

Université Grenoble I – Joseph Fourier Sciences et Géographie

3D Visibility: Analytical Study and Applications

Frédo DURAND

Dissertation presented in partial fulfillement of the requirements for the degree of
Docteur de l'Université Joseph Fourier, discipline informatique
Arrêté ministériel du 5 juillet 1984 et du 30 mars 1992
prepared at *iMAGIS-GRAVIR/IMAG-INRIA*. UMR CNRS C5527. to be defended on July
the 12th, 1999

Committee :

Dinesh	MANOCHA	Reviewer
Michel	POCCHIOLA	Reviewer
Seth	TELLER	Reviewer
Jacques	VOIRON	Examiner
Jean-Claude	PAUL	Examiner
Claude	PUECH	Advisor
Georges	DRETTAKIS	Co-advisor

Acknowledgments

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'était 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 discussions 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 à 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 à 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 à 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 à 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 paraît pas gagné). Pardon Alexiiis 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 *Fro-magerie 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 69 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.

TABLE OF CONTENTS

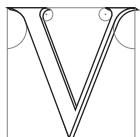
Introduction	7
I Contributions	15
1 Previous work	17
2 The 3D Visibility Complex	29
3 The Visibility Skeleton	57
4 Visibility Driven Hierarchical Radiosity	79
5 General Occlusion Culling using Extended Projections	113
II A Multidisciplinary Survey of Visibility	137
6 Introduction	139
7 Visibility problems	141
8 Preliminaries	157
9 The classics of hidden part removal	165
10 Object-Space	173
11 Image-Space	191
12 Viewpoint-Space	199
13 Line-Space	213
14 Advanced issues	225

15 Conclusions of the survey	231
Conclusions	235
A The 3D Visibility Complex	239
B The Visibility Skeleton	245
C Efficient extended projection using Open GL	249
D Some Notions in Line Space	251
E Online Ressources	253
Contents	255
List of Figures	263
Index	269
References	275

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)



isibility computations are central in many computer graphics methods. Some of the most common include the computation of the objects visible from a viewpoint, the computation of umbra and penumbra. Recent techniques such as global illumination simulations require a more global information, since the mutual visibility of all pairs of points of a scene must be determined. In many cases, visibility is the bottleneck, requiring the development of efficient solutions.

Unfortunately, visibility is intricate to comprehend. It is by nature global; a far away object can be visible, and spatially distant objects can have complex interactions. Previous approaches have tried to answer precise queries, and have not really sought to understand the coherence and the globality of the visibility properties of a 3D scene. This thesis first attempts to compensate for this lack of a framework permitting the description of visibility. We will then try to exploit this better understanding to develop solutions which are new and efficient.

We start this introduction with a brief historical background to place this thesis in the context of visibility computations. We then present our motivations and the points which make 3D visibility difficult. We outline our approach and our aims, before summarising our contributions. We conclude this introduction with a presentation of the structure of this document.

Context

The evolution of computer graphics has been different from that of classic painting. In painting, perspective projection was ignored for a long time, while visibility questions in fact never really arose: it is intuitive and evident for a human-being to determine if an object is visible as well as which object occludes another, especially if the scene lies in front of him. The qualitative aspect (what is visible?) is immediate, while the quantitative aspect (where do points project) was not solved until the Renaissance.

In computer graphics, perspective was implemented very early on, while treating occlusions proved much more problematic. The qualitative intuition is hard to translate into algorithms. The first images were displayed

in *wireframe*: all the edges of a polyhedron are rendered, even if they are behind another object¹. During the sixties and the seventies, hidden-part removal algorithms were developed. They were then named *visibility* algorithms. The question today seems solved in practice. The *z-buffer*, algorithm, which consists in storing and comparing object depth for each pixel, has imposed itself. Efficient implementations in specialized hardware allow personal computer to display thousands of polygons in real-time. We will however come back to the issue of hidden-part removal.

Shadow computations were the following challenge. Hard shadows are simple: they are caused by point light sources. They can be seen as the part of the scene visible from the source. Most of the methods were developed in the seventies, but the problem of hard shadow computation does not seem completely solved. Robustness and aliasing problems remain.

Extended light sources were first studied in the eighties. As opposed to point light sources, a point in the scene can see all, part of, or no point of the source, defining the lit region, the penumbra and the umbra. Soft shadow computation has often been reduced to blurring hard shadows or to sampling using point light sources. The computation of the limits of umbra and penumbra requires the treatment of complex interactions between obstacles. No method has really imposed itself, and soft shadows are rarely used outside research laboratories.

Ray-tracing was developed in the eighties and has had great success. This method reduces any visibility problem to an atomic query: the intersection between a light ray and the scene. This method is simple, highly flexible and versatile, at the price of a usually high cost.

Global illumination methods were introduced in the mid-eighties. They attempt to simulate all light interactions within a scene, in particular the interreflections. Indeed, when light reaches a surfaces, it is reflected, and the object acts as a light source, producing indirect lighting. Shading and shadowing then consist in considering any object as a light source. This requires the computation of visibility between any pair of points of the scene!

The size of the databases to display has dramatically increased in the nineties. CAD models are very large, for example the entire database of the latest Boeing B777 is composed of hundreds of millions of polygons. Users desire to walk through very large virtual environments such as entire buildings or even cities. Despite the ever increasing speed of 3D graphics cards, the direct display of the entire scene is impossible interactively (the latest graphics workstation by Silicon Graphics, the Onyx2 Infinite Reality, can only display 100,000 polygons at 30 frames per seconds).

Visibility computations also occur in other fields. Computer vision is often referred to as the inverse problem of computer graphics. An object recognition method can require the consideration of any possible view of the object, that is, performing a visibility computation for any possible viewpoint. The placement of cameras to monitor a task performed by a robot has to avoid occlusions caused by objects of the work cell. The visual inspection of an environment requires that any point be visible by a robot during its path.

As can be seen with this brief overview, visibility problems have evolved from point-based problems – view from a point, hard shadows – towards more global problems – visibility of an extended light source, mutual visibility of any pair of points. Moreover, the size of the scenes to handle has dramatically increased.

Motivations

Our motivations in this thesis are twofold and complementary: practical and theoretical. Visibility is a crucial practical problem for many methods. Its computation is often a bottleneck, impeding the performances of the methods used. It concurrently offers fascinating geometrical problems, whose understanding is still very limited. We nonetheless think that the vast amount of research dedicated on the subject makes a large overview necessary.

Practical needs

As a motivation, we are particularly interested in two applications of visibility calculation. They are motives and not restrictions. Our work is validated by these applications, but it remains a general study of visibility which has implications in other fields such as computer vision or robotics.

¹This representation, together with black and green displays, remains in many films the mark of a 3D model and the proof of an intensive and high-technology computer activity.

Lighting simulation is an exciting challenge for visibility calculation. Radiosity [SP94, CW93b] is a finite-element method which simulates all light interreflections within a scene. The lighting function is defined on a mesh which subdivides the polygons of the scene into patches. Light interaction must be simulated for each pair of patches, which implies the determination of their quantitative mutual visibility. The calculation cost is thus large, and is the bottleneck of the method [HSD94].

The precision of the computation is very important because it is used to determine the illumination value of the points of the scene. Errors yield artifacts which are very noticeable to the human eye. Current methods use sampling-based approximation which are inaccurate and costly.

The regular meshes usually used cause aliasing artifacts (jagged shadows) in penumbra. To obtain high quality shadows, the scene polygons have to be subdivided along shadow boundaries. This task is intricate even when only primary sources are considered, since many blockers can have complex interactions. This is why previous methods are complicated and prone to robustness problems. They are thus rarely used in practice. Indirect lighting further increases the complexity of the issue, since any polygon can be a secondary light source and cast shadows. This explains why no method handles indirect shadow boundaries for lighting simulation.

Recent advances based on a hierarchical approach [HSA91] permit the concentration of computational resources on “important” energy transfers. The criterion guiding the refinement is crucial for the efficiency of the method, and intuitively visibility should be accounted for. Unfortunately, setting the tedious and unpredictable setting of the thresholds has impeded the use of these techniques in industry.

To summarize, lighting simulation requires intensive and accurate visibility calculations between any pair of objects, together with the determination of shadow boundaries, with respect to potentially any object. Visibility problems are global and complex.

We are also interested in the display of very large scenes. In many cases, only a small number of objects are actually visible. Consider the example of a walk through a city. Nearby facades occlude most of the scene which thus does not need to be taken into account for display. *Occlusion culling* consists in quickly rejecting groups of invisible objects, the actual image is then computed using a z-buffer to resolve occlusion between the remaining objects.

Computations have to be conservative: an object which is visible must not be identified invisible, but the opposite may be true since a z-buffer will eventually be used. Too conservative calculations slow down the display (because too many objects are sent to the graphics hardware) but the image is correct. Numerous methods have been proposed to perform occlusion culling for each frame including recent hardware implementations.

However, in the context of network transmission or if the scene database is too large to fit into main memory, it is desirable to load only the visible objects. The slow speed of disk or network access and low bandwidth make it impossible to load the visible objects for each frame. More global visibility information on the neighbourhood of the viewpoint is necessary.

Previous work on this subject is restricted to cases such as the interior of buildings or to handling only the occlusion due to a single convex blocker at a time. The cumulative occlusion caused by multiple blockers is nevertheless crucial for a better efficiency. The size of the databases makes it impossible to perform complex geometrical computation for each object. The constraints are different compared to the case of global illumination: accuracy is less important than efficiency and the ability to handle large scenes.

A need for understanding

Most authors have attempted to *solve* visibility problems. Analysis or descriptive studies have rarely been proposed on the nature and on the properties involved in visibility problems. The search for direct solutions to concrete and precise problems has led to the reduction of visibility issues to problems which are better understood. This has led to efficient solutions, but the specificity of visibility issues has often been occluded. We believe that it is useful to better understand visibility, leading to more efficient treatment.

The community of computational geometry has been very interested in planar visibility. Work such as the visibility complex [PV96b] provide a powerful and elegant framework to apprehend visibility properties in the plane. Visibility problems and coherence are naturally and intimately expressed.

Unfortunately, theoretical literature is poorer when 3D visibility is concerned. Authors usually study the theoretical complexity of problems such as view computation or ray-shooting. The results available are often reductions of algorithms, and offer little enlightenment on the involved phenomena. Even in theory, the approach is mostly constructive and not analytical.

A need for a synthesis

Our research has led us to read many publications related to visibility in computer graphics, but also in computer vision, robotics and computational geometry. Many similarities exist in the problems and in the approaches but have noticed a lack of communication between these domains. This poor knowledge of the “neighbouring” work – ours primarily – sometimes lead to re-development of techniques, or worse, to ignoring solutions which could solve our problems.

The number of published papers and their spreading make it difficult not to miss entire fields of the issue, or not to get drowned by a sheer number of references.

We think it fruitful to write a survey which includes all these domains and which permits a global overview on visibility. In particular, we believe it is interesting to make the link between approaches and to interpret methods in different contexts. Such a synthesis may be useful to those beginning work on visibility and to those who search for information on problems and techniques studied in different communities.

Difficulty of visibility problems

Most previous work has considered visibility as a black box which gives answers when asked queries. They have tried to optimize the treatment of queries, and have rarely attempted to understand the structural aspects of visibility.

We borrow the metaphor of function or equation analysis from Koenderink and van Doorn [Kv76]. Most visibility methods permit the calculation of the values of the function: What is the point visible along this ray? Are these two points mutually visible? Nonetheless, these queries at a point do not provide more global or structural information which would permit a better description of the function: is it monotonic, continuous, are there inflexions, etc.?

In this section, we outline the conceptual obstacles which make the study of visibility a difficult topic. We first explain why a direct three dimensional approach fails. We then give intuitions which show that there is coherence in visibility. We describe the shortcomings of previous work which deal with visibility as if objects were transparent. We finally discuss why aspect graphs, though they are a major contribution to visibility study, offer a framework which is not completely satisfying.

Nature of visibility

Visibility has no spatial locality. From a point in space very distant objects may be visible while closer ones may be completely hidden. This is one of the pitfalls which explain the difficulty of studying visibility, and why spatial approaches do not treat the basis of the problem. Visibility issues have however often been reduced to three-dimensional problems, usually to spatial localisation or to collision detection.

Consider the example of ray-shooting. Classical methods accelerate it by optimizing the intersection of a ray with the scene. This is usually done using a spatial structure such as a regular grid or bounding boxes which permit the determination of the objects which are spatially close to the ray. The “collision” between the ray and the scene is sought. Such methods have proved useful at accelerating ray-tracing. However, no understanding is achieved. The problem has been circumvented by reducing it to a conceptually simpler problem: do two three-dimensional entities have an intersection?

Which is the nature of visibility properties? What is the framework which permits their study and the description of their structure? Visibility is naturally expressed using light rays. The example of ray-tracing is paradigmatic. An image corresponds to the set of rays going through the eye, the lit regions of a scene are those accessible by rays leaving the light sources.

Rays permit the expression of visibility problems under the following atomic form: Which is the first point hit by a ray going from a point in a given direction? Any visibility question can be reduced to this atomic property.

What is then the problem with three-dimensional approaches? Some of them perform computations on rays, such as ray-tracing. Rays are however not treated as proper entities, but as sets of 3D points. The intersection is computed with the *points* along the ray; the question of the object *seen* by the ray is not directly answered. An intermediate representation (set of points of 3D space) has been introduced.

Such an approach localises visibility problems in 3D space: is a point inside an object? The only structure or coherence which can be captured is purely spatial, related to spatial proximity. We have nonetheless seen that it is difficult to relate visibility and spatial proximity.

Visibility coherence

By coherence we mean the idea that two queries which are “close” will most of the time receive a “similar” answer. This is the mathematical notion of continuity.

Consider a simple example such as the view from a point. This view generally exhibits strong coherence. Consider two points which are close in the image, it is very likely that they correspond to the same object of the scene. To describe such coherence we say that the neighbourhood of a point usually sees the same object.

If the viewpoint is moved slightly, the view will certainly be very similar; There is also coherence. There will be a discontinuity when an object appears or disappears. Another example of coherence is given by soft shadows: the smooth variation from lit regions to umbra regions shows that the visible part of the extended light source varies in a *coherent* manner. This coherence of a view or of the visible part of a source has been widely studied in computer vision and in computer graphics.

This is similar to spatial proximity, since the neighbourhood in the image corresponds to a neighbourhood of three-dimensional points. However, proximity exists between the result of close queries, not between the result of the query and the point of view.

Line space and transparent object

Some approaches [CEG⁺96, MO88, Pel97b, Pel90, Pel93, Tel92a] have studied visibility by considering lines in space. Lines are considered as entities, and three-dimensional spatial technique are not used. A dualisation scheme permits the simplification of calculation and conceptualisation.

However, visibility is defined by the intersection of a line with the objects of the scene. All intersections of a line are taken into account. The notion of first object seen (of visible object in fact) cannot be expressed. Objects are treated as transparent and occlusion is ignored.

The notion of the origin of a ray is crucial. The technique of ray-classification [AK87] offers this possibility, since a ray is described by an origin (a 3D point) and a direction. The set of objects that a group of rays may see is computed. However, this calculation is reduced to the intersection of the embedding of the rays in 3D space. The aforementioned problem of spatial proximity prevents the extraction of structure or coherence.

Aspect graphs

Aspect graph constitute the most successful step towards an understanding of the visibility properties of a 3D scene. They permit the description of all the possible views of a scene, grouped by similarities. It is a fundamental data-structure, and we will use many concepts developed in this context.

In particular, aspect graphs permit the description of the qualitative changes of a view: the appearance or disappearance of a feature.

As an analysis tool, aspect graphs are limited to the study of the views of an object. For example, they do not permit the description or determination of the mutual visibility of objects or points of space.

Approach and aims

We now describe the approach which we have followed to address these issues, and then the criteria which have led our work and which illustrate our aims.

Approach

This document proposes principally a morphological and phenomenological study of visibility.

As can be guessed by reading this introduction, our study of visibility will be based on the notion of light ray, and on the expression of coherence among these light-rays. To better qualify coherence, we will be interested in the regions where coherence is lost.

When studying a question or a system, information is not encoded in “uniform”, continuous regions where by definition nothing occurs. The organisation or structure are defined by discontinuities, boundaries or *catastrophes* (in the sense of Thom [Tho72]). Most previous approaches consist in discovering a country by considering some sample points. We believe that the knowledge of its borders affords a more global understanding, with more detachment, which does however not obviate the need for a study of interior points.

We will attempt to understand for which sets of rays visibility properties remain invariant, and where they change. That is, which rays see the same object.

Our approach is *topological*. Topology is based on the notion of neighbourhood. It is particularly well suited to formally study the notion of coherence. It permits the description of the relationships between rays which see different objects. We will however not use a strict mathematical formalism, even though the underlying notions will be present. We have chosen a more descriptive and concrete language.

The visibility complex of Pocchiola and Vegter [PV96b] was our initial inspiration. Our aim is to extend these concepts to the third dimension.

Criteria

Our aim is first to develop a framework which describes the visibility properties of a three dimensional scene.

Any query or visibility property has to be easily and naturally expressed, be it simple (point seen by a ray) or more global (limits of shadows, parts of the scene visible from a volume, set of possible views of an object).

We want to explicit the notion of coherence. We have to account for the similarity between two close queries, and to explain where and how a change occurs.

It is desirable that this theoretical understanding lead to practical improvements, under the form of a generic data-structure. We want to avoid to focus on particular problems, even though our work is motivated by the aforementioned applications.

Validation has to be performed by a confrontation with concrete practical problems. The elegance and the theoretical complexity of an approach are not enough to conclude its actual efficiency.

Contributions

The contributions of this thesis can be organised into five points described below. The three first items are based on a study in line-space. We first describe a study and a data-structure in line space which are more fundamental and theoretical. This leads to the development of a practical generic visibility tool, which is then applied to lighting simulation. We have also developed an efficient visibility preprocess with respect to volumetric cells for the display of large databases, which reduces the problem to projections on planes. Finally, we have written a vast survey of work related to visibility.

- We have developed the *3D visibility complex*, a data-structure which describes all the visibility relations within a 3D scene composed of polygons and smooth objects. It is a structure in line-space, or as we will see it is more precisely a structure in maximal free segment space. Intuitively, the visibility complex groups light rays which “see” the same object. The boundaries of these groups correspond to visibility changes. In particular, they describe the limits of shadows, the locus of the appearance and disappearance of objects when an observer moves, the so-called *visual events*.

We analyse its complexity, both using classical theoretical bounds, and also with a probabilistic study based on a simplified model of “normal” scenes. We describe an output-sensitive construction algorithm. We present the interpretation of different queries with the visibility complex. We have moreover extended this approach to dynamic scenes.

- The analysis of the visibility complex and the understanding afforded have allowed us to develop a simpler data-structure which we call the *visibility skeleton*. The visibility skeleton is a graph in line-space which describes all *visibility events* of a scene.

We present a construction algorithm which avoids the direct treatment of visual events which is algorithmically complicated and prone to robustness problems because they describe surfaces in 3D space which are not always planar, and which have to be intersected with the scene. We compute *extremal*

stabbing lines which are the extremities of visual events in line-space. Visual events are simply deduced using a catalogue which reports their adjacencies with extremal stabbing lines. The strength of this method is that it requires only simple calculations on single lines. It is moreover very local, which explains its robustness: an error can incur incoherences only in the neighbourhood of an event, the rest of the process is not compromised. Our algorithm permits localised “on-demand” computations and does not require the construction of the entire structure.

This method has been implemented for polygonal scenes, and tests have been performed on scenes up to 1500 polygons. We have also implemented different queries such as the computation of the parts visible from a vertex, the limits of umbra and penumbra, blocker lists, and the limits of the occlusions caused by a blocker. Once the structure is built, these queries are very fast, on the order of milliseconds.

A method to incrementally update this structure after the motion of an object is also described.

- We have applied the visibility skeleton to lighting simulation using hierarchical radiosity. This permits the efficient and exact computation of the amount of light leaving a polygon which arrives at a point. We also use it to subdivide the mesh used to represent the lighting function along shadow boundaries. Any polygon can be considered as a source, which is crucial for scene mainly illuminated by indirect lighting.

As opposed to previous work, our multiresolution representation of the lighting function is not piecewise constant but piecewise linear and continuous. For this purpose, we have developed the use of lazy wavelets with hierarchical triangulations, to handle the irregular meshes due to the limits of umbra and penumbra.

The information encoded in the visibility skeleton is also used to decide at which level of the triangle hierarchy a light transfer should be simulated. We use a perceptual metric. The criteria which we use directly translate how much an artifact due to an approximation would be noticeable by a human observer. Previous methods require the setting of arbitrary and unpredictable thresholds.

Comparisons on a series of test scenes show that our methods provides a higher quality in a shorter time, compared to one of the most recent previous approaches. The price to pay is large memory consumption. We however propose improvements to decrease it.

- We have developed a preprocess for the display of large databases. The region where the observer moves is subdivided into cells. For each cell, we conservatively compute the set of potentially visible objects.

Our method relies on a projection onto planes. We define *extended projection* operators which yield conservative tests: an invisible object may be identified visible, but no visible object can be misclassified as invisible. Our method is the first to handle the cumulative occlusion due to multiple blockers in general scenes in the context of visibility with respect to a volume.

We also describe an *occlusion sweep* where the scene is swept by parallel planes leaving the cell, and where the occlusion due to small objects such as leaves within a forest aggregates.

These methods have been implemented and optimized for today’s graphics hardware for improved efficiency. We can for example walk through a city composed of more than two million polygons containing moving cars at 25 frames per second on an Onyx2.

- We present a vast survey of work related to visibility in different domains. We expose the situations in which visibility issues arise, before proposing a classification which permits a presentation of techniques which is not based on the domain in which they have been developed.

We attempt to propose several references and to underline similarities. We propose details on the most important techniques, to make this overview more concrete and to make it understandable by non-specialists of each aspect.

Thesis structure

The vast amount of work related to visibility has led us to write a survey whose size is beyond what should be reasonably dedicated to previous work in a thesis document. We have thus decided to include it as an independent part, and to write a brief summary of the work which are the most relevant to our research.

The first chapter presents a rapid overview of papers dealing with global visibility and with visibility in complex scenes. We then develop the *3D visibility complex* in chapter 2, then its simplification the *visibility skeleton* in chapter 3. Chapter 4 deals with the application of the skeleton to lighting simulation. Finally, in chapter 5 we describe our occlusion culling preprocess using extended projections.

Some of the material presented in this thesis has been published. In the introduction of each chapter, we indicate the additions and differences with the published papers, to help the reader who is familiar with our work focus on the most recent developments.

The structure of the survey proposed in the second part is described in chapter 6 and is not presented here.

We conclude with a summary of our contribution and a discussion of future work.

A French version of this thesis exists, but the survey of the second part has not been translated.

Part I

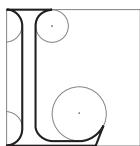
Contributions

CHAPTER 1

Previous work

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*



LITERATURE related to visibility is too vast to be exhaustively presented here. The interested reader will find a comprehensive survey of visibility problems and techniques in second part of this thesis. In what follows, we outline some situations where visibility calculations are involved in computer graphics, computer vision and robotics, and we present some important techniques to solve them. The emphasis is put on recent solutions, large scenes and *global* visibility computation.

By *global* problems or methods we mean those where visibility information is required with respect to a feature larger than a point: visibility from a surface, from a volume, or with respect to the entire scene (as opposed to classical hidden-surface removal which considers visibility with respect to a single point).

We only briefly outline the methods. The interested reader will find more details and a more comprehensive survey in the second part of this thesis.

We first survey the contexts in which visibility problems arise. We survey occlusion culling methods which reject parts of a large scene hidden from a point in section 2 while visibility with respect to volumes is covered in section 3. Limits of umbra are treated in section 4. Section 5 presents the *aspect graph*, a structure which encodes all possible views of an object. Methods in line space are surveyed in section 6, and section 7 deals with the update of visibility information for dynamic scenes. We conclude with a discussion of shortcomings.

1 Visibility problems

1.1 Computer graphics

An important great challenge of early computer graphics was the hidden surface removal problem, which was then a synonym for visibility. See *e.g.* [FvDFH90, Rog97, SSS74] for an overview of the classical techniques, while more theoretical results can be found in [Dor94, Ber93].

However, despite efficient dedicated, hardware implementations, the ever increasing size of the databases to display makes achieving real-time rendering problematic. Approaches have been developed which decrease the amount of geometry sent to the graphics hardware [Cla76, HG94]. The last decade has been particularly rich in such techniques. These include mesh simplification [HGar, Hop96, GH97a], image-based simplification [SLSD96, SS96a, SDB97], and *occlusion culling* which we will introduce in the next section. This latter technique consists of quickly rejecting parts of the geometry which are “obviously” not visible, for example if they are hidden by a nearby large wall.

Occlusion culling can also be exploited to cope with situations where the scene database is too large to fit into main memory, or if it has to be loaded from the network [FST92, Fun96c, COZ98]. Only visible portions of the model, or portions which may be visible in the next few frames are loaded.

The computation of shadows, and especially soft shadows which are caused by an extended light source is still a critical issue [WPF90, Hec87]¹. In the latter case, the source as seen from a point in the scene can be classified as invisible, completely visible or partially visible. This defines the lit part, the umbra and the penumbra. Fundamental issues are the determination of the limits of umbra and penumbra and the robust and efficient determination of the portion of the source visible from a point in the penumbra region.

Global illumination [CW93b, SP94, Gla95] simulates the propagation of light within a scene. If indirect lighting is considered, each surface of the scene is considered as a source and reflects a portion of its incoming light. *Radiosity* methods [CW93b, SP94] assume the objects of the scene to be *diffuse*, *i.e.* light is reflected equally in all directions. The scene polygons are subdivided into patches on which the illumination is assumed constant. The interaction between two patches is modeled by a *form factor* which is the proportion of light leaving one patch which arrives at the other. The computation of these form factors is the costliest part of the method because of the intensive visibility calculations involved [HSD94]. Moreover, a regular subdivision of the input scene induces artifacts such as jagged shadows. A subdivision along shadow boundaries is thus desirable to improve the quality of the resulting images.

1.2 Computer Vision

In computer vision, *model-based object recognition* [Pop94] attempts to match features of an image to features of objects. This requires an efficient representation of the visibility of an object from any viewpoint.

The position of sensors has a dramatic influence on many vision tasks. *Active vision* [AAA⁺92, RM97, TAT95] adapts sensor placement to given tasks. Optimal visibility of a scene or feature is often sought, which requires a complex optimization depending on visibility from many viewpoints.

1.3 Robotics

The deduction of the position of a robot from its current view [GMR95, SON96, TA96] is highly similar to the object recognition problem. Given a map of its environment, the robot has to recognise the structure of the visible walls.

Recently, the problem of finding an unpredictable intruder with a mobile robot has been raised, also referred to as *visibility-based pursuit-evasion* [GLLL98]. The path of the robot has to be planned such that it searches each region of the scene, and such that the intruder can not go from an un-visited region to a visited one without being seen.

¹In a discussion with Dan Wexler from Pacific Data Images, he stated that hard shadows are still a critical issue in production rendering. The problems of aliasing and numerical precision are still involved. Moreover, if tricks exist to obtain soft shadows, a method to efficiently and robustly handle penumbra is desirable.

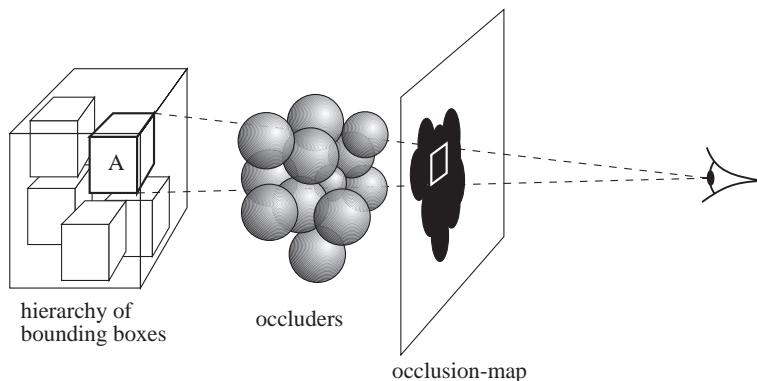


Figure 1.1: Occlusion culling using occlusion maps. Node A of the occludee hierarchy is tested against the occlusion map. Since its projection lies in the projection of the occluders, it is hidden.

2 On-line occlusion culling

The idea of *occlusion culling* has been proposed in the seventies [Jon71, Cla76], but it is only recently that the size of the 3D databases has imposed its practical use.

We present here *on-line* techniques, in which occlusion computations are performed for each frame with respect to the current viewpoint. The scheme is similar for all these methods, inspired by the efficient algorithm used for view-frustum culling [GBW90].

The scene is organised in a hierarchy of bounding volumes. Before rendering each node of the hierarchy, its bounding box is tested for occlusion. If the test fails, all children are discarded. Otherwise, they are recursively tested. Methods differ in the way the occlusion test is performed, and if occlusion caused by all objects are taken into account or if a subset of special occluders first has to be chosen.

2.1 Hierarchical z-buffer and Hierarchical occlusion maps

Greene *et al.* [GKM93, Gre96] proposed extension of z-buffer hardware to perform the occlusion test in image-space and take all occlusions into account. As the image is generated, the bounding box of each node of the hierarchy is sent to the hardware for depth-comparison with the current z-values (but no pixel is actually drawn). This test is optimized through the use of a depth pyramid which avoids the depth comparison for each pixel spanned by the bounding box. This explains the name of the method, the *Hierarchical Z-buffer*.

Nodes of the hierarchy are rendered roughly from front to back, and the depth pyramid is maintained as each object is rendered. This method provides *occluder fusion*, that is, the cumulative occlusion caused by multiple occluders is taken into account because of the natural aggregation in image-space.

Unfortunately, this requires a modification of the graphics hardware for efficient z-value queries and depth pyramid maintenance. Zhang *et al.* [ZMH97, Zha98b] propose the *Hierarchical Occlusion Map* which alleviates these requirements (Fig. 1.1). A (large) subset of nearby occluders is first rendered, and an occlusion map is read from the graphics buffer. Only one query is made to the hardware. This occlusion map is also organised into a pyramid for efficient occlusion tests. (Note however that the *overlap test* is differentiated from the *depth test*, *i.e.* it is first determined if a node projection overlaps the occluder projection, then if the node is behind the occluders. This is particularly important to perform approximate computations. See [ZMH97, Zha98b] for details).

2.2 Convex occluders and visual events

Coorg and Teller [CT96, CT97b] have chosen an object-space approach where occlusion with respect to some large convex occluders is handled. Each occluder defines a *shadow volume* [Cro77] or shadow frustum (infinite truncated pyramid whose apex is the viewpoint) against which the nodes of the hierarchy can be classified as visible, partially visible or occluded.

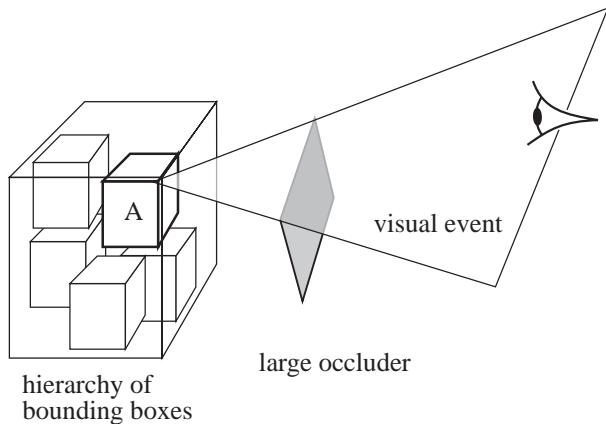


Figure 1.2: Occlusion culling and visual events. Node A of the occludee hierarchy will become completely hidden if ever the viewpoint crosses the wedge of the visual event.

Temporal coherence is used to maintain this classification. They note that for smooth observer motion, visibility changes occur only in the neighbourhood of the partially visible nodes. Moreover, these changes occur only when a *visual event* is crossed (see Fig. 1.2). They consider visual events defined by supporting or separating planes, that is, planes which are tangent to the occluder and a node.

They maintain a list of visual events, detect when the viewpoint crosses them, and consequently update the visibility status and the list of events.

The main drawback of this approach is that occluder fusion is not handled since only objects completely hidden by a single convex occluder are discarded. Related approaches can be found in [HMC⁺97, BHS98, HZ81, HZ82].

2.3 Occluder shadow footprints

Recently Wonka and Schmalstieg [WS99] have proposed an efficient method for city-like scenes. They exploit the fact that cities can be seen as *height fields*, i.e. they can be modeled by a function $z = f(x, y)$. The occlusion caused by a house occurs inside a shadow volume whose apex is the viewpoint. This shadow volume can also be modeled as a function $z = f(x, y)$ because a house lies on the ground. An object is hidden if it is below the shadow volumes of the houses.

They use the graphics hardware to represent and test those functions. An orthographic view from above the scene is used, the z-buffer storing the functions $z = f(x, y)$. Wedges defining the shadow volumes are rasterized, and nodes of the scene hierarchy are discarded if the z-test determines that they lie below the shadow volumes.

3 Shafts and portal sequences

As seen in the previous section, visibility with respect to points has often been characterized using frusta. We now present the extension of this concept to visibility from an area or from a volume, where a single apex is no longer sufficient to define portions of space where occlusion occurs.

3.1 Shaft culling

Haines and Wallace [HW91] have developed shaft-culling to speed-up form-factor visibility computation. They note that the blockers likely to cause occlusion between two objects lie in the approximate convex hull of the objects, called *shaft* in this context (Fig. 1.4). They devised an efficient algorithms to rapidly determine objects intersecting a shaft.

Zhao and Dobkin [ZD93] also proposed a shaft-method where triangles of a scene are preprocessed in a multidimensional space for efficient shaft-triangle intersection. See also [TH93] where shafts are used to maintain lists of blockers in a radiosity context.

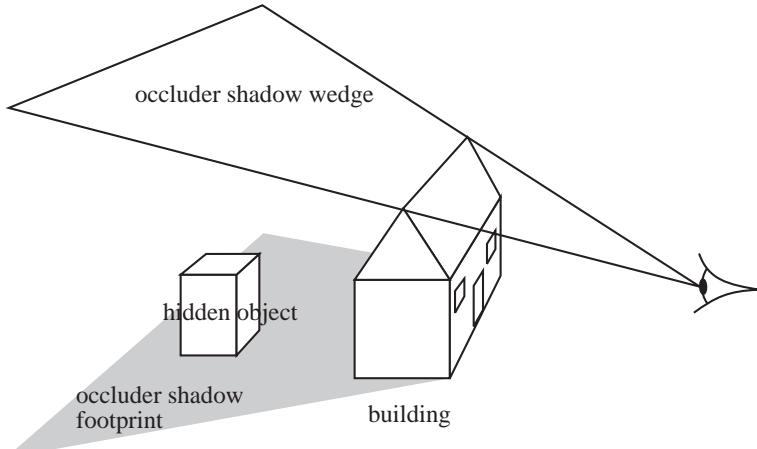


Figure 1.3: Occlusion culling using shadow footprints.

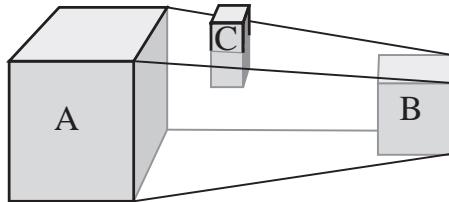


Figure 1.4: Shaft culling. Object C intersects the shaft between A and B, it is thus considered for visibility computation.

3.2 Architectural environments and portals

Airey [Air90] and Teller [Tel92b, TS91, TH93] have performed conservative visibility preprocessing in architectural scenes. The principle is to determine the portion of the scene which is visible from each room (also called *cells*). Note that visibility is here defined with respect to any point in the room.

The *portals* are exploited. A cell is visible from another cell only through a sequence of portals, that is, if there exists a sightline which *stabs* the portals. Visibility is propagated in a depth-first traversal. Visibility through a sequence of portals can be sampled using ray-casting or a s-buffer [Air90], computed using linear programming if the 2D floorplan of the scene is considered [TS91], or for general 3D portals by maintaining a list of separating and supporting planes [TH93].

This provides cell-to-cell visibility (which cells can a given cell see?). Teller [Tel92b, TS91] then refines Cell-to-object visibility by testing objects inside a visible cell against a shaft-like structure defined by the sequence of portals. Finally, at each cell, the *potentially visible set* (PVS) is stored. It contains all the objects which have been determined to be potentially visible from the cell.

During real-time rendering, the potentially visible sets are used to restrict the geometry sent to the graphics hardware. This information can be refined using eye-to-object visibility with a computation similar to that presented in section 2.2 which takes advantage of the precomputed sequence of portals. The PVS information can be used for database pre-fetching [FST92, Fun96c], which is a key-feature of the method.

This preprocessing has also been applied to radiosity simulations [ARB90, TH93, TFFH94, Fun96b] where it allows the computation of light interactions only for pairs of mutually visible polygons.

Luebke and George [LG95] also propose an on-line version where the screen-space bounding box of the portals is used to cull the geometry of adjacent rooms.

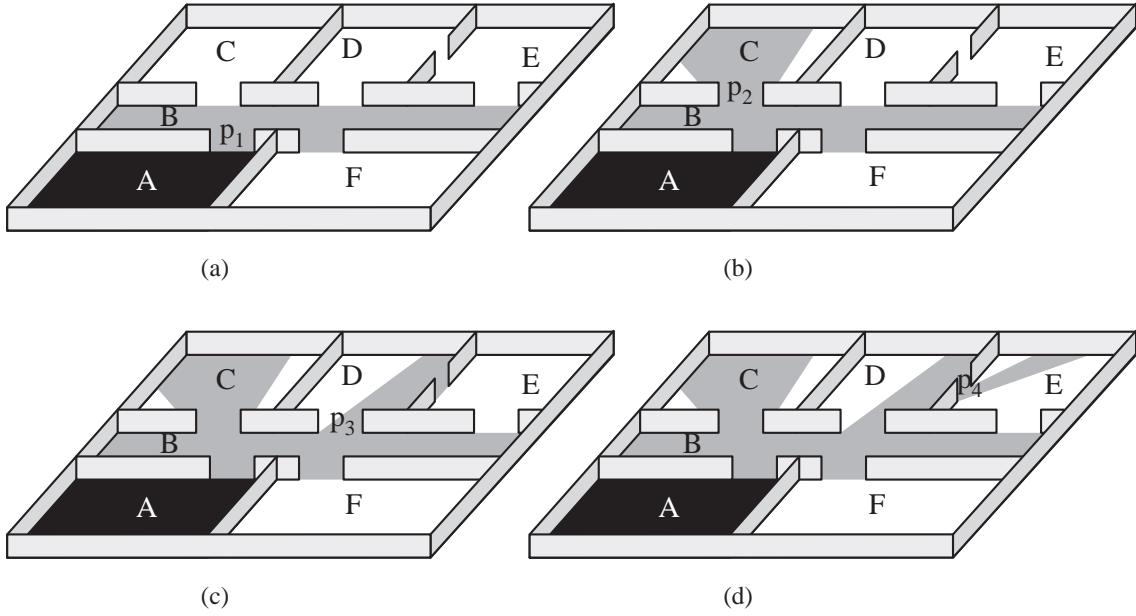


Figure 1.5: Visibility propagation through portals. The rooms visible from room A are computed with a depth-first traversal of their adjacencies. Visibility is propagated through portals p_i .

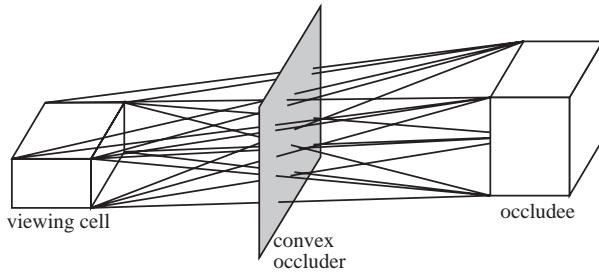


Figure 1.6: Off-line occlusion culling for convex occluders. If all rays between vertices of the viewing cell and the occludee are blocked by the same convex occluder, then the occludee is hidden from any viewpoint within the cell.

3.3 Off-line occlusion culling for convex occluders

Cohen-Or *et al.* [COFHZ98, COZ98] also propose an algorithm for *off-line* occlusion culling. They are not restricted to architectural scenes, but consider only occlusion caused by single convex objects. The region attainable by the observer is subdivided into cells, and for each cell the visibility of the objects of the scene is tested.

This is done by casting rays defined by pairs of vertices of the cell and the object bounding box. If all rays are blocked by a single occluder, then by convexity the occluder hides the object from any point inside the scene.

The obvious drawback of this approach is its inability to handle occlusion from multiple blockers. Moreover ray-casting a huge number of rays induces a high computation time.

4 Umbra and penumbra boundaries

The computation of soft shadows also involves visibility with respect to an extended region. The methods we expose here have first been developed as an extension of shadow volumes. The difference with the previous section is that the expected result is here the exact boundary of soft shadows on the objects of the scene.

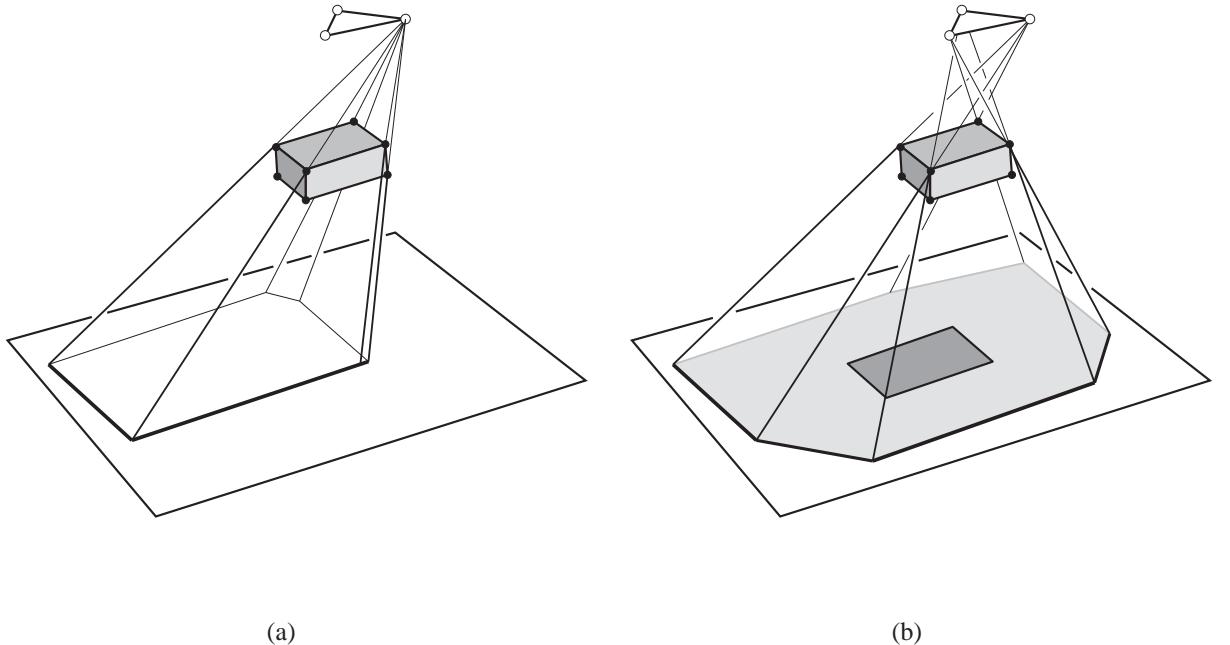


Figure 1.7: Umbra and penumbra boundaries for a convex blocker. (a) Shadow volume from one vertex of the source. (b) The umbra is the intersection of the shadow volumes, while the penumbra is their convex hull.

These boundaries are sought for two purposes: First to speed-up shadow computation by performing complex calculations of the visible part of the extended light source only in the penumbra. Their second application is the subdivision of the scene polygons along umbra boundaries for high-quality shadow rendering in lighting simulations.

4.1 Umbra and penumbra caused by a convex blocker

The pioneering work by Nishita and Nakamae [NN85, NON85, NN83] studies the shadow caused by a convex blocker with respect to a convex polygonal light source. They note that the umbra is the intersection of the shadow volumes with respect to the vertices of the source. The penumbra is the convex hull of the union of these shadow volumes (Fig. 1.7).

These volumes were used to speed up soft-shadow computations. More efficient construction algorithms have later been proposed [Cam91, YKSC98].

4.2 Discontinuity meshing

Campbell and Fussell [CF90] were the first to subdivide a mesh along shadow boundaries. They approximate an area light source using point-samples and then use BSPs.

Heckbert [Hec92b, Hec92a] has introduced the notion of discontinuity meshing. He proves that at a shadow boundary a C^2 discontinuity in the illumination function occurs. C^1 discontinuities may also arise for degenerate (but common) cases. He considers discontinuities generated by the interaction of edge-vertex pairs belonging to the primary source and a blocker. These are called *EV* discontinuities.

Similar approaches can be found in [LTG93, LTG92, Stu94]. Note that Hardt and Teller [HT96] also consider some discontinuities in the indirect lighting.

4.3 Complete discontinuity meshing with backprojections

Teller [Tel92a], Drettakis and Fiume [DF94, DS96] and Stewart and Ghali [SG94] have extended discontinuity meshing to handle all possible C^2 discontinuities. This requires the treatment of shadow boundaries generated

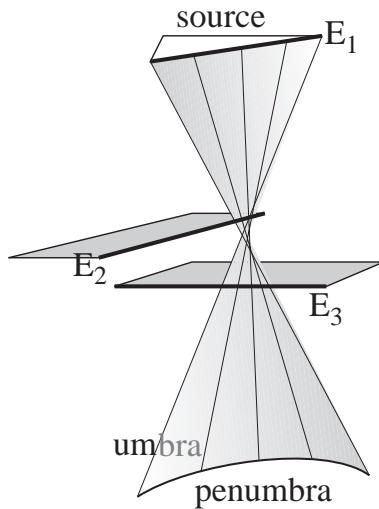


Figure 1.8: *EEE* shadow boundary. The interaction between E_1 , E_2 and E_3 determines the limit between the umbra and penumbra regions.

by the interaction of three edges, also called *EEE* (see Fig. 1.8). As opposed to *EV* boundaries, these are not line segments but conic curves.

Treating all discontinuities also permits a more precise classification of the penumbra region. Indeed, in each region bounded by visibility events, the visible part of the source is qualitatively invariant. The *backprojection* data-structure [DF94, DS96, SG94] encodes this structure of the visible part of the source. It allows fast and precise illumination calculations.

5 The Aspect graph

The *aspect graph* has been developed in computer vision for model-based object-recognition [Kv76, Kv79, EBD92]. It is a *viewer-centered* representation, meaning that it encodes informations on the possible views of the object rather than on its three dimensional structure. The principle is to partition the viewing space into regions where the view is qualitatively invariant.

If orthographic projection is used, a view is determined by the direction of projection: the viewing space is the direction sphere S^2 . In the case of perspective projection, the viewpoint can be any 3D point, the viewing space is thus \mathbb{R}^3 .

The changes in visibility are named *visual events*. Consider the case of a polygonal object. The interaction of an edge and a vertex is an example of visual event; A polyhedron A hidden behind another polyhedron B will become visible when a vertex of A is visually superimposed with a silhouette edge of B (see Fig. 1.9). Similarly, the conjunction of two edges can hide a third one, imposing the treatment of *EEE* visual events. Algorithms to compute the aspect graph of polyhedra include [GM90, GCS91, PD90].

These events are the same as those used in the discontinuity meshing literature. It is intuitive because they describe where the visibility of the source changes (visual events were first developed for aspect graphs and then adapted to shadow computation).

Curved objects can be handled as well. Visual events are described using singularity theory [Ker81, Rie92, Rie93, PPK92, Pet92]. See the second part of this thesis for more details.

