Tracing News*, 10(3), December 1997. http ://www.povray.org/rtn/.

[Wor97] Steve Worley. Fast soft shadows. *Ray Tracing News*, 10(1), January 1997. http ://www.povray.org/rtn/.

[WPF90] Andrew Woo, Pierre Poulin, and Alain Fournier. A survey of shadow algorithms. *IEEE Computer Graphics and Applications*, 10(6) :13–32, November 1990. http ://www.iro.umontreal.ca/labs/infographie/papers/.

[WREE67] C. Wylie, G.W. Romney, D.C. Evans, and A.C. Erdahl. Halftone perspective drawings by computer. In *FJCC*, pages 49–58, 1967.

[WS94] G. Winkenbach and D. H. Salesin. Computer-generated pen-and-ink illustration. *Computer Graphics*, 28(Annual Conference Series) :91–100, July 1994. http ://www.cs.washington.edu/research/graphics/pub/.

[WS99] Peter Wonka and Dieter Schmalstieg. Occluder shadows for fast walkthroughs ofurban environments. In *Eurographics'99*, August 1999.

[WW92] Alan Watt and Mark Watt. *Advanced Animation and Rendering Techniques : Theory and Practice*. Addison-Wesley, 1992.

[WW97] Daphna Weinshall and Michael Werman. On view likelihood and stability. *T-PAMI*, 19-2 :97–108, 1997.

[WWP95] Adam Worrall, Claire Willis, and Derek Paddon. Dynamic discontinuities for radiosity. In *Edugraphics + Compugraphics Proceedings*, pages 367–375, P.O. Box 4076, Massama, 2745 Queluz, Portugal, December 1995. GRASP- Graphic Science Promotions and Publications. http ://www.cs.bris.ac.uk/˜worrall/scope/port95.html.

[Yan85] Johnson K. Yan. Advances in computer-generated imagery for flight simulation. *IEEE Computer Graphics and Applications*, 5(8) :37–51, August 1985.

[YHS95] Seungku Yi, Robert M. Haralick, and Linda G. Shapiro. Optimal sensor and light source positioning for machine vision. *Computer Vision and Image Understanding : CVIU*, 61(1) :122–137, January 1995.

[YKSC98] Kwan-Hee Yoo, Dae Seoung Kim, Sung Yong Shin, and Kyung-Yong Chwa. Linear-time algorithms for finding the shadow volumes from a convex area light source. *Algorithmica*, 20(3) :227–241, March 1998.

[YR96] R. Yagel and W. Ray. Visibility computation for efficient walkthrough complex environments. *PRESENCE*, 5(1) :1–16, 1996. http ://www.cis.ohio-state.edu/volviz/Papers/1995/presence.ps.gz.

[Zat93] Harold R. Zatz. Galerkin radiosity : A higher order solution method for global illumination. In *Computer Graphics Proceedings, Annual Conference Series, 1993*, pages 213–220, 1993.

[ZD93] Ze Hong (Jenny) Zhao and David Dobkin. Continuous Algorithms for Visibility : the Space Searching Approach. In *Fourth Eurographics Workshop on Rendering*, pages 115–126, Paris, France, June 1993.

[Zha91] Ning Zhang. Two Methods for Speeding up Form-factor Calculation. In *Second Eurographics Workshop on Rendering*, Barcelona, Spain, May 1991.

[Zha98a] Hanson Zhang. Forward shadow mapping. In George Drettakis and Nelson Max, editors, *Eurographics Rendering Workshop 1998*, New York City, NY, June 1998. Eurographics, Springer Wien. SBN 3-211-83213-0, http ://www.cs.unc.edu/ zhangh/shadow.html.

[Zha98b] Hansong Zhang. *Effective Occlusion Culling for the Interactive Display of Arbitrary Models*. PhD thesis, University of North Carolina, Chapel Hill, 1998. http ://www.cs.unc.edu/ zhangh/research.html.

[ZMHH97] Hansong Zhang, Dinesh Manocha, Thomas Hudson, and Kenneth E. Hoff III. Visibility culling using hierarchical occlusion maps. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 77–88. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7, http ://www.cs.unc.edu/˜zhangh/hom.html.