Methods similar to the aspect graph have been used for robot self-localisation [GMR95, SON96, TA96] and for visibility based pursuit-evasion [LLG⁺97, GLL⁺97, GLLL98]

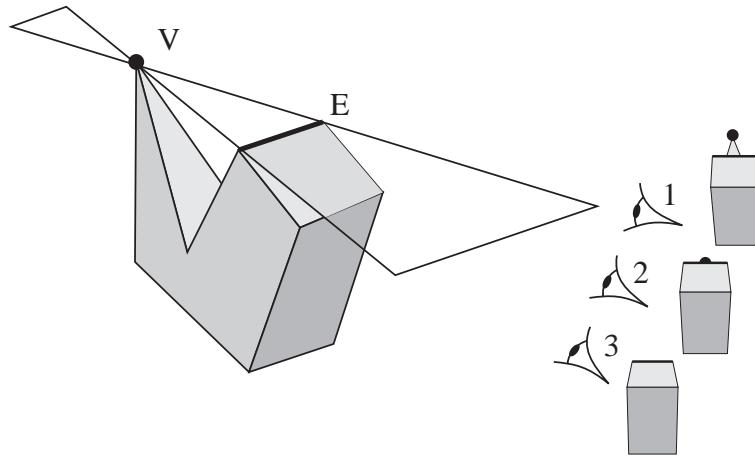


Figure 1.9: EV visual event. At viewpoint 2, vertex V becomes visible or hidden.

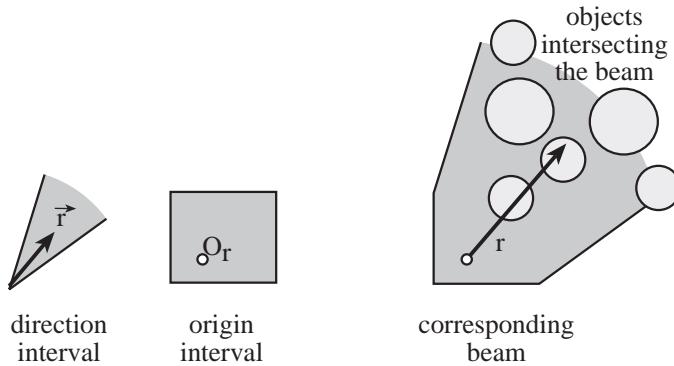


Figure 1.10: 2D equivalent of ray-classification. Ray r is defined by its origin O_r and its direction \vec{r} . It is tested only with the objects which intersect the beam defined by the intervals containing O_r and \vec{r} .

6 Line space

Visibility can be elegantly expressed in terms of lines or rays. We present approaches which perform computation directly in line or ray space.

6.1 Discrete approaches

Ray classification [AK87] has been developed to speed up ray-tracing. Ray-space is defined as a five dimensional space: three dimensions for the origin of a ray, and two for its direction. The space of rays is adaptively subdivided. An interval in ray-space defines a sort of beam in object space (see Fig. 1.10 for a 2D equivalent). Objects which potentially intersect the rays in the ray-interval are those which intersect the beam. When a ray is cast, intersections are computed only for the objects corresponding to its interval in ray-space.

Other approaches using a discretization of line or ray-space can be found in [LMW90, CCOL98, WBP98].

6.2 Plücker space

Plücker parameterization is a powerful duality which maps lines into a five dimensional space. Note that lines in space require only four dimensions, but no 4D parameterization is possible without singularities. The set of lines intersecting a given line is a hyperplane in Plücker space, which allows efficient stabbing computations. However, not all points in Plücker space correspond to a real line. Results obtained in Plücker space thus have to be intersected with the Plücker hypersurface which is the 4D locus of real lines.

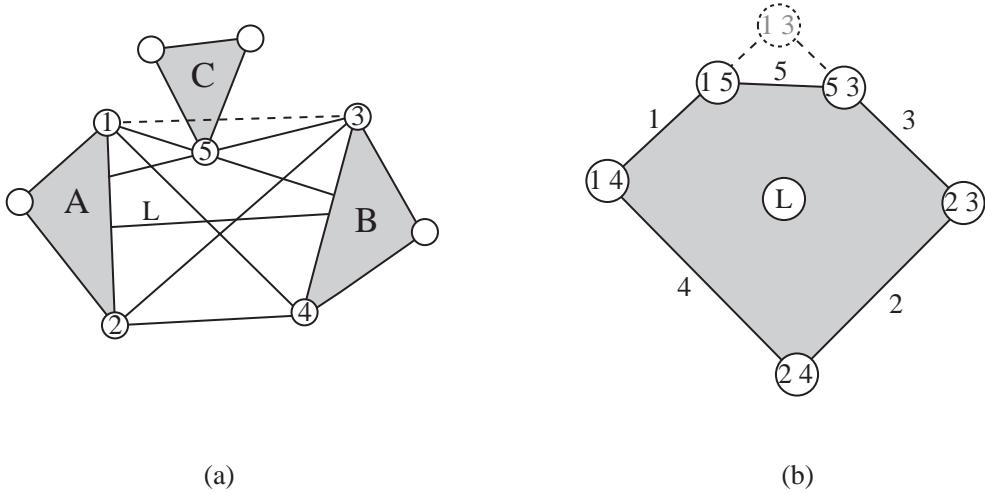


Figure 1.11: 2D visibility complex. (a) Scene. (b) Representation in a dual space of the face of the 2D visibility complex corresponding to segments between A and B .

Plücker coordinates have been used in computational geometry for example to compute lines stabbing a set of polygons or for ray-shooting [Pel93, PS92, CEG⁺96]. See [Pel97b] for a survey.

Teller [Tel92a] has developed and implemented an algorithm to compute the part of space visible through a sequence of polygonal openings. He builds the polytope in 5D space defined by the duals of the edges of the openings, and then intersects this polytope with the Plücker hypersurface. He obtains the set of *EV* and *EEE* events bounding the visible region. He however later noted [TH93] that this method is prone to numerical and degeneracy problems.

Teller and Hanrahan [TH93] also used Plücker coordinates to classify objects as visible, partially visible or hidden through a sequence of portals. This computation is more robust because no construction is performed, only predicates are evaluated.

6.3 The asp

The asp has been developed by Plantinga and Dyer [PD90] as an intermediate data-structure to build the aspect graph of polyhedra. They define the asp of a polygon as the set of lines intersecting it. They note that occlusions correspond to subtractions for the asp: if a polygon A occludes a polygon B , the asp of A has to be subtracted from the asp of B .

Two definitions of the asp exist. In the orthographic case, 4 dimensional line space is considered, while if perspective projection is used, 5 dimensional ray-space has to be considered.

The one-dimensional faces of the cellular decomposition in the orthographic case and the 2 dimensional faces in the perspective case correspond to the visual events required to compute the aspect graph. No full implementation is reported.

The use of the asp for view maintenance has also been proposed [PDS90, Pla93], but the implementation reported is limited to the rotation around one axis.

6.4 The 2D visibility Complex

Pocchiola and Vegter [PV96b, PV96a] have proposed the *visibility complex* of 2D scenes. It is based on the notion of *maximal free segments* which are segments whose extremities lie on objects of the scene and which do not intersect the interior of any object. They can be seen as rays which “see” in both directions. The visibility complex is a partition of the maximal free segments according to the objects at their extremities. The intuitive idea is to group all rays which see the same objects.

Consider the example in Fig. 1.11(a) which shows a simple scene composed of three triangles. A duality is used which maps lines to points (in our example, a line $y = ax + b$ is mapped to the point (a, b)). A face of

the 2D visibility complex corresponds to a set of segments which have two given objects at their extremities (for example, triangles A and B in Fig. 1.11(b)) Such a face is bounded by edges which correspond to segments going through a vertex of a polygon (or tangent to an object in the case of smooth objects). These edges are adjacent to vertices, which correspond to line going through two vertices of polygons.

Optimal constructions algorithms have been developed for curved [PV96b, PV96a] and polygonal objects [Riv95, Riv97a]. The 2D visibility complex can be used to compute and maintain views [Riv97c], or for 2D global illumination simulations [Ort97, ORDP96, DORP96].

7 Dynamic scenes

If objects in a scene move continuously, the visibility information will vary in a coherent manner. This is the so-called *temporal coherence*. Visibility is locally constant. We review here some methods which treat this coherence.

7.1 Shafts and motion volumes

A *motion volume* is similar to a shaft: it is the volume of space swept by a moving object in a given interval of time. Motion volumes have been used to limit the recomputation in radiosity updates [BWCG86, Sha97], in walkthroughs [SG96] and in sensor planning in robotics [AA95].

Shaw [Sha97] and Drettakis and Sillion [DS97] detect light transfers which need recomputation by testing the moving objects against the shafts joining the pairs of objects involved in the transfer.

7.2 BSP trees

Chrysanthou and Slater [CS92, CS95, CS97] update BSP trees by removing the dynamic objects, then re-inserting them. They mostly apply their technique for shadow computation with point and area light sources.

7.3 4D ray-tracing

Some approaches [Gla88, MDC93, BS96, Qua96, GP91] have been proposed to speed-up the ray-tracing of animated scenes where the motion of the objects is known in advance. Time is considered as a fourth dimension, and a ray-object intersection is valid for a certain time-interval.

7.4 Aspect graph and discontinuity meshing

Eggert *et al.* [EB93] propose the extension of the aspect graph to objects with moving parts. They note that the aspect graph changes when a temporal visual event is crossed, that is when the set of visual events of the scene is modified because of an *accidental configuration* of the object. Their analysis is however restricted to a simple case and no implementation is reported.

Loscos and Drettakis [LD97] and Worall *et al.* [WWP95, WHP98] take advantage of coherence in the update of discontinuity meshes. They note that the limits of penumbra and umbra move smoothly except at some events where a new shadow is cast on an object or when a shadow moves off an object. The treatment of these events for *EV* and certain *EEE* discontinuities is proposed.

8 Discussion

Despite the large amount of work dedicated to visibility, a number of questions remain without satisfactory answers. We have identified six major issues impeding the treatment of visibility. These are namely the understanding of visibility properties, the lack of general-purpose tools, the problem of visibility with respect to volumes, scalability, robustness and the treatment of dynamic scenes. We develop them below with previous work in mind.

- 3D visibility is poorly understood since most of previous theoretical work is restricted to the 2D case. Work in Plücker space, the aspect graph and the asp constitute first remarkable attempts to characterize global visibility of 3D scenes as a whole. Unfortunately the aspect graph seems restricted to the enumeration of all possible views –it is for example not suited to the extraction of certain kinds of mutual visibility information– and its storage complexity is large. The asp for orthographic projection does not encode information from within the scene, while its perspective version has redundancy because visibility does not usually vary for colinear rays. Visibility coherence is moreover not really explicit in the asp model. Plücker coordinates require five dimensions to parameterize a four dimensional space. Moreover, occlusion is not actually handled since the methods we have surveyed consider all intersections of lines.
- Though powerful techniques have been developed for specific cases, no general purpose tool has been developed. Some (mainly theoretical) work have shown that techniques in Plücker space have applications for different problems such as ray-shooting, mutual visibility or stabbing, but no implementation has been presented except the work by Teller on antipenumbra [Tel92a] and blocker classification [TH93].
- Although visibility from a point has received much attention, the extension to areas and volumes is more involved. While point-based occlusion culling techniques handle occluder fusion in general environments, cell-based methods are restricted to architectural scenes or to single convex occluders.
- The high cost of visibility calculations makes scalability issues critical. Quadratic or even linear growth are not acceptable in regard to the explosion of the size of 3D scenes. The resort to approximate calculations seems unavoidable, which raises the crucial point of error control. Some ideas have been proposed [SD95, SS96b, Sol98] but much remains to be done in this direction.
- Exact visibility computation such as the aspect graph or discontinuity meshing lead to involved geometric constructions prone to robustness problems. A major issue is the treatment of visibility events, since they define surfaces which can be ruled quadrics which have to be intersected with the scene. The issue of robustness and numerical precision lies beyond the scope of visibility alone and is relevant to any geometric method.
- Temporal coherence has received little attention compared to its potential benefits. Although everybody agrees that computation could be saved, the practical methods which have been proposed have not really capitalized on this evidence. They mainly consist in localizing a portion of space where computation has to be re-performed in the case of the motion of one object, or using the result of the preceding frame as an indication for the new calculation.

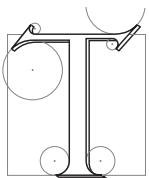
This thesis attempts to address these issues, focusing on the three first, even though the other points will also be explored to a lesser extent. Chapter 2 proposes an interpretation of the visibility properties of a 3D scene. A generic data-structure which encodes global visibility information is introduced in chapter 3 and applied to lighting simulation in chapter 4. Chapter 5 presents an efficient method to perform occlusion culling from a volume handling occluder fusion. The concluding chapter will propose a longer discussion of work which remains to be done.

CHAPTER 2

The 3D Visibility Complex

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*



HIS CHAPTER studies global visibility in line space. The most atomic visibility properties – mutual visibility of two points, ray-shooting– are naturally expressed in line-space. We thus explore the interpretation of line-object intersection in an appropriate dual space. Unlike standard ray-shooting acceleration techniques, our goal is not to study the fastest way to perform computations for each ray, but to envision the incidences between a scene and the entire set of rays in the scene to globally characterize visibility *coherence*.

Our general approach is to group the rays which “see” the same objects. This natural definition of coherence in visibility leads to the problem of defining the boundaries of such coherent sets. Their careful study proves fruitful for the understanding of many visibility properties. Particular attention will be given to the issue of line vs. ray visibility: whether to take into account all intersections of a line, or characterize only the first object intersected by a ray.

This discussion results in a new structure, which we call the *3D visibility complex*, which encodes all visibility information contained in a three dimensional scene. The approach is inspired by the equivalent 2D structure [PV96b, PV96a, DP95b], although the new approach has been developed from scratch with the specifically three-dimensional problem in mind. It will be used as a foundation for the practical tool we describe in the next chapter and apply to lighting simulation in chapter 4.

This chapter gathers material from two papers [DDP96, DDP97b]. A more detailed discussion and comparison with related work has been added (section 6). The catalogue of adjacencies has been added in appendix A as well as discussion of concave and smooth objects. In section 2.3 we have included a probabilistic study of the complexity of our structure in “normal” scenes, and in section 3 we discuss extensions of the 3D visibility complex to handle moving objects.

We first present the intuitions which have lead to the development of the structure. The definition is then formalized for scenes of polygons and smooth objects, and extended in the case of moving objects. An output-sensitive construction algorithm is presented as well as the interpretation of some visibility queries.

1 Introduction to the 3D Visibility Complex

In this discussion we will first consider scenes of general convex objects. We then generalize our approach to handle both smooth and polygonal objects. Visibility will be defined in terms of ray-object intersections. If we consider the objects to be transparent, a ray passing through them is not blocked and all the objects a line intersects must be considered. However, if we want to take occlusions into account, only the first object intersected by a ray is relevant. Nevertheless, considering rays induces redundant information since many colinear rays “see” the same object: Consider a ray r with origin A which intersects an object at point B (Fig. 2.1(a)). All the rays collinear to r with an origin A' between A and B “see” the same point B . We can thus group these rays into a segment S (Fig. 2.1(b)).

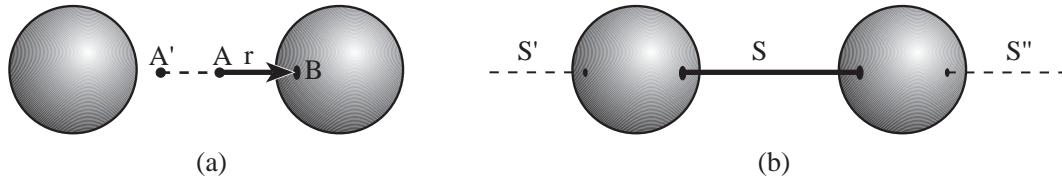


Figure 2.1: Maximal free segment. (a) All the rays collinear to r whose origin is between the two spheres “see” point B . (b) These rays are grouped into a *maximal free segment* S . Two other maximal free segments S' and S'' are collinear to S .

We will consider *maximal free segments* which are segments having no intersection with the inside of the objects and whose length is maximal (their two extremities lie on the boundary of two objects or are at infinity). In what follows we will often refer to them simply as *segments*. Examples of maximal free segments are given in Fig. 2.1(b). Segments can also be interpreted as rays which can *see* the two objects at their extremities. A 3D line can be collinear to many segments, separated by the objects the line intersects (see Fig. 2.1(b)). In this chapter, we will introduce concepts first in terms of line visibility (where all the objects intersected by a line are considered) and then in terms of segment visibility (where the occlusions are taken into account).

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

1.1 Duality

We use a duality which maps lines in 3D space onto points in a 4D space. We have chosen to decompose the 4 dimensions of line space into two dimensions of direction (the spherical coordinates (θ, φ) of the direction vector of the lines) and a projection (u, v) onto the plane perpendicular to the line and going through the origin. The axes of the planes are chosen such that u is along $\vec{t} \wedge \vec{y}$ ¹. The intersections of a line with two parallel planes could also be used. Nonetheless, we believe that such an approach makes the interpretation of lines sharing one coordinate harder, and as we will see, some visibility properties can not be easily described with this duality. We call the point in 4D space associated with a line the *dual* of the line.

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

Note that this duality will be used for illustrative and presentation purposes mainly. The concepts exposed in what follows are not inherent to any parameterization of lines (except when slices are involved). They could be exposed using only topological notions, although we believe that a dual space greatly helps their understanding.

¹Singularities occur at $\varphi = \pm\frac{\pi}{2}$, but since we use this duality for the purpose of presentation and visualisation we can ignore them without loss of generality.

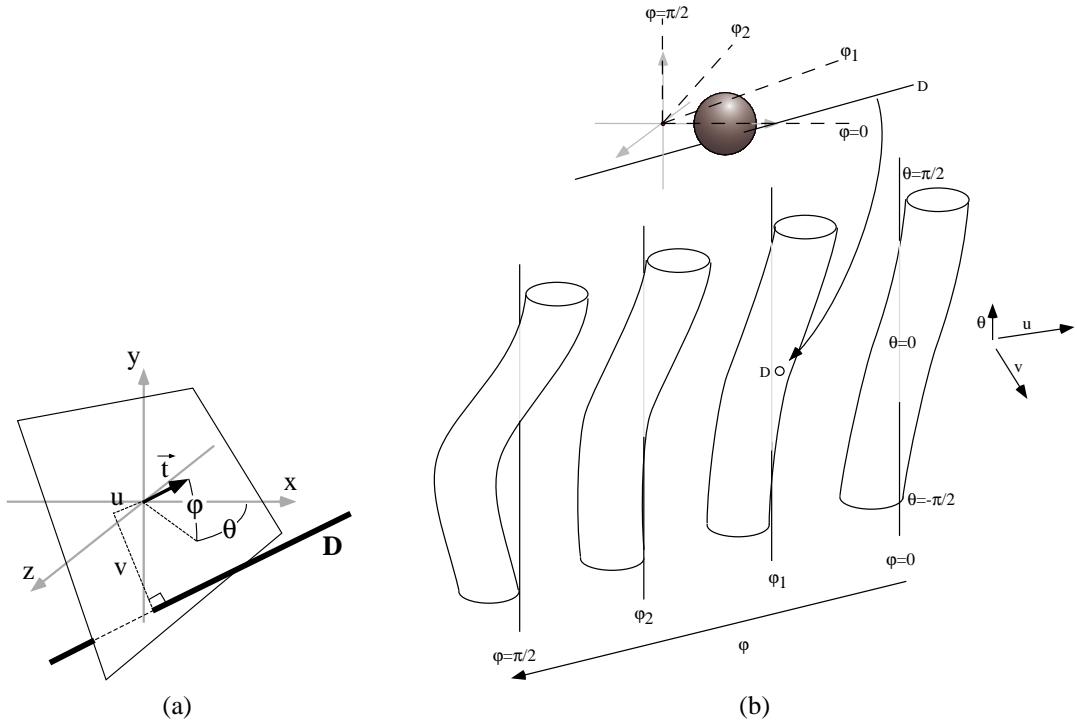


Figure 2.2: (a) Duality. (b) Tangency Volume of a sphere. The θ axis ($u = 0, v = 0$) is shown for each φ -slice providing a better 3D visualisation. In the left-hand φ -slice, which corresponds to the discontinuity in the duality for $\varphi = \frac{\pi}{2}$, the “cylinder” just turns around the θ axis. The line D intersects the object and has its dual inside the tangency volume.

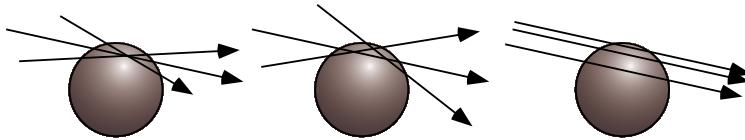


Figure 2.3: To remain tangent to an object, a line has three degrees of freedom. (a) Rotation along the contact point. (b) Rotation backward or forward. (c) Side translation.

1.2 Tangency volumes

Line visibility

Visibility changes whenever a line becomes tangent to an object. That is, a line which is tangent to an object is the limit between lines which intersect the object and lines which do not intersect it.

The set of lines tangent to one object is a 3-D set in the 4D dual space. This means, more intuitively, that a line has 3 degrees of freedom while staying tangent to one object (see Fig. 2.3). We will call the dual of the set of lines tangent to an object the *tangency volume* of this object.

Fig. 2.2(b) shows a representation of the tangency volume of a sphere. For each φ -slice, the set of tangents is a sort of 2D “tube”, forming a 3D structure in the 4D dual space. If we consider a 2D $\theta\varphi$ -slice (horizontal in Fig. 2.2(b)), the set of tangents sharing that direction is a circle in the dual space. This is general: because of the definition of u and v , the set of tangents to one object in one direction is the silhouette of the object in this direction. It is also called its *fold*.

If a line has its dual on the tangency volume, it is tangent to the object. If the dual is inside the 4D set bounded by the tangency volume, it intersects the object, similarly to line D on Fig. 2.2(b).

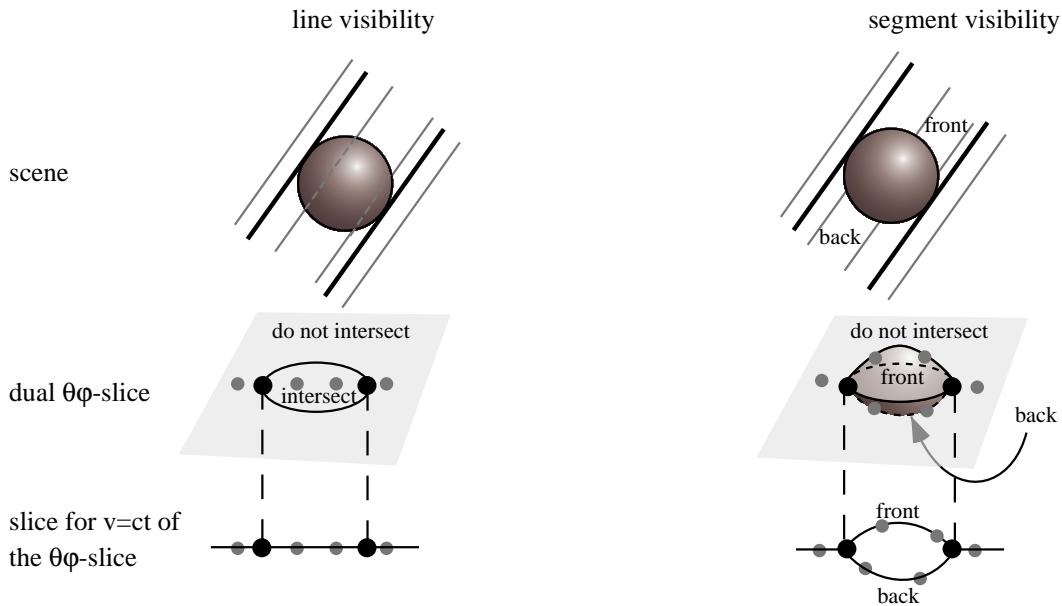


Figure 2.4: Visibility for $\theta = ct$ and $\phi = ct$. If we consider lines (on the left), visibility can be described by a planar structure (below). But if we consider segments (on the right) we have different levels on this plane depending on the side of the object. The set of segments which do not intersect and the sets of those that intersect the front or the back of the object share the same boundary, the tangents to the object which correspond to its silhouette. Recall that the Auxiliary Complex shown on the lower right is a 2D structure embedded into 3D, *i.e.* it is “empty”, since the points outside the surfaces have no meaning.

Segment visibility and auxiliary complex

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

Consider a $\theta\phi$ -slice such as that on the lower left of Fig. 2.4. The sets of lines that intersect, and those that do not intersect the object are bounded by the silhouette of the object. For segment visibility we have to consider the segments that see the front of the object and those that see its back. Since such segments are collinear to the same line, they are projected on the same point in the 4D line dual space. Consequently the set of segments that see the front and the set of segments that see the back of the object are projected onto the same position of the 4D dual space as shown in the right of Fig. 2.4. The silhouette, which is the set of tangents to the object for the chosen θ and ϕ , is incident to the three sets (front, back and no intersection). This means that a segment tangent to the object has topological neighbours that do not intersect the objects, some that see the front, and some that see the back.

To differentiate the segments, we add a pseudo-dimension. It is not a continuous dimension since we just have to sort all the collinear segments. If we impose $\theta = ct$, $\phi = ct$ and $v = ct$, the sets of segments can be represented by a graph shown on the lower right (it is in fact an embedding of a graph since the points on the arcs also have a meaning). Each tangent corresponds to a vertex of the graph. This graph is a 1D structure embedded in 2D. Similarly, for a $\theta\phi$ -slice, the sets of segments are represented by a 2D structure embedded into 3D. We call the partition of the segments of direction (θ, ϕ) according to their visibility the *auxiliary complex* for (θ, ϕ) (see also Fig. 2.6). It can be seen as a generalized orthographic view where all the objects, hidden or visible, are organised into layers.

In a similar manner, a ϕ -slice is in fact a 3D structure embedded into 4D, and the set of segments is a 4D space embedded into 5D. It is a 4-manifold² embedded in 5D.

²A n -manifold is a set for which the neighbourhood of each point is homeomorphic to \mathbb{R}^n . The space of rays is actually non-manifold because of the branchings of the tangency volumes. Non-manifolds are often also called manifolds for simplicity.

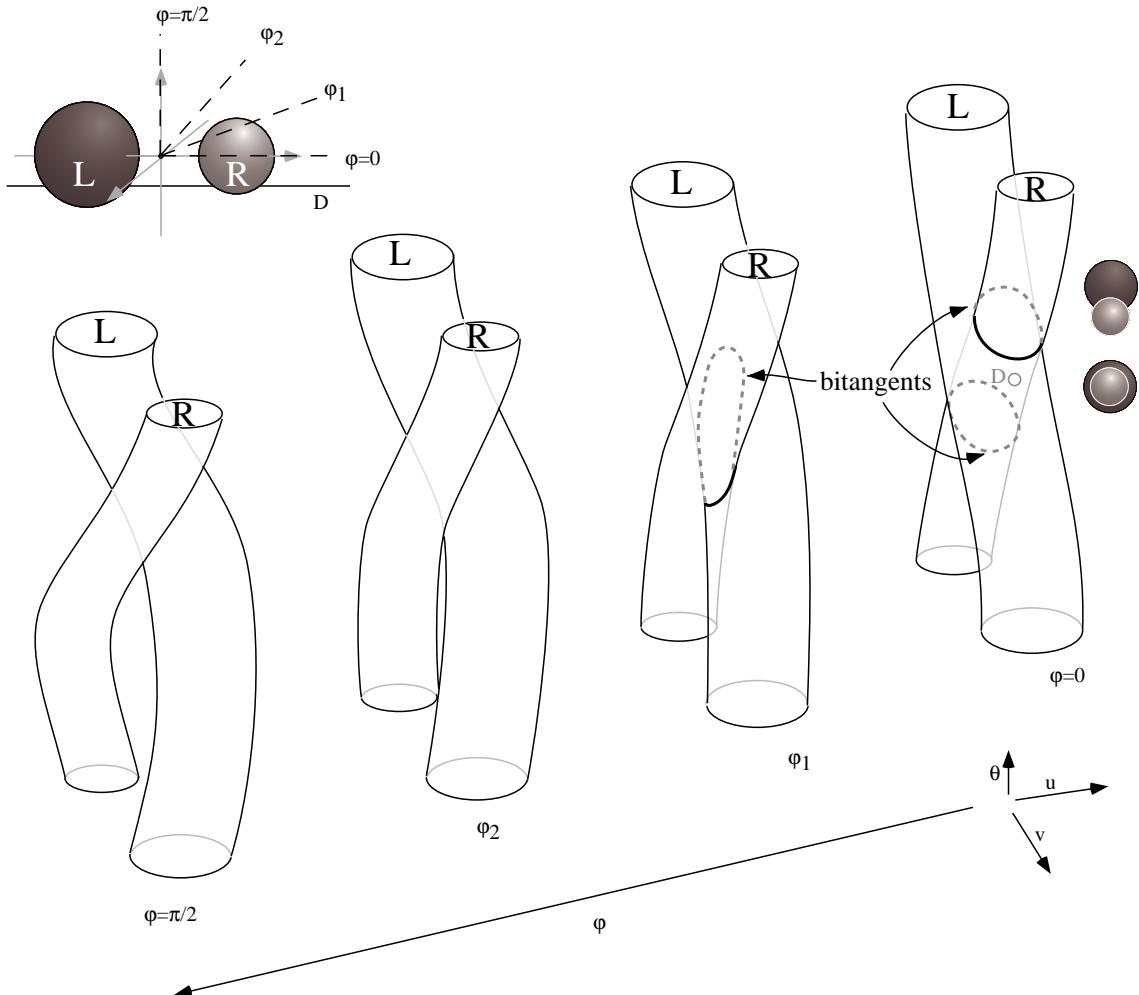


Figure 2.5: Dual arrangement for two spheres.

1.3 Bitangents

Line visibility

Now consider two objects. If a line has its associated dual point inside the tangency volumes of both objects, it intersects them both. The tangency volumes give us a partition of the dual space of the 3D lines according to the objects they intersect. We call this partition the *dual arrangement*. Its *topological faces* are 4D sets of lines which intersect the same objects. They are bounded by portions of the tangency volumes which are 3D. The intersection of two tangency volumes is a 2D set corresponding to the lines tangent to the two objects (bitangents). A bitangent corresponds to a *t-vertex* in an image (that is, the visual intersection of two object silhouettes).

For a φ -slice the set of bitangents is a space curve (shown as a dashed line in Fig. 2.5 on the two φ -slices on the right). It corresponds to the intersection of the two “cylinders” which are the φ -slices of the tangency volumes. The slice of a 4D face is a volume corresponding to the intersection of the inside of the two cylinders.

Segment visibility

An auxiliary complex for two objects is shown on Fig. 2.6 for a given direction. It is still delimited by the silhouette of the objects, but for example the silhouette of the upper sphere has no influence on the set B of segments that see the back of the lower sphere. Note that the two bitangents (shown in fat black lines) are

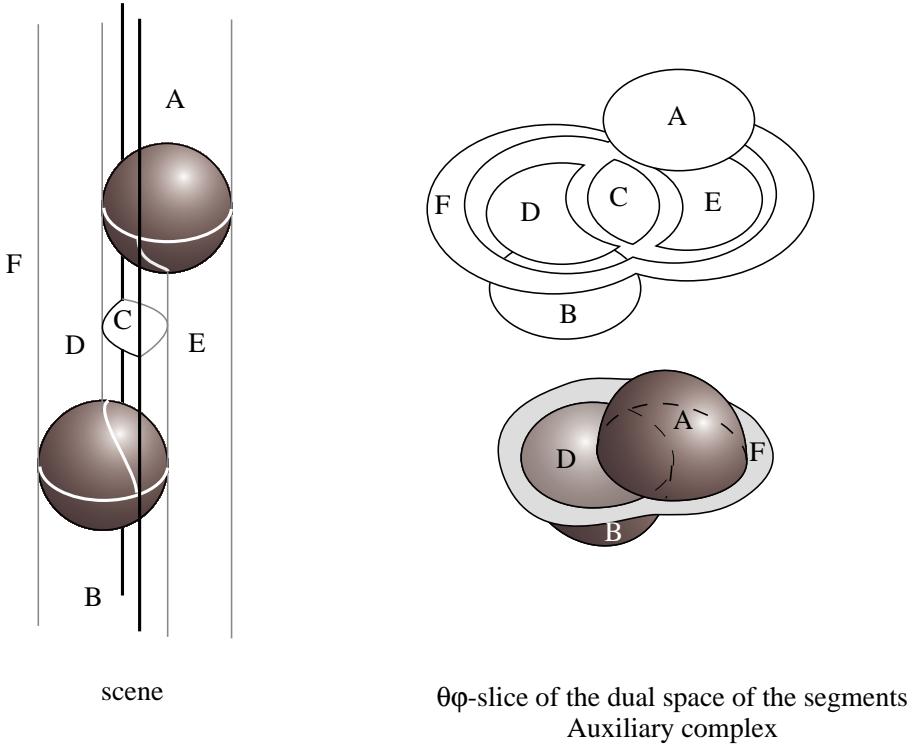


Figure 2.6: Auxiliary Complex for two spheres. Recall that the auxiliary complex is a 2D structure embedded in 3D. In the lower representation, only the points on the surfaces represented are associated with segments. In the upper view, the faces of the auxiliary complex have been moved out to make their incidences easier to understand. Face F is infinite.

incident to all faces.

The *3D visibility complex* is the equivalent of the dual arrangement for segment visibility. It is the partition of segments according to the objects at their extremities.

Fig. 2.7 is a φ -slice for $\varphi = 0$ of all the faces of the 3D visibility complex for the scene composed of two spheres of Fig. 2.5. The view in a given direction is shown on the left of the cylinders, and we consider the associated auxiliary complex shown six times on the top of the schema. Each time, a face is hatched and a volume is drawn below which corresponds to the φ -slice of the face of the visibility complex at $\varphi = 0$. Note that the union of these volumes is more than the entire 3D space, since a φ -slice of the complex is a 3D structure embedded into 4D which has branchings at the tangency volumes.

1.4 Tritangents

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

Fig. 2.8 shows part of the visibility complex of a scene of three spheres. On the φ -slice $\varphi = 0$ two orthographic views of the scene for $\theta = 0$ (View 0) and for $\theta = \theta_2$ (View 2) are drawn next to the corresponding θ in the φ -slice. The set F of segments that see the spheres R and B is shown by its two slices F_0 and $F_{\varphi 1}$. Note that it is the intersection of the tangency volume of R and B minus the tangency volume of G . The tritangents are the points in white. Note also that because of the occlusion by the sphere G , lines that are bitangents of the R and B do not correspond to bitangent segments. This is shown in Fig. 2.9 which is a zoomed view of the φ -slice $\varphi = 0$. The set of bitangents B_0 is cut because bitangent lines such as D intersect G and correspond to no bitangent segment. We can thus see that the tritangent T_0 and T'_0 are the intersection of the φ -slices of the

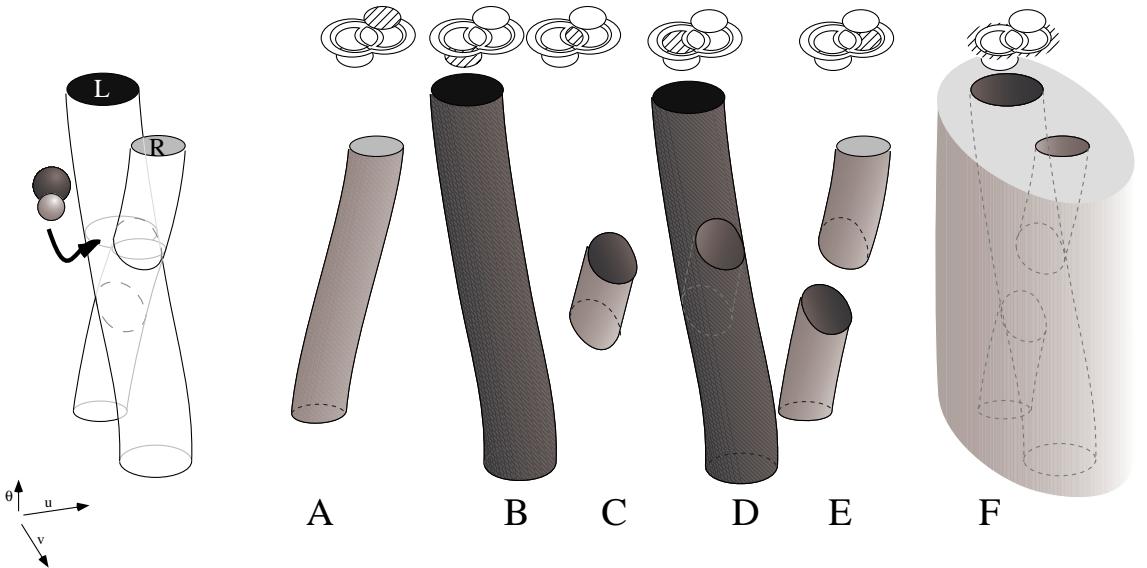


Figure 2.7: φ -slice for $\varphi = 0$ of the faces of the visibility complex of the previous scene. A is the set of segments that see the front of R , B is the set of segments that see the back of L . C is the set of segments between L and R . It can be interpreted as the intersection of set of lines that see L and the set of lines that see R , and in the dual space it has the shape of $A \cap B$. D is the set of segments that see the front of L . Since the visibility is occluded by R in this direction, D has the shape of $B - A$. Similarly, E is the set of segments that see the back of R . Finally, F is the set of segments that see none of the two spheres. It is the complement of $A \cup B$.

three tangency volumes, and are also incident to the three sets of bitangents B_0 , B'_0 and B''_0 .

Tritangents are an example of the so called *visual events* or *visibility events*. Visual events describe the *qualitative* (or topological) changes in visibility. Consider the example in Fig. 2.10. As the viewpoint moves downwards, sphere A becomes hidden by the conjunction of B and C . This occurs when the viewpoint lies on a tritangent.

Note that a scene does not necessarily contain tritangents in the general case (this is for example the case in Fig. 2.14(a) page 40).

1.5 Tangent crossing

There is another sort visual event called a *tangent crossing*. It corresponds to planes tangent to two objects, and to the lines going through the corresponding two points of tangency. Consider the case of a sphere hidden behind another sphere. If the viewpoint is moved, the sphere will become visible when the two spheres are visually tangent. The line going through the viewpoint and the point of visual tangency is contained in a plane tangent to the two spheres.

How is this interpreted in our dual space? These tangent crossing critical lines are subsets of bitangents. They correspond to minima or maxima with respect to θ in the φ -slices. To understand why, remember that a (φ, θ) -slice of the complex corresponds to an orthographic view. Consider a rotation at $\varphi = ct$ starting at a direction for which the two objects are distinct in the view. Bitangents (t-vertices) will appear in the orthographic view at a tangent crossing. The tangent crossing thus corresponds to the first θ at which a set of bitangents starts.

The same reasoning can be applied to show that tangent crossings are also minima or maxima in θ -slices. Note that if occlusion occurs because of a third object, the tangent crossing can be discarded and the minimum or maximum of a bitangent set can be another event, as is the case in Fig. 2.9 where the tritangent T'_0 is the maximum of the set B_0 .

Tangent crossings, as all visual events, have dimension 1. These events are crucial for dynamic maintenance of views, aspect graphs and discontinuity meshes.

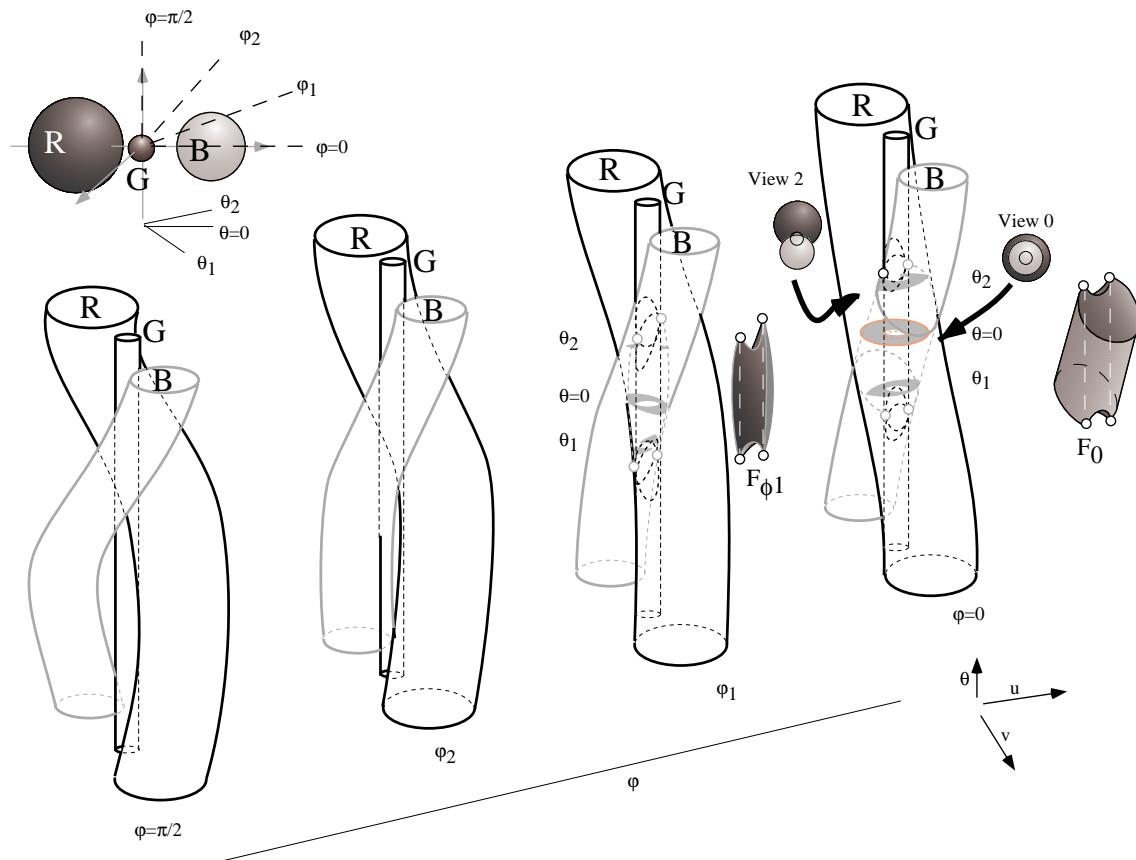


Figure 2.8: Visibility Complex of a scene of three spheres.

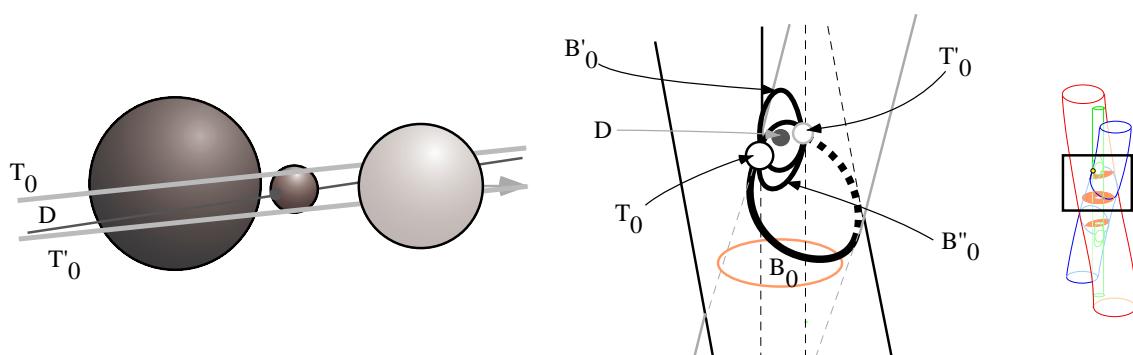


Figure 2.9: Zoomed view of the ϕ -slice $\phi = 0$.

1.6 The 3D Visibility Complex

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

Similarly, the *3D visibility complex* is the partition of the maximal free segments of 3D space into connected components according to the objects they touch. It is a 4D structure embedded into 5D. The dimensions and incidences of the boundaries of the faces are summarised in table 1.6.

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

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

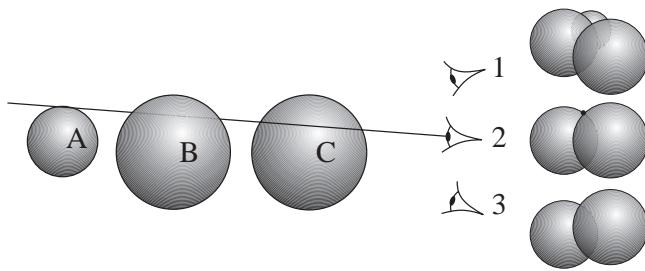


Figure 2.10: Tritangency visual event. At viewpoint 2, sphere A becomes hidden by sphere B and C . This occurs as the viewpoint lies on a line tangent to the three spheres.

Dimension	Scene configuration	φ -slice in the dual space	Name
4			face
3			tangency face
2			bitangency face
1			tritangency arc
1			tangent crossing
0			vertex

Table 2.1: Elements of the visibility complex

2 A definition for scenes of polygons and smooth objects

We now consider scenes of polygons and algebraic smooth convex objects. Concave objects and piecewise smooth objects will be discussed in appendix A. The algebraic objects are assumed to have bounded degree. In what follows, n represents the overall complexity of the scene which is the total number of objects, polygons, edges and vertices. The objects are assumed to be in general position; degeneracy issues are not addressed here.

We require the objects to be algebraic only to derive bounds on the complexity of our approach; The concepts which we derive are valid for all classes of smooth convex objects.

2.1 Critical segments

We define a segment to be in general position if it touches objects only at its extremities. A segment that touches objects in its interior will be called *critical*. At such an intersection there is a *local event*. If a segment touches more than one object in its interior, we call this a *multilocal event*. Critical segments are grouped into *critical segment sets*. We have seen that the dimension of such a set is the number of degrees of freedom of the set of segments which verify the event. We can also refer to the *codimension* of such a set, which is the complement to the dimension of the space (the number of fixed degrees of freedom)³. Codimensions are convenient because as we will see they are additive.

For the class of scenes we consider, there are two kinds of local events: tangency events (Fig. 2.11(a)) and vertex events (Fig. 2.11(b)). The object or the vertex are called the *generators* of the event. To stay tangent

³The codimension we exhibit are different from those found in aspect graph literature where the total space has a different dimension. In the case of orthographic projection, viewpoint space has dimension 2. Tangents are thus stable (in any view there is a tangent), and have codimension 0.

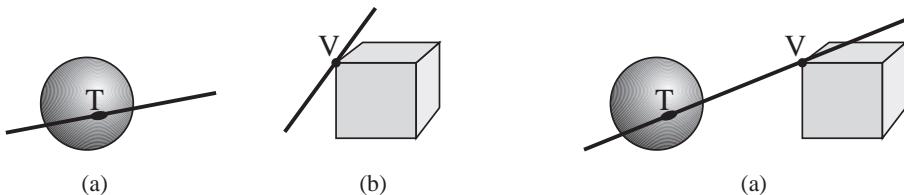


Figure 2.11: Critical segments. (a) Tangency local event. (b) Vertex local event. (c) $T+V$ multilocal event.

to a 3D object, a segment has three degrees of freedom. It is of codimension 1. It is of course the same when a segment goes through the edge of a polygon. We call this a T event for tangency (referred to as E from edge when only polygons are considered). A segment that goes through a vertex has two degrees of freedom (rotation), and thus has codimension 2. We call it a V event.

The combination of many local events causes a multilocal event, and the codimensions are added. We use the notation $+$ to describe such a combination. For example, a segment that is tangent to an object and that goes through a vertex belongs to a $T+V$ critical segment set of codimension $1+2=3$ (it is a 1D set) as illustrated in Fig. 2.11(c).

We have seen that at a *tangent crossing* a segment is tangent to two objects and belong to one of their common tangent planes. In this case, the common tangent plane adds one codimension and we use the notation $++$ (following [Ker81, Pet92]). For example $T++T$ critical segment sets have codimension $1+1+1=3$ (1D set)⁴.

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

Events caused by faces are considered as $T+T$ events since they involve two polygon edges. In the same way, segments going through a polygonal edge are $V+V$ events. The reason why the case of vertices (which could be seen as two edge events) is distinguished is that they introduce “discontinuities” at the end of polygonal edges and require a specific treatment as we shall see in section 4.4. They are in fact degeneracies of smooth objects.

2.2 The 3D Visibility Complex for scenes of polygons and smooth objects

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

The different faces of lower dimension correspond to critical segments as summarized in table 2.2.

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

The $\Omega(n)$ bound is trivial since there exist at least one tangency volume for each 3D object. The $O(n^4)$ bound comes from the number of $T+T+T+T$ events. However, we have to prove that the size of the complex is dominated by their number (for example, the size of a tetrahedralisation is not necessarily dominated by the number of vertices, it can be quadratic [BY98], chapter 13). For this purpose, we first prove that the number of faces of dimension 4, 3 and 2 is bounded by a constant factor of the number of faces of dimension 1.

Proof

The number of $(k+1)$ -faces adjacent to a k -face is bounded. For example a 1-face $T_1+T_2+T_3$ is adjacent to five 2-faces: two faces T_1+T_2 (there are two different faces because one extremity of the segments can lie on the object tangent at T_3 or not. See Fig. 2.12), T_1+T_3 and two T_2+T_3 . The other adjacencies are summarized in appendix A.

⁴One may think of the example of two parallel cylinders and notice that lines contained in a bitangent plane have two degrees of freedom. This case is not considered here because it is degenerate.

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

Table 2.2: Faces of the visibility complex of polygons and smooth objects.

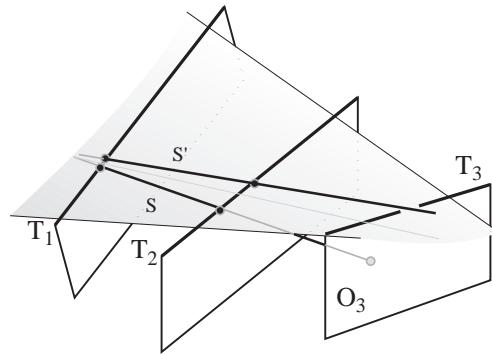


Figure 2.12: Adjacencies at a $T_1 + T_2 + T_3$ face. Two different $T_1 + T_2$ bitangent are adjacent: one with extremity O_3 (such as S) and one lying above T_3 (such as S').

Each 4-face is adjacent to at least one 3-face, a 3-face to at least one 2-face, and a 2-face to at least one 1-face. However, a 1-face may not be adjacent to any 0-face. We sketch the demonstration. For a given face F of the complex (composed of segments), we consider the associated critical line set S : that is, the set of lines with the same tangency properties, a $T_1 + T_2$ face is associated with the set of all lines tangent to T_1 and T_2 , whatever their other intersections. This set of lines contains a line set S' with one more codimension (one of the lines tangent to one object is also tangent to a second object, one of the lines tangent to two objects belongs to one of their common tangent planes, and one line going through a vertex is tangent to an object). Consider a continuous path from the line associated with a segment s of F to one of S' , and the corresponding continuous path over the segments (the notion of corresponding path in segment space is not always properly defined because of the branchings due to tangency. In our case however, if tangency occurs, one codimension is

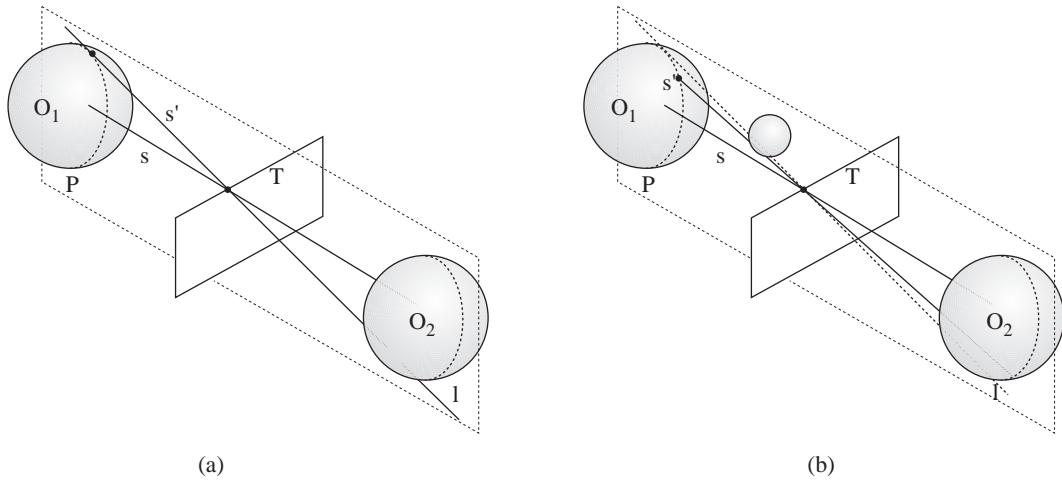


Figure 2.13: Construction of a segment of a $T + T$ 2-face adjacent to a T 3-face.

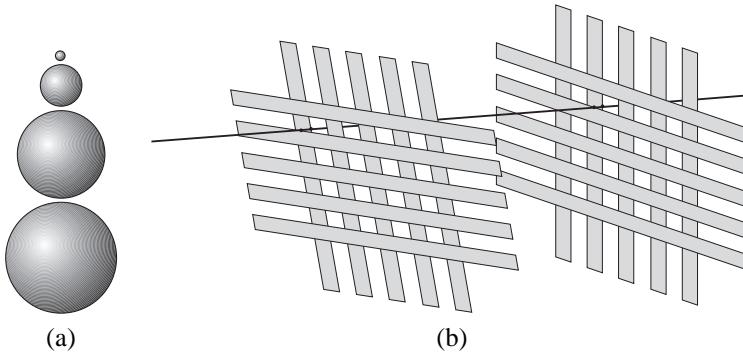


Figure 2.14: Lower and upper bound scenes for the visibility complex (a) Scene with an $O(n)$ Visibility Complex. (b) Scene with an $O(n^4)$ Visibility Complex (an example of $T + T + T + T$ critical segment is shown).

added and we have our adjacent face). If all the segments of this path have the same extremities, F is adjacent to the face with one more codimension associated with S' , otherwise when the extremity changes there is a tangency local event and one more codimension.

Consider for example a T 3-face and a maximal free segment s of this face. s is supposed to have no extremity lying at infinity. Consider a plane P containing s . We define l to be the first line (in a polar order) tangent to both the object at T and the object O_1 at one of the extremities of s (see Fig. 2.13). Consider a continuous path $l(t)$ in the line space from the line defined by s to l . A corresponding continuous path can be defined over the segments by considering for each $l(t)$ the segment $s(t)$ tangent to the object at T . If $s(t)$ has always the two same objects at its extremities, it belongs to the same T 3-face as s and this two face is thus adjacent to a $T + T$ 2-face (Fig. 2.13(a)), otherwise when the object at one extremity changes the segment is tangent to two objects and there is also a $T + T$ 2-face (Fig. 2.13(b)). The other cases can be demonstrated in a similar manner.

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

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

We show in figure 2.14(a) an example of a scene with a visibility complex of size $O(n)$: there is one $T + T$ face for each pair of neighbour spheres. Note there is no $T + T + T$ -face in this case because of the decreasing radius of the spheres. There is no 0-face because there can be no $T + T + T$ nor $T + T + T$ segment. The scene in figure 2.14(b) is the same as in [PD90] and has an $O(n^4)$ visibility complex. There are two “grids”, each one composed of two very slightly perturbed (to avoid degeneracies) orthogonal sets of $\frac{n}{4}$ parallel

rectangles (this is also valid with thin ellipsoids). Consider a rectangle in each of the four sets: there is always a $T + T + T + T$ critical segment.

2.3 Probabilistic complexity

In many cases, theoretical bounds unfortunately do not tell much about the practical complexity of a problem or method in the case of “normal” scenes. Some probabilistic approaches using a model of “reasonable” scenes in the spirit of de Berg *et al.* [dBKvdSV97] should be made.

As a first step, we propose to study the practical number of $T + T + T$ line sets in a scene. The $O(n^3)$ bound is based on results on infinitely thin objects, and is not realistic as will be illustrated in the next chapter. In typical scenes, the size of the object is bounded, and the complexity is increased by adding smaller details or by placing many objects next to each other, not by adding many interleaved infinite lines! We propose here a simplified probabilistic model for these “normal” scenes, and show that under our assumptions the number of $T + T + T$ events is $O\left(n^{\frac{7}{3}}\right)$. This bound gives a better intuition of the actual complexity of normal scenes.

We consider a scene model where objects of bounded extent are uniformly distributed inside a finite sphere. R is the diameter of the scene, and r is the maximum diameter of an object. The density of objects is constant as their number varies: If n is the number of objects we have $R = \Theta(\sqrt[3]{n})$. We will study the mean number of $T + T + T$ events. For this we will first study the probability that, given two objects, a third object generates a $T + T + T$ event with them. We will show that this can be expressed as a volume ratio. We will then integrate over all objects to get a bound on the mean total number of $T + T + T$ events. To simplify our calculation, we use bounding spheres enclosing the objects.

Probability given two objects

Consider two objects A and B separated by a distance x . We want to obtain a bound on the probability $P_{AB,x}$ that a third object C generates a $T + T + T$ event with A and B . This will happen only if the sphere bounding C lies inside the hourglass volume formed by two cones and a cylinder tangent to the two spheres enclosing A and B (Fig. 2.15(a)). The probability that a sphere of radius r intersects this volume is equal to the probability that the center of the sphere intersects an hourglass volume dilated by r . Since we assume that the centers are uniformly distributed, this probability is equal to the ratio between the volume of the dilated hourglass and the volume of the sphere bounding the scene.

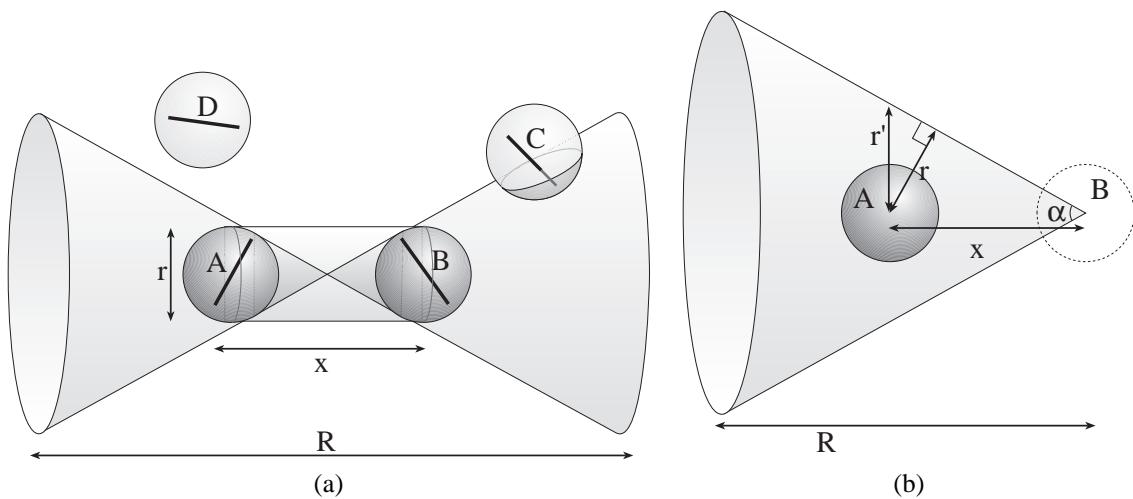


Figure 2.15: Probabilistic complexity of $T + T + T$ events. (a) Hourglass volume. (b) Construction of one cone of the dilated hourglass.

We now compute the volume of the dilated hourglass. Recall that x is the distance between the centers of the bounding spheres of A and B . We distinguish the case where $x \leq 2r$ and the case where $x > 2r$. In the first

case we trivially bound the volume by the volume of the scene sphere, *i.e.* $P_{AB,x \leq 2r} = 1$. We will see that this case is asymptotically negligible.

In the latter case, we bound the volume of the dilated hourglass by the sum of a cylinder of length x and diameter $2r$ and two cones of height R , defined as shown in Fig. 2.15(b). The volume of the cylinder is $\text{vol}_{\text{cylinder}} = \pi r^2 x$.

Let us now bound the volume of one cone. Simple trigonometry gives:

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

r' is a decreasing function of x , we will bound it by its value when $x = 2r$, that is lower than $1.16r$. The volume of one cone is:

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

The total volume of the dilated hourglass is then bounded by:

$$\begin{aligned} \text{vol}_{\text{hourglass}} &\leq 2\text{vol}_{\text{cone}} + \text{vol}_{\text{cylinder}} \\ &\leq 2 \frac{1.35 \pi r^2 R^3}{3x^2} + \pi r^2 x \end{aligned}$$

We divide by the volume of the scene sphere to obtain a bound on the probability:

$$\begin{aligned} P_{AB,x} &\leq \frac{\text{vol}_{\text{hourglass}}}{\text{vol}_{\text{sphere}}} \\ &\leq \frac{5.4r^2}{x^2} + \frac{6r^2 x}{R^3} \end{aligned}$$

Probability for all the objects

We now have a bound on the probability, given a pair of objects at distance x that a third object C generates a $T + T + T$ event with them. We have to compute the total mean number of $T + T + T$, that is consider every possible C , and every possible pair A, B at every possible distance x .

We multiply by n to obtain the mean number of $T + T + T$ events generated by a given pair A, B and all the other objects.

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

Until now, we have considered that the distance x between A and B is fixed. Consider now a random pair A, B . We have to take into account the probability that they lie at distance x . Using a result of integral geometry ([San76] page 212) on the probability distribution of distances of pairs of points inside a sphere, we have

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

where $\lambda = \frac{x}{R}$ (that is, the probability that two random points in a sphere lies at a distance between x and $x+dx$ is $\mu(x)dx$).

The intuition behind this formula is that, once one point is fixed, the second point is at a distance between x and $x+dx$ if it lies between the two corresponding spheres (Fig. 2.16). The surface of this sphere explain the term λ^2 . However, this sphere has to be intersected with the sphere bounding the scene, which explains the following polynomial term.

To obtain the total mean number of $T + T + T$ we integrate over x and multiply by the number of pairs of objects, n^2 . Recall that we have distinguished the case $x \leq 2r$ where the probability is bounded by 1 and the case $x > 2r$ where the probability is given by a ratio of volumes.

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

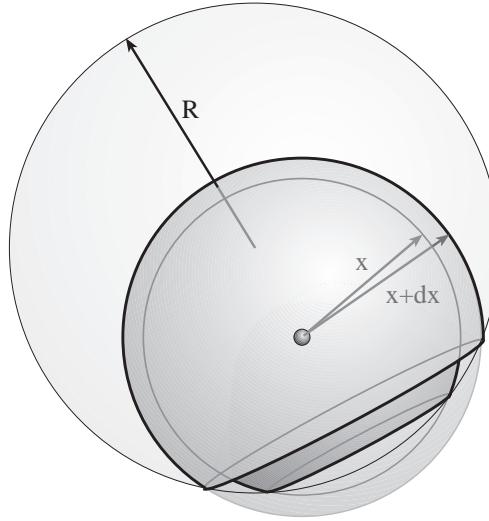


Figure 2.16: Probability distribution of distances of pairs of points inside a sphere.

We now derive bounds on the two integrals in inequality 2.1. The first integral is:

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

The second integral in inequality 2.1 is bounded by:

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

Expanding the first term inside the integral gives:

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

The second term in inequality 2.3 gives:

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

Combining the results of equations 2.1, 2.2, 2.4 and 2.5 we obtain:

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

The mean number of $T + T + T$ events is thus $O\left(n^{\frac{7}{3}}\right)$ (that is, $O(n^{2.33})$) in our model instead of $O(n^3)$ when objects of unbounded extent are considered. We will see in the next chapter that on the scenes we have tested, the number of $T + T + T$ events is about quadratic, confirming our probabilistic bound.

Similar arguments can be used to show that the number of $T + T + T + T$ 0-faces is $O\left(n^{\frac{8}{3}}\right)$ (i.e., $O(n^{2.67})$) instead of $O(n^4)$.

3 Extension to temporal visibility

In this section we describe the extension of the Visibility Complex to dynamic scenes. We show that a similar framework permits the description of all changes in visibility as objects move smoothly. We consider any smooth movement of the objects (including deformations), and all objects can be dynamic. The temporal dimension is considered continuous, as opposed to the regular succession of frames usually used for animation (even though a representation at certain time steps will be necessary for clarity).

3.1 Temporal visual event

We adopt the same approach as for the static case: we try to determine when visibility changes in a dynamic 3D scene. We are interested in the topological changes of the visibility complex as the objects move. Consider the situation depicted in Fig. 2.17(a). Sphere A moves upward. At time t_1 , there is no segment with extremities on A and C because of the occlusion by B. At time t_3 , some segments see A and C because A has moved. The limit configuration occurs at time t_2 where a plane is tangent to the three spheres. Segment $A++B++C$ corresponds to this tritangent plane. It is a *temporal visual event*. It is similar to the 0-faces of the static complex, with one codimension added.

In Fig. 2.17(b) we represent the 0 and 1-dimensional faces of the static visibility complex for some timestep. At time t_1 the complex only has two 1-faces and no 0-face. At time t_3 six $T++T+T$ 0-faces have appeared, as well as some new $T++T$ and $T+T+T$ 1-faces. There has been a topological change in the structure of the 3D visibility complex.

3.2 The Temporal Visibility Complex

We add the temporal dimension to the space of maximal free segments. A temporal segment is defined as a maximal free segment in space together with a time value. Temporal segment space is thus a 5D manifold embedded in 6D.

We define the *Temporal Visibility Complex* as the partition of the temporal maximal free segments according to the object that they touch.

The k-faces of the Temporal Visibility Complex correspond to the (k-1)-faces of the static Visibility Complex. However, their codimensions are preserved (since temporal segment space has one more dimension, and so have the faces). The static Visibility Complex is in fact a time-slice of the Temporal Visibility Complex. The 0-faces of the Temporal Visibility Complex correspond to the *temporal visual events*. These correspond to changes in the topological structure of the visibility complex. Maintaining the Visibility Complex as objects move is equivalent to a time-sweep of the Temporal Visibility Complex.

Fig. 2.17(c) shows the 0, 1 and 2-faces of the temporal complex for the scene in Fig. 2.17(a). At the temporal visual event, the slice of the temporal complex is modified, new $T+T++T$ faces appear as well as

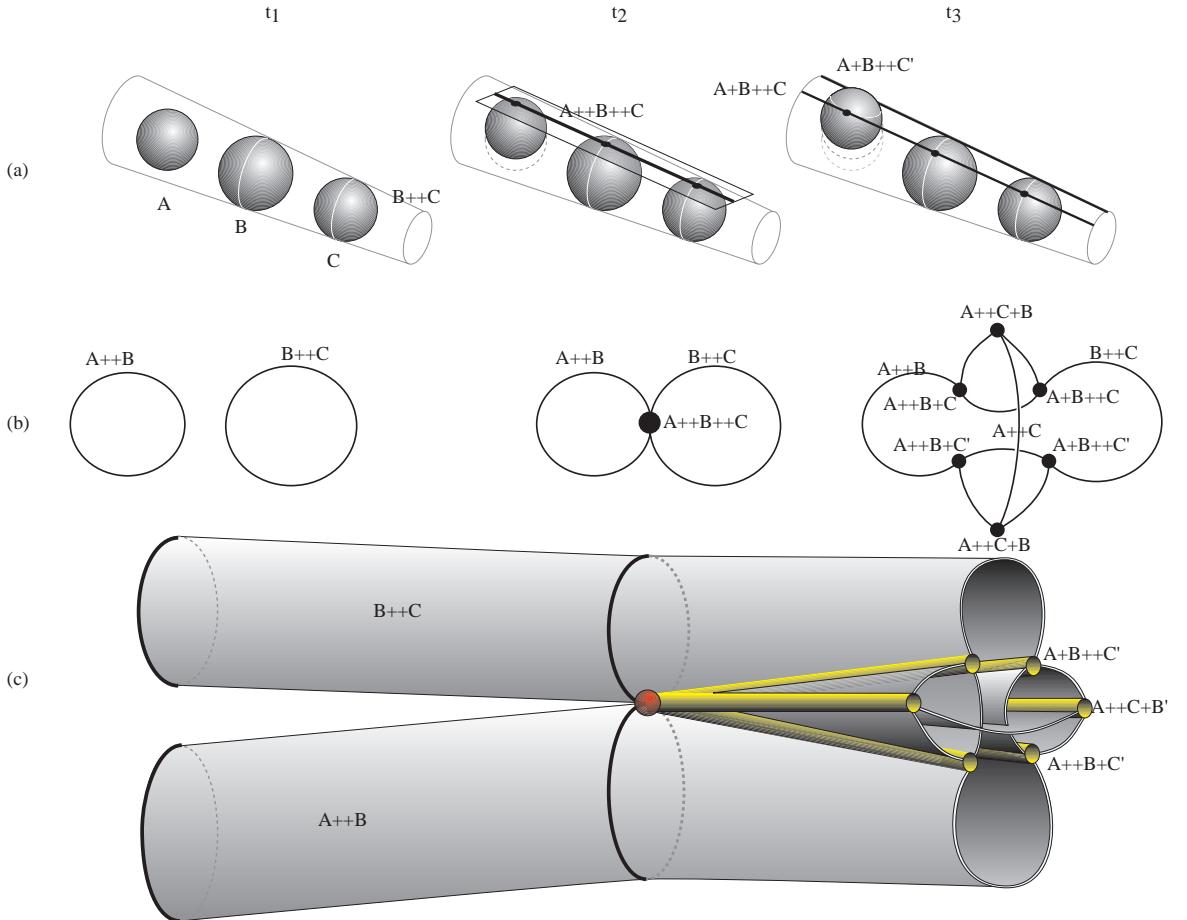


Figure 2.17: Temporal visual event and Temporal Visibility Complex. (a) Situation in 3D space. As A moves upward, it becomes visible from C at time t_2 where a segment ($A + B + +C$) lies in a plane tangent to the three spheres. The envelope of $B + +C$ is represented as a cone. Only two $T + +T + T$ 0-faces are represented, but there are actually 6 of them. (b) 1 and 0-dimensional faces of the visibility complex for each timestep. (Note that this is a projection, not a slice of the dual space. 1D sets are actually represented as 1D curves). At time t_3 we have not labeled all the 1-faces. Note that the graph of 1-faces is not planar (it is 4-dimensional), to draw it we cannot avoid the crossings of $A + +C$. (c) Structure of a part of the Temporal Visibility Complex in temporal segment space. Only 0, 1, and 2-dimensional faces are represented. The faces of the static complex represented in (b) are slices of the temporal complex. The 1-faces of the temporal complex are represented as thin cylinders. Note that they correspond to the 0-faces of the static complex and that they appear only at t_2 . Note also that due to the non-planarity of the slice, we cannot represent the 2-faces of the temporal complex without some intersections in the middle.

some 1-faces of the static complex. Temporal visual events are summarized in table 2.3 for scenes of polygons and smooth objects. They are very similar to the 0-faces of the static visibility complex.

Our definition is independent of the motion of the objects. Deformations can occur, and no rigidity or linearity is required. The handling of the removal or addition of objects is however more involved. These correspond to degenerate temporal events. Concave objects also induce a more complex catalogue of temporal events. In particular, in the case of object deformation, the changes in the differential properties of the objects must be taken into account.

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

Table 2.3: Vertices of the temporal visibility complex. These configurations can occur because of the movement of any (or all) of the involved objects.

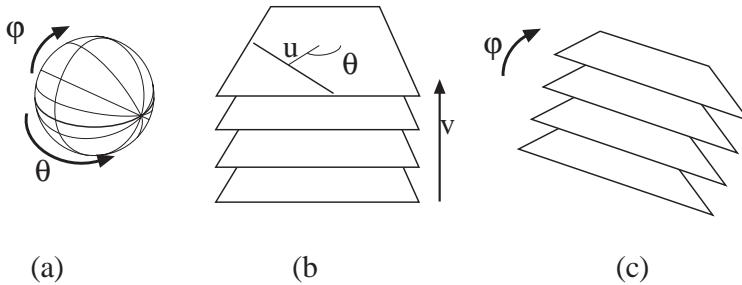


Figure 2.18: (a) Parameterization of the directions. (b) Initial v sweep. (c) ϕ sweep.

4 Output-sensitive sweep

We now present an output-sensitive construction algorithm for the static Visibility Complex. Our algorithm is a double sweep with a preprocessing phase. First the scene is swept by a horizontal plane and a 2D Visibility Complex [PV96b] of the φv -slice is maintained (figure 2.18(b)). We then sweep φ (figure 2.18(c)), but some 0-faces can not be detected during this sweep and have to be preprocessed.

4.1 Sweeping the initial slice

To build the initial φ -slice, we first maintain a φv -slice of the 3D visibility complex which corresponds to the 2D visibility complex [PV96b] of the sweeping plane.

The 2D visibility complex

We first briefly review the 2D visibility complex [PV96b, PV96a, Riv97a, DP95b] introduced in the previous chapter and its relations with the 3D visibility complex. The 2D visibility complex is the partition of the segments of the plane according to the objects they touch. Its 2D faces are connected components of segments touching the same objects (they are φv -slices of the 4-faces of the 3D visibility complex). They are bounded

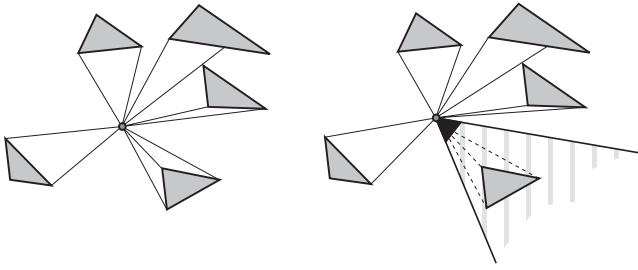


Figure 2.19: When the first vertex of a polyhedron is swept, the 2D view is computed in the sweeping plane and is restricted for each edge adjacent to the vertex by considering the angle formed by the direction of the two adjacent polygons.

by edges which correspond to segments tangent to one object (φv -slices of the 3-faces T) and vertices which are free bitangents of the 2D scene (φv -slices of 2-faces $T + T$).

Since a view around a point corresponds to the extremities of the segments going through this point, it corresponds to the traversal of the 2D visibility complex along the 1D path of these segments. The object seen changes when the path traverses a new face, which occurs at an edge of the 2D complex. In the case of a polygon, the chain of edges of the 2D complex going through one of its vertices is the view around this vertex.

Sweep

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

When the sweeping plane starts intersecting an object, we have to “insert” this object in our 2D complex. This is done by computing a 2D view around the point of tangency or around the vertex using the current 2D visibility complex. This can be done in $O(v \log n)$ where v is the size of the view using the techniques described in [Riv97b]. When the path of this view crosses an edge of the 2D complex it corresponds to a new $T + T$ or $V + T$ face of our 3D complex. In the case of the first vertex of a polygon, the view has to be restricted for each edge of the polyhedron, corresponding to the view seen by a vertex of the 2D slice (see Fig. 2.19).

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

When a vertex in the middle of a polyhedron is encountered the 2D views around the points corresponding to the edges under the vertex have to be merged, and then the view around this vertex has to be restricted for each edge above the vertex, in the same manner as first vertex sweep-events, see figure 2.20. Each operation is linear in the size of each view.

As the plane moves, three slices of objects can share a tangent (corresponding to a $T + T + T$ face of the 3D complex), in which case the 2D visibility complex is updated using the technique of [Riv97b]. Basically, for each bitangent we compute the value of v where it will become tangent to a third object and store these sweep-events in a queue which requires time $O(\log n)$ whenever a bitangent is created.

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

4.2 Principle of the φ sweep

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

During the φ -sweep (Fig. 2.23(c)) we maintain this φ -slice as well as a priority queue of sweep-events. In what follows, we will only describe the update of the 1-faces of the visibility complex, the update of the higher dimensions is done at each sweep-event using a catalogue of adjacencies of the 1-faces which for reason of

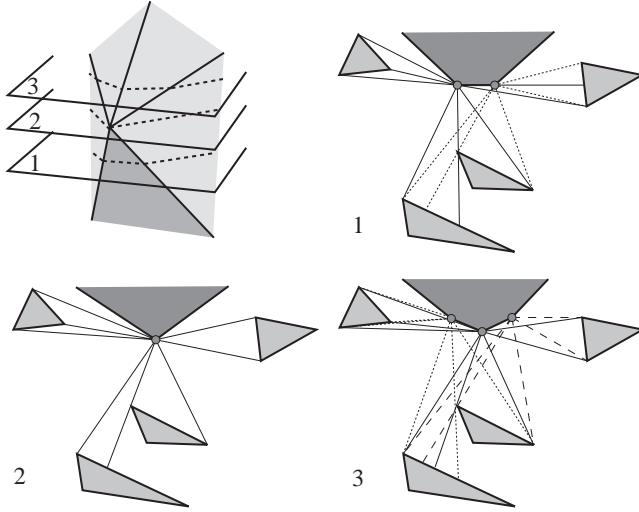


Figure 2.20: Fusion-restriction of a view around edges when a vertex is swept

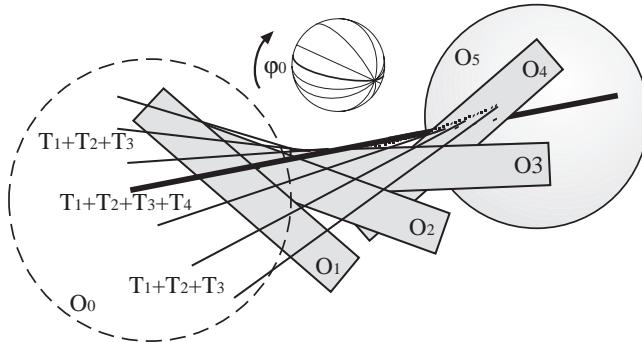


Figure 2.21: $T + T + T$ critical line sets adjacent to a $T + T + T + T$ critical line.

place is reported in appendix A. As stated before, the number of adjacent upper-dimensional faces is bounded; their update does not affect the complexity.

This φ sweep can also be understood by projecting the 1-faces and the vertices of the visibility complex onto the direction sphere S^2 represented in Fig. 2.18. The sweep then corresponds to the rotation of a great circle. Note that some intersection on S^2 are not relevant for the skeleton (this is the same difference as a t-vertex in a view and an actual vertex of the scene). This representation can be used as a educational tool because it permits the parallel between our sweep and the panning of an orthographic view.

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

4.3 Regular 0-faces

Consider a $T_1 + T_2 + T_3 + T_4$ segments with extremities O_0 and O_5 and elevation angle φ_0 (Fig. 2.21). Consider the 1D critical line set $T_1 + T_2 + T_3$. We locally parameterize it by φ and call it $l(\varphi)$. The ruled surface described by $l(\varphi)$ cuts O_4 at φ_0 . Two 1-faces of the complex are associated with $l(\varphi)$, one for $\varphi < \varphi_0$ and one for $\varphi > \varphi_0$; one has O_5 at its extremity, the other O_4 . It is the same for $T_2 + T_3 + T_4$. Moreover the two 1-faces before φ_0 are adjacent to a 2-face $T_2 + T_3$. In the φ -slice, this 2-face is a 1D set bounded by the slices of $T_1 + T_2 + T_3$ and $T_2 + T_3 + T_4$. This 1D set collapses at φ_0 , it is thus a regular sweep-event. It can be detected by considering the adjacent $T + T + T$ faces in the φ -slice and maintaining a priority queue.

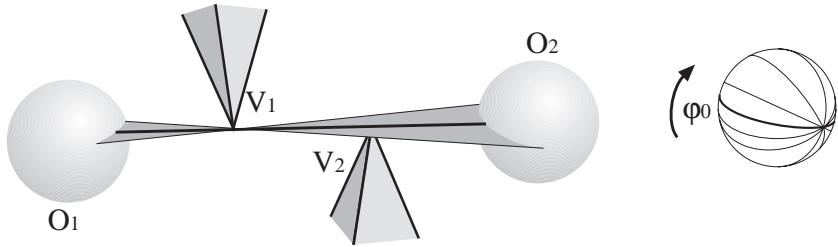


Figure 2.22: None of the $T + V$ critical segment sets adjacent to this $V + V$ critical segment exist before ϕ_0

The $T + T + T$ faces can be handled the same way because they are adjacent to a pair of $T + T$ and a pair of $T + T + T$ 1-faces, and the faces of a pair are associated with the same line set.

The projection of regular events onto S^2 is similar to that illustrated in Fig. 2.23(a).

4.4 Irregular 0-faces

Unfortunately, all the 0-faces are not regular sweep-events. The $T + T + V$ and $V + V$ events cannot always be detected in this way. The main reason is that vertices represent discontinuities at the end of edges, and we have no guarantee that a 1-face adjacent to such a 0-face exists for $\phi < \phi_0$. See Fig. 2.22 where the four $T + V$ faces appear at ϕ_0 ; this corresponds to the situation (b) of Fig. 2.23 in dual space⁵.

These events thus have to be preprocessed by considering all the VV pairs and all the Object-Object- V triples. This determines a *potential* 0-face, which is a line with coordinates $(\theta_0, \phi_0, u_0, v_0)$. ϕ_0 is used to insert it in a sweep event queue. Note that this is just a *potential* 0-face which then has to be tested for occlusion when the corresponding ϕ_0 is swept. If an object lies between the generators, the 0-face has to be discarded.

Fortunately, at least one slice of an adjacent V 2-face exists before such 0-faces appear (face V_1 in Fig. 2.22) because lines going through a vertex span all directions. This 2-face permits an efficient occlusion test.

This face is found using a search structure over the 1D V components of the ϕ -slice ordered by their generators and the angle θ . The 0-face is then tested for occlusion: we test if the second generator of the 0-face (V_2 here) lies between the extremities (O_1 and O_2) of the 2-face (V_1 in our case). The 0-face can then be inserted if the test succeeds.

4.5 Non monotonic 1-faces

There is another kind of irregular sweep-event. A 1-face of the complex can appear during the sweep without a 0-face event. This is obviously the case for $T + T + T$ events since they can be adjacent to no 0-face, but this can also be the case for $T + T + T$ events. Consider the associated line set; It is not necessarily monotonic with respect to ϕ (see Fig. 2.23(c)). These sweep-events also have to be preprocessed and inserted in the ϕ -slice with a search over the 1D components.

This search is however not as straightforward as in the V case, since the slices of $T + T$ faces are not as intuitive to order. They also can be ordered by their generators, but for a given pair (T_1, T_2) , the set of bitangent segments of a ϕ -slice is not a function of θ , *i.e.*, for a given θ there can be more than one bitangent segment. This number varies and it is thus hard to classify the different bitangents sharing the same θ coordinate.

Since an ordering is hard to obtain along θ , we use the v coordinate. For a given v there are at most four bitangent segments in a ϕ -slice. This is true because a ϕv -slice corresponds to a 2D visibility complex, and because two convex objects in the plane have four common tangent lines. Moreover, since a 2D object lies entirely on the positive or negative half-plane defined by such a line, we can classify these lines according to the side of each of the objects. This defines four categories of bitangents. For each category, there is at most one bitangent for each value of v in a ϕ -slice.

We thus use a search tree on the $T + T$ faces of the current ϕ -slice sorted by their generators, their category among the 4 possible and their v values.

⁵This also explains why cells of the aspect graph are not necessarily convex: these irregular events correspond to reflex vertices on S^2 or to reflex edges in 3D space.

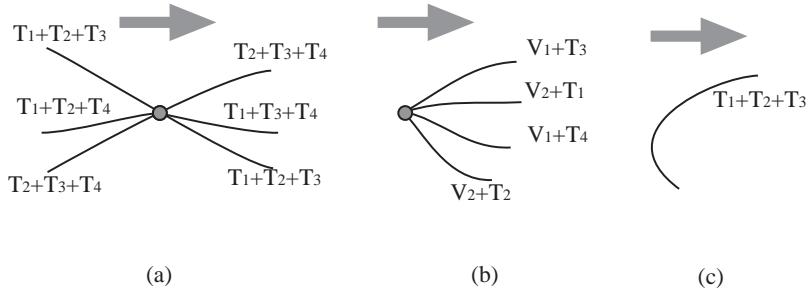


Figure 2.23: Different sweep-events represented in the dual space. The $T + T + T + T$ event (a) is regular, but the $V + V$ event (b) has to be preprocessed as well as the null derivative with respect to φ of the $T + T + T$ events (c).

4.6 Complexity of the algorithm

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

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

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

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

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

5 Applications of the approach

5.1 View computation

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

In Fig. 2.24, the surface described by the lines going through viewpoint V is represented by its φ -slices which are curves. The intersections of these curves with the tangency volumes are the points of the view on the silhouette of the objects, such as D_1, D_2, D_3, D_4 and D_5 . However, all the intersections do not necessarily correspond to a silhouette since the objects are not transparent, and points such as D' must not be taken into account. Consider the φ -slice $\varphi = 0$ and the slice V_0 of the lines going through V with $\varphi = 0$. Fig. 2.25 shows the φ -slices of the faces of the visibility complex and their traversal. We traverse the visibility complex up and down along V_0 . Initially, the segments see nothing, since we are in the face F (which is the set of segments which have both extremities at infinity). At D_1 , we leave face F and have to choose between face A and E . Since V lies in the front of the sphere R , we now traverse A from D_1 to D_2 . D' lies on no boundary of face A and is thus not considered. We then traverse face D and finally face F again. Once the φ -slice has been traversed, the intersections with the boundaries of the faces are maintained while φ is swept. Visibility changes will appear when V_φ meets a bitangency edge or a new tangency volume.

For a walkthrough, the view can be maintained since the events where the visibility changes correspond to intersections of the surface described by V with the 1-faces of the visibility complex. This approach is similar to the one described in [CT96, CT97b] where conservative visibility events are lazily computed.

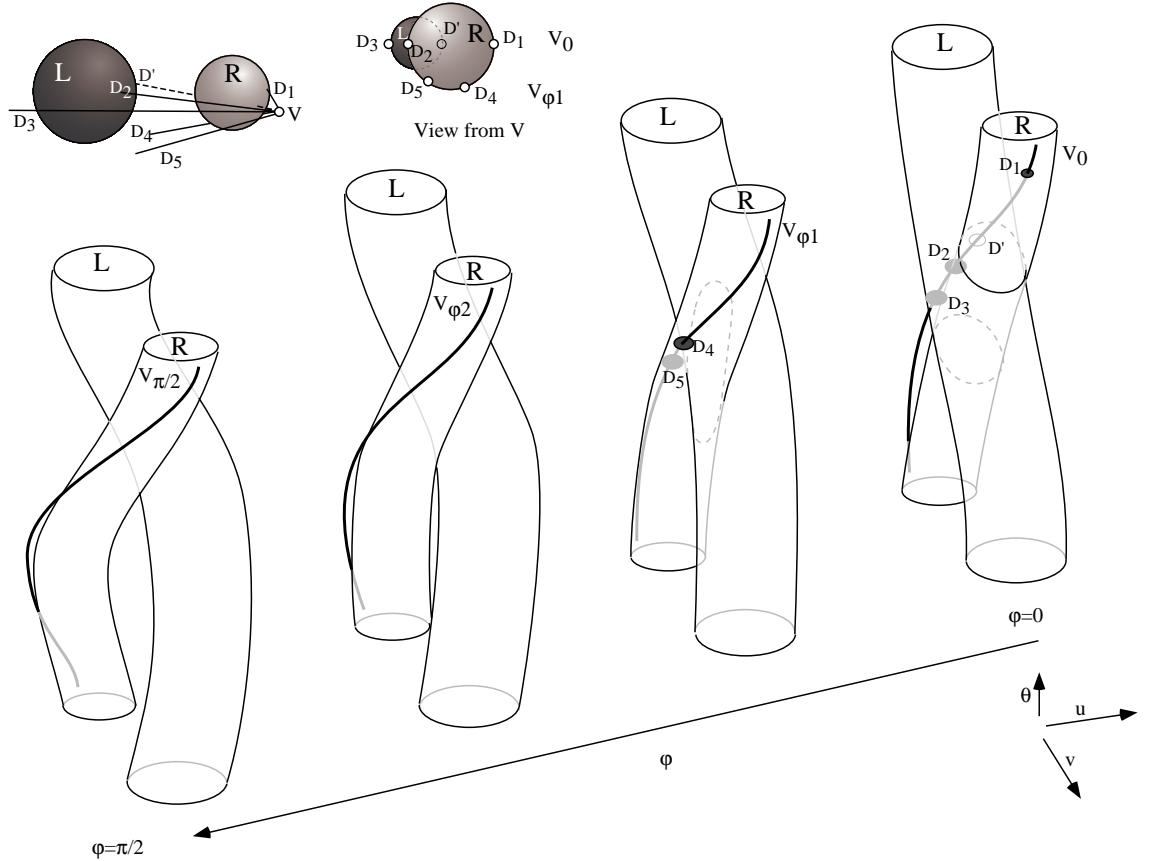


Figure 2.24: The View around a point is the intersection of the visibility complex and the surface described by the set of segments going through this point.

The recent technique of *multiple center of projection images* [RB98] in fact corresponds to the intersection of the complex with other 2D manifolds.

5.2 Form-factors

The form factor ⁶ F_{ij} involved in radiosity computation is the proportion of light that leaves patch i which arrives at patch j . It can be expressed as the measure of lines which intersect i and j divided by the measure of lines which intersect i . In the dual space, it is the measure of the face F_{ij} divided by the measure of the inside of the tangency volume of i . See [ORDP96, DORP96] for the equivalent interpretation of the form factors with the 2D visibility complex.

Unfortunately, although a simple formula exists to compute exact form factors in 2D in the presence of occluders, the only analytical result for the form factor between two 3D polygons holds only for full visibility and is quite intricate [SH93]. Exact computations can however be performed for point-to-polygon form factors, as will be shown in chapter 4. Stochastic approaches based on integral geometry [San76, Sbe93] could also be explored, by sampling the 4D faces of the complex, or by globally sampling line-space.

5.3 Computer vision and robotics

Recall that the aspect graph [PD90, GM90, EBD92] (chapter 1 section 5) is a powerful *viewer-centered* data-structure which encodes all possible views of an object. It is based on the notion of visual events, which partition the space of all possible viewpoints into cells where views are qualitatively equivalent.

⁶The same notation is used for the form factor and for the face between i and j though the form factor is a scalar and the face is a set of segments.

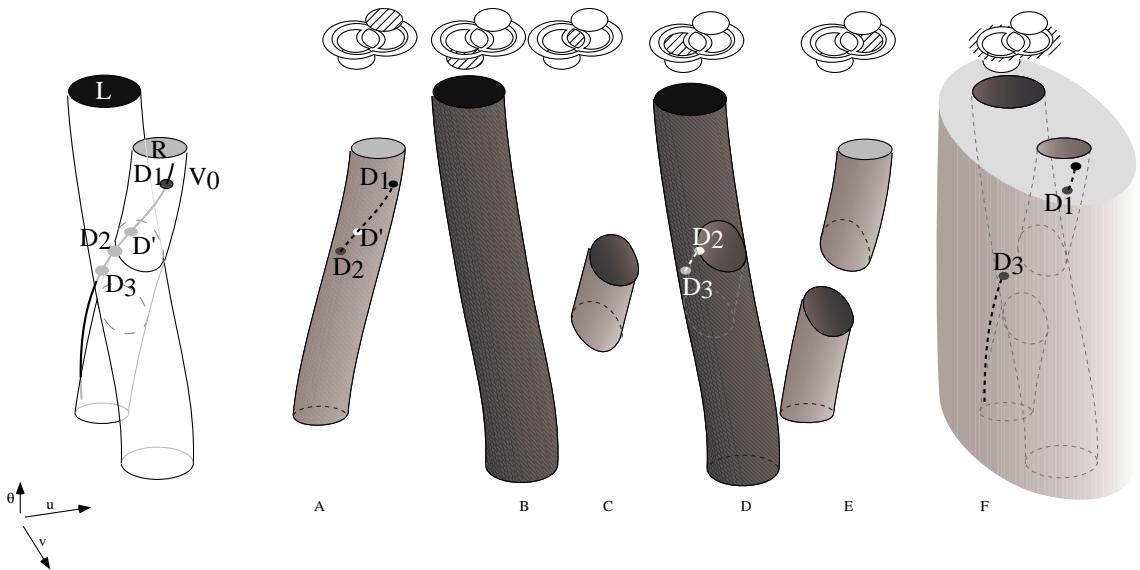


Figure 2.25: Traversal of the φ -slice $\varphi = 0$ of the complex to compute the view around point V .

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

Visual events are also involved in the computation of the *visual hull* [Lau94] or in the planning of visibility based pursuit motions in robotics [GLLL98]. The complex could thus be used as an intermediate data-structure for these problems for which mainly 2D results are known.

5.4 Umbra and penumbra

In the same way, the 1-faces inside the tangency volume of a light source correspond to the *discontinuity surfaces* of the discontinuity meshing methods (section 4.3 of chapter 1), that is, the limits of umbra and penumbra. The visibility complex gives all the events to compute a discontinuity mesh where all the objects are considered as sources.

In the context of hierarchical radiosity [HSA91], whenever a link between two objects i and j is to be refined the boundary of the face F_{ij} of the visibility complex provides all the visibility information pertinent to this energy exchange. This information can be used to effect progressive discontinuity meshing and to improve the quality of the form-factor calculation.

Chapter 4 will introduce an efficient method based on the visibility complex concepts which uses visibility to guide the meshing and the refinement of a hierarchical lighting simulation, resulting in high quality images.

6 Related approaches

Global visibility structures have been proposed in the literature, some of them based on concepts similar to the visibility complex. We review them and discuss similarities and differences.

6.1 The aspect graph

Visual events considered in the aspect graph literature correspond to the 1-faces of the visibility complex: the topology of a view for example changes when a vertex and an edge are aligned from the viewpoint. The aspect graph is in fact the arrangement of these events in the viewing space.

This explains its larger size with respect to the complex. Consider the polygonal case. There can be $O(n^3)$ $T + T + T$ events. In the visibility complex, this induces a size $O(n^4)$ because these events are split only by $T + T + T + T$ vertices. In the case of the aspect graph for orthographic projection, the construction of the 2D arrangement on S^2 can lead to as many as $O(n^6)$ intersections between pairs of events. In the perspective case where the viewing space is \mathbb{R}^3 , the size is $O(n^9)$. The visibility complex in fact implicitly contains the information encoded in the aspect graph.

The aspect graph for orthographic projection in fact also deals with a 4 dimensional information: 2 for the space of viewpoints, and 2 for each view. However, the separation of those 4 dimensions makes it hard to express many visibility properties such as the mutual visibility of two objects.

6.2 The asp

The *asp* data structure [PD90] (see section 6.3 of chapter 1) like the visibility complex is a line-space data-structure. It has been developed as an intermediate data-structure to build the aspect graph.

Two different definitions are considered depending on whether orthographic or perspective projection is considered. In the orthographic case oriented lines are considered and the *asp* is a 4D cellular decomposition of line-space according to the first object intersected (starting from infinity). Information about visibility from inside the convex-hull of the object is not encoded.

If perspective projection is used, the *asp* is a 5D cellular decomposition of ray-space according to the first object a ray intersects. The same information is encoded as in the visibility complex, but the information is redundant because many colinear rays “see” the same object.

The visibility complex can be seen as offering the advantages of the two versions of the *asp*. It encodes the optimal amount of information, by considering maximal free segments. The fifth dimension is “used” only where necessary, where the visibility of rays changes, *i.e.* along tangency volumes. It is thus a 4D structure embedded in 5D. Moreover, no version of the *asp* for curved object has been presented.

6.3 Plenoptic function, lumigraph and light-field

The *plenoptic function* [AB91] is a function in 5D ray-space which describes the light traveling at a point in a given direction. If light traveling in free-space is assumed invariant and if the observer is constrained outside the scene, it can be simplified into a function in 4D line space called *lumigraph* [GGSC96] or *light field* [LH96]. They are in a sense very similar to the *asp* concept, except that color information is encoded and that a discrete sampling is used instead of an analytic subdivision. However, the underlying multidimensional ray-spaces are the same.

Langer and Zucker [LZ97] use similar topological concepts as ours to describe the light field of a scene in a shape-from-shading context. As for the visibility complex, they note that the manifold of rays is 4 dimensional with some branchings. They do not however propose the analysis of the changes in visibility, nor the adjacencies between classes of rays.

6.4 Plücker space

Plücker parameterization is a powerful duality which represents lines in a five dimensional space where hyperplanes can be used to characterize line-line intersections (see section 6.2 of chapter 1 and the second part of this thesis). It has been extensively used to compute visibility in polygonal scenes.

Note that the fifth dimension in Plücker space does not correspond to the pseudo-dimension which we have introduced in section 1.2. Not all points in Plücker space correspond to real lines: only those lying on the so-called *Plücker hypersurface*. The fifth dimension alleviates the problem of singularities which are always present in a 4D parameterization of lines (as is also the case for any 2D parameterization of the S^2 sphere). For example our (ϕ, θ, u, v) parameterization has singularities at the poles of the direction sphere.

Arrangements have been described in Plücker space which are very similar to the dual arrangement which we have presented, *e.g.* [Pel93, Tel92b, Tel92a]. An arrangement of hyperplanes is computed in 5D and intersected with the Plücker hypersurface to obtain a 4D structure. This latter structure is in fact exactly the dual arrangement expressed in a different parameterization. This illustrates our claim in the introduction: the concepts described in this chapter do not rely on a particular parameterization of lines.

However, approaches in Plücker space have considered only equivalents of the dual arrangement. They consider only line visibility, occlusion is not really taken into account.

6.5 The 2D visibility complex

The 2D version of the visibility complex has been the first inspiration of this work. We now compare the 2D and the 3D data-structures, which illustrates the large differences between 2D and 3D visibility.

First of all, the space of lines in 3D is 4 dimensional. The increase in the dimensionality of the problems is 2 and not only 1. Similarly, the theoretical complexity is $O(n^4)$ instead of $O(n^2)$ in 2D. In 2D, only tangent and bitangent segments are considered, while in 3D the vertices of the complex are segments tangent to four objects.

The property which explains the most why 3D visibility is much harder than in 2D, is *separability*. In 2D, a line separates the plane into two half-planes. No such property holds in 3D because lines are no longer hyperplanes.

The consequence is that some convexity or monotonicity properties which hold in 2D do not hold in 3D. This is especially the case of the faces for the visibility complex. In 2D, depending on the duality, the faces are at least monotonic with respect to direction. This has a number of useful consequences, including the possibility to perform efficient optimal sweeps or walks along the complex.

In 3D, we have seen that some 0-faces are irregular for the sweep of the complex. The faces cannot be made monotonic with respect to one parameter. They moreover can have a non 0-genus (they can have holes, like a torus). This makes our algorithm far from optimal. This also explains the difficulty in designing an efficient view extraction algorithm.

7 Conclusions

7.1 Summary

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

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

Using a probabilistic approach, we have shown that in scenes with reasonable assumptions, the number of tri-tangency events is $O(n^{2.33})$ instead of $O(n^3)$.

We have also defined the *Temporal Visibility Complex* which encodes all the visibility of a time-varying 3D scene.

The 3D visibility complex is a very promising data structure for numerous computer graphics applications: we have briefly outlined its potential use for the visibility computation of a view, its use in form-factor computations and discontinuity meshing as well as the computation of aspects or backprojections. We will see in the next chapter that it leads to practical solutions by adapting a simplified representation.

7.2 Discussion

The Visibility complex is a useful framework because it permits an efficient encoding of visibility information. We believe it is a valuable tool for the interpretation of visibility problems. By providing different insights to these questions, we hope it will permit a better understanding and the development of new methods, as shown by the next chapter.

Its direct practical interest is however more questionable. The implementation of a 4D cellular subdivision is an intricate task, and traversing the adjacencies of such a complex is not straightforward.

Moreover, sweep constructions, though efficient, are prone to numerical precision and robustness problems. If an event of the sweep is not correctly handled, lost coherence will make the entire algorithm fail.

Our thoughts on a possible implementation of the visibility complex led us to extract the most important information and develop a much simpler data structure together with a more robust algorithm based on practical scene properties and spatial acceleration structures, which we present in the next chapter.

7.3 Future work

The faces of the Visibility Complex directly translate the intricate nature of 3D visibility. They have no convexity nor monotonicity property, as illustrated by the irregular events of our sweep algorithm. Enforcing such properties could be achieved by appropriately subdividing the faces of the complex. The cylindrical partitions of Mulmuley [Mul91] or the cylindrical algebraic decomposition of Collins [Col75] are two examples of such approaches. Unfortunately, this would increase the complexity of the structure.

The separation between a directional (θ, ϕ) component and a spatial component (u, v) for maximal free segments should be studied further. As we have seen, this permits the interpretation of the $T + +T$ events in our dual space. We moreover think that this will also help us to obtain some monotonicity properties, because the tangency volumes have an infinite tube structure along the directional component. For every (θ, ϕ) pair there exists a (u, v) belonging to the tangency volume of any object, while the opposite is not true.

The use of randomized localisation techniques [Cla87] could be studied to obtain efficient queries. This requires the treatment of hyperplanes, which can be achieved for polygonal scenes using Plücker coordinates. Such an approach is very similar to the work by Pellegrini [Pel93, Pel90, PS92, Pel94], except that he considers line visibility. Encoding segment visibility in this context is however by no means a trivial matter, but it is the only way to alleviate the fixed $O(n^4)$ cost.

The definition of an efficient view computation retrieval using the complex should be explored. We have not obtain such result because of the aforementioned problems of non-monotonicity. View maintenance raises the same issues.

The design of an efficient view-computation algorithm is highly related to the problem of an efficient construction of the complex. For polygonal scenes, another approach would sweep the scene by planes rotating around the edges of the polygons while maintaining a 2D visibility complex. The sweeps have to be synchronised, and only the portion of the scene visible from each vertex has to be maintained in the corresponding 2D complex. The definition of a topological order to sweep the 0-faces of the complex is yet another issue required to reach an optimal output-sensitive algorithm.

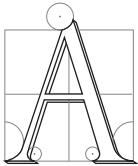
In the case of algebraic scenes, the work of PetitJean [Pet95, Pet96] on enumerative geometry could be used to obtain tight bounds on the size of the complex depending on the degree of the objects.

CHAPTER 3

The Visibility Skeleton

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

Eugène DELACROIX



ALTHOUGH it encodes all visibility information of a 3D scene, the Visibility Complex has many shortcomings which prevent its direct implementation as a practical tool. The four dimensional adjacencies of its faces are intricate to encode and to traverse. In addition, the sweep construction algorithm proposed in the previous chapter is prone to degeneracy problems and has a fixed cubic cost which is not acceptable in practice. Moreover, simple access to the visibility information is not really provided by the visibility complex.

In this chapter we present a data-structure which has been developed to simplify the visibility complex and turn the concepts presented previously into a practical tool. It can be seen as a simplification of the 3D visibility complex. In particular, the visibility skeleton only requires the construction of the 0 and 1-dimensional faces of the complex. It can also be seen as a new structure from first principles of visual events and critical lines. We adopt the latter presentation because it results in a more self contained chapter and because it permits the presentation of some of the notions of the visibility complex from a new angle.

The emphasis is in the development of a practical and easily implementable tool. The queries presented at the end of this chapter, as well as the use of the skeleton for global illumination presented in the next chapter show that the data-structure is versatile and efficient.

Part of the work developed in this chapter has been presented at Siggraph'97 [DDP97c]. Differences with the original paper include improvement to the data structure presented in a later paper [DDP99] and a presentation of the update for dynamic scenes in section 7.

1 Motivation

Previous algorithms have been unable to provide efficient and robust data structures which can answer global visibility queries for typical graphics scenes. In what follows we present a new data structure which can provide *exact* global visibility information. Our structure, called the *Visibility Skeleton*, is easy to build, since

its construction is based exclusively on standard computer graphics algorithms, *i.e.*, ray casting and line-plane intersections. It can be used to solve numerous different problems which require global visibility information; and finally it is well-adapted to *on-demand* or *lazy* construction, due to the locality of the construction algorithm and the data structure itself. This is particularly important in the case of complex geometries.

The central component of the Visibility Skeleton are *critical lines* and *extremal stabbing lines*, which, as will be reviewed in what follows, are the foci of all visibility changes in a scene. All modifications of visibility in a polygonal scene can be described by these critical lines, and a set of *line swaths* which are necessarily adjacent to these lines. We present the construction of the Skeleton, and the implementations of several applications. As an example, consider Fig. 3.1(a), which is a scene of 1500 polygons. After the construction of the skeleton, many different queries can be answered efficiently. We show the view from the green selected point to the left wall which only required 1.4 ms to compute; in Fig. 3.1(b), the complete discontinuity mesh on the right wall is generated by considering the screen of the computer as an emitter which required 8.1 ms.

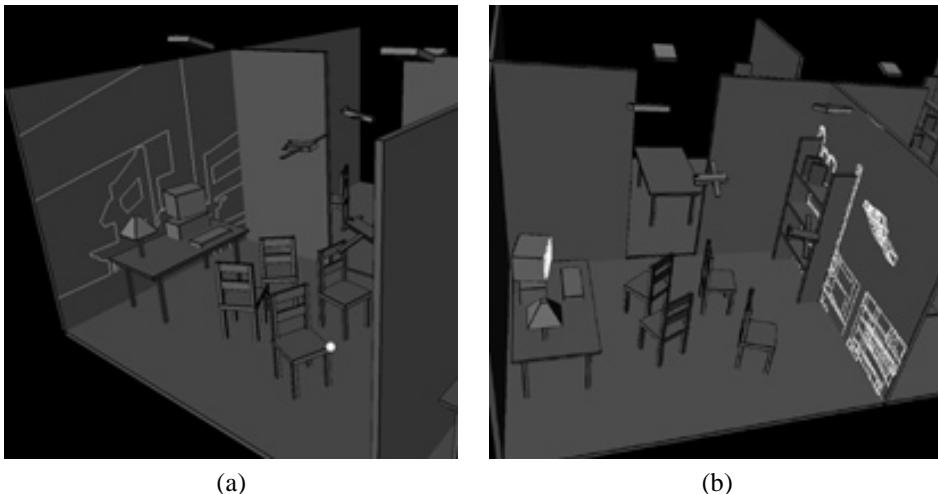


Figure 3.1: (a) Exact computation of the part of the left wall as seen by the green vertex on the chair. (b) Complete discontinuity mesh on the right wall when considering the computer screen as a source.

In what follows, we will provide a complete description of all possible extremal stabbing lines, and all the adjacent line swaths in section 2. In section 3 the actual data structures are presented in detail while the construction algorithm is described in section 4. The results of our implementation are presented in section 5, giving the complete construction of the Visibility Skeleton for a suite of test scenes. We show how the Skeleton is then used to provide exact point-to-surface visibility information for any vertex in the scene, to calculate the complete discontinuity mesh between any two surfaces in the scene, extract exact blocker lists between two objects, and compute all visibility interactions of one object with all other objects in a scene, which could be used for dynamic illumination updates in scenes with moving objects. Section 6 addresses the issues arising when treating more complex scenes, and in particular we present a first attempt at on-demand construction. The results of the implementation show that this allows significant speedup compared to the construction of the whole structure. In section 7 we sketch how the structure can be extended to environments in which objects move.

2 The Visibility Skeleton

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

2.1 Visual events

We have seen in chapter 1 that in previous global visibility algorithms, in particular those relating to aspect graph computations (*e.g.*, [PD90, GM90, GCS91]), and to antipenumbra [Tel92a] or discontinuity meshing

[DF94, SG94], visibility changes have been characterized by *critical lines sets* or *line swaths* and by *extremal stabbing lines* which are the loci of *visual events*

Following [Pel90] and [Tel92a], we define an extremal stabbing line to be incident on four polygon edges. There are several types of extremal stabbing lines, including vertex-vertex (or *VV*) lines, vertex-edge-edge (or *VEE*) lines, and quadruple edge (or *E4*) lines. As explained in Section 2.3, we will also consider here extremal lines associated to faces of polyhedral objects. These correspond to the vertices of the 3D visibility complex.

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

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

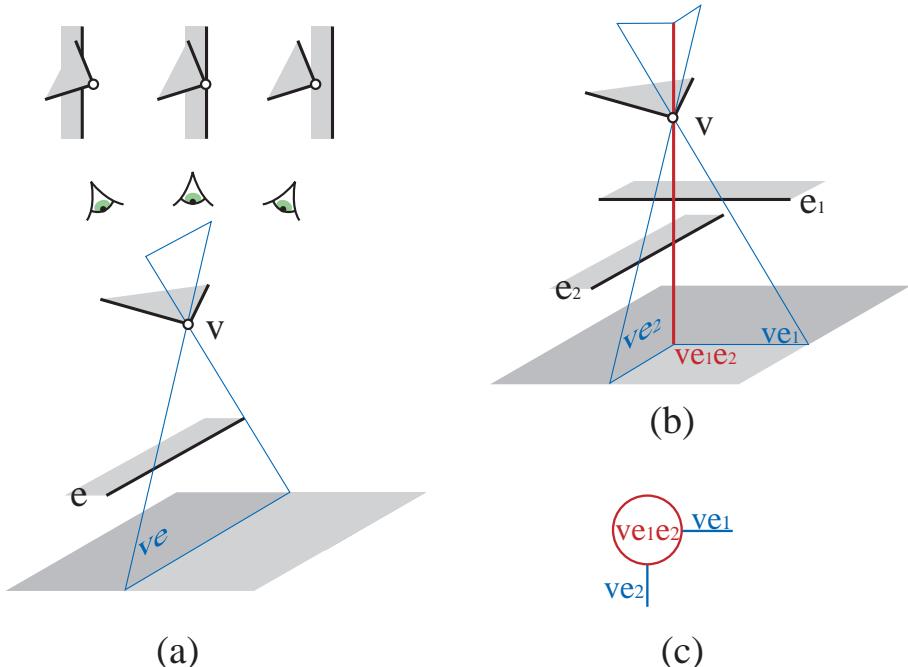


Figure 3.2: (a) While the eye traverses the line swath *VE*, the vertex *v* passes over the edge *e*. (b) Two line swaths meet at an extremal stabbing line (c) and induce a graph structure.

We start with an example: after traversing an *EV* line swath from left to right as shown in Fig. 3.2(a), the vertex as seen from the observer will lie upon the polygon adjacent to the edge and no longer upon the floor. This is a visibility change, *i.e.* a visibility event as we have seen in the previous chapter. The topology of the view is modified whenever the vertex and the edge are aligned, that is, when there is a line from the eye going through both *e* and *v*.

This *EV* line swath is a one dimensional (1D) set of lines, passing through the vertex *v* and the edge *e*, thus it has one degree of freedom (for example a parameter varying over the edge *e*). When two such *EV* surfaces meet as in Fig. 3.2(b) a unique line is defined by the intersection of the two planes defined by the *EV* surfaces. This line is an extremal stabbing line; it has zero degrees of freedom.

In what follows we will develop the concepts necessary to avoid any direct treatment of the line swaths themselves since sets of lines or the surfaces described by these sets are difficult to handle, in part because they can be ruled quadrics. All computations will be performed by line – or ray – casting in the scene.

We will be using the extremal stabbing lines to encode all visibility information, by storing a list of all line swaths adjacent to each extremal stabbing line. In our first example of Figure 3.2(b), the *VEE* line *ve₁e₂* is adjacent to the two 1D elements *ve₁* and *ve₂* described above; *i.e.*, the swaths *ve₁* and *ve₂*. Additional adjacencies for the *VEE* line *ve₁e₂* are implied by the interaction of *ve₁* and *e₂* (Fig. 3.3(a)).

To complete the adjacencies of a *VEE* line, we need to consider the *EEE* line swaths related to the edges

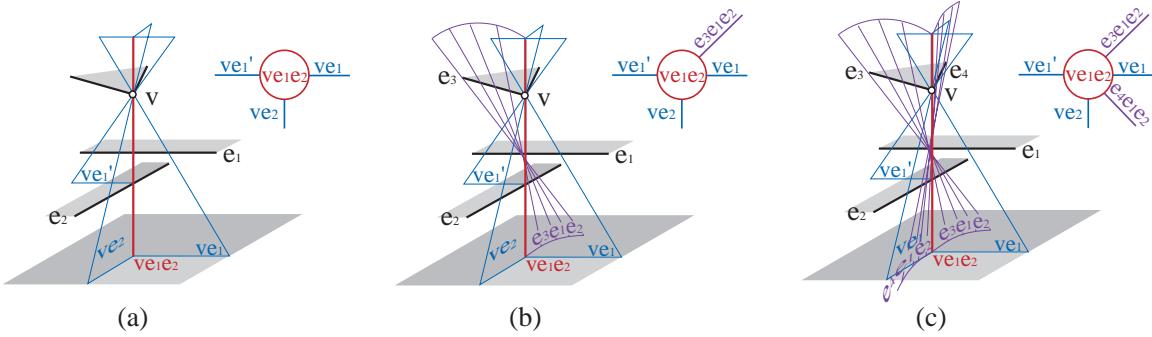


Figure 3.3: (a) An additional *EV* line swath is adjacent to the extremal stabbing line, (b) (c) and two *EEE* line swaths.

e_1 and e_2 , and the two edges e_4 and e_3 which are adjacent to the vertex v (Fig. 3.3(b) and (c)).

The simple construction shown above introduces the fundamental idea of the Visibility Skeleton: by determining all the appropriate extremal stabbing lines in the scene, and by attaching all adjacent line swaths, we can completely describe all possible visibility relationships in a 3D scene. They will be encoded in a graph structure as shown on Fig. 3.3, to be explained in Section 2.3. Consider the example shown in Fig. 3.3(a): The node associated to extremal stabbing line ve_1e_2 is adjacent in the graph structure to the arcs associated with line swaths ve_1 , ve_1' and ve_2 .

2.2 The 3D visibility complex, the *asp* and the visibility skeleton

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

The zero and one-dimensional components of the visibility complex are in effect the same as those introduced above, which we will be using for the construction of the Visibility Skeleton. Similar constructions were presented (but not implemented to our knowledge for the complete perspective case) for the *asp* structure [PD90] for aspect graph construction.

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

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

The *Visibility Complex* and *asp* are intricate data-structures with complicated construction algorithms since they require the construction of a 4D or 5D subdivision. In addition they are difficult to traverse due to the multiple levels of adjacencies. The approach we present here is different: we have developed a data structure which is easy to implement and use.

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

2.3 Catalogue of visual events and their adjacencies

The Visibility Skeleton is a graph structure. The nodes of the graph are the extremal stabbing lines and the arcs correspond to line swaths (see Fig. 3.3). In this section (and in Appendix B) we present an exhaustive list of all possible types of arcs and nodes of the Visibility Skeleton.

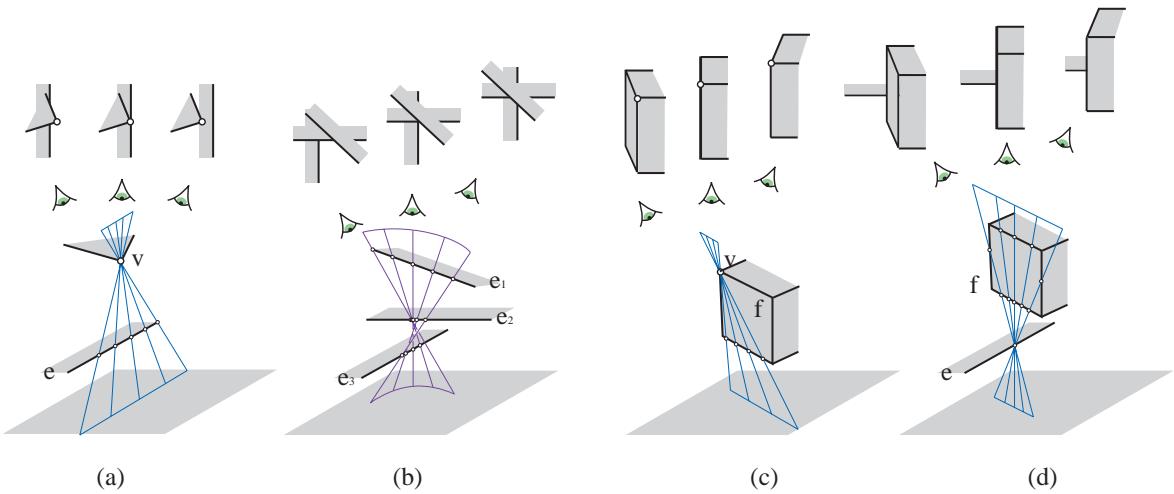


Figure 3.4: (a) Same as Fig. 3.1(a): *EV* line swath. (b) As the eyepoint crosses the *EEE* line swath, the order of the visual intersection of e_1 , e_2 and e_3 is modified. When the eyepoint is on the swath, the three edges are visually aligned. (c) In front of the *FV* swath we see the front side of f , on the swath we see a line and behind we see the other side of f . (d) The *FE* swath is similar to the *FV* case.

1D elements: Arcs of the visibility skeleton

In Fig. 3.4, we see the four possible types of 1D elements: an *EV* line swath (shown in blue), an *EEE* line swath (shown in purple) and two line swaths relating a polygonal face (F) to one of its vertices (Fv) or an edge of another polygon(*FE*) (both are shown in blue). In the upper part of the figure we show the view (with changes in visibility), as seen from a viewpoint located above the scene. It moves from left to right and in front of, on, or behind the line swath.

Note that the interaction of an edge e and a vertex v can correspond to many ve arcs of the skeleton, because they have different polygons at their extremities. These arcs are separated by nodes. Consider, for example, arcs ve_1 and ve_1' adjacent to node ve_1e_2 in Fig. 3.3(a).

0D elements: Nodes of the visibility skeleton

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

The simplest node corresponds to the interaction of two vertices shown in Fig. 3.5(a).

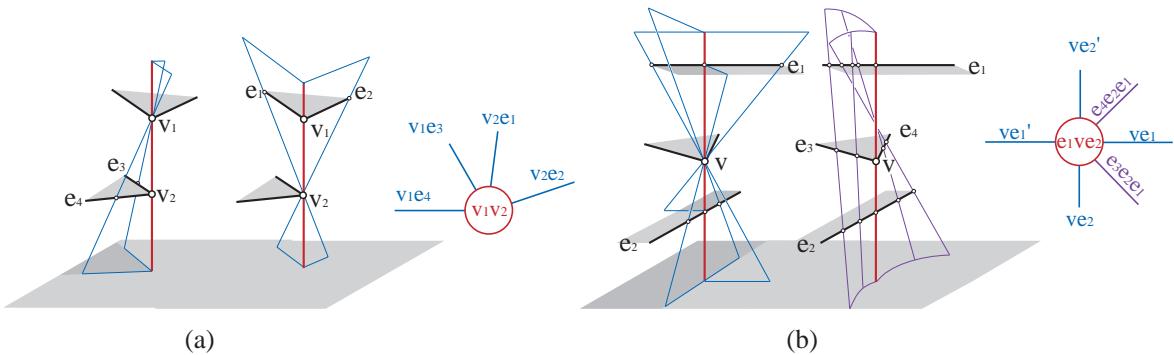


Figure 3.5: (a) A *VV* node v_1v_2 is adjacent to four *EV* arcs defined by a vertex and an edges of the other vertex: v_1e_3 and v_1e_4 and e_1v_2 and e_2v_2 . (b) An *EVE* node e_1ve_2 : each edge defines two *EV* arcs with v depending on the polygon at the extremity, and to two *EEE* are defined by e_1 , e_2 and the two edges adjacent to v .

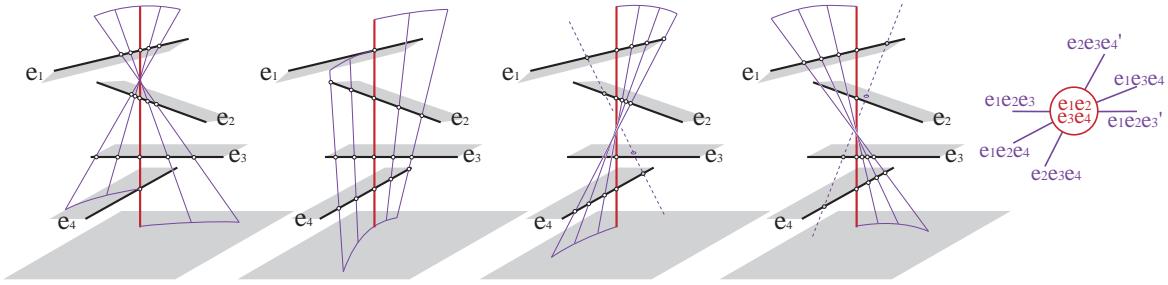


Figure 3.6: An $E4$ node is adjacent to six EEE arcs.

The interaction of a vertex v and two edges e_1 and e_2 can result in two configurations, depending on the relative position of the vertex with respect to the edges. The first node was presented previously in Fig. 3.3 and the second is shown in Fig. 3.5(b).

The interaction of four edges is presented in Figure 3.6, together with the six corresponding adjacent EEE arcs. Face related nodes are given in detail in appendix B: EFE , FEE , FF , E and Fvv (see Fig. B.3 to B.4).

3 Data structure

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

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

3.1 Initial data structure

The simplest element of the structure is the node. The *Node* structure contains a list of arcs, and pointers to the polygonal faces F_{up} and F_{down} (possibly void if the corresponding extremity of the node lies at infinity) which block the corresponding extremal stabbing line at its endpoints P_{up} and P_{down} (Fig. 3.7(a)).

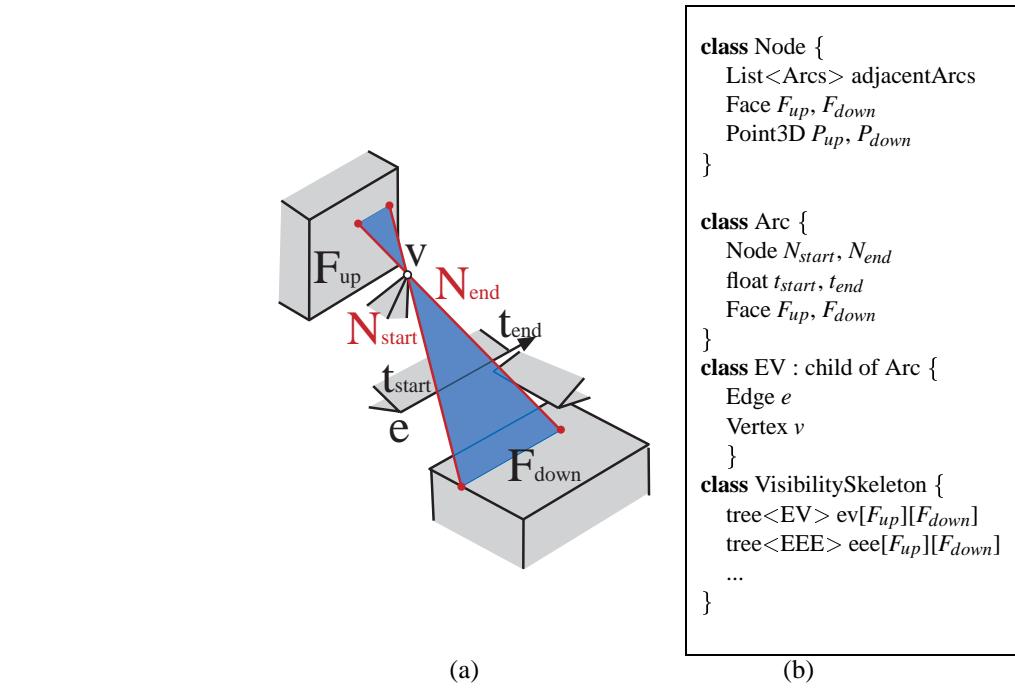
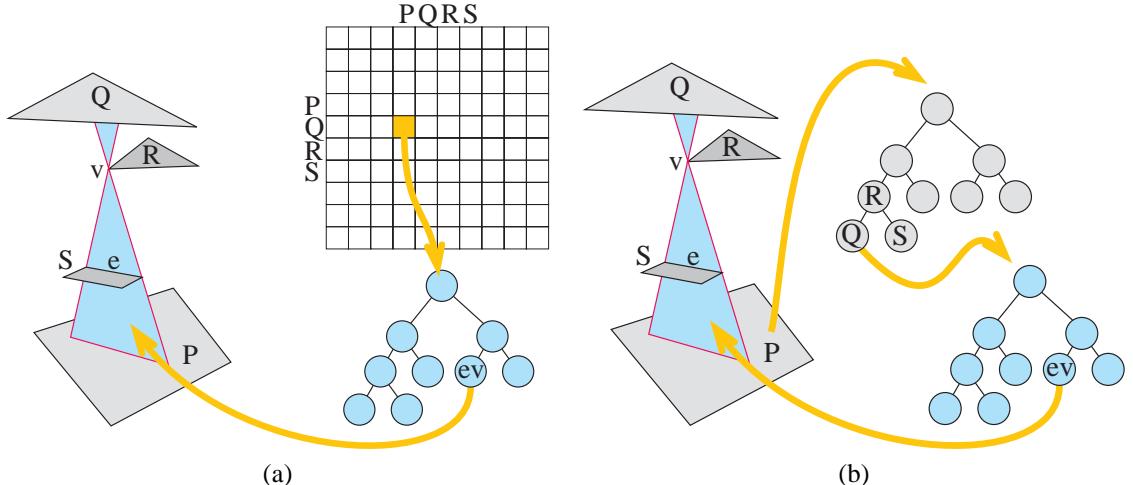
The structure for an *Arc* is visualized in the Fig. 3.7(a). The arc represented here (swath shown in blue) is an EV line set. There are two adjacent nodes N_{start} , N_{end} , represented as red lines. All the adjacency information is stored with the arc.

The lines of a swath are parameterized by a parameter t , which is chosen along one of the generating edges (we attach an arbitrary coordinate system to each edge). If there is ambiguity because the arc is generated by more than one edge, the edge with the smallest identification number is chosen. This parameterization is crucial for the insertion of nodes during the construction of the skeleton. Note that an extremal stabbing line has different parameters, depending on the arc which is considered. Each arc is characterized by the two values t_{start} and t_{end} of its two adjacent extremal stabbing lines.

Details of the structures *Node* and *Arc* are given in Fig. 3.7(b).

To access the arc and node information, we maintain arrays of balanced binary search trees corresponding to the different type of swaths considered. For example, we maintain an array ev of trees of EV arcs (see Fig. 3.7(b)). These arrays are indexed by the unique identifiers of the endpoints of the arcs. These can be faces, vertices or edges (if the swath is “interior”, that is if the lines traverse the polyhedron adjacent to one of its generators).

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

**Figure 3.7:** Basic Visibility Skeleton Structure.**Figure 3.8:** (a) Summary of the initial visibility skeleton structure. A two-dimensional array indexed by the polygons stores balanced binary search trees where the arcs of the skeleton are stored. (b) Summary of the new visibility skeleton structure which reduces the memory requirements. Each polygon stores a search tree indexed by the polygons it can see. For each pair of mutually visible polygons, a search tree of arcs of the skeleton is stored.

3.2 Reducing memory requirements

The storage of the arcs of the Skeleton in a two dimensional array incurs an $O(n^2)$ cost in memory. In the scenes we present in section 5 half of the memory is used for the array, in which more than 95% of the cells are empty! It is even more problematic when the scene is highly occluded such as in the case of a building where each room sees a only fixed number of other rooms: the number of arcs is only $O(n)$. Moreover, for our lighting simulation, we will need to subdivide the initial polygons into sub-patches and incrementally compute visibility information between some pairs of sub-patches, but not all.

For these reasons we propose to store the set of critical line swaths between two polygons on the polygons themselves. Each polygon P stores a balanced binary tree; each node of this tree contains the set of arcs between P and another polygon Q . This set is itself organized in a search tree (see Fig. 3.8(b)). Each set of arcs is referenced twice, once on P and once on Q . In the same manner, each vertex V has a search tree containing the sets of arcs between V and any polygon Q (which represents the view of Q from V). These sets of arcs are closely related to the notion of *links* in hierarchical radiosity as we will see in chapter 4.

For the scenes tested, this approach results in an average memory saving of about 30%. Moreover, we ran our modified version of the visibility skeleton on a set of scenes consisting of a room replicated 2, 4, and 8 times, showing roughly linear memory growth. Using a binary tree instead of an array incurs an additional $O(\log n)$ time access cost, but this was not noticeable in our tests.

We will see in the next chapter that the Visibility Skeleton can be used for lighting simulation. To permit the subdivision of surfaces required to represent visual detail (shadows etc.) on scene polygons, visibility skeleton information must be calculated on the triangles created by the subdivision and the corresponding interior vertices. The process is presented in detail in chapter 4.

Since the visibility information is now stored on polygons and vertices (instead of in a 2D array), the generalization to subdivided polygons is straightforward. On each sub-patch or sub-vertex, we store the visibility information only for the patches it interacts with.

4 Construction algorithm

4.1 Finding nodes

Before presenting the actual construction of each type of node, we briefly discuss the issue of “local visibility”. As has been presented in other work (*e.g.*, [GM90]), for any edge adjacent to two faces of a polyhedron, the intersection of the two negative half-spaces of the polygonal faces is locally invisible. Thus when considering interactions of an edge e , we do not need to process any other edge e' which is “behind” the faces adjacent to e . This results in the culling of a large number of potential events.

The general principle of the node detection method is as follows: we first find a potential extremal stabbing line generated by a given set of generators. That is, we find a line which intersects the generators. We then test it for occlusion, because an object can lie between the generators, discarding the line. This is the difference between line-visibility and segment-visibility discussed in the previous chapter.

Trivial nodes

The simplest nodes are VV , Fvv and Fe . For these, we simply loop over the appropriate scene elements (vertices, edges and faces). The appropriate lines are then intersected with the scene using a traditional ray-casting operation to determine if there is an occluding object between the related scene elements, in which case no extremal stabbing line is reported. Otherwise the elements and points at the extremities of the lines are returned as well as the appropriate location in the overall arc tree array.

VEE and EEEE nodes

We consider two edges of the scene e_i and e_j . All the lines going through two segments are within an extended tetrahedron (or double wedge) shown in Fig. 3.9, defined by four planes. Each one of these planes is defined by one of the edges and an endpoint of the other.

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

We next consider a third edge e_k of the scene. If e_k cuts a plane of the wedge, a potential *VEE* or *EVE* node is created: If edge e_k of the scene intersects the plane of the double wedge defined by edge e_i and vertex v of e_j , there is a ve_ie_k or $e_iv_e_k$ event (Fig. 3.9(a)).

We call the intersection of e_k with the extended tetrahedron its *restriction*. There are at most three parts for a restriction of an edge which lie inside the double wedge (because there can be 4 intersections with the double

wedge). Moreover, the computation of the limits of the restriction of e_k inside the double wedge also give us the potential EVE and VEE lines going through e_k , one of the edges e_i or e_j and a vertex of the other edge.

The restriction of e_k also corresponds to the restriction of the $e_i e_j e_k$ line set, with respect to the line set generated by the entire lines supporting e_i , e_j and e_k .

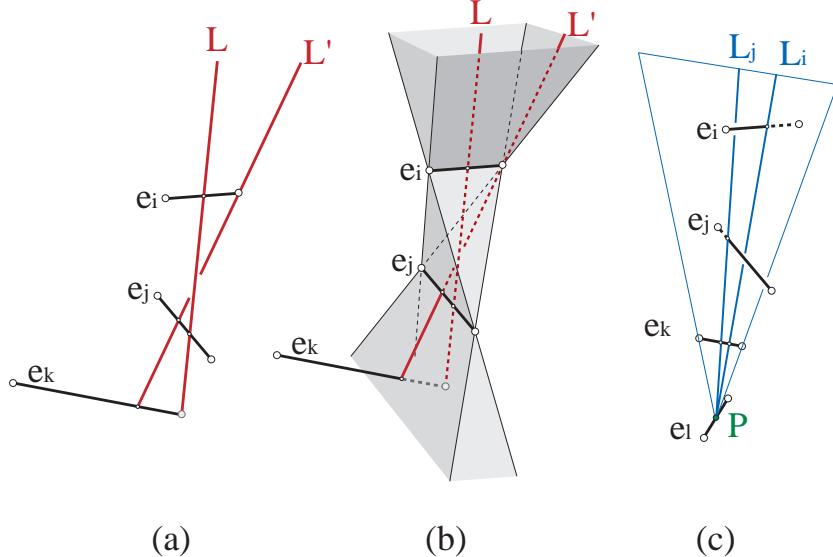


Figure 3.9: (a) (b) VEE enumeration and EEE restriction. (c) E4 computation: find the root of the line going through $e_j e_k e_l$ that also goes through $e_l e_k e_i$.

We next proceed with the definition of the E4 nodes. The intersections of e_k and the planes of the double wedge have *restricted* the third edge e_k (Fig. 3.9). To obtain a line going through e_i , e_j and e_k we need only consider the restriction of e_k to the double wedge. This process is re-applied to restrict a fourth edge e_l by the wedge of e_i and e_j , by that of e_i and e_k and by that of e_j and e_k . This multiple restriction process eliminates a large number of candidates.

Once the restriction is completed, we have two EEE line sets, those passing through e_l , e_i and e_j and those passing through e_l , e_i and e_k . A simple binary search is applied to find the point on e_l (if it exists) by which a potential E4 extremal stabbing line $e_i e_j e_k e_l$ passes. We perform this search for a point P of e_l by searching for the root of the angle formed by the two lines L_i and L_j defined by the intersection of the plane (P, e_i) with e_j and with e_k . This is shown on Fig. 3.9(c).

A more robust algorithm such as the one given in [TH91] could be used, but the simpler algorithm presented here seems to perform well in practice. This is true mainly because we are not searching for infinite stabbing lines, but for restricted edge line segments. The potential VEE and E4 enumeration algorithm is given in Fig. 4.1.

Acceleration using space subdivision

We have developed an acceleration scheme to avoid the enumeration of all the triples of edges. For each pair of edges, we quickly reject most of the third potential edges using a regular grid. Instead of checking if each cell of the grid intersects the extended tetrahedron, we use the projection on the three axis-aligned planes. We project the extended tetrahedron (which gives us an hourglass shape) onto each such plane.

To compute this 2D hourglass, we use the bounding boxes of the two edges e_i and e_j (see Fig. 3.11). The 2D hourglass is then defined by the separating and supporting lines of the projection of the bounding box. These hourglasses are then conservatively rasterized at the resolution of the grid (each pixel containing a point of an hourglass is marked valid).

We then perform the actual edge-tetrahedron intersection (the restriction) only for the edges contained in the 3D cells whose three projections intersect the three pixelized hourglasses (see Fig. 3.11).

```

potential VEE and EEEE enumeration
{
  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
}

```

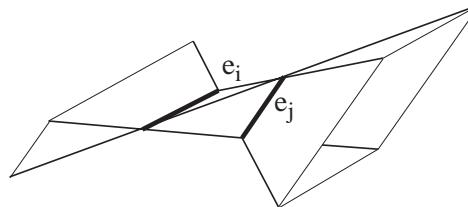
```

EEE restriction
{
  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
}

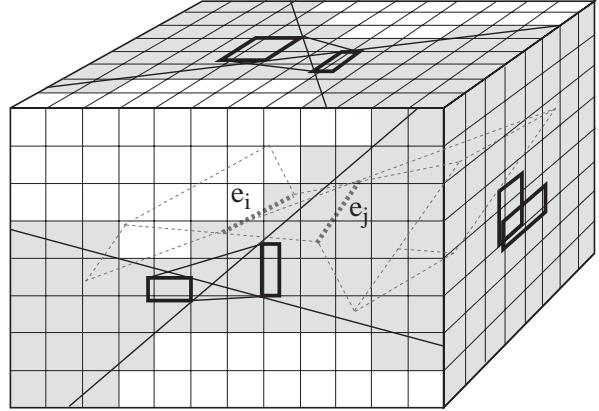
```

(a)

(b)

Figure 3.10: Enumeration of Potential VEE and E4 Nodes.

(a)



(b)

Figure 3.11: Acceleration using a regular grid and hourglasses. (a) Two edges and the corresponding extended tetrahedron. (b) We use projections along the three axis of a regular grid. The projection of the bounding box of the edges is used to compute an hourglass shape. The valid voxels are those projecting on a grey “pixel” on each axis-aligned plane.

Non-trivial face nodes

To calculate the non-trivial face-related nodes (FvE , FEE and FF), we start by intersecting the plane of each face f_1 with every edge of the scene. For edges intersecting the face we attempt to create an FvE node (Fig. B.3). That is, we perform the occlusion test (is there an object between the face and the edge?), and if no occlusion occur we actually create the node.

For each pair of edges intersecting the plane of the face, we search for a potential FEE node. To do this we determine if the line joining the two intersections intersects the face f_1 . If this is the case, the potential FEE line is then tested for occlusion.

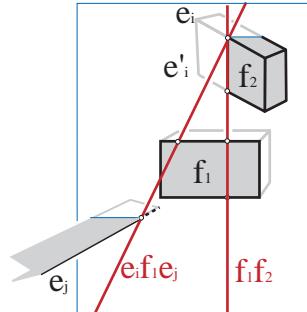
The last operation required is the verification of the existence of an FF node. This case occurs if the faces adjacent to the edge of the intersection cause an FF (face f_2 in Fig. 3.12 (b)). The construction for the FEE and FF nodes is described in Fig. 3.12 (a).

```

Find Face Nodes {
  foreach face  $f_1$  of the scene
    foreach edge  $e$  of the scene
      compute the intersection of the edge  $e$  with the plane of  $f_1$ 
      foreach intersection  $P_i$ 
        create a  $FvE$ 
        foreach intersection  $P_j$ 
          if  $(P_i P_j)$  intersects  $f_1$ 
            create  $FEE$ 
          foreach of the 2 faces  $f_2$  adjacent to the edge of  $P_i$ 
            find  $P_j$  the intersection of a second edge of  $f_2$  with  $f_1$ 
            if  $(P_i P_j)$  intersects  $f_1$ 
              create  $FF$ 
}

```

(a)



(b)

Figure 3.12: Finding Face Nodes.

4.2 Creating the arcs

```

Creation of a Visibility Skeleton Node
{
  foreach adjacent arc  $n$ 
    compute  $t$ 
    foreach arc  $a$  with same extremities and same generators
      if  $a \rightarrow t_{start} < t < a \rightarrow t_{end}$ 
        AddNodeToArc( $n, a$ )
    if no arc found
      create new Arc
}

```

```

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

```

Figure 3.13: Node Creation.

The creation of the arcs of the Visibility Skeleton is performed while detecting the nodes. When inserting a new node, we treat all the adjacent arcs from the corresponding catalogue presented in Section 2.3. If the arc has already been created (because of a previous node) we just link it to the new node, otherwise we create the arc and link it to the node (it has thus a null node at its extremity, waiting for the insertion of a new node).

For this purpose, for each of these arcs a we calculate the arc parameter t corresponding to the node to be inserted (as defined in Section 3.1), and proceed as explained in Fig. 4.2. We access the list of arcs a' in the Skeleton with the same extremities (thus in the same cell of the array or in the same search tree) and which have the same generator elements (*vertices and edges*) as the arc a . If the value of t indicates that the node is contained in the arc a' , this means that a had already been created, and is equal to a' . We then determine whether this node is the start or the end node of the arc (according to the chosen arc parameterisation). This is explained in more detail in the following paragraph. If this position is already occupied we split the arc, else we assign the node to the corresponding extremity of the arc.

We have seen above that each time an arc adjacent to a node is considered, we have to know if it is its *start node* or its *end node* according to the parameterization of the arc. That is, does this node terminate the arc in the direction of the increasing or decreasing t ? In some cases this operation is trivial, for example for a $v_1 v_2$ node and one if its adjacent $v_1 e$ arcs, we simply determine if v_2 is the starting vertex of e (recall that edges are arbitrarily oriented, and that our parameterization t is based on the intercept on the edges). In other cases, this can be more involved, especially for the *E4* case. This case and the necessary criteria for the other cases are summarized in Table 2 of Appendix B.

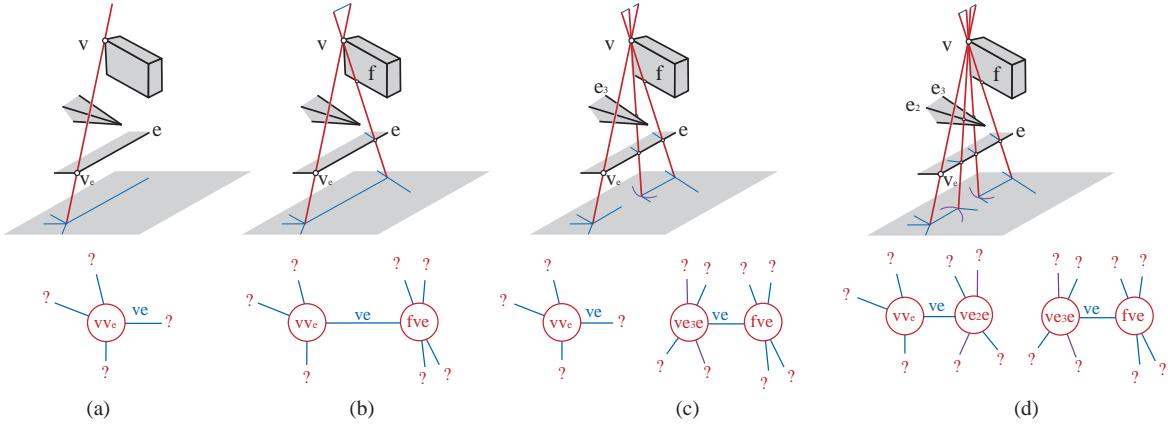


Figure 3.14: Example of node insertions: (a) Insertion node vv_e . (b) Insertion of node f_v_e . Arc ve has now two ending nodes. (c) Insertion of node ve_3e . Arc ve is split. (d) Insertion of node ve_4e , the two arcs ve have their actual adjacent nodes.

In Fig. 3.14, we illustrate an example of the construction algorithm. Initially a trivial vv_e node is created. The second node identified is vfe , which is adjacent the arc ve . Thus the arc ve is adjacent to both vv_e and vfe . The third node to be created is ve_3e . When this node is inserted, we realize that the start node for ve already exists, and we thus split the ve arc. This splitting operation will leave the end of the ve arc connected to vv_e undefined. The final insertion shown is ve_2e which will fill an undefined node previously generated.

4.3 Treating degenerate configurations

Computational geometry often makes the assumption that the scenes considered are in a “general configuration”. Unfortunately, computer graphics scenes are very often highly degenerate: many points are aligned, segments or faces are parallel or coplanar and objects touch each other.

This results in degenerate visibility events; *e.g.*, VVV extremal stabbing lines passing through three aligned vertices, or $E5$ stabbing lines going through five edges (these degenerate events will also be discussed in the context of dynamic scenes in section 7).

These degenerate configurations cause duplicate line swaths and result in numerical instabilities in the occlusion test of a potential extremal stabbing line. This line may then be randomly discarded. Inconsistencies can thus appear in the neighbourhood of the corresponding nodes of the graph. A consistent policy has to be chosen to include these nodes and their adjacent arcs or not.

We first have to identify the occurrence of these problems. When a potential extremal stabbing line is tested for occlusion, we also check for grazing objects (typically edges collinear to the line or coplanar faces). This requires a simple modification to the point-in-polygon test used for the ray-casting occlusion test of the potential extremal stabbing lines. We thus detect the intersection with a silhouette edge or vertex.

We also have to deal with the aforementioned degenerate extremal stabbing lines. A first possibility is to explicitly create a catalogue of all these degeneracies. This approach however quickly makes the implementation intractable because of the large number of different cases. We have chosen to always consider the simplest configuration, that is the one in which we have the smallest number of visual events. For example, if four edges E_1, E_2, E_3 and E_4 are parallel in that order, we consider that E_2 occludes E_1 and then E_3 occludes E_2 etc. The configurations to be treated are thus simpler and correspond to the standard Skeleton catalogue of events. The problems of numerical precision are treated using a consistent ϵ threshold for equality and zero tests.

We do not claim that the problem of numerical precision and robustness have received a definitive, general and reliable treatment, as will be discussed in the conclusions chapter. Our implementation however has permitted the computation of the visibility skeleton for scenes where the typical degeneracies encountered in computer graphics occur.

5 Implementation and first applications

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

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

5.1 Implementation and construction statistics

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

We treat touching objects by detecting this occurrence and slightly modifying the ray-casting operation (to avoid ray-leaks through the contact between two objects). We also reject coplanar edge triples. Other degeneracies such as intersecting edges are not yet treated by the current implementation.

We present statistics on the size of our initial structure and construction time in Table 5.1 and Fig. 3.15. Evidently, these tests can only be taken as an indication of the asymptotic behavior of our algorithm. As such, we see that our test suite indicates quadratic growth of the memory requirements and super-quadratic growth of the running time. In particular, for the test suite used, the running time increases with $n^{2.4}$ on average, where n is the number of polygons.

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

With the initial structure (*i.e.*, with the 2D arrays), a large percentage of the memory required is used by these arrays (*e.g.* for scene (d) of Table 5.1, the arrays need 53.7Mb out of a total 135Mb). Since these arrays are very sparse (*e.g.* 99.3% empty for scene (d)), it is clear that storage requirements can be greatly reduced. The memory requirements are greatly decreased by our improved data-structure which replaces the arrays with binary trees stored at the polygons. An average gain of 30% has been observed on our test scenes.

In the case of densely occluded scenes, the memory requirements grow at a slower rate, on average much closer to linear than quadratic with respect to the number of polygons. As an example, we replicated scene (a) 2, 4 and 8 times, thus resulting in isolated rooms containing a single chair each. The memory requirements (excluding the arrays) are 1.2Mb, 2.8Mb, 8.6Mb and 17.3Mb, for respectively 78, 150, 300 and 600 polygons. The improved structure leads to a linear growth.

The theoretical upper bounds are very pessimistic, $O(n^4)$ in size because every edge quadruple can have two lines going through it [TH91], and $O(n^5)$ in time because such potential extremal stabbing lines have to be ray-cast with the whole scene. But such bounds occur only in uncommon (“exotic”) worst case scenes such as grids aligned and rotated or infinite lines. It is clear that our construction algorithm would be very inefficient for such cases. More efficient construction algorithms are possible (such as the construction of the visibility complex presented in the previous chapter, which requires time $O(k + n^3 \log n)$ where k is the size of the complex, which is also the size of the skeleton), but these approaches suffer from all the problems described previously in Section 2.2 and in the conclusions of the previous chapter.

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

5.2 Point-to-area form-factor for vertices

The calculation of form factors has become central in many global illumination methods. The form factor between two surfaces is the ratio of light leaving one surface which arrives at one other. In many radios-

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

Table 3.1: Construction statistics (all times on a 195Mhz R10000 SGI Onyx 2). The storage is indicated for our initial data structure (*i.e.* including the two-dimensional array). It is scene dependent and is greatly reduced with our improved data-structure.

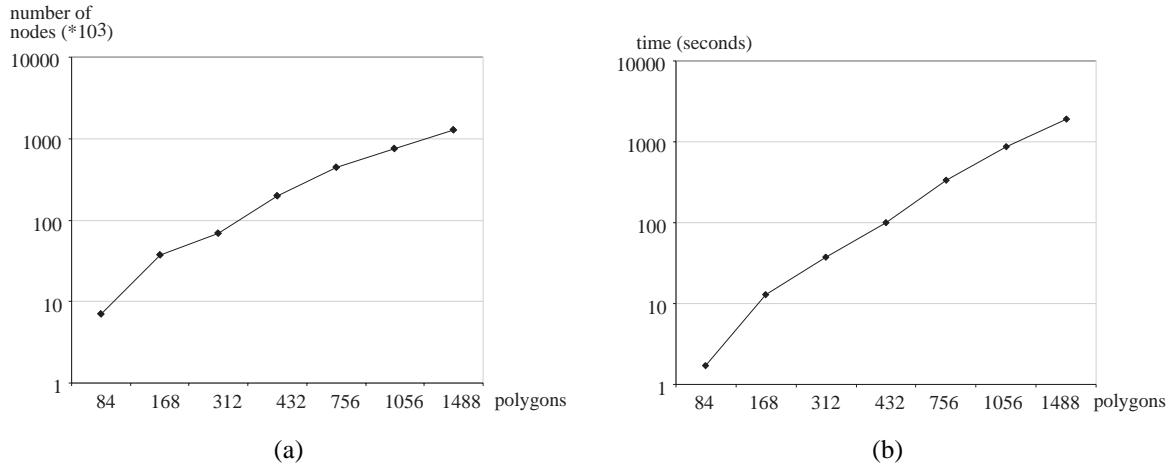


Figure 3.15: (a) Logarithm-logarithm graph of the number of nodes. The slope of about 2 indicates a quadratic growth. (b) Logarithm-logarithm graph of the computation time. The slope of about 2.4 indicates that the growth is in $n^{2.4}$.

ity lighting simulation systems, point-to-area calculations are used to approximate area-to-area calculations [CG85, BRW89], and in others the actually point-to-area value is computed at the vertices [WEH89].

In both theoretical [LSG94] and experimental [DS96] studies, previous research has shown that error of the visibility calculation is a predominant source of inaccuracies. This is typically the case when ray casting is used. Lischinski *et al.* [LSG94] have developed a very promising approach to bounding the error committed during light transfer for hierarchical radiosity. For it to be useful for general environments, access is required to the exact visibility information between a point on one element with respect to the polygon face it is linked to. This information is inherently global, *any* two surface elements of the scene can interact.

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

To calculate the view of a polygonal face from a vertex v , with respect to a face f , we access all the *EV* arcs of the skeleton related to the face f . This is simply the traversal of the line of our global two-dimensional array of arcs, indexed by f (or the traversal of the search tree stored at f for the improved structure). For each entry of this list (many of which are empty), we search for the *EV* arcs related to v . These *EV* arcs are exactly the visible boundary of f seen from v . This is done efficiently because the arcs in the search trees are sorted in a lexicographic order where the first key is the generators. In particular, we have chosen to use the vertex identifier before the edge to order *EV* arcs.

An example is shown in Fig. 3.16(a) and (b). For these scenes, containing 312 and 1488 polygons, the extraction of the point-to-area boundary takes respectively 1.2 ms and 1.5 ms (all query times are given without the cost of displaying the result).

The next chapter will present a global illumination simulation algorithms which uses the visibility skeleton to calculate exact point-to-area form factors.

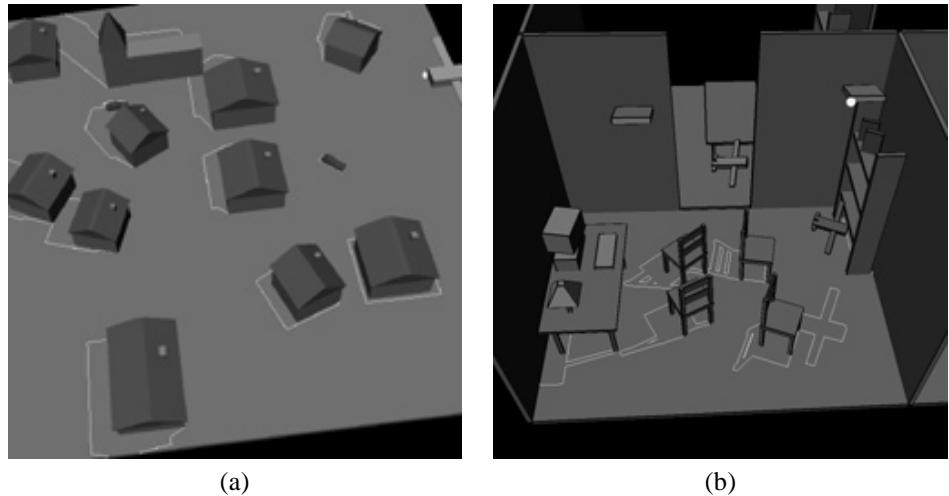


Figure 3.16: (a) Part of the floor visible from a vertex of the airplane. (b) Part of the floor seen by a vertex of the right-hand light source.

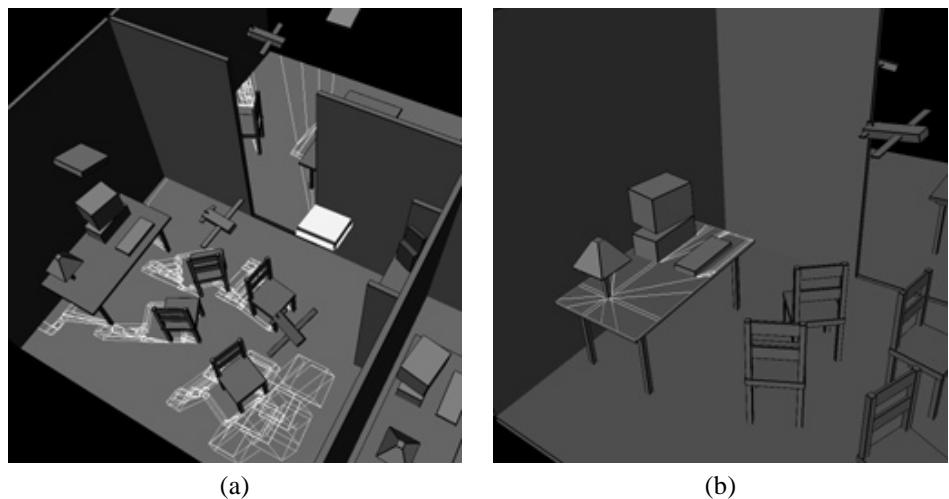


Figure 3.17: (a) The discontinuity mesh with respect to the right source. (b) Discontinuity mesh between the lamp and the table.

5.3 Global and on-the-fly discontinuity meshing

In radiosity calculations, it is often very beneficial to subdivide the mesh of a surface by following some [Hec92a, LTG93], or all [DS96] of the discontinuity surfaces between two surfaces which exchange energy. The partial [Hec92a, LTG92] or complete [DF94, SG94] construction of such meshes has in the past been restricted to the discontinuity mesh between a primary light source (which is typically a small polygon) and the receivers (which are the larger polygons of the scene). For all other interactions between surfaces of scenes, the algorithmic complexity and the inherent robustness problems related to the construction of these structures has not permitted their use [Tam93].

For many secondary transfers in an environment, the construction of a global discontinuity mesh (*i.e.*, from any emitting/reflecting surface to any other receiving surface in a scene), can aid in the accuracy of the global

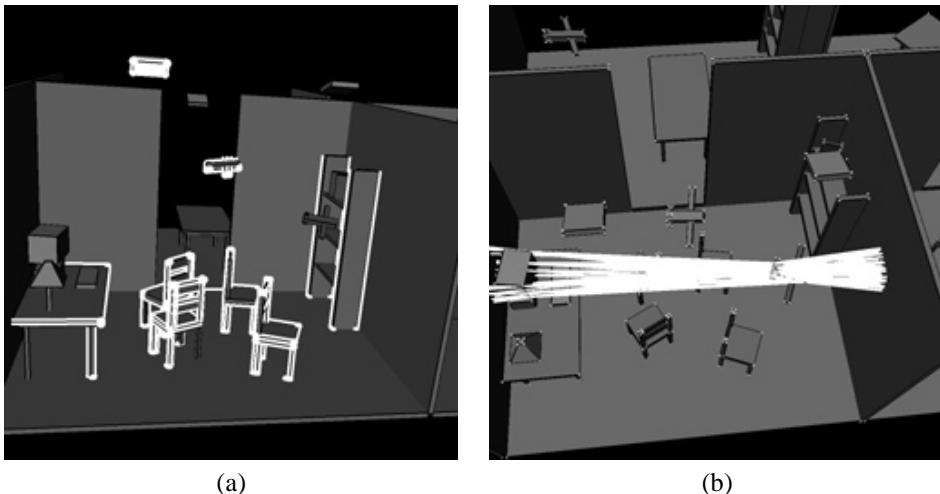


Figure 3.18: (a) List of occluding blockers between the left light source and the floor. Note that the objects on the table that are invisible from the floor are not reported as blockers. (b) Limits of the occlusions caused by a part of the plane between the computer screen and the right wall.

visibility computation. This was shown in the high-quality radiosity system by Hardt and Teller [HT96]. In their case, the discontinuity surfaces are simply intersected with the scene polygons for subdivision, while the visibility information encoded by the line swath is not considered. Moreover, they do not treat *EEE* swaths. With the Visibility Skeleton, the complete global discontinuity mesh between two surfaces can be efficiently computed.

Consider a polygon f_i as a light source, and a polygon f_j as a receiver. The discontinuity surfaces caused by f_i on f_j correspond to the arcs of the visibility skeleton which have one generator on f_i and one extremity on f_j , together with the arcs which have one extremity on f_i and one on f_j . The latter category simply corresponds to the arcs stored in the two dimensional array indexed by f_i and f_j (or in the node of the search tree corresponding to f_i and f_j). The former kind of arcs are stored in the row of the array indexed by f_j , and can be found using the search trees of arcs in a way similar to view extraction.

However, for view extraction only one type of generator is searched, a vertex, since an arc is never generated by more than one vertex, as opposed to *EEE* which are generated by three edges. Enumerating the relevant arcs no longer corresponds to a search on the first key in the search tree. Multidimensional search trees could be used to cope with this. We have nevertheless chosen a simpler approach.

We add an additional two-dimensional array $DM(i, j)$, storing all the arcs with one generator on face f_i and one extremity on f_j . Insertion into this array of lists and well as subsequent access is performed in constant time. To extract the discontinuity mesh between two surfaces f_i and f_j we simply access the entry $DM(i, j)$, and the cell of the original array storing the arcs with extremity f_i and f_j , and consider the corresponding arcs. In Fig. 3.17(a), the complete discontinuity mesh between the source and the floor is extracted in 28.6 ms. The mesh caused by the small lamp on the table in Fig. 3.17(b) was extracted in 1.3 ms (recall that the arrangement is not built).

This additional array is replaced by a search tree stored at the polygons for the improved structure. It will prove particularly useful for lighting simulation, because all the information related to the light transfer between f_i and f_j is stored at the corresponding polygons (see the next chapter).

The resulting information is a set of arcs. These arcs can be used as in Hardt and Teller [HT96] to guide subdivision, or to construct the arrangement of the discontinuity mesh on-the-fly, to be used as in [DS96] for the construction of a subdivision which follows the discontinuities. The adjacency information available in the Skeleton arcs and nodes should permit a robust construction of the mesh arrangement.

5.4 Exact blocker lists, occlusion detection

When considering the interaction between two surfaces f_i and f_j , it is often the case that we require access to the exact list of blocker surfaces hiding one surface from the other. This is useful in the context of blocker list maintenance approaches such as that presented by Teller and Hanrahan [TH93].

The Visibility Skeleton can again answer this query exactly and efficiently. We use the cell of the initial array of the arcs having one extremity on f_i and one on f_j , as well as the cell $DM(i, j)$, and we traverse the related arcs. All the polygons related to the intervening arcs are blockers. It is important to note that this solution results in the *exact* blocker list, in contrast with all previous methods. Consider the example shown in Fig. 3.18(a) where we compute the occluders between the left ceiling lamp and the floor in 4 ms.

The *shaft* structure [HW91] would report all objects on the table even though they are hidden by the table. In this case the Visibility Skeleton reports the exact set of blockers.

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

Similarly, the Skeleton allows for the detection of the occlusions caused by an object. This can be very useful for the case of a moving object m allowing the detection of the form factors to be recomputed. To detect if the form factor F_{ij} has to be recomputed we perform a query similar to the discontinuity mesh between two polygons: we traverse the corresponding cell of the array and $DM(i, j)$, and search for an arc caused by an element (vertex, edge or face) of m . This gives us the limits of occlusions of m between f_i and f_j . Moreover, by considering all the arcs of the skeleton, we report all the form factors to be recomputed, and not a superset. Fig. 3.18(b) shows the occlusions caused by the body of the plane between the screen and the right wall. This computation required 1.3 ms.

6 Dealing with spatial complexity: on-demand construction

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

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

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

We propose two approaches (but only the first one has been implemented): a source driven computation, and an adaptive subdivision of ray-space in the spirit of [AK87].

In the context of global illumination, the information related to “sources” (emitters or reflectors) is crucial. Thus the part of the visibility skeleton we compute in an on-demand construction is related to lines cutting the sources. The node detection has to be modified: every time a double wedge or a face does not cut the source, the pair of edges or the face is discarded, and if a potential node is detected, the ray-casting is performed only if the corresponding critical line cuts the source.

We use our grid-acceleration scheme here too: for the *EEE* restriction, for an edge e_i , we form an edge paironly for the edges e_j that lie inside the hourglass defined by the source and the first edge. A similar scheme is applied for *VV* detection.

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

For scene (g) of Table 5.1, the part of the Visibility Skeleton with respect to one of the sources is computed in 4 min. 15 s. instead of 31 min. 59 s. for the entire scene.

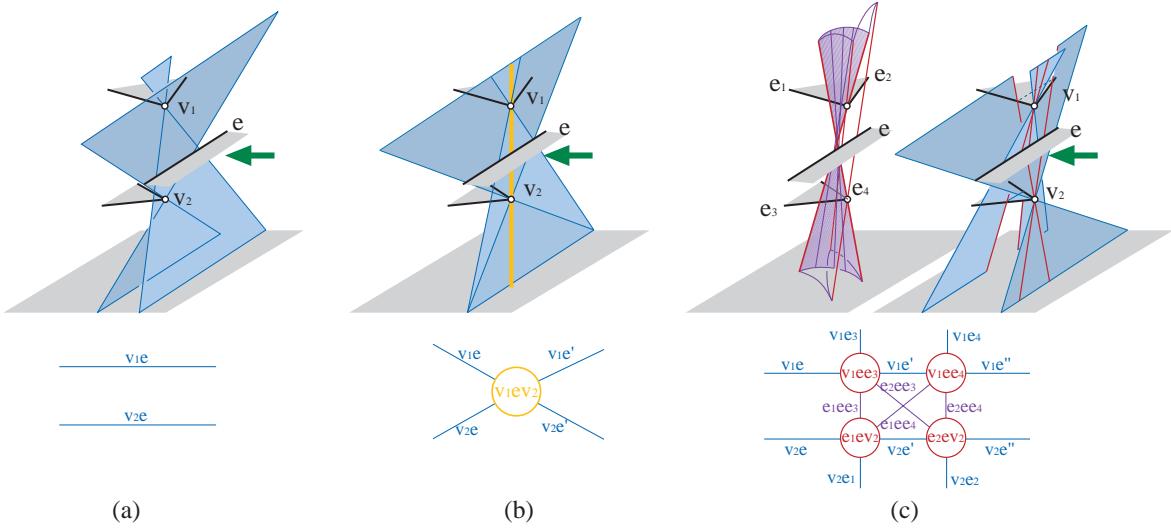


Figure 3.19: Dynamic update of the skeleton. Edge e moving from right to left causes a VEV temporal visibility event which is the meeting of two EV with the the two same extremities and with a common generator (here the edge e). Four nodes are created, the EV arcs are split into three parts and eight arcs are created. These events and the topological visibility changes are local in the visibility skeleton.

When the number of sources becomes large, most of the time would be spent in checking if lines intersect the sources or if they have already been computed. If we need visibility information only between two objects, not between an object and the whole scene, we propose the use of ray classification [AK87] together with the notions of dual space introduced in the previous chapter to build the visibility skeleton only where and when needed. The idea (which is not currently implemented) is to parameterize the lines of the 3D space (which is a set in 4D space), for example by their direction and projection on a plane or by their intersections with two parallel planes. We then perform a subdivision of the space of lines with a simple scheme (e.g., grid, hierarchical subdivision) and compute the nodes of the complex located inside a given cell of this subdivision.

7 Dynamic scenes

We present in this section issues concerning the handling of dynamic scenes. These are only sketches of algorithms, no implementation actually exists. We believe that the importance of this topic justifies an exposition, even in this preliminary form.

7.1 Temporal visual events

When an object moves, the topology of the visibility skeleton is modified only when a *temporal visibility event* occurs, as presented in section 3 of the previous chapter.

We present here the example of a VEV event, and show how the Visibility Skeleton is modified. Next section will show how the occurrence of such an event can be detected. Other kinds of visual events can occur ($EEEEEE$, $EEEV$, FFE , $FEEE$, FEV , ...) and can be treated in a similar way.

On Fig. 3.19 the edge e moves from right to left. The two EV arcs v_1e and v_2e meet at the temporal visibility event v_1ev_2 .

The two EV arcs are split into three parts because of the occlusion caused by the other vertex. Four nodes v_{1ee3} , v_{1ee4} , e_1ev_2 and e_2ev_2 are created as well as four EEE arcs e_1ee3 , e_1ee4 , e_2ee3 and e_2ee4 . The reciprocal change can happen if the edge moves in the other direction.

These changes are local in the visibility skeleton, even if the corresponding edges, vertices and faces are far away in the scene.

7.2 Temporal visual event detection

Detection of a *VEV* event could be done as follows. The two meeting *EV* always share the two same polygons at their extremities. This is always true, crossing arcs always have the extremities of the extremal stabbing line corresponding to the temporal visibility event. They share a common generator, and of course at least one of these arcs is related to the dynamic object. Thus a dynamic update of the Visibility Skeleton should, for each arc related to the dynamic object, find the next potential temporal visibility event.

Consider the arc v_1e related to the dynamic object. In order to meet v_1e another arc should have the two same extremities as v_1e and should be related to either e or v_1 . Such arcs can be found quickly by searching in the same list of the array if it is organised as a search structure on both the edge and the vertex (for example as a KD-tree). Then a potential temporal event is computed.

All the actual temporal events are then sorted, and processed in order. When an event occurs, the visibility skeleton is locally updated. The temporal event list has to be updated too. The modified arcs are checked for meeting with other arcs, and the temporal events related to destroyed arcs are removed.

If most of the objects in the scene move, we search for each list of the arrays of arcs if there are meeting arcs in it. All arcs having a generator (face, edge, vertex) in common are checked for meeting.

7.3 Potential applications: form factor maintenance

As an example of a useful application, let us show how this can be applied in a context of global illumination of dynamic scenes. In radiosity simulation with moving objects, some form-factors should be recomputed: those ones related to the moving object, but also form-factors between patches partly occluded by the moving object. Moreover, pairs of objects may become mutually visible or mutually invisible, requiring to set or discard a representation of the corresponding light transfert. The Visibility Skeleton proves to be an efficient tool for detecting all these modifications.

The mutual visibility of two polygons f_1 and f_2 is affected by an object, iff there is an arc of the skeleton related to this object in the lists of arcs between f_1 and f_2 . This gives an exact characterization of form factors to be updated.

The need for form-factor recomputation will start and end a iff a temporal visibility event occurs which has a generator or extremity on each face. For example, in Fig. 3.19, the form factor between the two faces adjacent to v_1 and v_2 starts to need recomputation because of the occlusion caused by e . The recomputation can be made locally in the corresponding view itself: consider for example the point-to-area form factor between v_1 and the down face. At any time the quantity to remove from the form-factor corresponds to the area that becomes hidden by e .

See the work by Orti *et al.* [ORDP96] for a development of the 2D equivalent.

8 Conclusions

8.1 Summary

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

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

The implementation shows that despite unfavorable asymptotic complexity bounds, the algorithm is manageable for the test suite used, both in storage and in computation time. In addition, we have developed and

implemented a first approach to on-demand or lazy construction which opens the way to hierarchical and progressive construction techniques for the Skeleton.

8.2 Discussion and future work

Robustness

Robustness and scalability are the two major issues of future work. The locality of our construction algorithm limits the effect of errors, and our treatment of degeneracies permits the handling of most of the cases encountered in computer graphics scenes. Nevertheless, their implementation has been tedious and we can not claim that all situations are handled.

We do not believe that robustness problems should be treated using exact arithmetic. Practical scenes exhibit many degeneracies such surface contact or parallelism which should be taken into account and not treated as if objects were in general position. If two object touch, we do not want to compute a visual event between them because the last digits are slightly different. We want to detect that this is a degenerate case and conclude that no visibility is possible through them. This does not prevent the use of precise (if not exact) arithmetic to detect the degenerate cases. However, a specific treatment should then be applied.

Unfortunately this makes the implementation very tedious, since it dramatically increases the number of different cases to handle. To cope with this, a general definition of extremal stabbing lines could be used. Define an extremal stabbing line as a line going through n edges (with $n \geq 4$) (see Fig. 3.20). The adjacent arcs can be enumerated by considering all triples of its generating edges. The configuration of each triple of edges can then be tested (including tests to detect degeneracies) to decide if the arc should be discarded or treated. This permits the use of a single procedure where our implementation requires specific code for each sort of extremal stabbing line. Specific code should however be written for each arc configuration (EV , EEE , FE , or Fv).

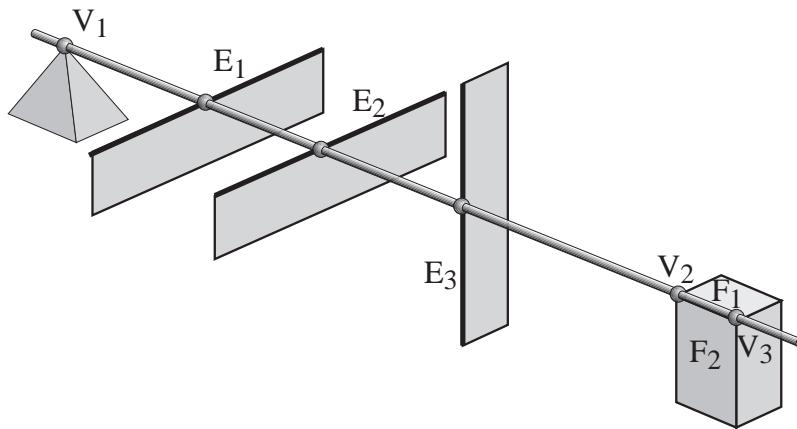


Figure 3.20: General definition of extremal stabbing lines.

Consider the example in Fig. 3.20. The generators are first sorted along the line. The arcs generated by V_1 are enumerated first. Two V_1E_1 arcs are created, depending on the right extremity. V_1E_2 is discarded because of the occlusion by E_1 . For triples or pairs of generators, the procedure simply consist in testing occlusion by another generator in between, then finding the extremities (which is an extremity of the extremal stabbing line or a face adjacent to one generator).

Another approach to improve the robustness of our construction is to *a posteriori* repair the skeleton. As we have seen, an error incurs only local incoherences in the skeleton. A post-process could be used which links in a coherent (if not exact) way the pendant “hanging” arcs and nodes of the skeleton, or which removes the unnecessary features. The next section will also discuss some aspects of the robustness issue.

Scalability

Though the practical behaviour of our construction algorithm is much lower than its theoretical complexity, a quadratic cost is not acceptable for practical use with large scenes. We think that a visibility data-structure should be developed which allows a trade-off in precision as the objects get more distant from each other. A hierarchical approach should be explored, computing an exact visibility skeleton for clusters of objects, and another data-structure between different clusters. We have already studied such an approach in 2D [Dur95].

However, an approximate visibility structure between clusters of objects is hard to define because the definition of approximate visibility itself is far from straightforward. A cluster of objects is not opaque, transmittance should be considered. The work by Soler [Sol98, SS96b] on approximate visibility is an interesting step towards refinement in visibility computation.

The Visibility Skeleton is a graph structure. Graph literature has certainly a lot to tell us. What are the properties of the skeleton as a graph? Graph compression techniques could be explored. Consider the example of two triangles with full visibility. This situation could be detected and factorized to compress the visibility information.

Simplifying the skeleton through vertex removal or collapse operations is an interesting issue. The effect of such operations on the visibility information encoded should be carefully analyzed, especially their global consequences. For example, when considering the umbra of a highly tessellated object, some events could be collapsed. This however eliminates some visibility information at the vertices of the object.

Of course the application-specific restriction can help guide simplification as well as the definition of approximate visibility. Discontinuity meshing and visible part computation for lighting simulation provide a perfect context since hierarchical frameworks have been developed for global simulation [HSA91, SAG94, Sil95].

Moreover, we believe that scalability and robustness should be studied together, since a hierarchical approach naturally defines a notion of scale which can be used to set thresholds for degeneracy detection. The same ϵ can be used to decide that a visibility information is not “noticeable” and that two features are in a degenerate configuration.

Other issues

The Visibility Skeleton can be defined for scenes of curved objects as the one and zero-dimensional faces of the corresponding 3D Visibility Complex. Visual events are described by the theory of the singularities of smooth mappings (see appendix A). Unfortunately, the extension of our construction algorithm is not straightforward. The major issue is the parameterization of arcs, which can no longer be based on intercepts along edges, making it difficult to define an order for the search trees, especially in the case of concave objects.

The update of the skeleton after the addition or removal of objects is different from the maintenance in the case of smooth motion. Because of its locality, our construction algorithm could be adapted to recompute visibility only where relevant to the modified object.

The visibility skeleton permits in a sense to “emulate” the visibility complex thanks to the search trees. The zero and one dimensional boundaries of a 4-face of the complex are encoded in one binary tree in the skeleton. For a given pair of polygons, the related 4-faces are encoded in the corresponding cell of the array in the initial structure or stored at the polygons in the second. This should be explored further.

In the next chapter, we apply the visibility skeleton to the concrete problem of global illumination using a hierarchical radiosity method. We show that it permits efficient and accurate simulation, providing high quality shadows even in hard configuration such as scenes lit by many sources or illuminated mainly by indirect lighting.

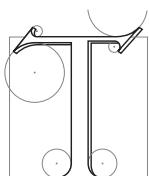
The applications of the skeleton to other techniques should be developed. In particular in computer vision, the aspect graph [Kv76, Kv79, EBD92], the visual hull [Lau94, Lau95, Lau97, Lau99] or the computation of occlusion-free viewpoints [TTK96] could benefit from the visual events encoded by the skeleton. In robotics, visibility based pursuit-evasion motion planning [LLG⁺97, GLL⁺97, GLLL98] also requires a partition of the scene by visual events. However, the exponential cost of the graph-search algorithm implied by the existing 2D techniques makes their direct extension to 3D a challenge.

CHAPTER 4

Visibility Driven Hierarchical Radiosity

Les reflets des parties éclairées rejoaillissent 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 Urbinae*



HIS CHAPTER describes the use of the Visibility Skeleton to answer visibility queries encountered in lighting simulation. Recent advances in global illumination, such as hierarchical radiosity [HSA91] and its combination with discontinuity meshing [LTG93, DS96] have resulted in high quality lighting simulations. These lighting simulations are *view independent* and are suitable for walkthroughs. The quality of the resulting illumination is important everywhere in the scene, since the user can, for example, approach a shadow of an object and see its details.

Despite the high quality of existing techniques, certain aspects of these algorithms are still suboptimal. In particular, deciding when a light-transfer is *refined* appropriately, and thus computed with higher precision is a hard decision; current algorithms ([HSA91, LSG94, GH96] etc.) include methods based on error bounds which in many cases prove insufficient. Creating a *mesh* to represent lighting variations accurately (notably for shadows) is hard; discontinuity meshing approaches [LTG93, DS96] have proposed some solutions for these issues which are however often limited in their applicability. Recent approaches (*e.g.*, [CSSD96, UT97]) avoid this problem by performing a view-dependent, ray-casting “final gather”; view-independence and the capacity for interactive display and walkthroughs are thus sacrificed. Accurate *visibility* calculation is also fundamentally hard, since we have to consider the potential interaction between all polygons in the scene for global illumination.

The above three problems, *visibility*, *refinement* and *meshing* are accentuated in the following two lighting configurations: scenes lit by multiple sources and scenes lit mainly by indirect illumination. In this chapter we present a new algorithm which addresses the three shortcomings mentioned above. For all three problems, previous approaches lack information on accurate *global visibility* relationships in the scene. This information is provided by the *Visibility Skeleton*. The Skeleton allows the fast computation of exact point-to-polygon form-factors for any point-polygon pair in the scene. In addition, all visibility information (blockers and all discontinuity surfaces) is available for any polygon-polygon pair.

This global visibility information allows us to develop an intelligent refinement strategy, since we have

knowledge of visibility information for *all* light transfers from the outset. We can *rank* discontinuity surfaces between any two hierarchical elements (polygons or patches resulting from their subdivision), using perceptually-based techniques [GH97b]; thus only discontinuities which are visually important are considered. An appropriate mesh is created using these discontinuities; illumination is represented very accurately resulting in high-quality, view-independent meshes. To achieve this in the context of a hierarchical radiosity algorithm, we have introduced a hierarchy of triangulations data structure. Radiosity is gathered and stored at vertices, since the Skeleton provides us with the exact vertex-to-polygon form-factor. An appropriate multi-resolution push-pull procedure is introduced. The high-quality mesh, the exact form-factor calculation and the hierarchical triangulation result in lighting simulation with accurate visibility.

Our approach is particularly well-suited for the case of multiple sources since the discontinuity ranking operates simultaneously on *all* light energy arriving at a receiver. Indirect illumination is also handled very well, since visibility information, and thus the refinement and meshing strategies as well as the form-factor computation apply equally well to all interactions, *i.e.*, both direct (from the sources) and indirect (reflected light). Examples of these two cases are shown in Fig. 4.1. In Fig. 4.1(a) we see a scene lit by 10 separate light sources, where the multiple shadows are visible but the mesh complexity is reasonable (see Table 4.2, in Section 6.2). In Fig. 4.1(b) we see a room lit mainly by indirect light; notice the high quality shadows created entirely by indirect light (*e.g.*, on the far wall from the books and lamp).

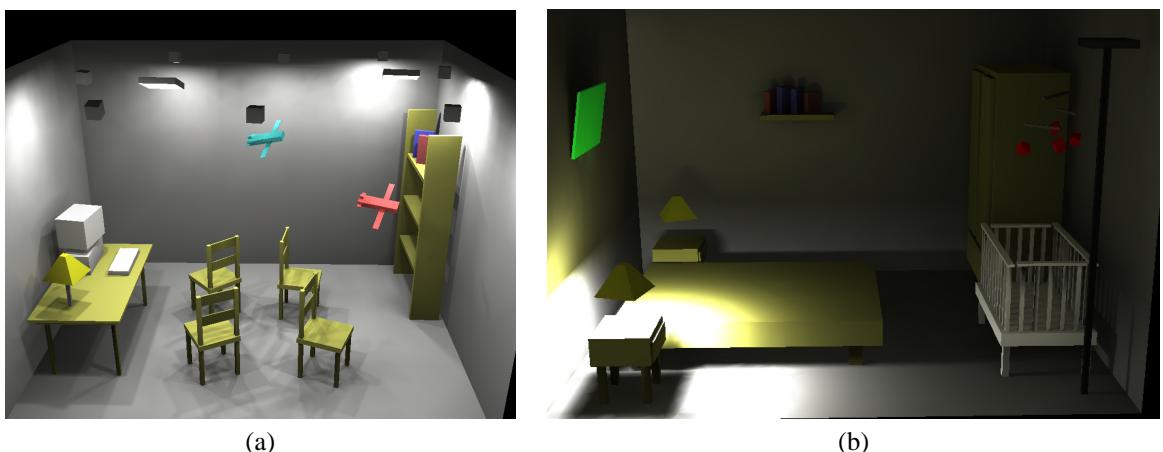


Figure 4.1: Images computed using our new hierarchical radiosity algorithm based on the Visibility Skeleton and hierarchical triangulations. (a) A scene with multiple sources. The skeleton construction took 2min 23s and the lighting simulation 8min. (b) A scene mainly lit by indirect light. The skeleton construction took 4min 12s and the lighting simulation 6min 58s. Note the shadows caused by indirect illumination, cast by the books on the back wall.

The new algorithm we present here is in a certain sense an extension of hierarchical radiosity, using visibility structures, advanced meshing techniques and perceptually-based subdivision. We briefly review hierarchical radiosity methods, accurate visibility techniques and related visibility-based refinement for lighting algorithms and finally perceptually-based refinement for illumination.

The work presented here has been accepted for publication in *ACM Transaction on Graphics* [DDP99]. This chapter adds a description of possible improvements to decrease the memory consumption of our method in section 7.

The chapter is organised as follows. After a presentation of previous work, the details of our multiresolution structure, and the novel push-pull algorithm are presented in Section 2. In Section 3 we present the new hierarchical radiosity algorithm using accurate global visibility and we present the new point-polygon and polygon-polygon link data structures. In Section 4 we present the corresponding refinement processes and the visibility updates required for their use. In Section 5 polygon subdivision and the perceptually-based refinement criterion are described. We then present results of our implementation, as well as a discussion of possible improvements. We conclude with a discussion of relative limitations and advantages of our approach.

1 Motivation and previous work

1.1 Hierarchical radiosity

The hierarchical radiosity algorithm [HSA91] allows efficient calculation of global illumination. Lighting calculations are limited to a user-specified level of accuracy, by means of hierarchically subdividing the polygons in the scene into a quadtree, and creating light-transfer “links” at the appropriate levels of the hierarchy. In the original hierarchical radiosity solution [HSA91], radiosity is considered constant over each quadtree element. The rectangular nature of the quadtree, and the constant reconstruction result in the need for very fine subdivision for high quality image generation (high quality shadows etc.).

Higher-order (non-constant) methods have also been introduced, notably in the context of wavelet-based solutions [GSCH93]. The wavelet-based radiosity solutions presented to date typically operate on discontinuous bases, resulting in visible discontinuities if the solution is displayed directly (*e.g.*, [CSSD96]). Zatz [Zat93] used a Galerkin-type method and shadow masks to improve the quality of the shadows generated. To avoid the problem of discontinuous representations the “final gather” step was introduced by [Rei92] and used for wavelet solutions (*e.g.*, [CSSD96]). A final gather step consists of creating a ray-cast image, by querying the object-space visibility and lighting information to calculate illumination at each pixel [UT97]. This approach allows the generation of high quality images from a coarse lighting simulation, at an additional (frequently high) cost. The solution thus becomes view-dependent, and interactive display and walkthrough capability are lost.

More recently, Bekaert *et al.* have presented an efficient algorithm which combines hierarchical radiosity and Monte-Carlo radiosity [BNN⁺98]. However, the stochastic nature of the algorithm makes it difficult to refine along shadow boundaries.

1.2 Accurate visibility and image quality

The accurate calculation of visibility in a lighting simulation is essential: both the numerical quality of the simulation and the visual quality of the resulting image depend on it. The exact computation of visibility between two patches in a scene or between a patch and a point requires the treatment of the visual events studied in the previous chapters. In the case of the view of a light source, these boundaries correspond to the limits of umbra and penumbra. By choosing certain of these boundaries and using them to guide the (irregular) mesh structure, *discontinuity meshing* lighting algorithms have been introduced resulting in more visually accurate images (*e.g.*, [Hec92a, LTG92]).

The determination of the visible part of an area light source in computer graphics is exactly the calculation of the aspect of the light at a given point, in the sense of the aspect graph literature [PD90, GM90, EBD92]. Algorithms performing this operation by building the complete discontinuity mesh and the *backprojection* data structure (encoding the source aspect) have been presented (*e.g.*, [Tel92a, DF94, SG94]). The full discontinuity mesh and backprojection allows the computation of the exact point-to-area form-factor with respect to an area light source. Nonetheless, these methods suffer from numerical problems due to the required intersections between the discontinuity surfaces and the scene polygons, complicated data-structures to represent the highly irregular meshes and excessive computational requirements.

We have chosen to use the Visibility Skeleton because of its flexibility, relative robustness (compared to discontinuity meshing) and ease-of-use.

1.3 Visibility-based refinement strategies for radiosity

In the Hierarchical Radiosity algorithm, mesh subdivision is effected through link refinement. The original algorithm used a *BFV* criterion (radiosity times form-factor modulated by a visibility factor V for partially occluded links). The resulting meshes are often too fine in unoccluded regions, and do not always represent fine shadow details well.

Refinement strategies based on error bounds [LSG94, GH96] have improved the quality of the meshes and the simulation compared to the *BFV* criterion. Conservative visibility determination in architectural scenes [TH93], accurately characterises links as *visible*, *invisible* or *partially visible*. This triage guides subdivision, allowing finer subdivision in partially illuminated regions.

Discontinuity meshing clearly improves the visual quality of images generated by lighting simulation [LTG92, Hec92a, DF94], since the mesh used to represent illumination follows the actual shadow boundaries, instead of finely subdividing a quadtree which attempts to approximate the boundary. One problem is the extremely large number of discontinuities. Tampieri [Tam93] attempted to limit the number of discontinuity lines inserted, by sampling the illumination along the discontinuities and only inserting those with radiosity values differing more than an predefined threshold. In the context of progressive refinement radiosity, Stuerzlinger [Stu94] only inserted discontinuities at a second level of a regular quadrilateral adaptive subdivision, once a ray-casting step has classified the region as important. The only discontinuities inserted were those due to the blocker identified by the ray caster.

Several methods combining discontinuity meshing with hierarchical radiosity have been presented [LTG93, DS96, HT96, BP95]. Hardt and Teller [HT96] present an approach in which potential discontinuities from all surfaces are considered, without actually intersecting them with the blockers. Potential discontinuities are ranked, and those deemed most important are inserted at the lowest level of the quadtree. In [DS96] the backprojection information is used in the complete discontinuity mesh creating a large number of small triangles. Exact form-factors of the primary source are then computed at the vertices of these triangles. The triangles are then clustered into a hierarchy. Standard [HSA91] constant-element hierarchical radiosity follows. The previously cited problems of discontinuity meshing, the expensive clustering step and the fact that the inner nodes of the hierarchy often overlap, limit the applicability of this approach to small models. The only other hierarchical radiosity method with gathering at vertices is that of Martin *et al.* [MPT97], which requires a radiosity value at each vertex, and a complex push procedure.

The algorithm of Lischinski *et al.* [LTG93] is much more complete and relevant to our work. The basis of this approach is to separate the light simulation and rendering steps. This idea is similar in spirit to the use of a “final gather” step, but in a view independent manner. They first compute a “global pass” by creating 2D BSP trees on scene polygons subdivided by choosing important discontinuities exclusively due to the primary sources. The 2D BSP tree often incurs long splits and consequently long or thin triangles, which are inappropriate for high quality lighting simulation. The second, view independent, “local pass” recomputes illumination at the vertices of a triangulated subdivision of the leaf elements of the BSP tree. To achieve high quality images, the cost of triangulation and shading (light recomputation at vertices using “method D” [LTG93]), is higher than that of the actual lighting simulation (if we ignore the initial linking step).

1.4 Perceptually based refinement

Recently, perceptually-based error metrics have been used to reduce the number of elements required to accurately represent illumination (*e.g.*, [GH97b, HWP97]). Tone-reproduction approaches [TR93, War94] are used to map calculated radiosity values to display values which convey a perceptual effect closer to that perceived by a real world viewer. Since display devices have limited dynamic range compared to real world luminance values, the choice of this mapping is very important. The tone reproduction mappings of [TR93, War94] depend on two parameters: a world adaptation level which corresponds loosely to the brightness level at which a hypothetical observer’s eye has adapted, and a display adaptation level which corresponds to the brightness displayed on the screen. Choice of these parameters affects what will be displayed, and, more importantly, which differences in radiosities will actually be perceptible in the final image. Most notably, one can define a “just noticeable difference” using this mapping. In the context of lighting, a just noticeable difference would correspond to the smallest difference in radiosity values, which once transformed via tone reproduction, will be visible to the viewer of the display. Display adaptation is typically a fixed value (*e.g.*, half the maximum display luminance [War94]), while the world adaptation level can be chosen in a number of different ways. Using static adaptation [GH97b], one uses an average which is independent of where the observer is looking, while dynamic adaptation (which is closer to reality) changes depending on where the observer is looking. Gibson and Hubbold [GH97b] use tone reproduction to guide subdivision in a progressive refinement radiosity approach, thus allowing a subdivision only if the result will be “just noticeable”.

Hedley *et al.* [HWP97], in a similar spirit, use a tone mapping operator to determine whether a discontinuity should be inserted into a lighting simulation mesh. This is performed by sampling across the discontinuity (in a manner similar to that of Tampieri [Tam93]), but also orthogonally across the discontinuities. This results in an important reduction of discontinuities without loss of visual quality.

1.5 Discussion

Previous radiosity methods (surveyed above) provide view-independent lighting simulations which are acceptable for many situations. In particular, quadtree based hierarchical radiosity provides fast solutions of moderate quality, even for scenes mainly lit indirectly. Nonetheless, in walkthroughs the observer often approaches regions of shadow, and in these cases the lack of shadow precision is objectionable. Previous approaches based on discontinuity meshing subdivide the mesh along shadow boundaries and alleviate this problem for direct lighting, but rapidly become impractical for scenes with many lights, or for which indirect lighting is dominant. Their limitations are due to the sheer number of discontinuity surfaces that need to be considered when computing indirect illumination and the complexity of the meshes which result. These issues are discussed in [LTG92] and [Tam93]. The solutions adopted to date have restricted the use of discontinuity information to those from primary light sources [LTG93, DS96]; for subsequent light bounces (secondary, tertiary etc.), approximate ray-casting approaches are used for visibility computations in light transfer.

The new algorithm presented in this chapter allows the generation of accurate shadows for a more general class of scenes, including those with dominant indirect illumination. The Visibility Skeleton allows us to select and insert discontinuity lines for all light transfers, and to calculate exact point-to-area form-factors rapidly, using the visibility information provided. These choices required us to develop a new hierarchical radiosity algorithm, with gathering at vertices, based on embedded *hierarchical triangulations* allowing the mesh to follow discontinuity lines. We moreover describe a refinement criterion based on accurate visibility information and a perceptual metric which obviates the setting of arbitrary and intricate thresholds.

2 Irregular hierarchical triangulations and lazy wavelets

In previous work (*e.g.*, [Hec92a, LTG92]) it has been shown that the creation of a mesh well adapted to the discontinuities in illumination results in images of high visual quality. Incorporating such irregular meshes into a hierarchical radiosity algorithm presents an important challenge. As mentioned in chapter 1, most previous algorithms [LTG93, DS96] addressing this issue have restricted the treatment of discontinuities to those due to direct (primary) illumination.

The core of the problem is that two conflicting goals are being addressed: that of a simple regular hierarchy, permitting straightforward manipulations and neighbor finding and that of an essentially irregular mesh, required to represent the discontinuity information. The first goal is typically achieved using a traditional quadtree structure [HSA91] and the second typically by a BSP-type approach [LTG93].

In previous approaches, discontinuity information and accurate visibility were incorporated into constant-element hierarchical radiosity algorithms. In the case of the Skeleton, this would be wasteful, since we have all the necessary information to compute *exact* form-factor from any polygon in the scene to any vertex (see Section 4 to see how this is also true for vertices resulting from subdivision). Gathering to vertices introduces one important complication: contrary to elements whose level in the hierarchy is clearly defined, vertices are shared between hierarchy levels.

As a solution to the above issues, we introduce hierarchical triangulations and lazy wavelets for hierarchical radiosity. Our approach has two major advantages over previous hierarchical radiosity methods: (i) it adapts well to completely irregular meshes and this in a local fashion (triangulations contained in triangulations), avoiding the artifacts produced by splitting edges of a 2D BSP tree and (ii) it allows gathering to vertices by a “lazy wavelets”-type (or sub-sampling) construction (see the book by Stollnitz *et al.* [SDS96] pp 102–104 and 152–154). It preserves a linear approximation to radiosity during the gather and the push process of the solution.

Note however that the piecewise linear representation will be used only for lighting representation during the “push” and display process. Gathering and “pull” will be performed on mean values on triangles. The simulation will be improved compared to standard Haar pyramid used in classical hierarchical radiosity because the values computed at a higher level will be linearly interpolated during the push process, resulting in a smoother representation of the mean values at the lower levels.

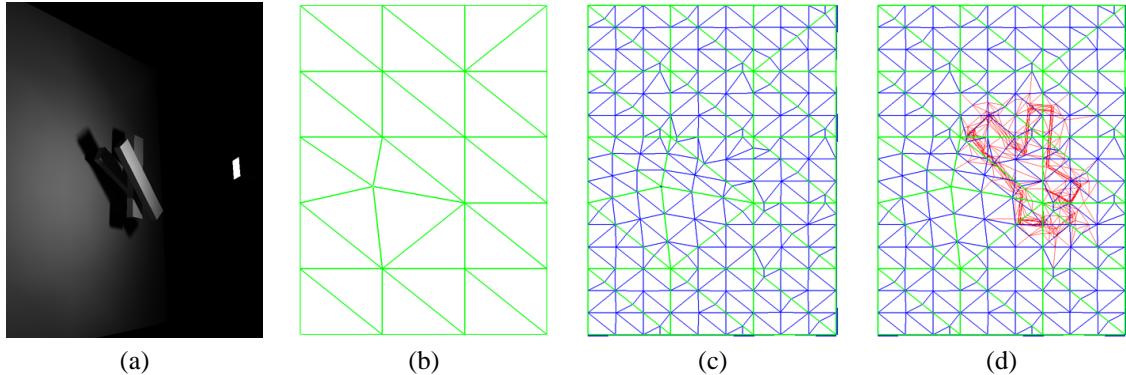


Figure 4.2: Hierarchical Triangulation Construction. Notice how the triangles are overall well shaped but also well adapted to local detail. (a) Scene geometry: the leftmost polygon is illuminated by the area source on the right pointing leftwards. (b) First level of subdivision for the leftmost polygon (green). (c) Second level (blue). (d) third level (red).

2.1 Hierarchical triangulation

Our hierarchical triangulation construction has been inspired by that of de Floriani and Puppo [DP95a]. The principle is straightforward: it is a hierarchy of triangulations where each triangle of a level can be subdivided into a triangulation.

We start with an initial triangulation, which is a constrained Delaunay triangulation. The constrained Delaunay triangulation allows the insertion of constrained edges into the triangulation, which are not modified to satisfy the Delaunay property and thus remain “as is”. Each triangle of the initial triangulation can be subdivided into a sub-triangulation by the addition of new vertices and constrained edges, and so on recursively. At each level, a constrained Delaunay triangulation is maintained.

An example of such a construction is shown in Fig. 4.2, clearly showing the first advantage mentioned above. As we can see, the triangulation maintains well-shaped triangles everywhere in the plane, while providing fine details in the regions where this is necessary. The representation of such detail induces irregular subdivision at the finer levels.

Our hierarchical triangulation is “matching”, in the sense that edges split across two levels of a triangulation are done so at the same point on the edge. At the end of each subdivision step an “anchoring” operation is performed by adding the missing points in the neighboring triangles, thus resulting in a conforming triangulation across levels required for the push phase of hierarchical radiosity.

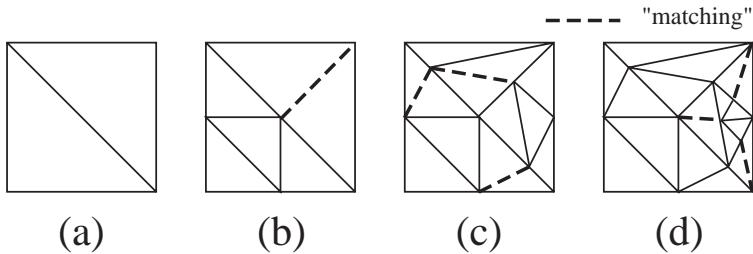


Figure 4.3: The “matching” constraint for the Hierarchical Triangulation. The sequence shows subsequent segment insertions. The dashed lines show the insertions performed to enforce the “matching” constraint.

As mentioned above, vertices are shared between different levels of the triangulation. The initial level of a triangulation is an *HPolygon*, which contains an *HTriangulation* child once subdivided, where the prefix *H* represents the hierarchical nature of the construction.

To transmit neighborhood information between levels (for the matching operation), we use a special *HEdge* structure. An *HEdge* is shared between hierarchy levels by all edges which correspond to the same segment. It contains pointers to sub *HEdge*’s when it is subdivided. To perform a matching operation we determine whether

the edge on which we insert a point p has already been split. We then add the new points corresponding to the previously split vertices, and split the *HEdge* at the point p . The neighbouring triangle can thus identify the newly inserted sub-*HEdge*'s from the shared *HEdge*. For example, in Fig. 4.3, after the subdivision of the lower left triangle in Fig. 4.3(a), the *HEdge* shared between the two triangles notifies the upper right triangle that the edge has been split, and facilitates the matching operation as shown in Fig. 4.3(b).

2.2 Piecewise linear multiresolution representation

The second advantage, that of linear reconstruction of illumination across irregular meshes, requires the use of a “lazy-wavelet” or “sub-sampling” type construction. Lazy wavelets provide an elegant formalism for a simple approach: a piecewise linear approximation is refined through the addition of new sampling points [SDS96]. The principle of lazy wavelets is illustrated in Fig. 4.4.

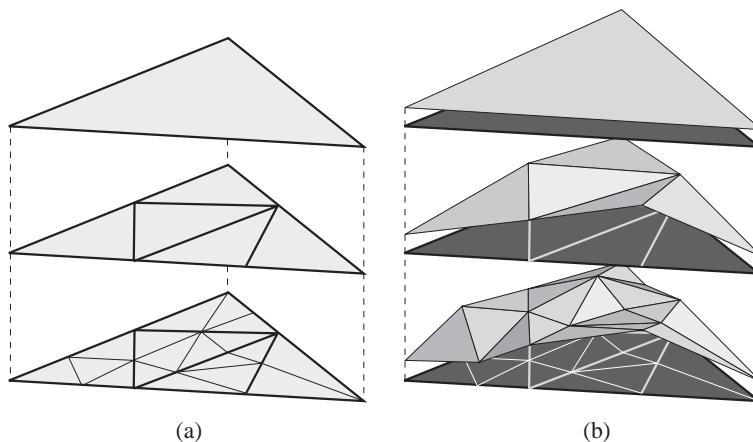


Figure 4.4: Principle of our multiresolution representation. (a) Hierarchical triangulation (the base triangle is on the top, and finer meshes are represented below) (b) Illumination function representation using lazy wavelets. Details are added to the function at vertices resulting from the subdivision of the triangles.

As mentioned above, in our hierarchical triangulation representation of radiosity, vertices will be shared between hierarchy levels. As a consequence, traditional push-pull procedures [HSA91] cannot be directly applied.

To understand why, consider the 1D example shown in Fig. 4.5. Segment $v_a v_b$ is illuminated by two light sources S_1 and S_2 . Assume that initially both light transfers are refined, and vertex v_1 is added. This results in the configuration of level 1 (Fig. 4.5). The light transfer with S_2 is further refined with the addition of v_2 on the right, thus splitting segment $v_1 v_b$. Finally, the light transfer from S_1 is refined on the left, with the addition of v_3 .

To determine the light contribution of S_1 in the interval $[v_1, v_b]$ we interpolate between the values transferred by $S_1 \rightarrow v_1$ and $S_1 \rightarrow v_b$, which are “represented” at level 1. However, for light S_2 , we must interpolate in the subinterval $[v_1, v_2]$ using the transfers determined by $S_2 \rightarrow v_1$ and $S_2 \rightarrow v_2$, and in the subinterval $[v_2, v_b]$ using the values determined by $S_2 \rightarrow v_2$, $S_2 \rightarrow v_b$, all of which are “represented” at level 2. Thus v_1 is shared between level 1 and level 2. As a consequence traditional push-pull procedures with gathering at elements rather than vertices cannot work, since they require that an element clearly belong to a certain hierarchy level.

A naive solution would be to duplicate vertex v_1 to differentiate exchanges simulated at different levels of the hierarchy. This however is not sufficient, since it is unclear how to perform the push operation. In particular, assume that we had one representation of v_1 for level 1 and one for level 2. It is unclear where the transfers $S_1 \rightarrow v_1$ and $S_2 \rightarrow v_1$ should be stored. If $S_1 \rightarrow v_1$ is stored at level 1, we can interpolate correctly in the interval $[v_1, v_b]$ to perform the push onto vertex v_2 . However the value will no longer be available at level 2 to enable the interpolation between v_3 and v_1 . In a symmetrical manner, we need $S_2 \rightarrow v_1$ to perform the interpolation at level 1 for the interval $[v_a, v_1]$ and the push on v_3 , and at level 2 for the interpolation in the interval $[v_1, v_2]$. With gathering at vertices and linear interpolation, it no longer makes sense to speak of a transfer at a given level of the hierarchy.

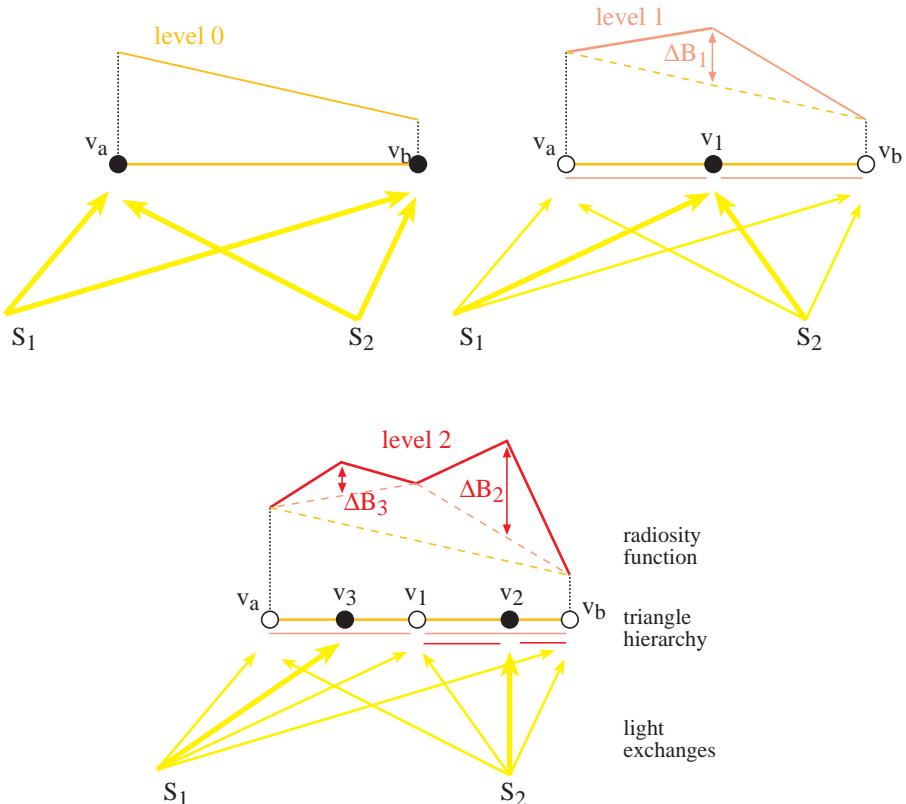


Figure 4.5: Consistent multiresolution representation with lazy wavelets. Instead of storing radiosity values, we store the difference of the radiosity values at refined vertices.

We use lazy wavelets to provide a solution to these problems. Instead of storing the actual radiosity value, at refined vertices we store the *radiosity difference* as shown in Fig. 4.5. This is the difference between the radiosity value at the current level and the interpolated value of the immediate ancestor. This provides a multi-resolution representation, since certain light transfers are refined more in the appropriate regions with the addition of new links.

The push procedure is then straightforward: To compute the total radiosity at a vertex, we interpolate the value of its ancestor, and add the radiosity difference. We obtain the total value at this vertex, which is thus recursively pushed down the hierarchy in a breadth-first manner.

This construction is directly applicable to the 2D case, by using barycentric coordinates (or bilinear for quadrilaterals) for the interpolation. We thus can simply perform a push operation on a hierarchical triangulation with gathering at the vertices.

However that it is slightly more involved to compute the difference of a light transfer than the total light transfer. Section 3.2 will deal with this problem through the use of “negative” links.

The pull computation is simpler, since we pull values to the triangles. At each triangle leaf, the value given is simply the average of values at the vertices (after the push). An intermediate node receives as a value the area-weighted average of its children triangles, as in standard hierarchical radiosity.

The advantages of this approach are that we can now create a consistent multi-resolution representation of radiosity over the hierarchical triangulation, while gathering at vertices. In addition, the push operation maintains a linear reconstruction of the radiosity function down to the leaf level.

The result is better than the smoothing post-process used by previous approaches, since in our case the interpolation is performed at all levels of the hierarchy, not only at the leaves. A typical artifact induced by classical smoothing is that the quadtree structure may be apparent with slightly smoothed discontinuities at the central horizontal and vertical lines. This comes from the fact that a transfer from a source S is computed at the

second level, while the wall is refined more finely for other transfers. The smoothing is performed only on the leaf values, which means that the values of the contribution of S are smoothed only for the leaf elements which are adjacent to the central lines, while it should be performed on the entire polygon.

3 Visibility-driven hierarchical radiosity: Algorithm and data structures

The hierarchical triangulation structure is one of the tools required to effect visibility-driven hierarchical radiosity. In particular, we can efficiently represent the irregular lighting discontinuities in a hierarchical structure. In addition, the information contained in the Visibility Skeleton provides *exact* and *global* visibility information. As a consequence, we can compute exact (analytical) area-to-point form factors for any light transfer, direct (primary) or indirect. The information contained in the Skeleton arcs (*i.e.* the visibility events affecting any light transfer) also allows the development of intelligent refinement criteria, again for any exchange of light.

In what follows we present our new algorithm which uses the Skeleton and the hierarchical triangulations for efficient refinement and accurate light transfer.

3.1 Algorithm outline

Our new algorithm is outlined in Fig. 4.6. It begins with the creation of the Visibility Skeleton for the given scene, using the improved link-based approach (Section 3.2 of chapter 3). After this step, we have all the information available to calculate form-factors from each polygon to each (initial model) vertex in the scene. In addition, polygon-polygon visibility relationships are available directly from the skeleton, thus obviating the need for initial linking (*i.e.* only necessary links are created). After computing the form-factors of the initial polygons to the initial vertices, a “gather” step is performed to the vertices, followed by a “push-pull” process. In practice we perform a fixed number of iterations; however it would be possible to iterate to convergence, since these iterations are not computationally expensive.

Even at this very initial phase, the form-factors at the vertices of the scene are exact. To bootstrap subdivision, we first insert the maxima of the light source illumination functions into large receiver polygons (procedure *insertMaxima()*, see also Section 5.2).

```
visibilityDrivenHierarchicalRadiosity
{
    computeSkeleton()           // compute the Visibility Skeleton
    computeCoarseLighting()     // 3 gather push-pull
    insertMaxima()              // insert the maxima of light sources into meshes
    while( !converged() ) do
        subdividePolygons()      // Refine the polygons using visibility info
        refineLinks()             // Refine the links using visibility info
        gatherAtVertices()        // Gather at the vertices of the Hierarchical Triangulation
        pushPull()
    endwhile
}
```

Figure 4.6: Visibility driven hierarchical radiosity

Once the system has been initialized in this manner, we begin discontinuity based subdivision (*subdividePolygons()*) and link refinement (*refineLinks()*). Using the global visibility information, we are capable of subdividing surfaces by following “important” discontinuities. After the completion of each subdivision/refinement step, a gather/push-pull operation is performed, resulting in a consistent multi-resolution representation of light in the scene.

In the following discussion we use the terms “source” and “receiver” for clarity. A source is any polygon in the scene which emits or reflects light. For secondary or tertiary illumination, for example, “sources” will be

polygons other than the primary light sources (*e.g.*, the walls, ceiling or floor of a room).

In the rest of this section, we present the link data structures and discuss issues related to form-factor calculation and multi-resolution link representation. In Section 4 we describe the refinement process for links, and the details of visibility updates; in Section 5 we present the polygon subdivision strategy and the perceptually-based refinement criterion used to effectively perform the subdivision.

3.2 Link data structures and form-factors

The central data structures used for our lighting solution are the links used to perform subdivision and light transfers. In contrast to previous hierarchical radiosity methods, two distinct link types are defined: point-polygon links which are used to gather illumination at vertices, and polygon-polygon links, which are used to make refinement decisions and to maintain visibility information while subdividing.

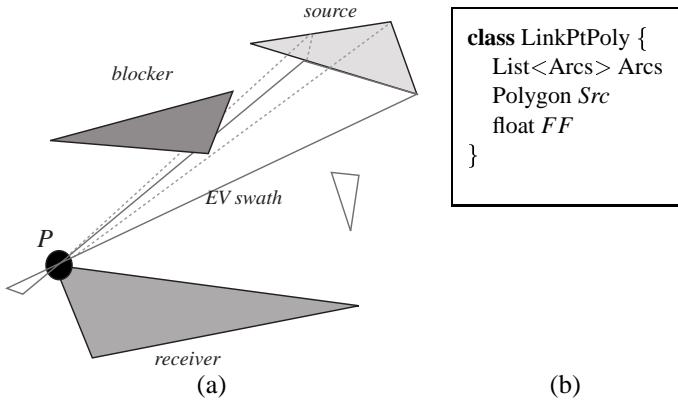


Figure 4.7: (a) A point-polygon link used to gather illumination at vertex P . All the arcs of the skeleton between P and the polygon *source* are stored with the link, *e.g.*, the *EV swath* shown. (b) The corresponding data structure.

Point-polygon links

As mentioned above, the skeleton provides all the information required to calculate the exact area-to-point form-factor from any polygon in the scene to any vertex. By updating the view information as shall be discussed below (Section 4.3), we extend this capacity to new vertices created by subdivision.

There are numerous advantages to calculating illumination at vertices. When computing mean radiosity on patches, the result can be displayed as flat shaded polygons. As we have seen, to provide a more visually pleasing result, the radiosity values are usually first *extrapolated* to the patch vertices and then interpolated. Inevitably, this introduces many artifacts in the approximation of the original radiosity function. In addition, it is much cheaper and simpler to compute exact polygon-to-vertex form-factors than polygon-to-polygon form-factors. Computing radiosity at vertices was first introduced by Wallace *et al.* [WEH89] in the context of progressive refinement radiosity. For hierarchical radiosity, the fact that vertices can be shared between different levels renders gathering at vertices more complicated.

A point-polygon link and the corresponding data structure are shown in Fig. 4.7. The point-polygon links are stored at each vertex of the hierarchical triangulations. A point-polygon link stores the form-factor calculated, as well as the arcs of the visibility skeleton (visibility events of the view) between the point and the polygon. An example is shown in Fig. 4.7, where the *EV swath* is stored with the link between point P and the source polygon.

The point-area form factor is computed analytically using the formula in *e.g.* [BRW89]. Consider Fig. 4.8.

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

The sum is evaluated using the arcs of the skeleton stored in the point-polygon link. \vec{R}_i and \vec{R}_{i+1} correspond to the two nodes (extremal stabbing lines) of the arc.

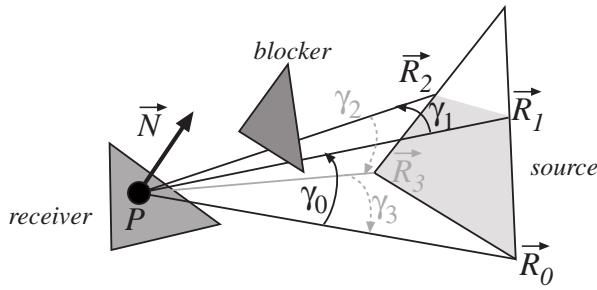


Figure 4.8: Geometry for the calculation of a form factor

Fig. 4.9 shows an example of form-factor computation with the Visibility Skeleton; the computation is exact. For comparison, the average (relative) error is given using ray-casting and a jittered grid sampling on the source (both the kernel and visibility are evaluated by Monte-Carlo). 36 rays are needed to have a mean error of 10%; numerical error on the form-factor is being measured. As expected from stratified sampling the convergence rate is about $O(n^{-\frac{3}{4}})$ [Mit96], since the function to be integrated is only piecewise continuous because of the visibility term. In Section 6.2 we will show the effect of this accuracy on the image quality.

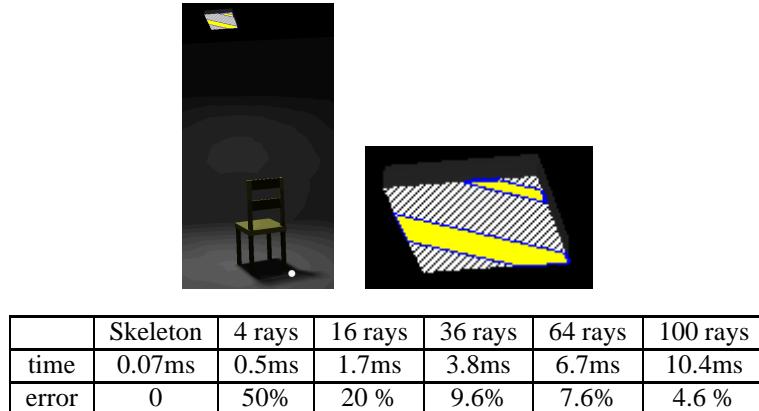


Figure 4.9: Example of Form-Factor computation from the white point on the floor to the area light source using the visibility skeleton and ray-casting with jittered sampling. The hidden part of the source is hatched. The Visibility skeleton timing does not include the visibility update (about 0.13ms per link on average for this image). All timings on a 195Mhz R10k Onyx 2.

Multi-resolution link representation

To maintain the multi-resolution representation of radiosity in the hierarchical triangulation, we require the representation of ΔB as described in Section 2.2, for the push phase of the push-pull procedure.

When a new vertex is inserted into a receiver polygon, “negative links” are created, from the source to the three vertices of the triangle containing the newly inserted vertex¹. These links allow the direct computation of ΔB as follows:

$$\Delta B = B_l - \sum_{i=0..2} c_i B_{nl}^i, \quad (4.1)$$

where B_l is the radiosity gathered from the positive link, B_{nl}^i is the radiosity gathered from the negative links and c_i are the barycentric coordinates of vertex P_i . An example of negative links is shown in Fig. 4.10(b).

The entire gather/push-pull process is illustrated in Fig. 4.11. The field *child* of an *HPolygon* is the associated triangulation.

¹In practice, these are simply pointers to the previously existing links.

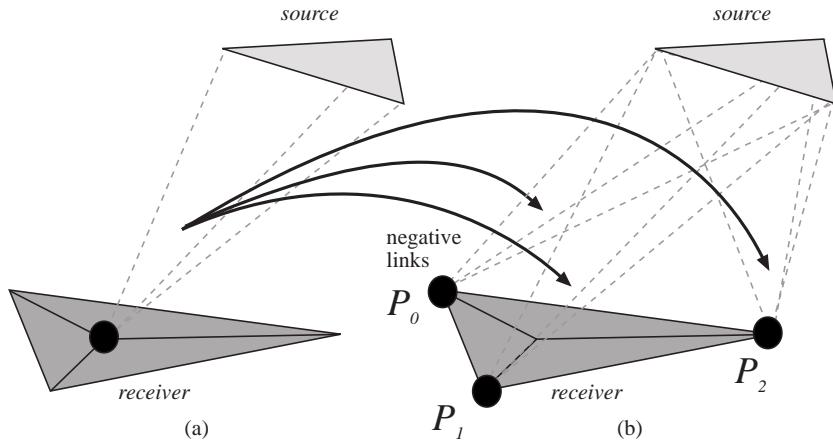


Figure 4.10: (a) A newly inserted point (in black) and the point-polygon link to the source; the vertex points to the (b) three new negative links to the source used for the ΔB representation.

```

Gather ()
{
    for each vertex  $v$ 
         $\Delta B_v = \text{gather}(\text{positive links}_v) - \text{gather}(\text{negative links}_v)$ 
}
Push (HPolygon poly)
{
    if child(poly) == NULL return
    for each vertex  $v$  in triangulation child(poly)
         $B_v = \Delta B_v + \text{interpolation}(\text{poly}, v)$ 
    for each triangle  $t$  in triangulation child(poly)
        Push(t)
}
Pull (HPolygon poly)
{
    if child(poly) == NULL
         $B_{poly} = \text{average of } B_v, \text{ for all } v \text{ vertex of poly}$ 
        return
    for each triangle  $t$  in triangulation child(poly)
        Pull(t)
         $B_{poly} = \text{average of } B_t, \text{ for all } t \text{ in child(poly)}$ 
}

```

Figure 4.11: Gather and push-pull

Polygon-polygon links

The polygon-polygon link is used mainly to determine how well the light transfer is represented, in a manner similar to that of the links in previous hierarchical radiosity algorithms. This information is subsequently used in the refinement process as described below. It also encodes information which is used for visibility updates.

A polygon-polygon link stores visibility information via pointers to the point-polygon links (two sets of three links for a polygons pair) between each polygon and the vertices of the other polygon. A polygon-polygon link also stores the set of visual events which have the two corresponding polygons at their extremities, and those with one generating edge on one polygon and one extremity on the other. A polygon-polygon link is illustrated in Fig. 4.12, with the corresponding point-polygon links from the source to the receiver.

In the case of a subdivided polygon, all the neighboring triangles of a vertex v share all the point-polygon links related to v .

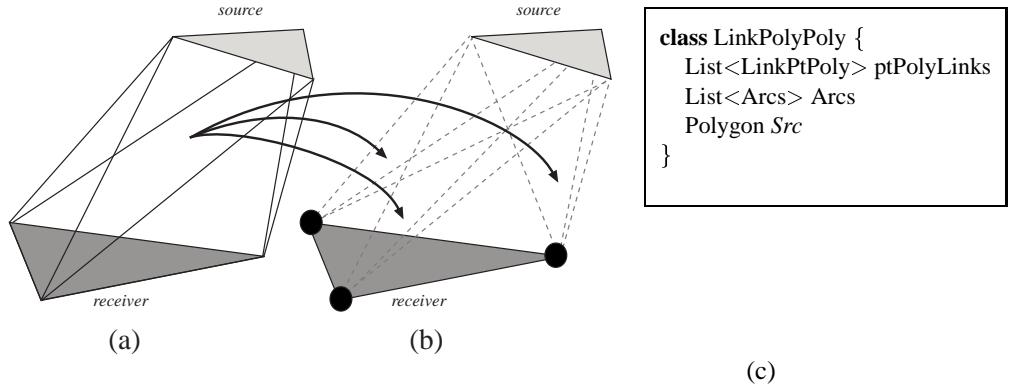


Figure 4.12: (a) A polygon-polygon link used to estimate illumination transfer between two polygons (b) The polygon-polygon links store pointers to the 6 corresponding point-polygon links: 3 of them (*source* → *receiver*) are shown here (c) The corresponding data structure.

4 Link refinement

Now that the link data structures have been described in detail, we can present the link refinement algorithm. The process is slightly more involved than in the case of standard hierarchical radiosity (*e.g.*, [HSA91]), because the subdivision is not regular and the existence of the two link types requires some care to ensure that all updates are performed correctly. This section describes how the links are actually refined.

4.1 Refinement overview

Consider a light exchange from a source polygon to a receiver polygon. Because we gather radiosity from the polygon at the vertices, two kinds of refinement can be necessary.

- *Source-refinement* if the radiosity variation over the source polygon is too high;
- *Receiver-refinement* if the sampling on the receiver is too coarse.

The link refinement algorithm is straightforward: for each polygon and each of its polygon-polygon links, the link is tested for refinement. If the test succeeds, the link is refined and the new point-polygon and polygon-polygon links are created. Finally, the visibility of the link is updated. The refinement test uses a perceptually-based refinement criterion, based on the visibility information contained in the Visibility Skeleton (see Section 5.4). Note that since we compute exact point-to-area form factors, the source refinement cannot be caused by the inaccuracy of the form-factor computation. It can only happen because the radiosity of the source is not uniform, *i.e.*, if a receiver-refinement has occurred on the source in another exchange.

4.2 Source and receiver refinement

The first type of refinement is that of a source. If the representation of radiosity across the source is considered insufficient for the given transfer (*i.e.*, the variation of radiosity is too high across the source), the link will be refined. The geometric subdivision of the source has occurred at a previous iteration, typically due to shadowing. New polygon-polygon links are created between the original receiver and the sub-triangles of the source. New point-polygon links are created for each vertex and each source sub-triangle, and the corresponding visibility data is correctly updated (see Section 4.3).

The second type of refinement is that of the receiver. For example, in Fig. 4.13, a point is added to the receiver. As a consequence, the triangulation is updated and three new polygon-polygon links are added. One of these is shown in Fig. 4.13(b). In addition, a new point-polygon link is created, from the point added on the receiver to the source (Fig. 4.13(c)).

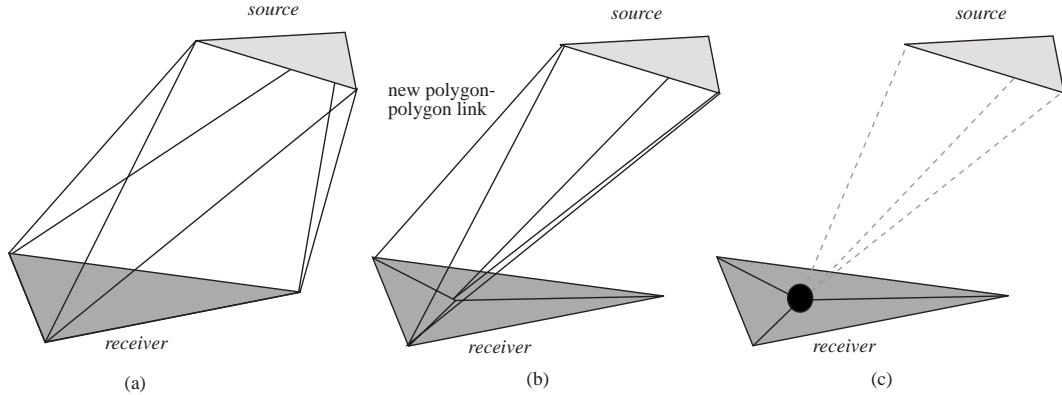


Figure 4.13: Receiver refinement: (a) Original polygon-polygon link (b) Insertion of a point on the receiver and one of the three new polygon-polygon links created. (c) The additional point-polygon link to the source.

4.3 Visibility updates

Each refinement operation requires an equivalent update in the visibility information contained in the point-polygon links. We again distinguish the two main cases, source refinement and receiver refinement.

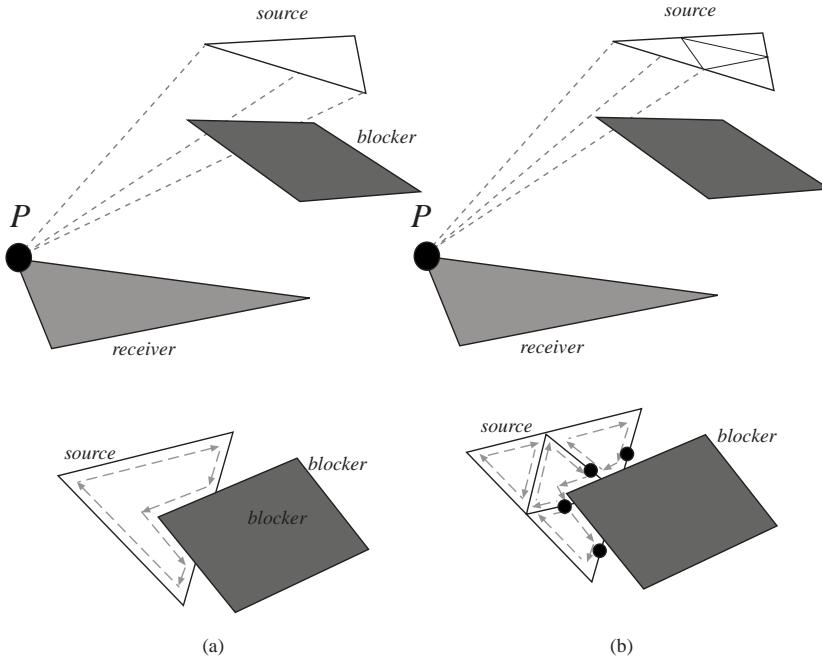


Figure 4.14: Source visibility updates. The dashed arrows (lower part) represent the limits of the visible part of the source used to compute the form-factors. (a) The point-polygon link before subdivision and below the corresponding view of the source (b) One of the 4 new point-polygon links due to subdivision and the four new views. The black circles correspond to new nodes of the Skeleton.

In the case of source refinement we need to update the existing visibility information contained in the new point-polygon links. Since the visibility information of such a link can be represented by the view of the source from the receiver point of the link, all that needs to be done is the update of the link with respect to the new source sub-triangles. For example, in Fig. 4.14(a) the original view from point P is shown in the lower part of the figure. Once the polygon-polygon link is subdivided, four new views are computed, shown in the lower part of Fig. 4.14(b). The new point-polygon links now contain the references to the skeleton arcs (swaths),

corresponding to the parts of the view affected. For example the leftmost source sub-triangle is completely unoccluded from P and thus no arcs are stored. For the others, the intersections of the previously existing arcs and the source sub-triangles result in new skeleton nodes (corresponding to the black circles in Fig. 4.14(b)). The corresponding arcs are then subdivided. The new nodes are adjacent to these subdivided arcs. Note that all visibility/view updates are performed in 2D.

Refining a receiver is more involved. When adding a point to a polygon, a new view needs to be computed. We use the algorithm of the Skeleton construction which is robust. The only difference is that we use the blocker lists defined by the arcs stored in the initial point-polygon links instead of the entire model. Since the number of polygons in any given blocker list is relatively small, the cost of computing the new view is low. An example is shown in Fig. 4.15(a)-(b), where the point P is added to the receiver. In Fig. 4.15(b) we see the point and the new point-polygon link, and in Fig. 4.15(c) the newly calculated view is illustrated. The black circles correspond to newly created nodes of the skeleton.

The case of full visibility is detected using the information contained in the polygon-polygon links. The visibility update is then optimized: no new arc is computed and the unoccluded form-factor is used, thus saving time and memory.

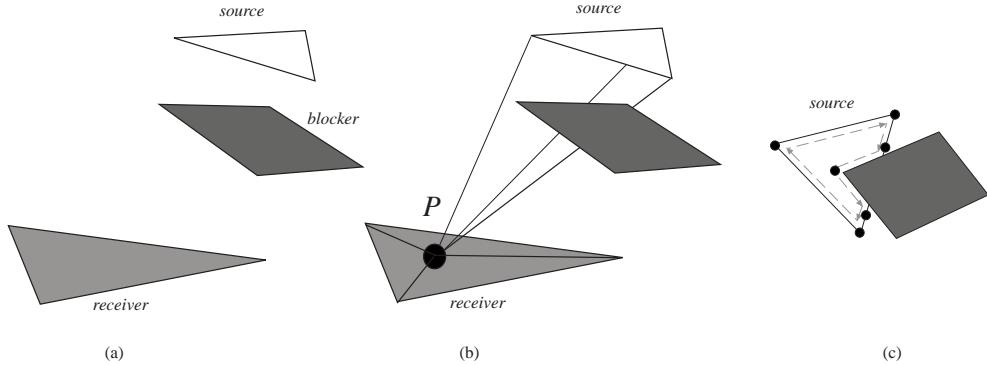


Figure 4.15: Receiver visibility updates: (a) The initial configuration. Blocker information is contained in the source-receiver link. (b) A point is added to the receiver, creating a new point-polygon link. (c) The new view of the source computed at P . The blocker lists are updated using this computation.

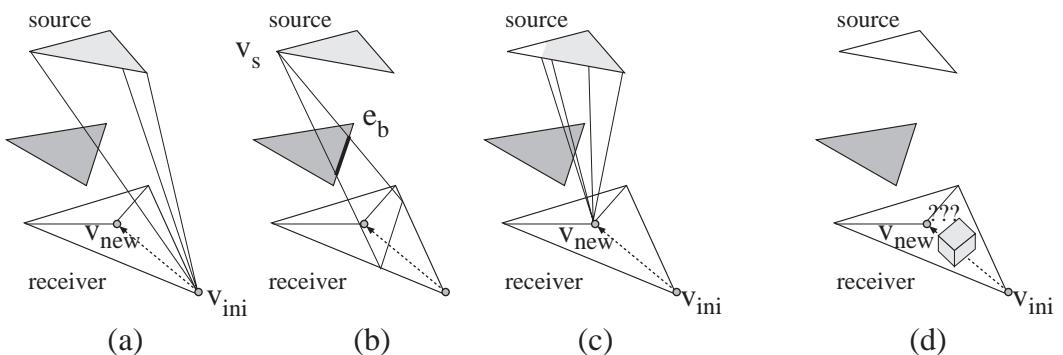


Figure 4.16: Receiver refinement with visibility events (this solution was not implemented). (a) We start with a view at one of the initial vertices. (b) We walk across the receiver to the new vertex. Here we cross a $v_s e_b$ event. v_s begins to be hidden by the blocker. (c) We obtain the view at the new vertex. (d) In the case of touching objects, no information can be kept while crossing the interface between the two objects.

4.4 Alternative visibility updates

Visibility updates could be performed using the information encoded in the Skeleton, starting from the view at one of the initial vertices, then walking to the new vertex and updating the view each time a visibility event is crossed (see Fig. 4.16). This method is however hard to implement, and suffers from robustness problems if the visibility events are not crossed in a coherent (if not exact) order. Moreover, the case of touching objects complicates the problem furthermore since no information can be kept while walking “under” a touching object.

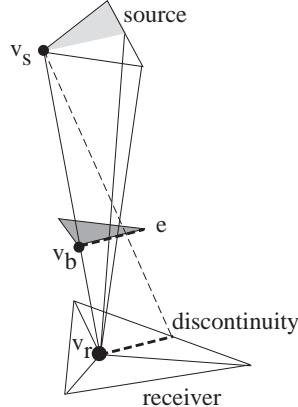


Figure 4.17: Degeneracy due to discontinuity meshing. The receiver is split along discontinuity $v_s e$, causing a degenerate $v_s v_b v_r$ extremal stabbing line.

4.5 Treating degeneracies

Subdividing along discontinuities induces degenerate viewpoints. For example in Fig. 4.17, we subdivide the receiver along the discontinuity $v_s e$. The view from v_r has a degenerate $v_s v_b v_r$ extremal stabbing line. To treat it coherently, we store with each vertex of the triangulation the extremal stabbing line which caused it (which is possibly null). This is a simpler and more robust alternative to the ray-casting modified for grazing objects described in Section 4.3. The treatment of the degeneracy then proceeds in the same manner.

A different alternative would have been to slightly perturb the point position to avoid those degeneracies. Two reasons have prevented us from doing so. First, discontinuity meshing allows us to delimit regions of umbra (full occlusions), regions of full visibility, and regions of penumbra. The two first region types require coarser subdivision than the latter. If we perturb the point position, some regions which should have been totally in the umbra will have a very small part in the penumbra, and need more subdivision. Second, point perturbation would cause numerical precision problems.

5 Polygon subdivision

We have now seen how link and visibility information is updated during the light propagation process. Evidently, link updates are a consequence of a *refinement* decision, based on an appropriate criterion. We have chosen to use a *perceptually-based* refinement criterion. In what follows, we first review basic concepts of perceptual mapping which we use for our refinement criteria. We next present the polygon subdivision process, and then detail the perceptually based refinement criterion which we have used for our algorithm.

5.1 Perceptual just noticeable difference

The work in the field of perception provides us with two important features. First, it permits the conversion of radiometric quantities into displayable colors while preserving the subjective impression a viewer would have when observing the real scene. Second, it allows us to use error thresholds related to the error an observer is able to perceive, which are thus easy to set.

The human eye can deal with a very high dynamic range, while computer displays are usually limited to a 1 to $100\text{cd}/\text{m}^2$ range [GH97b]. The eye adapts itself according to the luminosity of the scene being looked at. This explains why we are able to see dark night scenes as well as very luminous sunny scenes. The tone mapping operation deals with the transformation of high range radiometric quantities into low range display colors, while trying to provide the viewer with the same impression as the real scene. One obvious and simple method is to divide all quantities by the maximum radiosity of the scene. The problem with this approach is that if the light source intensity is halved, the scene will look exactly the same, though we would expect it to seem darker.

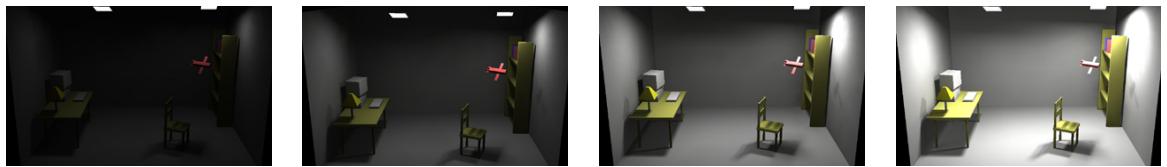


Figure 4.18: Effect of Ward's tone-mapping on the same scene with different light source intensities. Note how details remain perceptible while the impression of darkness or luminosity is preserved.

Ward's contrast preserving tone mapping operator [War94] deals with this problem. A simple scaling factor sf is used for the whole scene which depends on the maximal displayable luminance L_{dmax} and the world adaptation level L_{wa} which is usually the logarithmic average of the scene luminosity without primary light sources. In what follows, all intensities are expressed in $\text{candelas}/\text{meter}^2$ (a *candela* is a *lumen/steradian* [War94]). The scaling factor sf is then given by

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

Fig. 4.18 demonstrates the effect of this operator on a given scene with different source intensities.

We use a technique similar to that of Gibson *et al.* to compute the adaptation level [GH97b]. We use a static adaptation level which is the average radiosity of the scene. However, since we use hierarchical radiosity as opposed to progressive radiosity, we do not have to rely on an estimate involving the average luminance and reflectance. Instead, at any step we use the average radiosity value of the polygons of the scene. This is why we start the radiosity computation with several gather steps to compute a coarse estimate of the light distribution.

The use of a global static adaptation level is only a coarse approximation of the human visual system adaptation. As shown by Gibson *et al.* it gives a fairly good estimate of the dynamic adaptation level [GH97b]. More elaborate solutions could be explored, such as the use of local adaptation levels computed using the average radiosity in the neighbourhood of an object, and more involved tone-mapping operators could also be used [TR93, FPSG96] but this is beyond the scope of this thesis.

Once the tone mapping operation has been applied, the admissible error can be set as a given percentage of the maximum displayable intensity L_{dmax} . Psychovisual studies [GH97b, Mur87] have shown that the human eye is able to distinguish a difference of 2%: this is the *just noticeable difference*. We will call the allowed error ϵ_{percep} , and in practice we will use $\epsilon_{percep} = 2\%$ for all our refinement criteria.

5.2 Polygon subdivision

Our experiments have shown that subdividing along the discontinuities during the first few subdivisions results in the creation of triangles with poor aspect ratios, inducing very visible artifacts. For this reason, subdivision of the polygons is performed using a two step strategy:

- *During the first two subdivisions:* The polygons are subdivided in a regular grid-like manner. In particular, a regular grid is created as a function of size of the polygon being subdivided.
- *During the third and subsequent subdivisions:* Insert shadow discontinuities or other illumination detail. Discontinuities are added as constrained edges, and result in a modified triangulation.

```

subdividePolygons() {
    for each polygon  $r$  and each poly-poly link  $s$  to  $r$ 
        if shouldRefine  $\text{Link}(s, r)$ 
            refine source
    for each polygon  $r$  and each poly-poly link  $s$  to  $r$ 
        if shouldRefine  $\text{Link}(s, r)$ 
            if iteration < 3
                regularSubdivision(  $r$  ) // perform grid-like subdivision
            else
                find and insert discontinuities in  $r$ 
                complete subdivision at this level // create sub-triangles in meshes
}

```

Figure 4.19: Polygon subdivision

This approach is similar in spirit to the approaches of Stuerzlinger [Stu94] and Hardt and Teller [HT96] where the discontinuity meshing is, however, used for display purposes only. The polygon subdivision algorithm is outlined in Fig. 4.19. In contrast to standard hierarchical radiosity, we cannot subdivide the polygons on-the-fly when a link needs subdivision, because polygon subdivision is not uniform and has to be performed along discontinuity curves. For this reason, we first consider all the polygon-polygon links for a given polygon to decide if it requires subdivision, and to determine which discontinuities will be inserted. After all discontinuities for a given receiver have been inserted, the constrained Delaunay triangulation is completed.

5.3 Maxima insertion

If we consider the radiosity gathered from a light source as a function defined over a receiver, it has been shown that subdividing a mesh used to represent illumination at the maximum of the function can increase the accuracy of the radiosity solution [DF93]. We thus first compute the maxima of the unoccluded radiosity functions of the light sources before the first refinement.

The maxima are computed only for important light-transfers (estimated using the disk-disk formula [HSA91] and the perceptual metric). Given a receiver and a polygon considered as a source, we use a gradient-descent algorithm to locate the maximum. Once the maximum is found, we compute the contribution of the source at this point; if it is above ϵ_{percep} the maximum is stored to be subsequently inserted in the mesh. The radiosity of the receiver polygon is updated to take this maximum into account. That is, a gather is performed at the maximum (before a link is created from it) to obtain a better estimate of the light distribution that will be used for the first refinement.

The maxima are inserted as a separate initial step during the first subdivision. The points of the regular subdivision which are too close to a maximum are not inserted. An example was shown in Fig. 4.2(b), where the maximum corresponds to the point on the lower left which is not exactly on the grid. We thus obtain nearly regular meshes with well shaped triangles.

The maxima-search process is applied iteratively to take indirect illumination into account. The insertion of maxima of indirect sources is very important for example in Fig. 4.24, where the table (illuminated by the lamp) is the most important light source for the upper part of the left wall.

5.4 Refinement criterion

We distinguish two refinement criteria (or *oracles*): a radiometric criterion which accounts for the variation of the unoccluded radiosity, and a visibility (discontinuity) criterion. Moreover, the discontinuity criterion also guides the choice of the discontinuity curves to be inserted.

The radiometric oracle estimates if the linear interpolation of the light transfer is “accurate enough”. We sample the unoccluded form-factor (see [BRW89] and Section 3.2) at the center of the patch and at the edge mid-points, and compare this to the linearly interpolated value. If the perceptually transformed difference is larger than ϵ_{percep} , we proceed with subdivision.

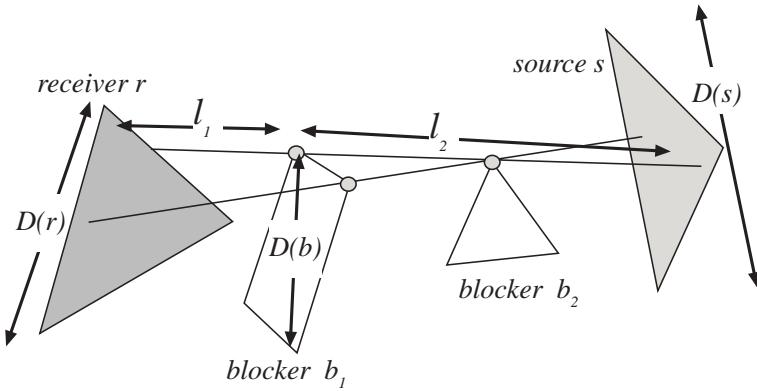


Figure 4.20: Refinement criterion geometry

The principle of our visibility oracle is to estimate (as a percentage of the maximum displayable intensity) the “shadow amount” cast by the blockers, that is the radiosity that would be transferred without the blocker. Our refinement criterion thus has three steps: unoccluded estimate, “shadow amount” estimate and “shadow sharpness” estimate.

Consider a receiver and a source. Recall that a source is any polygon in the scene, considered as a source at this step of the refinement process. In what follows we refer to Fig. 4.20, for the definition of all geometric quantities.

First, we compute an estimate of the unoccluded light transfer B_{unoc} using the disk-disk formula [HSA91]. As above, if the estimate is less than ϵ_{percep} , the link will not be subdivided.

Second, we consider each visibility event between the source and the receiver, and estimate the “shadow amount”. To do this, we estimate the part of the source potentially hidden by the blocker by using the projected diameter of the blocker on the source to estimate its projected umbra:

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

The estimated percentage of occlusion is then:

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

(clamped to 1). The “shadow amount” is:

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

If $shadow$ is below ϵ_{percep} , the visibility event is ignored.

Third, we estimate the sharpness of the shadow. The extent of the zone of penumbra is approximated by projecting the diameter of the source onto the receiver:

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

If the size of the receiver is bigger than the zone of penumbra, then the receiver may contain regions where the source is completely visible, and regions where the blocker projects entirely on the source. In the latter case, the fraction of occlusion is maximal and approximated by $occlu$. The variation of radiosity on the receiver is thus approximated by $shadow$. Otherwise, we make the approximation that the radiosity varies linearly in the penumbra, and the variation of radiosity on the receiver is then:

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

All the links containing visibility events with $\Delta(B) > \epsilon_{percep}$ will be subdivided. As explained above, in the first two iterations subdivision will be regular. During the third and later iterations, subdivision is performed by

inserting the discontinuities with the highest $\Delta(B)$. This is the *ranking* phase of our algorithm, similar in spirit to that of [HT96].

$\Delta(B)$ is computed using l_1 and l_2 at the two node extremities of the visibility event, and taking the maximum. The evaluation of these oracles is very rapid since the links and events are pruned as soon as we can decide that they will not cause subdivision.

6 Implementation and results

6.1 Implementation

We have used the C++ Visibility Skeleton implementation of the previous chapter, with the extensions and changes for the storage of visibility information resulting from subdivision. The scene polyhedra are represented using a winged-edge data-structure and a pre-processing step is performed to detect touching objects, which is a necessary step for the treatment of degeneracies.

The hierarchical triangulation has been incorporated into the same system. We use the public domain implementation of [GS85] by Dani Lischinski [Lis94] for the constrained Delaunay triangulation. Each subdivided polygon contains a triangulation QuadMesh.

On our test scenes, the algorithm spends most of its time on the visibility update, especially the calculation of the views at new vertices for the receiver refinement. For example, for the Desk scene of Fig. 4.21, for the last iteration, the computation of the criterion and the refinement of the mesh took 15 seconds, updating the visibility took 64 seconds, and the gather/push-pull took 2.5 seconds.

We have chosen not to use textures in our examples since they usually hide the accuracy of the lighting simulation.

6.2 Results

We present results for four different scenes. The first scene is a simple “Desk” scene, containing 438 polygons and two large, powerful light sources (see Fig. 4.21). This scene is used to illustrate the general functionality of our algorithm. The second scene contains the same geometry, but with 8 additional small, powerful light sources. The two large light sources have been turned down in intensity; we call this scene “Many Lights” (see Fig. 4.22). This scene shows how our approach treats the case of multiple light sources effectively. The third scene has been chosen to demonstrate the performance of our algorithm for mainly indirect lighting. We chose a common example of a bedroom lit exclusively by a small downwards-pointing, bed-side lamp. Most of the room is lit indirectly; this scene is called “Bed” (see Fig. 4.24). Finally, simply in the interest of showing a completely different type of scene, we show the result of our approach on a “Village” scene, containing buildings and cars. The scene is lit overhead by a rectangular light (see Fig. 4.27). In what follows we present various performance statistics as well as an informal comparison with hierarchical radiosity using quadtree subdivision, but with improved refinement and error bound strategies ([GH96, LSG94]).

All times presented are in seconds on an R10000 195 MHz Silicon Graphics Onyx 2 workstation.

Before presenting the results for the complete algorithm, we present some interesting statistics concerning the importance of accurate visibility for form-factor computation.

Importance of accurate visibility

We have run some tests with approximate visibility to judge the importance of the exact computation of the form-factors on the quality of the images. We have slightly modified our implementation to compute the form-factors using ray-casting on a jittered grid sampling of the source. In Table 4.1, the same method is used for discontinuity-based mesh subdivision for all cases shown. It is performed using the Skeleton, and the cost of the skeleton construction and update is not included in the ray-casting timings, which report exclusively the cost of form-factor computation. As mentioned before (Fig. 4.9), inexact form-factor computation introduces significant error. The visual consequences of this can be seen in Table 4.1. Moreover, the computation overhead is significant if high quality is required because at least 64 rays are needed per form-factor.

Image		
Method	exact (Skeleton)	16 rays
Total time	1min 19	1min 17
Image		
Method	36 rays	64 rays
Total time	2min 02	3min 04

Table 4.1: Importance of the form-factor accuracy on a small scene of 246 polygons. The number of rays for the indirect illumination is set to 4, while only the number used for direct illumination varies. In inset we show in false color the difference with the skeleton solution in the perceptually uniform *CIE L*a*b** color space.

Note that the effect on the images is particularly dramatic, because the subdivision induced by the discontinuity meshing is not uniform. The thin triangles introduce very visible artifacts. These results confirm those observed in [DS96].

Scene	Pol	Skel	1st	2nd	3rd	Total	Mem(ini/tot)	links	tris
Desk	444	2min 08	22s	16s	1min 14	4min	40/200MB	378K	46K
Many	492	2min 23	2min 38	55s	4min 27	10min 23	47/365MB	1546K	104K
Bed	534	4min 12	1min 25	58s	4min 35	11min 10	56/400MB	383K	43K
Village	312	45s	12s	7s	24s	1min 28	15/43MB	134K	28K

Table 4.2: Timing and memory results for the test scenes. The memory statistics shown are the initial memory usage for the skeleton *before* any subdivision, and the total memory used after the subdivision for lighting.

General solution

The images of Fig. 4.21 show the initial steps of the algorithm as described previously. Fig. 4.21(a) is the result of three gather steps on the initial unsubdivided scene. Note that at this point we already have a very crude approximation of the global distribution of illumination in the scene. In Fig. 4.21(b) we see the first step which is a regular grid together with the maxima of the light sources inserted into the mesh. Fig. 4.21(c) and (d) show the evolution of the algorithm after two iterations. The shaded images without the meshes are shown in (d). In (e) we show the discontinuities actually inserted. These include discontinuities for all light transfers (direct and indirect) and that their number is much lower than that for a discontinuity meshing type approach (about 40% of the discontinuities caused by direct sources have been inserted).

In Table 4.2 we show the statistics of scenes computed using our method. For the “Desk” scene, we see that the total solution, including illumination, requires 4 minutes of computation. The quality of the solution is very high, including well-defined shadows on all surfaces. Note high quality shadows on the chairs and the table. The total number of point-polygon links is 378,746, and the number of leaf triangles is 46,058.

Treating many lights

One scene type for which our approach performs particularly well is that of multiple sources. This is demonstrated by our second test scene containing 10 lights and the same geometry as “Desk”. Fig. 4.22(a) shows an overview of the scene as rendered by our new approach, and Fig. 4.22(c) shows a closeup of the floor. The shadows due to the multiple sources are well represented in the areas when appropriate. The perceptually based ranking algorithm has correctly chosen the discontinuities that are of importance, since the combined influence of all sources is taken into account. This is shown by the small number of discontinuities present on the floor in Fig. 4.22(d). From Table 4.2 we see that 1.5 million links were used in this scene and the total computation time was 10 minutes 23 seconds. Only 10% of the direct discontinuity segments have been inserted.

As an informal comparison, we have compared to an implementation of hierarchical radiosity with clustering with the refinement proposed by Gibson and Hubbard [GH96] using the error bound propagation of Lischinski *et al.* [LSG94]. For the Many lights scene computation with 1 million links, the computation time is almost 2 hours (Table 4.3). In addition, the quality of the results is lower, since the multiple shadows are much less sharp, or even missing (see Fig. 4.23). A much larger number of links would be necessary to compute an image of similar quality to Fig. 4.22 using hierarchical radiosity. Despite the fact that this method uses approximately the same number of elements (110K vs. 104K for our method), the quality of the resulting images is much lower.

Scene	Poly	1st	2nd	3rd	4th	Total	Mem	links	elems
Many	492	1 hr 25	22 min	10 min	-	1 hr 57	147 MB	1098K	110K
Bed	534	11 min	37 min	6 min	25s.	54 min	94 MB	903K	32K

Table 4.3: Comparative Timing and memory results for the test scenes using Hierarchical Radiosity with error bounds [LSG94, GH96].

Indirect illumination

Accurate and efficient computation for indirect lighting is another challenge for our approach. It is for this type of scene that we see the power of our accurate form-factor and discontinuity ranking method. Previous approaches require significantly longer computation time to achieve this level of precision for secondary illumination.

This is illustrated with our third test scene (Fig. 4.24 and 4.25), in which light arrives from the bedside lamp which is pointing downwards only (no light leaves from the sides or the top of the lamp). Thus everything in the room above the level of the lamp is lit indirectly.

The algorithm uses a relatively small number of point-polygon links (383,715), and manages to represent shadows generated by secondary illumination. Notice for example the shadows of the right hand lamp or the books on the far wall in Fig. 4.24(d); these are caused by illumination of light bouncing off the bedside table and the bed.

Another informal comparison is presented, using the same algorithm as described above (based on [GH96, LSG94]). Using almost a million links, hierarchical radiosity takes a slightly less than one hour, and produces lower quality results (see Fig. 4.26(a) and (b)).

Moreover, the advantages of our linear lazy-wavelet representation are well illustrated on the overall view of the hierarchical radiosity solution. The left part of the back wall is much lighter than the right part, with a strong discontinuity in-between revealing the quadtree nature of the mesh. This is because interpolation is applied as a post-process at the finest level of subdivision; exchanges simulated at higher level are thus not correctly interpolated.

Village scene

A final scene of a village is shown in Fig. 4.27, to show that the algorithm can be used for different scene types. Here the scene is lit overhead by a rectangle and also by the head and rear lights of the cars.

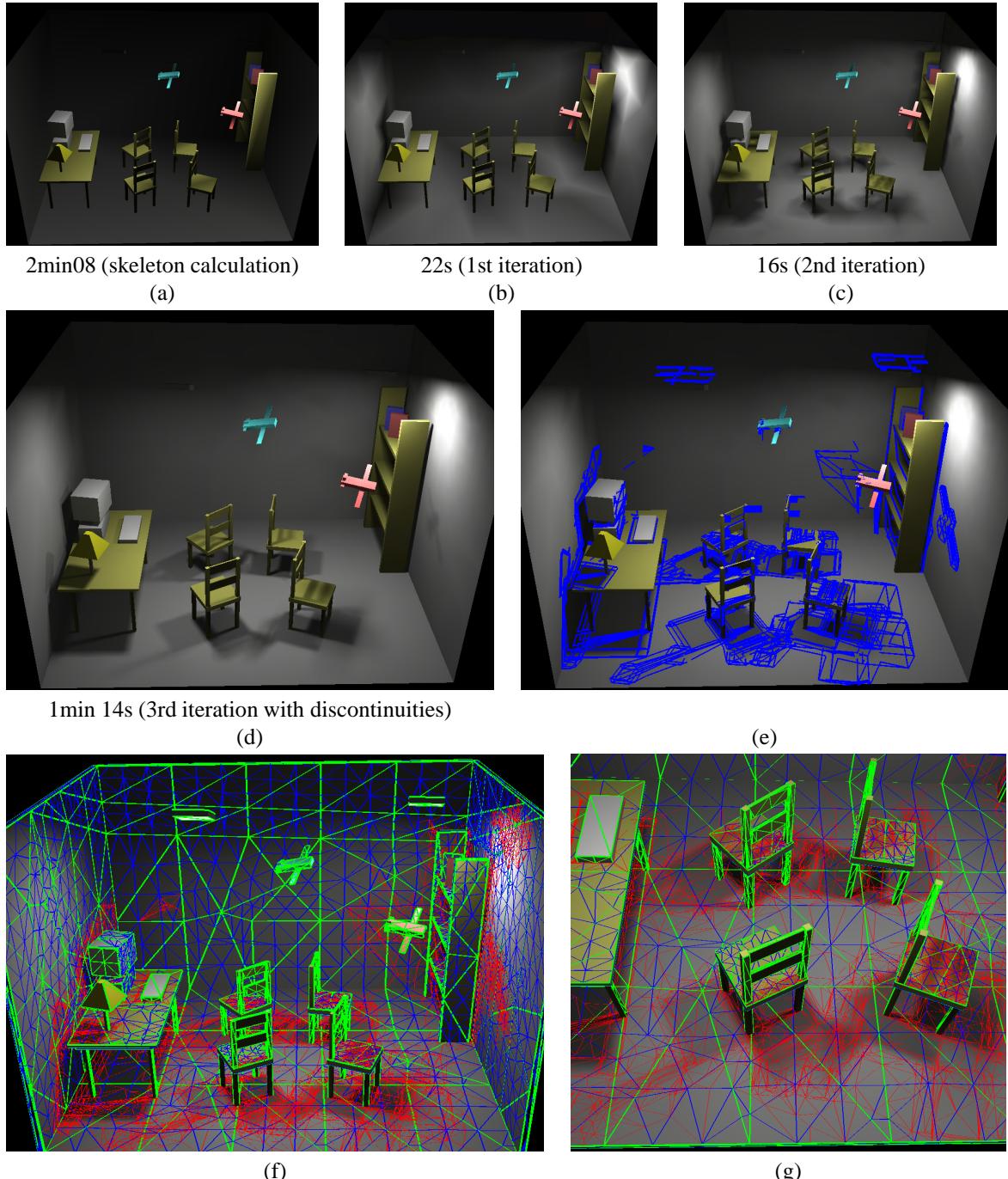


Figure 4.21: Initial Desk Scene. In (a) we show the initial, unsubdivided scene. In (b) we show the first step which includes the grid and the maxima, in (c) we show the second iteration and (d) show the results of the third iteration which includes the discontinuity meshing. (e) shows the discontinuities actually inserted. (f) and (g) show the hierarchical triangular mesh (first level in green, second in blue, and third in red).

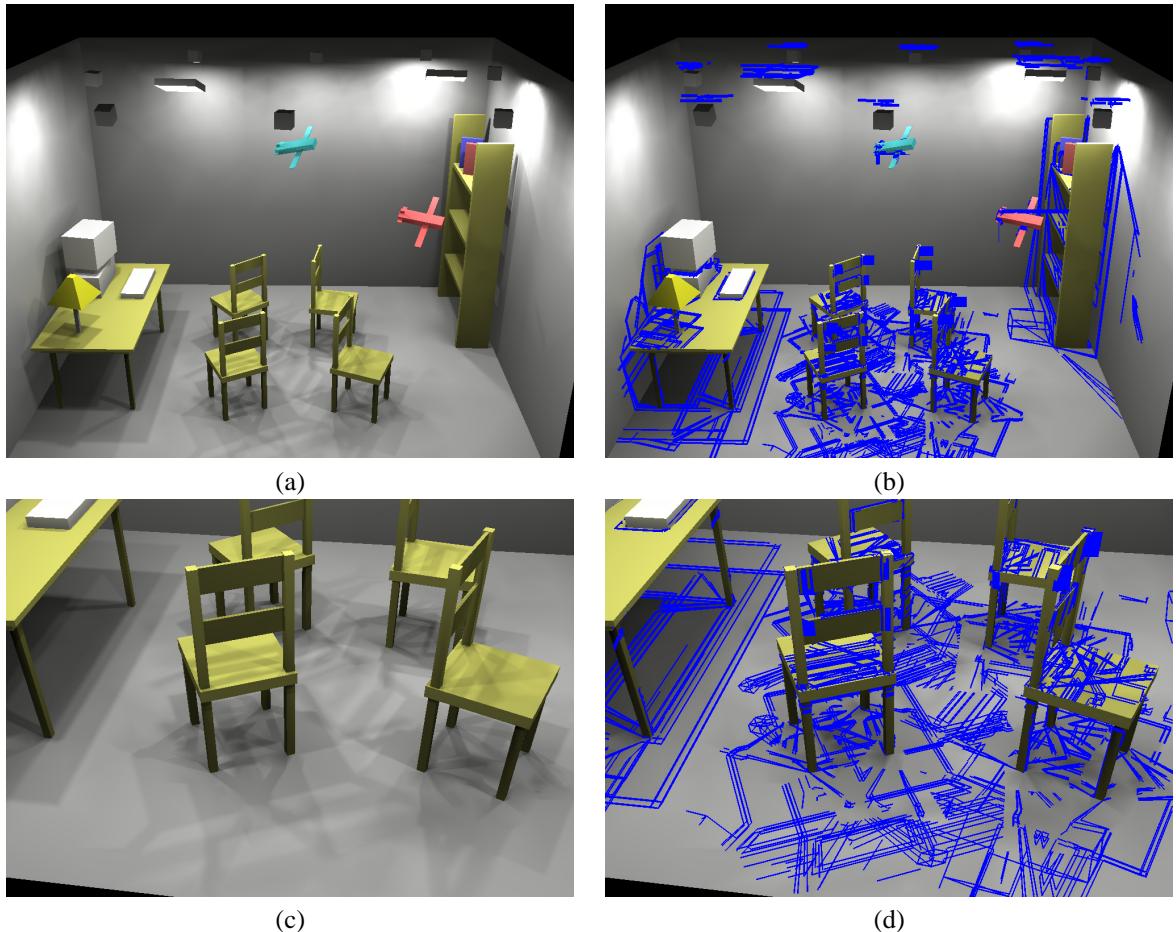


Figure 4.22: Many Lights scene: (a) the final image, (b) the discontinuities actually inserted. (c) and (d) a closeup view of the floor.

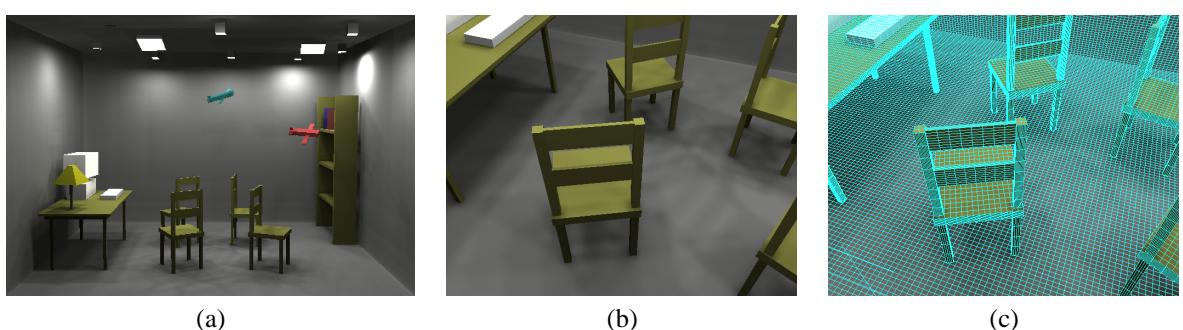


Figure 4.23: Hierarchical Radiosity comparative results for Many Lights scene: (a) the final mesh, (b) a general view of the rendered scene (c) a closeup view of the floor.

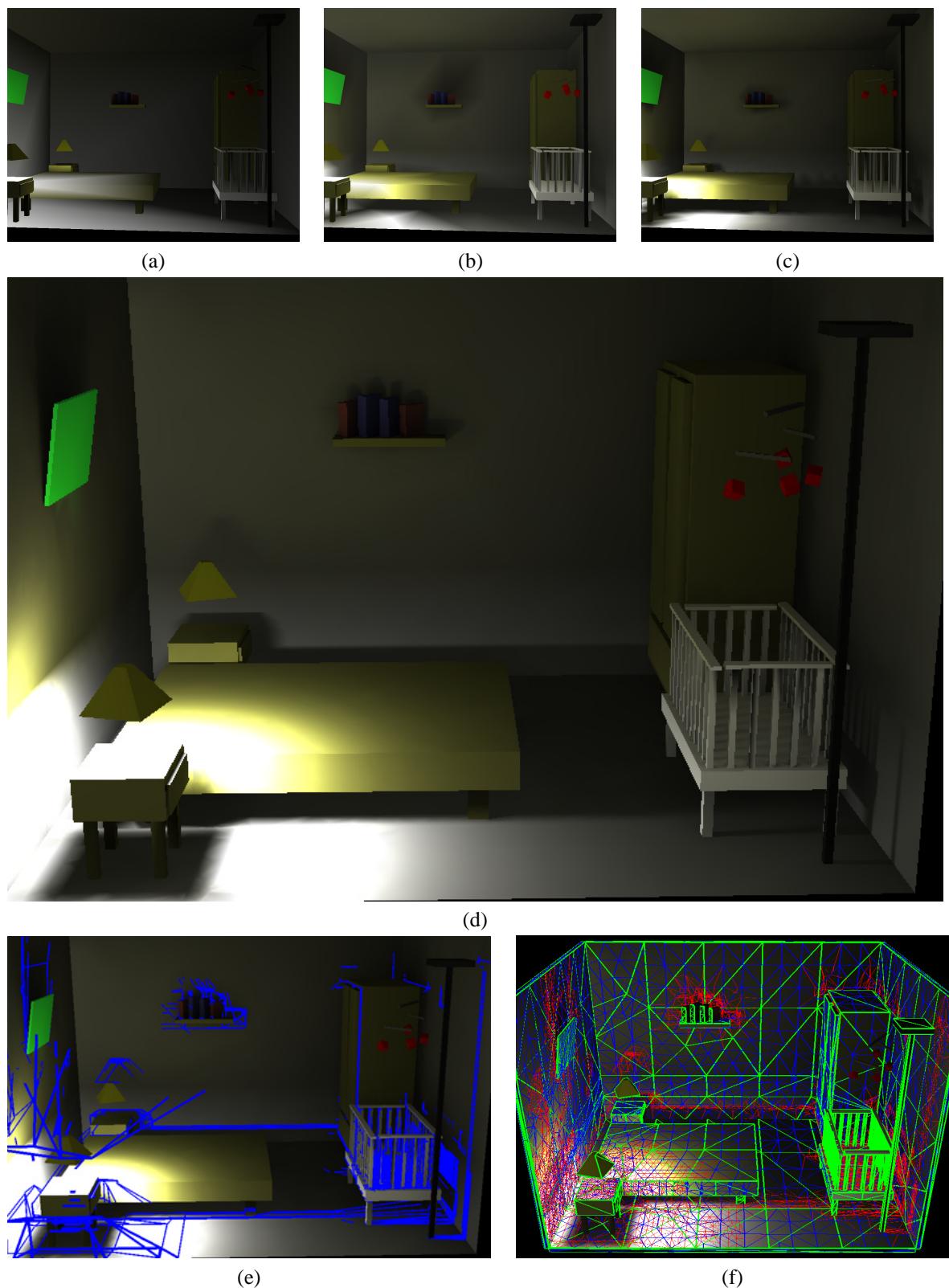


Figure 4.24: Indirect lighting scene: (a) Initial solution, (b) first iteration (c) second iteration (d) final image (e) discontinuities inserted (the discontinuities inserted on the front wall are represented though this wall is backface-culled) (f) hierarchical triangulation.

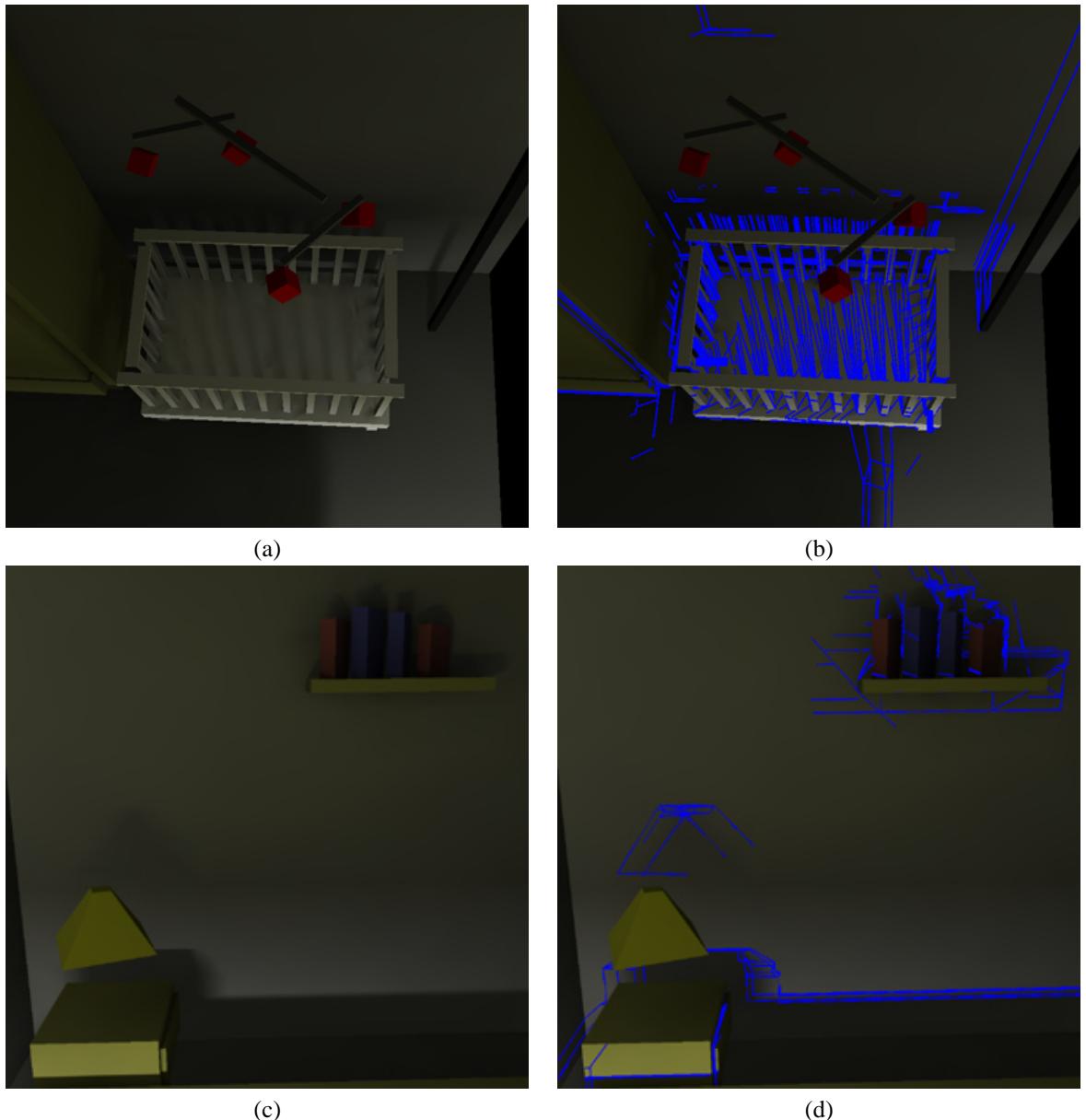


Figure 4.25: Indirect lighting scene: (a) and (b) closeup of the right wall (c) and (d) closeup of the back wall. The lower part of the wall is directly illuminated by the left lamp (which is not visible on this image), while the upper part is indirectly illuminated by the left table. Note the indirect shadows cast by the books and the right lamp.

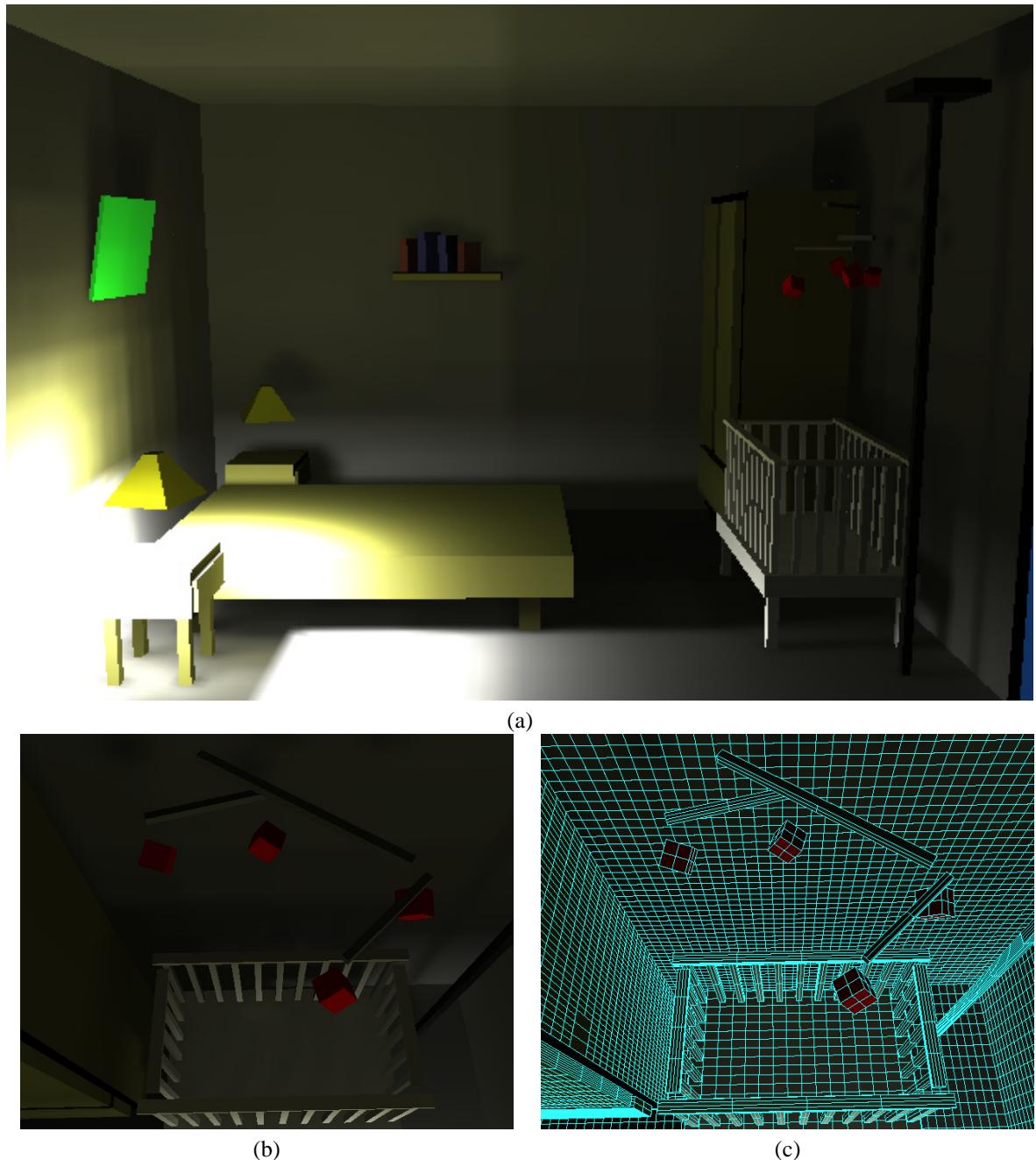


Figure 4.26: Hierarchical Radiosity comparative results for Indirect Lighting scene: (a) the final image, (b) and (c) a closeup view of the right-hand wall.

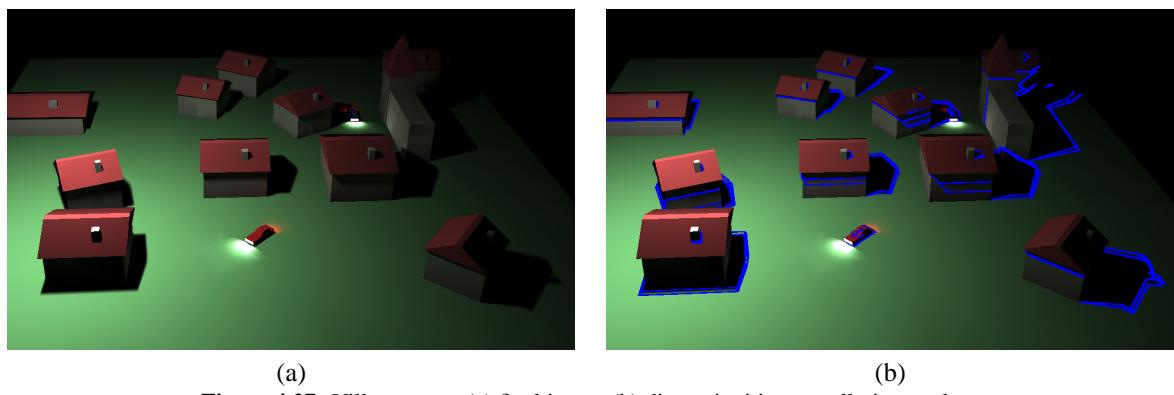


Figure 4.27: Village scene (a) final image (b) discontinuities actually inserted.

7 Improvements

The large memory consumption of our method mainly comes from the visibility information resulting from subdivision of the initial polygons². In this section we propose some improvements to decrease these requirements. They have not yet been implemented, but we believe that the practical importance of the issue makes their presentation valuable. We first present a scheme which saves time and memory for certain link refinements. We then propose a method which is a tradeoff between time and memory.

7.1 Taking advantage of invariant backprojections

As observed in section 4.3, when receiver refinement is performed on a link with full visibility, no visibility information has to be recomputed. In this section we extend this to any link refinement where visibility is qualitatively invariant.

This is based on the work on complete discontinuity meshing with backprojections [DF94, SG94]. Recall that a complete discontinuity mesh subdivides the polygons of the scene into cells where the visible part of the source is qualitatively invariant. The backprojection encodes the structure of this visible part (*i.e.*, the chain(s) of edges and vertices bounding it). Inside the cell of such a mesh, visibility needs no update. These cells are bounded by the same visual events encoded by the visibility skeleton since they are the locus of the changes in visibility.

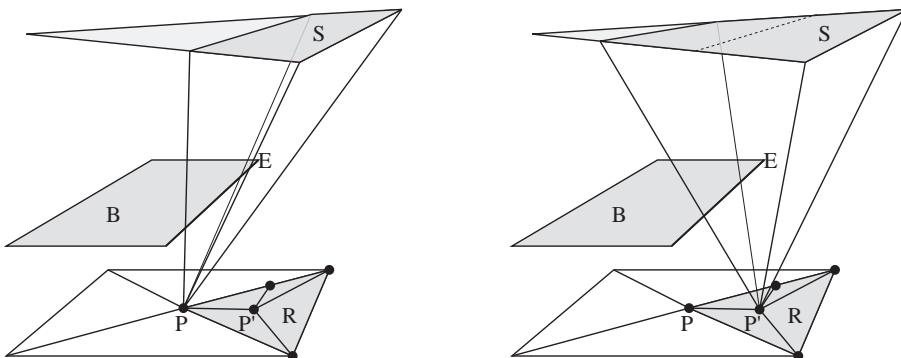


Figure 4.28: Taking advantage of invariant backprojections. The link between triangle R and source S contains no visual event. The view of S from P is qualitatively equivalent to the view from P' .

The visibility skeleton implicitly contains the information of a complete discontinuity mesh. Moreover, the information is available for any pair of polygons. Consider a polygon-polygon link where one polygon S is considered as a source while R is the receiver. The visible part of S is invariant from the points of R if no visual event related to S lies inside R ; this is exactly the information encoded in the polygon-polygon link (see Fig. 4.28).

Testing if the visibility is invariant between two (sub)-polygons thus simply consists in checking if the corresponding polygon-polygon link contains a visual event.

If no event is found, visibility is invariant and we just need to compute the numerical value of the form-factor for points inserted by subdivision. We use the information encoded at the vertices of the higher level (P in Fig. 4.28) to compute the form factor at the new vertices (P' for example). We use the formula of section 3.2 and Fig. 4.8. The values of the angles subtended by the edges are different from the initial vertices and need recomputation.

This reduces both the computation of the new visual events generated by the new points and their storage.

7.2 Time-memory tradeoff

We now show how to decrease memory requirements even if the visibility of a source varies over a receiver. We start from a simple observation: Consider Fig. 4.29 where a link between two polygons R and S is represented.

²In addition, in our implementation the mesh information is very costly in memory because we have used code from different sources requiring different data-structures, resulting in a highly redundant mesh data-structure.

A first receiver refinement results in the subdivision of R . One of these triangles R_i is then further subdivided. In our original method presented in section 4.2, the visibility between each R_j and S or from any new point P is deduced from the visibility information between R_i and S . However, R_j can also be considered as a sub-element of R , and P is also a point inside R . Visibility can be computed from the polygon-polygon link between R and S , as if the first refinement had directly resulted in the smaller sub-elements.

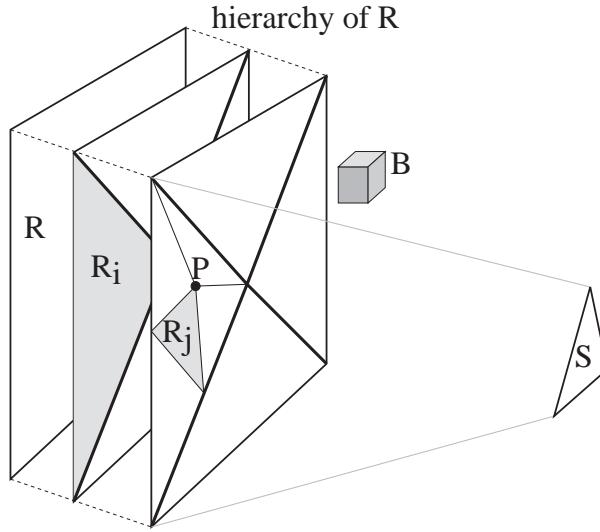


Figure 4.29: Time-memory tradeoff. Information between R_j and S can be deduced from the link between R_i and S , or from the link between R and S . The latter solution is more time consuming, since blocker B is unnecessarily considered. However it consumes less memory since the information between R_i and S need not be stored.

This latter method has the disadvantage that more blockers usually lie between R and S than between R_i and S (blocker B in our example). The visibility update will thus be more costly. However, it avoids the storage of the visibility information between R_i and S . We have a typical time vs. memory tradeoff.

In the scene we have tested, we estimate that it is for example useful to store this information in the first refinement of a link related to a wall or to the floor since many blockers are involved. In the subsequent refinements however, the number of blockers is usually low and the information of the upper level of the hierarchy could be used without incurring a time-penalty which is too large.

A simple criterion to decide whether to store visibility information or not is to consider the difference of the number of blockers between the parent link and the child link. Note that this criterion is very similar to the method proposed in the previous section: if the visibility is invariant, the number of blockers is constant and no information needs to be stored. A memory budget could also be assigned for the visibility information. Once it is filled, no more information is stored.

The issue of the refinement criteria proposed in section 5.4 then has to be handled with care since visibility information is no longer available in its entirety. Visual events are still encoded, but they are stored at a higher level of the link hierarchy, and there is no guarantee that they lie in the correct sub-polygon. The criteria can be computed with all visual events stored at the higher level, if one of them should cause refinement, we test if it actually lies inside the sub-polygon.

We believe that these two improvements will decrease the memory consumption of our method. The first solution will also lower computation time, and we believe that the second scheme will not incur a large time penalty.

8 Discussion

8.1 Summary

We have presented a new hierarchical radiosity algorithm using the Visibility Skeleton. We have also introduced update algorithms permitting the maintenance of consistent views at vertices added to a polygon due to subdivision, as well as the resulting sub-faces.

The visibility skeleton permits the computation of exact point-to-polygon form-factors for any vertex/polygon pair in the scene. It also provides detailed visibility information between any (sub)polygon-(sub)polygon pair.

We have introduced a novel hierarchical radiosity algorithm based on a “lazy wavelet” or “sub-sampling” type multi-resolution representation. The basic data structure used is a non-uniform hierarchical triangulation, which consists of a hierarchy of embedded constrained Delaunay triangulations. By maintaining radiosity differences at subdivided vertices, we introduce a linear “push” step, resulting in higher quality radiosity reconstruction at the leaves. A new, perceptually-based, discontinuity driven refinement criterion has also been introduced, resulting in hierarchical subdivision of surfaces well adapted to shadow variations. The results of our implementation show that we can generate accurate high-quality, view-independent solutions efficiently. The results also show that our approach is particularly well suited to previously hard-to-handle cases such as multiple light sources and scenes lit almost entirely by indirect illumination.

Memory consumption is the major drawback of our approach, but solutions based on a time-memory trade-off have been proposed. They are based on the detection of refinements where visibility is invariant or where it varies little.

8.2 Limitations

Two major limitations of this work can be identified, the first is high memory consumption (even with the proposed improvements) and the second is numerical robustness problems of the algorithms used.

The memory usage of the initial skeleton data structure is high, and can often have quadratic storage growth in the number of input polygons, depending on the how complex the visibility relations are between polygons. Even for simple environments, our method uses very large amounts of memory (see Table 4.2). To make our approach practical for large scenes, it is evident that we need to adopt one or a combination of the following strategies: lazy or on-demand skeleton construction, divide-and-conquer strategies (similar to *e.g.*, [HT96]) or a clustering approach allowing a multi-resolution representation. Some ideas in these directions can be found in the section 8.4 on future work.

Numerical robustness and the treatment of degenerate cases are important issues. As discussed in the previous chapter, despite the simplicity of the construction algorithm degenerate cases can cause problems with the skeleton construction. Moreover, in the case of subdivision, many visual events coincide, causing problems of coherence both for the ray-tracing step (for node creation) and the adjacency determination. These problems are particularly evident in the case of view updates. A coherent and consistent treatment of degeneracies is planned, but is a research topic in itself and beyond the scope of this thesis. Insertion of points in the mesh also causes problems, especially during subdivision due to numerical imprecision. Symbolic calculations instead of numerical intersections could potentially resolve most of these problems.

8.3 Advantages

The visibility-driven hierarchical radiosity algorithm introduced here has many advantages. First we achieve visually accurate shadows using discontinuities and exact point-to-polygon form-factors, for both direct and indirect illumination. The new hierarchy of triangulations data-structure, the novel two link types and the multi-resolution point-area link representation allow accurate linear reconstruction of radiosity over irregular meshes. The global treatment of visibility and discontinuities permits the definition of an efficient refinement oracle. Using a perceptually based method to estimate shadow importance, our refinement algorithm has proven to be very efficient for previously hard-to-handle scenes such as scenes lit with multiple light sources and scenes lit mainly by indirect light. As part of an informal comparison, we have seen that Hierarchical Radiosity uses more computation time to produce much lower quality results, as would be expected.

Approaches such as that of [LTG93] based on discontinuity meshing have difficulty with large numbers of light sources, since the number of discontinuities becomes unmanageable very quickly. This has consequences

both on computation time and on robustness in the construction of the discontinuity mesh. For similar reasons, no discontinuity-based *hierarchical* lighting algorithm has been proposed previously in which discontinuities are treated for indirect light transfers.

8.4 Future work

Scalability

The improvements we have proposed to decrease memory consumption should be implemented and tested.

The advantage of the skeleton construction is that it is local, and thus can be built in a “lazy” or even “on-demand” fashion. Using to-be-defined criteria, we could compute only the parts of the visibility skeleton related to “important” light transfers. This information could be deleted once used, thus dramatically reducing the memory requirements.

The hierarchical visibility structure discussed in the future work section of the previous chapter could be directly applied to global illumination simulation. The framework of radiosity with clustering [SAG94, Sil95] is the natural extension of hierarchical radiosity. It could benefit from a multiresolution yet accurate visibility data-structure.

Error control

The information encoded in the skeleton also permits the computation of the gradient or higher order derivatives of the lighting function[Arv94, HS95, HS99]. Such an information could be useful to derive more efficient refinement criteria, or to interpolate the lighting function.

Error control techniques for radiosity [ATS94, LSG94, HS99] have often neglected the error due to visibility because it is very intricate and also because exact visibility information is hard to obtain. The visibility skeleton provides the necessary information, making it possible to accurately control the precision of the simulation.

A complete error control of the simulation is however rarely necessary. Nonetheless, this kind of approach can be simplified and approximated to obtain some criteria which are more global. The oracles we have implemented perform an estimate on the direct interaction between two patches. The influence is nonetheless more global, since this light energy is then reflected to all the scene.

Meshing

The quality of a mesh is crucial for finite element methods. Thin triangles should be especially avoided. Delaunay triangulation, particularly constrained Delaunay triangulation, does not ensure such a property. Some standard meshing techniques (see e.g. [She96]) should be used to further improve our meshes, by first inserting some new points on constrained edges which do not verify the Delaunay property.

Moreover, a Delaunay triangulation does not provide the best interpolation of a function. Consider the example in Fig. 4.30 which represents a function defined on a square sampled at the corners. Two triangulations are possible, depending on the diagonal which is chosen. The criterion should not be based on a Delaunay property (both are equivalent) but should take the gradient of the function into account [ARB90, Sch93]. Such a scheme could improve the quality of our lighting representation.

Perceptual criteria

The criteria we have developed can also be adapted for standard radiosity systems where the skeleton information is not available. In particular, the separation between a pure radiometric oracle (variation of unoccluded radiosity) and a visibility oracle (“amount of shadow” and variation in penumbra) is a promising approach for efficient refinement. The use of the perceptual metric moreover permits an intuitive and predictable setting of the simulation.

More involved metrics could be used [Mys98, BM98, RPG99, FPSG97, PFFG98]. In particular, the spatial frequency component should be taken into account, since the human eye is less sensitive to error in quickly varying regions, for example a carpet with a very noisy texture.

We believe that the use of perceptual metrics is a powerful technique to optimize computation in computer graphics. Animation techniques could also benefit from these efficient criteria, spending more computation resources in part of the scene/movement which are most noticeable by a human observer.

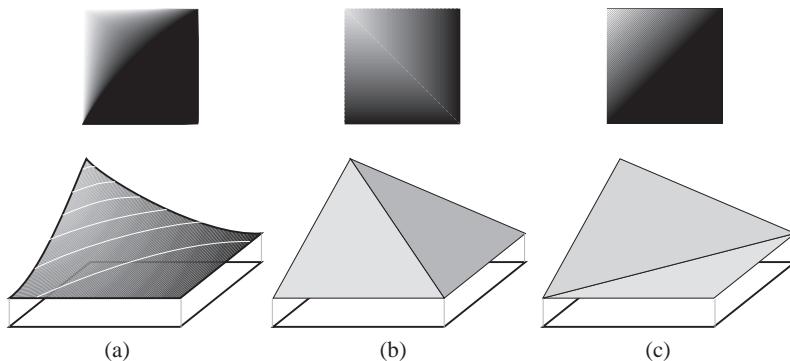


Figure 4.30: Optimizing a triangulation. (a) Exact function. (b) Bad triangulation. (c) Good triangulation.

Other issues

The skeleton could also be used for Monte-Carlo methods. In the case of standard Monte-Carlo techniques, the inherent random nature of the sampling makes it hard to take coherence into account. However, more recent approaches such as Metropolis light transport [VG97] could be coupled with the skeleton for a better exploration of the path space. It could also be used to find some good initial samples in path space.

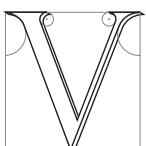
Extending the skeleton and the resulting illumination algorithm to dynamic scenes is another promising research direction.

CHAPTER 5

General Occlusion Culling using Extended Projections

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 Urbinate*



ISUALIZATION of very complex geometric environments is a major goal of interactive rendering systems. Despite the impressive improvement in processor power and graphics hardware speed of modern workstations, it is still currently difficult or even impossible to visualize very complex models (millions of polygons) in real time, on mid- or even high-range graphics workstations. We have identified three important restrictions which often impede visualization of complex scenes: (i) the limited speed of the graphics display (*i.e.*, the number of primitives the hardware can render per frame), (ii) the fact that different classes of models (*e.g.*, architectural scenes, scenes with concave objects, etc.) often have very different visibility properties, making solutions which work for all cases hard to develop and (iii) the fact that the model of the entire scene often is too large to fit in main memory or to be transmitted over a network.

The new approach we present here is a *preprocess* of a complex scene, which generates a decomposition of space into cells. A novel algorithm is introduced, using *extended projections* to compute an extended depth map with respect to all viewpoints in a cell. The new approach incorporates the combined effect of multiple occluders (*occluder fusion*) and can handle concave objects. The extended depth maps are used to conservatively¹ calculate the potentially visible geometry for each cell. The visibility cell structure is built adaptively, using recursive subdivision and hierarchical testing for potentially hidden objects; it is then used by an interactive viewer of complex scenes. Our algorithm opens the way to overcoming the restrictions mentioned above, permitting highly interactive display of very complex, general (not application-specific) models on mid-

¹Note that throughout this chapter we employ the term *conservative* to the limit of the resolution of the image to which we rasterize the projections.

and high-range graphics workstations. Potential applications for our new approach include video-games and network rendering, for which the reduction of the size of the geometry sent to the pipeline is paramount.

This chapter starts with a presentation of our general method. We then present some methods for the actual computation of extended projections in section 2, before we propose an important improvement for some special cases of occluders in section 3. We then introduce a reprojection operator and the occlusion sweep in section 4. Section 5 presents our adaptive preprocessing, while section 6 presents the results of our implementation. Finally section 7 proposes a discussion and some future work issues.

1 Extended projections

We present a preprocessing method which subdivides the regions where an observer can move into viewing cells. For each viewing cell we compute a set of objects which are potentially visible from all the points inside the cell (the so-called *PVS*, potentially visible set). For this, we perform a conservative occlusion test for each object of the scene. Our occlusion test uses a representation of the occlusion caused by occluders which is based on *extended projections* onto a plane.

1.1 Principle

Our method can be seen as an extension to volumes of on-line occlusion culling with respect to a single viewpoint. In image-space point-based occlusion culling [GKM93, ZMH97], occluders and occludees are projected onto the image plane. Occlusion is detected by testing if the projection of the occludee is contained in the projection of the occluders (overlap test) and if the occludee is behind (depth test). In some methods [GKM93] both tests are performed in a single pass using the depth-buffer.

We extend these single viewpoint methods to volumetric viewing cells. *Extended projection operators* are defined for occludees and occluders in the next section. An occludee is hidden with respect to all viewpoints within the viewing cell if: (i) the extended projection of the occludee is contained in the extended projection of the occluders and (ii) if the occludee is behind the occluders. The extended projection operators are different for occluders and occludees.

The intuition behind our extended projection operators is to underestimate the projection for the occluder, and to overestimate them for occludees. The extended projection of for the entire cell is an underestimate with respect to each individual viewpoint for the occluder, and an overestimate for the occludee.

Even though we describe our method for a single plane, six planes will actually be necessary to test occlusion in all directions. The important issue of the position of the projection plane(s) is deferred until section 5.2.

We define a view as a perspective projection from a point onto a projection plane. However, in what follows, the projection plane will be shared by all viewpoints inside a given cell, resulting in sheared perspective, as if one were moving inside a room and describing what is seen through a fixed window. Note that visibility is equivalent to the case in which a plane perpendicular to the optical axis is used and that these projections can be handled by a standard perspective projection matrix.

1.2 Extended projections

Definition 1 *We define the extended projection (or Projection) of an occluder onto a plane with respect to a cell to be the intersection of the views from any point within the cell.*

Definition 2 *The extended projection (or Projection) of an occludee is defined as the union of all views from any point of the cell.*

Fig. 5.1 illustrates the principle of our extended projection.

In what follows, we will simply use *Projection* to refer to an extended projection. The standard projection from a point will still be named *view*.

This definition of *Projection* yields conservative occlusion tests. To show this consider first the case of a single occluder. Assume that an occludee is behind the occluders. It is declared hidden if its *Projection* is contained in the *Projection* of the occluder. This means that the union of its views is contained in the intersection

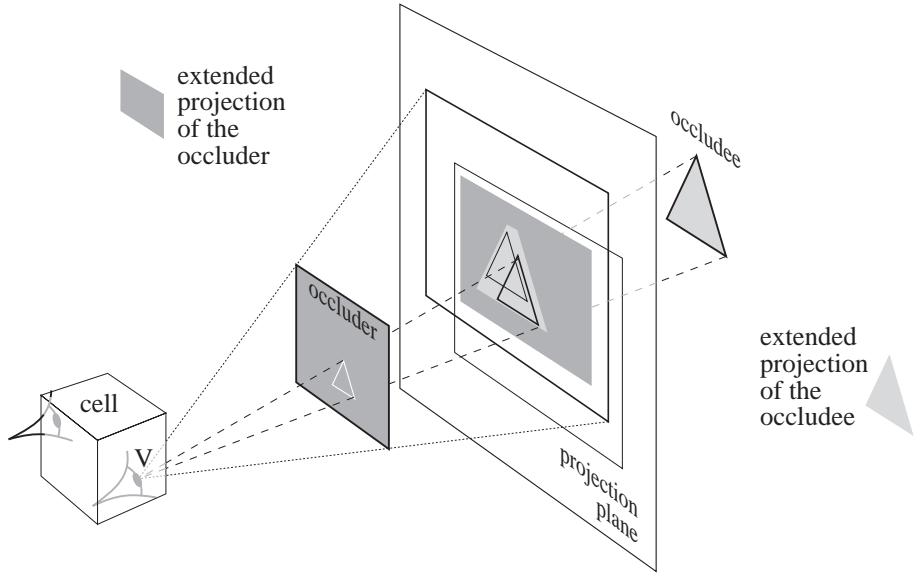


Figure 5.1: Extended projection of an occluder and an occludee. The view from point V is shown in bold. A view from another viewpoint is also shown with thinner lines. The extended projection of an occluder on a plane is the *intersection* of its views. For an occludee, it is the *union* of the views.

of the views of the occluder. From any viewpoint V inside the cell, the view of the occludee is contained inside the view of the occluder. To summarize, for any viewpoint V in the cell we have:

$$\text{view}_V^{\text{occludee}} \subset \bigcup_{W \in \text{cell}} \text{view}_W^{\text{occludee}} \subset \bigcap_{W \in \text{cell}} \text{view}_W^{\text{occluder}} \subset \text{view}_V^{\text{occluder}}$$

The occludee is thus hidden (if the depth test is satisfied). See Fig. 5.1 for an illustration.

Consider now the case of an occludee whose Projection is contained in the cumulative Projection of two (or more) occluders. This means that from any viewpoint V in the cell, the view of the occludee is contained in the cumulative view of the occluders. We have:

$$\text{view}_V^{\text{occludee}} \subset \bigcup_{W \in \text{cell}} \text{view}_W^{\text{occludee}} \subset \bigcup_{\substack{\text{occluders } W \in \text{cell}}} \bigcap_{W \in \text{cell}} \text{view}_W^{\text{occluder}} \subset \bigcup_{\text{occluders}} \text{view}_V^{\text{occluder}}$$

The occludee is thus also hidden (Fig. 5.2). Our Projection operators handle *occluder fusion*, that is, occlusion caused by the joint effect of multiple occluders.

Unfortunately there is no one-to-one correspondence between a point in a Projection and a projected point of an object, as with standard perspective view. Consider the situation depicted in Fig. 5.3(a). Depending on the viewpoint V , a point P in the Projection of the occluder corresponds to the view of different points of the occluder.

Note that convexity is not required in any of our definitions, just as for point-based occlusion-culling. Nonetheless, we will see that it is easier to compute a Projection in the convex case, and in section 3 we show that the efficiency of Projections can be improved if occluders are convex or planar.

Let us give another interpretation of these operators in the special case where the projection plane is behind the objects. The Projection of an occluder is in fact its *umbra* if the cell is considered as a light source. A point P in the Projection of an occluder A is in the view of A from any viewpoint V in the cell (see Fig. 5.3(a)). Thus, for any point V in the cell, the ray originating from V going through P intersects A . P is in the umbra of A on the projection plane. Similarly, the Projection of an occludee can be seen as its *penumbra*.

1.3 Depth

The overlap of Projections is not sufficient to conclude that an occludee is hidden; A depth comparison has to be performed. It is however more involved than in the single viewpoint case since a simple distance between

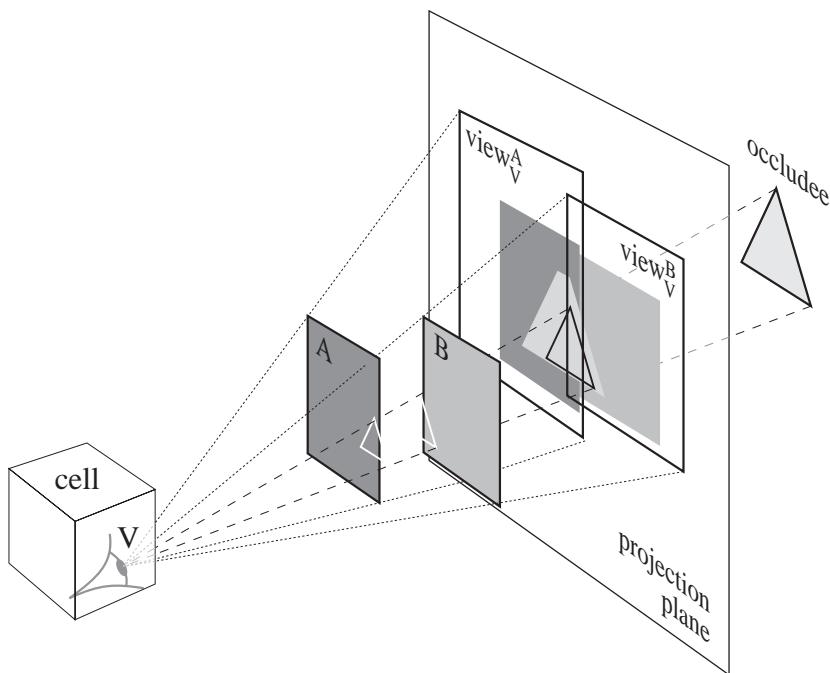


Figure 5.2:

Extended projections handle *occluder fusion* of two occluders A and B . We show the example of a view from point V . The view of the occludee is contained in the cumulative view of the two occluders, as determined by the extended projection occlusion test. Note that a single occluder does not hide the occludee.

the viewpoint and a projected point cannot be defined. This is because a set of viewpoints are considered, and also because a point in a Projection does not correspond to the view of a unique point of an object as seen in Fig. 5.3(a).

Our definition of depth must be consistent with the properties of occlusion; For each ray emanating from a viewpoint inside the cell and going through the projection plane, depth must be a monotonic function of the distance to the viewpoint. The most obvious definition of depth is along the direction orthogonal to the projection plane. We chose the positive direction leaving the cell and we place the zero at the projection plane. Objects between the projection plane and the cell thus have negative depth. Occluding objects will have smaller depth values than hidden objects.

We now have a unique depth function which is coherent for all viewpoints in the cell. Consider a point P in the Projection of an occluder as illustrated in Fig. 5.3(b). It corresponds to many points of the occluder (e.g., A_1 and A_2) as defined by the intersection of all lines going through this point and the cell.

Definition 3 *We define the extended depth (or Depth) of a point in the Projection of an occluder as the maximum of the depth of all the corresponding projected points.*

Similarly, we define the extended depth (Depth) of a point in the Projection of an occludee as the minimum depth of its projected points.

In what follows, we will refer to this quantity simply as *Depth*. If the Depth of a point in the Projection of an occluder is smaller than the Depth in the Projection of an occludee, all the corresponding points of the occludee are behind the occluder from any viewpoint inside the cell. Our definition thus yields conservative depth tests.

Consider now the case of multiple occluders. We want to consider the occlusion caused by the occluder closest to the cell. To perform the depth test, we thus consider the occluder with the *minimum Depths*. Recall that for each occluder the Depth of a point P is the *maximum* depth of the corresponding projected points. The Depth of a point in the cumulative Projection of multiple occluders is thus the *minimum of the maxima* of the depth of the projected points.

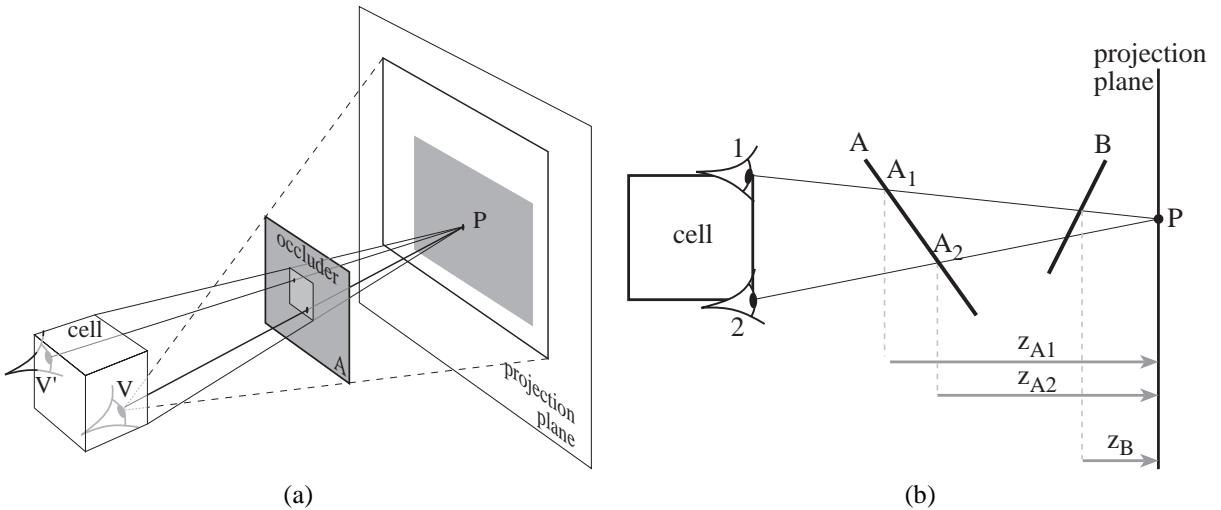


Figure 5.3: (a) Inverse of a point P in a Projection. We show the point of the occluder corresponding to P in the view from V . For the second viewpoint V' , P corresponds to the view of another point of the occluder. The light region of the occluder corresponds to the set of points which project on P . (b) We consider depth along the direction orthogonal to the projection plane (Note that the depth in this figure are all negative since zero is defined on the projection plane). The Depth of a point P in the Projection of an occluder A is the maximum of the depth of the corresponding points (z_{A1} here). In the case of multiple occluders, the occluder closest to the cell is considered (A here).

For this purpose we define the extended depth map or *Depth Map*. For each point of the projection plane, the value of the Depth Map is the Depth of the occluder closest to the cell which projects on it. If no occluder projects on a point, an “infinite” value is used.

A simplified version of the depth test can be used. If the occluders are all in front of the projection plane and the occludees are all behind, then the overlap test is enough to determine occlusion. This will prove useful for occluder reprojection. We will not use this simple case until section 4. Until then, we consider the use of a Depth value for each point of an occluder Projection.

1.4 Overview

We now present an overview of our method. We first compute the Projections of the occluders on a projection plane. Then, for each occludee we test if its Projection overlaps the Projection of the occluders, and whether it lies behind, that is, whether it has larger Depth values. For each point inside the Projection of an occludee we have to test if its Depth is larger than the Depth of an occluder Projection.

This permits a simple algorithmic description of our occlusion test: For each point of the Projection of the occludee, test if the Depth is larger than the corresponding value of the Depth Map.

If we use the simplified depth test (if the projection plane is between the occludees and occluders), we similarly define the extended occlusion map or *Occlusion Map*. Each point of the projection plane is assigned a boolean value which is true only if this point belongs to the Projection of at least one occluder.

1.5 Implementation choices

Until now, our definitions have been quite general, and do not depend on the cell, plane, occludee nor on the way the overlap or depth test are performed. We now present the choices we have made for our implementation.

The viewing cells are non-axis-aligned bounding boxes. The projection planes will be restricted to the three directions of the cell (note that these three directions depend on the cell). The occludees are organised in a hierarchy of axis-aligned bounding boxes.

We will use a bitmap representation of the Depth Map. This is our only concession to conservatism (it can be alleviated as will be discussed later). It allows the use of the graphics hardware which simplifies most of the

computation, and it avoids the robustness issues inherent to geometrical computations.

We store a Depth value for each pixel. As described above, we consider the minimum of the Depths of this pixel. Occluder fusion is handled by the natural aggregation in the Depth Map. Following [GKM93] we organize the Depth Map into a pyramid for efficient testing. We call it the Hierarchical Depth Map. Similarly, we will also use a Hierarchical Occlusion Map in the spirit of Zhang *et al.* [ZMH97].

2 Computation of the extended projections

We have defined *extended projection operators* (Projection) as well as *extended depth* (Depth) which permit conservative 3D occlusion tests on a projection plane. These tests take *occluder fusion* into account. We represent the occlusion on the plane using a *Hierarchical Depth Map*. We now describe how to actually compute the Projections of the occludees and occluders.

We first present an approximate yet conservative way to compute the Projection of occludees. We then give a method for the Projection of convex occluders, and a method to handle concave occluders.

2.1 Occludee Projection

Recall that the Projection of an occludee is the union of its views. Our cells are convex as is also the bounding box of an occludee. The Projection of such a box reduces by convexity to the 2D convex hull of its views from the vertices of the cell.

This computation could be implemented using standard geometric algorithms, by computing the views, their 2D convex hull and rasterizing it for depth test. We however use a conservative approximation which simplifies the computation and the rasterization phase.

We use the bounding rectangle of the Projection on the projection plane as shown in Fig. 5.4. The goal is to overestimate the Projection of the occludee. To do this, we split the problem into two simpler 2D cases. We project onto two planes orthogonal to the projection plane and parallel to the sides of the cell. The 2D projection of the cell is a rectangle, while the 2D projection of the occludee bounding box is a hexagon in general (Fig. 5.4 shows a special case for simplicity).

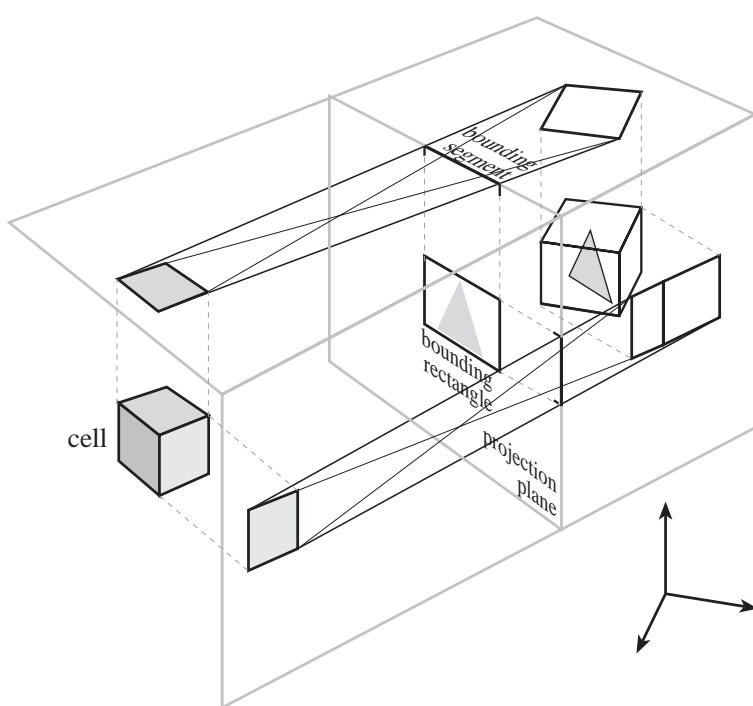


Figure 5.4: Occludee Projection is reduced to two 2D problems.

We then compute the separating and supporting lines of the rectangle and hexagon. The intersections of these lines with the projection plane define a 2D bounding segment. A bounding rectangle is defined by the cartesian product of the two 2D segments as illustrated in Fig. 5.4. We use this bounding rectangle as an overestimated approximation of the occludee Projection. Separating lines are used when the occludee is between the cell and the projection plane, while supporting lines have to be used if the occludee lies behind the plane.

This method to compute an occludee Projection is general and always valid, but can be overly conservative in certain cases. In section 3 we will present an improvement of this Projection for some particular configurations.

2.2 Convex occluders Projection using intersections

The Projection of convex occluders has properties which are similar to the occludee bounding box projection. By convexity of the cell and occluder, the intersection of all possible views from inside the cell is the intersection of the views from the vertices of the cell (see Fig. 5.5(a)). This Projection can be computed using standard geometric intersection computation.

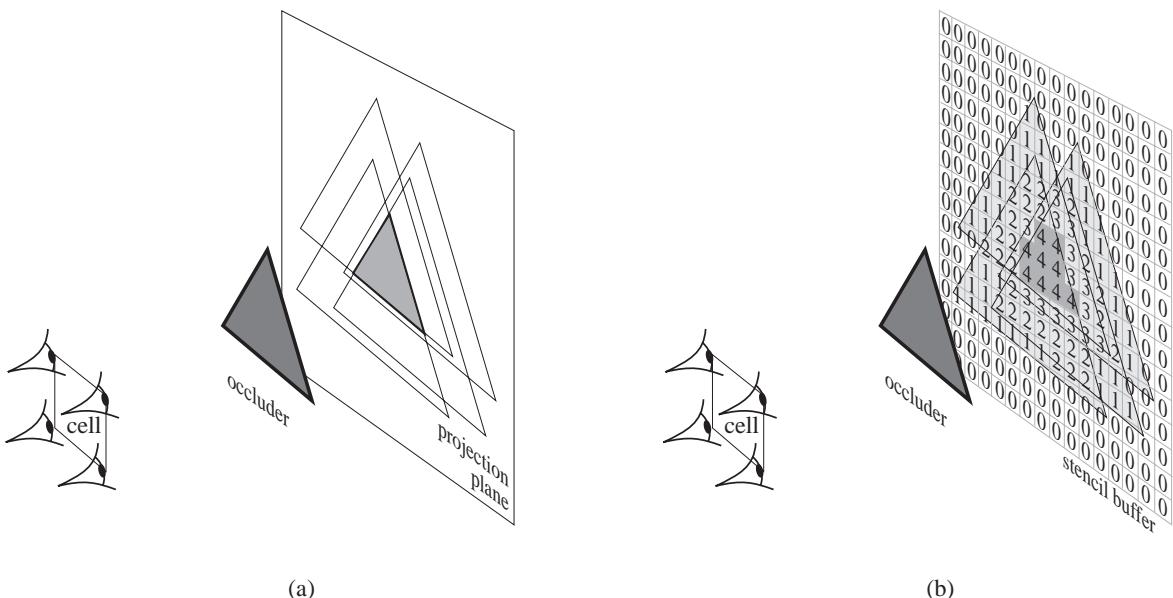


Figure 5.5: Convex occluder Projection. (a) The Projection of an occluder is the intersection of the views from the vertices of the cell (we have represented a square cell for simplicity). (b) Computation using the stencil buffer. The values of the stencil buffer indicate the number of views projecting onto a pixel.

We have nevertheless developed an efficient method which takes advantage of the graphics hardware. It is a *multipass* method using the *stencil buffer*. The stencil buffer can be written, and compared to a test value for conditional rendering. Its original use was to prevent the rendering of 3D scenes on pixels representing a cockpit [MBGN98].

The basic idea of our method is to project the occluder from each vertex of the cell, and increment the stencil buffer of the projected pixels without writing to the frame-buffer. The pixels in the intersection are then those with a stencil value equal to the number of vertices, as illustrated in Fig. 5.5(b).

The consistent treatment of Depth values (as described in section 1.3) in the context of such an approach requires some care and is described in appendix C.

2.3 Concave occluder slicing

Concave polygonal meshes can be treated with the previous method, by Projecting each individual triangle. However, some gaps will appear between the Projections, losing the connectivity of occluders. We now present

a first method to treat concave occluders as single entities. Other approaches will be discussed in section 7.3.

Our method is based on this simple observation: the Projection of an object lying in the projection plane is the object itself. We thus consider the intersection of concave objects with the projection, which we call a *slice* of the object.

Slicing concave objects is straightforward: projection plane(s) are placed in such a way that they cut the object. The object is then intersected with the projection plane and a 2D contour found. It is important to slice the object at a point where the resulting contour is closed. The contour is then scan-converted onto the projection plane with a null Depth value to compute a Depth Map, or in Black to compute an Occlusion Map.

Evidently, a complex object can be sliced at a number of positions. Combining the results of the sliced occlusion maps requires conservative *reprojection* onto a different plane, as will be introduced in section 4.

3 Improved Projection of occludees

The Projection operators we have presented are *conservative*, meaning that they never erroneously identify an object which is actually visible as invisible. However, the opposite can occur, some objects may be declared potentially visible while they are not visible from any point inside the viewing cell.

In this section we present an improvement in the case of convex or planar occluders for configurations in which our initial Projection yields too conservative results. We first present some occlusion properties which allow the improvement of our Projection. We discuss improved Projection in 2D which is simpler. We then explain why the 2D results cannot be trivially extended to 3D and explain how these 2D results can nevertheless be used to improve 3D computation.

3.1 Some properties of umbra

In Fig. 5.6(a) we show in 2D a situation in which our Projection is too restrictive. The Projection of the occludee is not contained in the Projection of the occluder, even though the occludee is evidently hidden from any viewpoint in the cell. This configuration occurs because the occludee is between the occluder and the projection plane and because the occluder is convex. In what follows we discuss only the case where the occludee is between the projection plane and the cell. If the occludee is behind the plane, the Projection which we have presented in section 1.2 yields satisfying results.

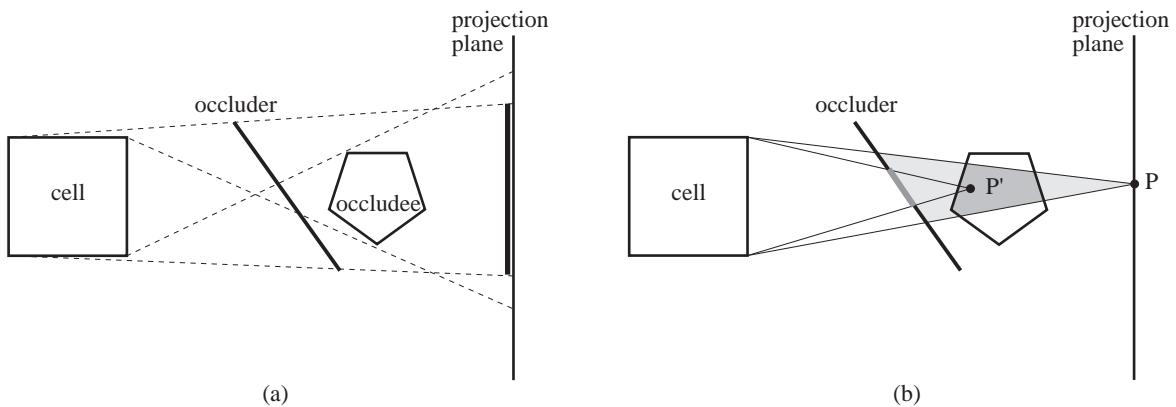


Figure 5.6: (a) Configuration where initial Projections is too restrictive. The Projection of the occludee is not contained in the Projection of the occluder, even though it is obviously hidden. (b) Any point P' inside the cone defined by P and the cell and behind the occluder is hidden.

Intuitively, testing the occludee penumbra against the occluder umbra is not sufficiently restrictive. As we shall see, substituting the penumbra with the occludee umbra for this configuration results in the required answer.

To cope with this case, we will define an improved Projection for occludees. It will be valid only in the case of convex or planar occluders, but occluder fusion will be handled. Note that the restriction is on the type

of *occluders* but that the improved Projection is defined for the *occludees*. We base our improved Projection on some properties of shadows of convex objects.

In what follows we use the term *cone* to name a general cone defined by an apex and a set of points (the term “pyramid” could also be used). We also use “cone” to mean a 2D wedge, which makes terminology more coherent with the 3D case. Similarly we will still use “projection plane” even though a line is actually considered in 2D.

Before defining the actual improved Projection we introduce two important properties which we will use to demonstrate that our improved Projection provides valid occlusion results.

Property 1 *For a given point P in the occluder umbra region with respect to a cell, all points P' in space behind the occluder, which are also contained in the cone defined by P and the cell, are hidden with respect to the cell.*

This property is illustrated in Fig. 5.6(b). The proof is evident for convex occluders since the cone defined by P' and the cell is contained in the cone defined by P . The section of the occluder which occludes P is a superset of the section which occludes P' . Note that this property is not valid for general concave occluders.

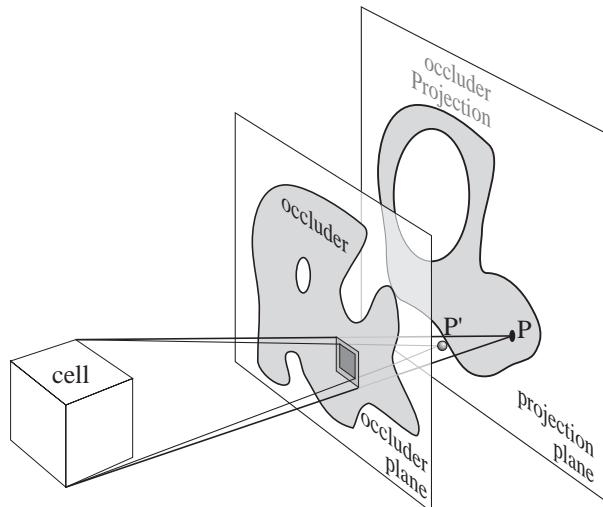


Figure 5.7: Case of a concave planar occluder. The intersection of the occluder and cone (defined by point P and the cell) is shown in dark grey. It is convex and equal to the intersection with the plane of the occluder.

However, the 3D case of concave planar occluders is similar to the convex case. Consider Fig. 5.7. If a point P is in the umbra of such an occluder, then the intersection of the occluder and the cone defined by P and the cell must be equal to the intersection of the cone and the plane of the occluder (otherwise there is a “hole” and P is not in umbra). This intersection is thus convex. The intersection of the cone defined by P' and the cell is a subset of this plane-cone intersection. P' is thus also hidden. The plane of the occluder need not be parallel to the projection plane, and the occluder can be concave and have holes.

Property 2 *To yield valid occlusion tests, the improved Projection of the occludee being tested must have the following property: The union of cones defined by each point of the improved Projection of the occludee and the visibility cell must contain the occludee.*

Consider the situation in Fig. 5.6(b). The points of the occludee contained in the cone defined by P and the cell are occluded by property 1. Thus, if the occludee is contained in the union of cones defined by the cell and points in the improved extended projection, any point of the occludee is in one of these cones and can be identified as hidden by property 1.

Note that occluder fusion is still taken into account: all points P defining the cones need not be hidden by the same convex or planar occluder.

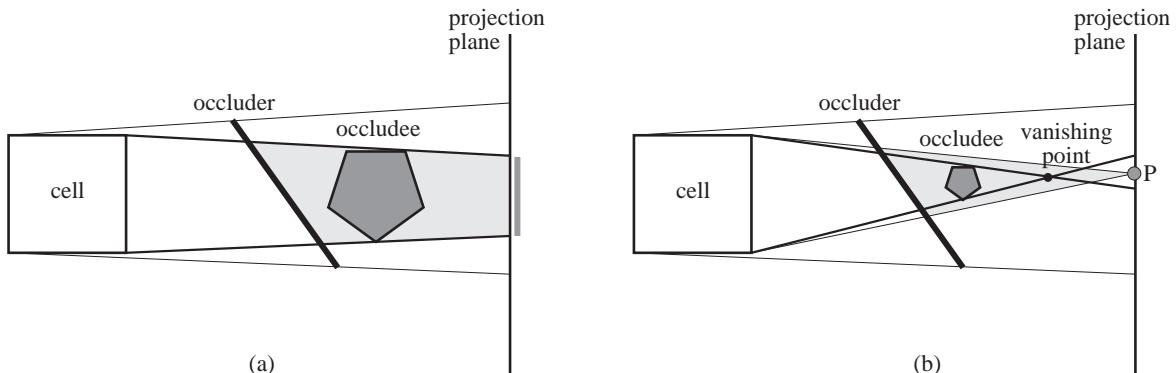


Figure 5.8: Improved Projection in 2D. (a) The 2D improved Projection is defined by the intersections of the projection plane with the supporting lines of the cell and the occludee. (b) When the vanishing point of the umbra of the occludee is in front of the projection plane, any point between the intersections of the supporting lines can be used.

3.2 2D improved Projection

An improved Projection respecting property 2 can be defined in 2D by considering the supporting lines of the cell and the occludee as illustrated in Fig. 5.8(a). The intersection of these lines with the projection plane define the limits of the improved Projection of the occludee.

However, if the occludee is too small, the two supporting lines intersect in front of the projection plane at the *vanishing point*. In this case, any point P between the intersections of the two supporting lines and the projection plane satisfies property 2, as illustrated in Fig. 5.8(b). Any cone defined by such a point and the cell contains the vanishing point and thus the object. In practice, we use the mid-point in our calculations.

Note that this computation using supporting lines is the same for objects behind the projection plane, since the supporting lines then define the limits of the union of the views of the occludee. As opposed to standard Projection presented in section 1.2, we no longer consider separating lines. In the case where the occludee is in front of the projection plane, we have substituted the penumbra by the umbra.

We now summarize our 2D *improved Projection* for any occludee (in front or behind the projection plane): We compute the intersection with the projection plane of the upper and lower supporting lines defined by the cell and the occludee. If the plane is beyond the vanishing point (*i.e.*, if the intersection of the lower line is above the intersection of the upper line), we consider the mid-point of the two intersections, otherwise we consider the entire segment. The point or segment are then used in the occlusion test. This is illustrated in Fig. 5.8. The 2D improved Projection of an occludee is a segment or a point, depending on whether the projection plane is behind the vanishing point.

3.3 3D improved Projection

Unfortunately, in 3D supporting planes cannot be used as simply as supporting lines, and the vanishing point is ill-defined. Even if the umbra volume of an occludee intersects the projection plane, the union of cones defined by the cell and points of the plane in umbra are not guaranteed to contain the occludee.

We thus project on two planes orthogonal to the projection plane and parallel to faces of the cell, as illustrated in Fig. 5.9. On each plane we use our 2D improved Projection. The cartesian product of these two 2D improved Projections defines our 3D improved Projection.

This improved Projection verifies property 2. Consider a point P' of the occludee. Its projection P'_i onto each of the two planes is inside a cone defined by a point P_i of the 2D improved Projection and the projection of the cell. The 3D point P' is thus in the cone defined by the cell and the cartesian product P of P_1 and P_2 (see Fig. 5.9).

This is true because the cell is the cartesian product of its 2D projections since it is a box and the 2D planes are parallel to the cell faces. Thus a cone defined by a point P and the cell is the cartesian product of the cones defined by the 2D projections of the cell and the 2D projection of P .

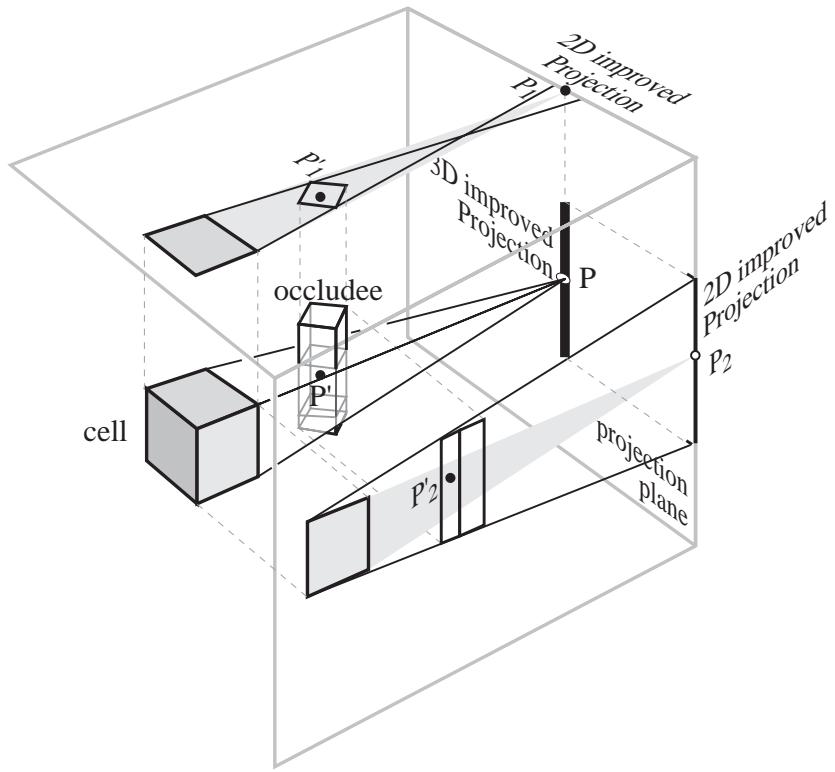


Figure 5.9: The 3D improved Projection is the cartesian product of two 2D improved projections. Any point P' of the occludee is contained in a cone defined by one point P of the 3D improved projection and the cell. This cone can be constructed by considering the two corresponding 2D projections.

The 3D improved Projection is the cartesian product of 2D improved projections which are points or segments. It is a rectangle (segment \times segment), a segment (segment \times point) or a point (point \times point).

Our 3D improved Projection thus yields conservative occlusion tests in the case of convex or planar blockers. As mentioned previously in the 2D case, occluder fusion is still handled since all cones containing the occludee need not be hidden by the same occluder.

4 Occluder reprojection and occlusion sweep

The choice of the projection plane has a great influence on the efficiency of occlusion testing using Projection, and especially on the handling of occluder fusion (as will be discussed in section 5.2). Contradictory plane locations can be required to handle the fusion of different groups of occluders as illustrated in Fig. 5.10. To cope with this, we use a first projection plane, compute a, Occlusion Map where occluder Projections aggregate, then re-Project this Occlusion Map on a new projection plane. The natural aggregation in the first Occlusion Map results in a larger *equivalent occluder* whose occlusion can be tested on the second projection plane. The same applies when slices of concave objects are used. In a certain sense, our scheme can be seen as an image-based simplification and aggregation of occluders.

We first justify why and to what extent re-projection is valid. We then explain how Occlusion-Maps can be efficiently reprojected onto another parallel projection plane. We finally extend this concept to an *occlusion sweep* where a plane is swept from the cell while aggregating occlusion.

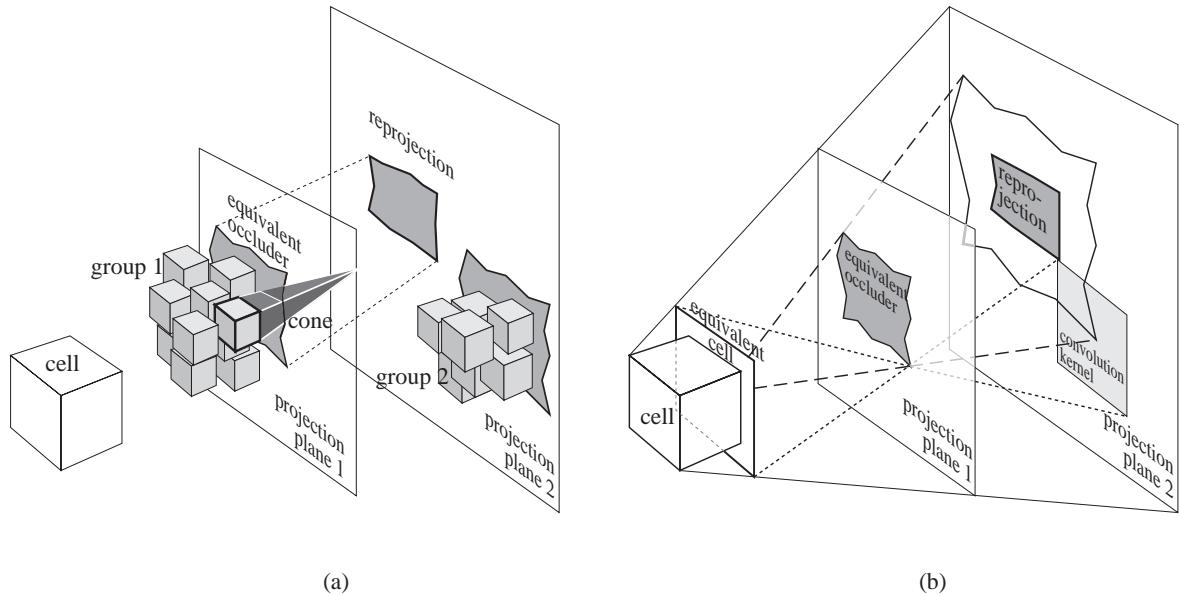


Figure 5.10: (a) If projection plane 2 is used, the occlusion of group 1 of occluders is not taken into account. The shadow cone of one cube shows that its Projection would be void since it vanishes in front of plane 2. The same constraints apply for group 2 and plane 1. It is thus desirable to Project group 1 onto plane 1, and *re-Project* the aggregate equivalent occluder onto plane 2. (b) Occluder reprojection. The extended occlusion map of plane 1 is projected on plane 2 from the center of the equivalent cell. It is then convolved with the inverse image of the equivalent cell.

4.1 Validity

We now show that the Projection of several occluders can be used as a single conservative equivalent occluder, *i.e.*, an occludee hidden by this Projection is also hidden by the occluders. We prove the following more general property.

Property 3 Consider an extended light source, any object A (convex or concave) and U the umbra cone of A . Then the shadow of any subset U' of U lies inside U .

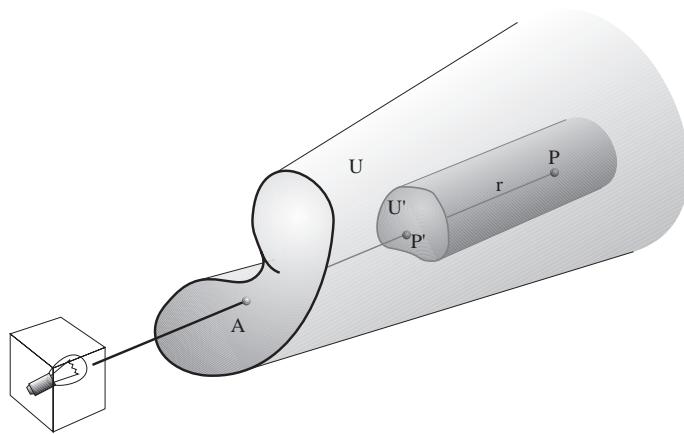


Figure 5.11: Umbra of the subset of an umbra. Point P is in the umbra of U' which is a subset of the umbra U of A . It is thus also in the umbra of A .

To prove this, consider a point P in the umbra of U' (Fig. 5.11). Any ray r going through P and the source intersects U' . Consider an intersection point P' . Since $P' \in U' \subset U$, then P' is in the umbra of A . Thus any ray

(r for example) going through P' and the source intersects A . We have shown that any ray going through P and the source intersects A . P is thus in the umbra of A .

Note that property 3 presupposes neither convexity nor planarity of the object A .

If the cell is considered as a light source, this proves that any subset of the umbra of a set of occluders is a conservative version of these occluders. As we have seen, the Projection of an occluder which lies in front of the projection plane is its umbra on the plane. This Projection can thus be re-Projected as a new occluder. If the occluder lies behind the projection plane, its Projection does not lie inside its umbra because the projection plane is closer to the viewing cell than the occluder. Thus property 3 does not apply.

We thus only consider occluders behind the projection plane, which is why we do not consider their Depth and use Occlusion Maps instead of Depth Maps in the initial projection plane. An Occlusion Map defines a planar occluder which is well suited for our improved Projection. Reprojection will be performed only onto planes which are behind the initial plane. However the depth of the initial projection plane can be used for the Depth Map of the final projection plane.

The reprojection scheme we are about to present is in fact an extended projection operator for the special case of planar blockers parallel to the projection plane.

4.2 Reprojection

We base our reprojection technique on the work by Soler and Sillion [SS98a, Sol98] on soft shadow computation. They show that in the case of planar blockers parallel to the source and to the receiver, the computation of soft shadows is equivalent to the convolution of the projection of the blockers with the inverse image of the source. This convolution can be computed efficiently using Fast-Fourier-Transform. They extend this scheme to the general case by projecting the actual geometry onto three parallel planes, computing the soft shadow using convolution, then re-projecting it on the actual receiver. This computation is approximate but yields impressive results.

We are nearly in the ideal case for the Soler and Sillion algorithm: our blocker (the occlusion map of the initial projection plane) and the receiver (the new projection plane) are parallel. However our light source (the cell) is a volume. We define an equivalent cell which is parallel to the projection planes and which yields conservative Projection on the new projection plane. Its construction is simple and illustrated in Fig. 5.10(b). We use the fact that our projection planes are actually bounded. Our equivalent cell is the rectangle defined by the face of the cell closest to the plane and the supporting planes of the cell and the final projection plane (the projection rectangle in fact).

Any ray going through the cell and the projection plane also intersects our equivalent cell. Thus if an object is hidden from the equivalent cell, it is also hidden from the cell.

The convolution method computes continuous grey levels. To obtain a binary Occlusion Map, thresholding is performed. In practice, since we use hardware convolution on integers, we use as scaling factor together with clamping (multiplying by 255 corresponds to setting to 255 all values which are not null).

Fig. 5.10 shows that the re-Projection of the extended projection from plane 1 to plane 2 allows us to handle the cumulative occlusion of both group 1 and 2 and their occluder fusion. Recall that a Depth map can be used on the final plane. The Depth of the re-Projected equivalent occluder is the depth of the initial plane.

4.3 Occlusion sweep

To handle the case where multiple concave or small occluders have to be considered, we generalize re-Projection to the *occlusion sweep*. It is a sweep² of the scene by parallel projection planes leaving the cell on which occlusion is aggregated due to re-Projection.

The principle is simple as illustrated in Fig. 5.12. We Project the convex occluders which lie in front of the current projection plane P onto P . We also compute the slices of the concave objects which intersect the plane. This results in an Occlusion Map against which the occludees are tested.

We then advance to the following projection plane. We re-Project the Occlusion Map of the previous plane using the convolution technique described in the previous section. We compute the new slices of concave

²Our use of the term “sweep” is different from that defined in computational geometry or used in chapter 2. We compute no sweep events, and the location of the planes are independent of the scene.

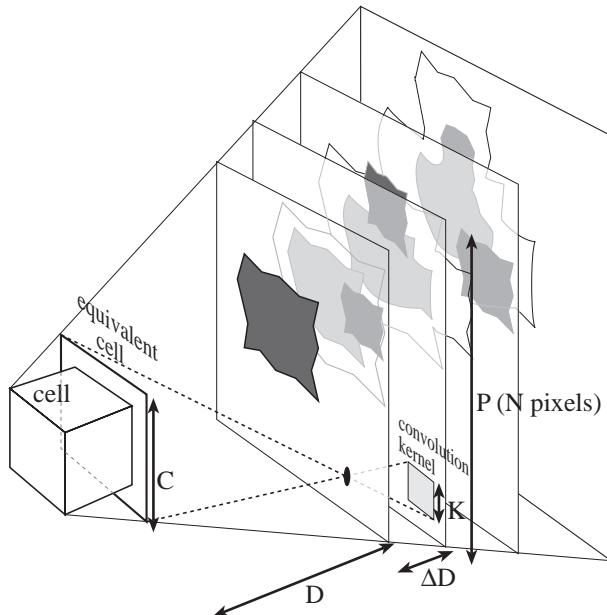


Figure 5.12: Occlusion sweep. The scene is swept by a set of parallel planes leaving the cell. Occluders are not represented to simplify the figure. On each projection plane we show in light grey the re-Projection of the previous plane (the re-Projection before convolution is indicated with a thin line). New Projections are shown in dark grey.

objects, and Project the convex occluders which lie between the two planes. This defines the Occlusion Map of the new projection plane. We again test the occludees and the sweep proceeds.

The distance ΔD between two projection planes is chosen to make optimal use of the discrete convolution. We want the size of the convolution kernel to be an integer number K of pixels. This means that we want the inverse projection of the equivalent cell onto the new projection plane to have size K (Fig. 5.12). Let D be the distance between the initial plane and the equivalent cell, and C the size of the equivalent cell. N is the resolution of the Occlusion Map, and P is the size of the projection plane. Applying Thales theorem gives:

$$\Delta D = \frac{D(K-1)P}{CN}$$

$(K-1)$ is used because a discrete kernel of size K actually “removes” $\frac{K-1}{2}$ pixels around each pixel. In practice we use $K = 5$ pixels. Note that this formula results in planes which are not equidistant. This naturally translates the fact that occlusion varies more quickly near the viewing cell.

The occludees need not be tested against each projection plane. We tests only the closest occludees for each projection plane. Occludees which are farther away are tested less frequently (*e.g.*, every 5 or 3 planes).

This sweep process however remains more costly than our standard Projection method using a single plane, because several re-Projections have to be performed, and because the occludees have to be tested many times. Nevertheless, it permits the treatment of very hard configurations such as the occlusion caused by the aggregation of the leaves in a forest.

5 System

Some aspects of our method have remained vague. In what follows we describe which objects are chosen as occluders and how the projections planes are set. We then present our adaptive cell preprocessing, before treating the case of moving occludees. We finally describe how the visibility information computed during the preprocess is used for interactive rendering.

5.1 Occluder selection

For each viewing cell we choose the set of relevant occluders using a heuristic similar to those presented by Coorg and Teller [CT97b], Zhang *et al.* [ZMH97, Zha98b] and Hudson *et al* [HMC⁺97]. The importance of an occluder is judged based on its approximate solid angle. In practice, it is computed at the center of the viewing cell.

To optimize this selection, we use a preprocess similar to the one proposed in previous approaches [HMC⁺97, ZMH97, Zha98b]. For each occluder, we precompute the region of the scene for which it may be relevant. We store this information in a kd-tree. Then, for each cell, we use the occluders stored at the nodes of the kd-tree that the cell spans.

To improve the efficiency of occlusion tests, we have also implemented an adaptive scheme which selects more occluders in the direction where many occludees are still identified visible. As the occludees are tested, we store together with the Depth Map a *failure map* which indicates which regions of the projection plane need be filled by occluder Projections. Each time the occlusion test of an occludee fails, we update the failure map. For each pixel of the Projection of the occludee, we add the number of polygons of the occludee in the failure map. We then use this information to decrease the solid-angle threshold for occluder selection in this direction.

This scheme requires a two pass occlusion test: During the first pass, the failure map is built, which is used during the second pass to improve the occluder selection. The Depth Map is updated with the Projection of the new occluders, and only occludees which have not been identified hidden are tested in the second pass.

The Depth Map itself could also be used directly to improve occluder selection without performing a first occludee test. Following an idea in [ZMH97, Zha98b], holes in the Depth Map could be detected and used to indicate directions where occluders should be searched for. This solution has not yet been implemented.

5.2 Choice of the projection plane

Until now, we have discussed our method for an arbitrary projection plane. However, the efficiency of the method highly depends on the choice of this plane. We first try to understand which constraints govern the selection of this plane, before proposing the simple heuristic we have implemented. Recall that our projection planes are constrained to the three directions of the viewing cell. We base our discussion on a 2D example for simplicity, the properties we discuss are similar in 3D.

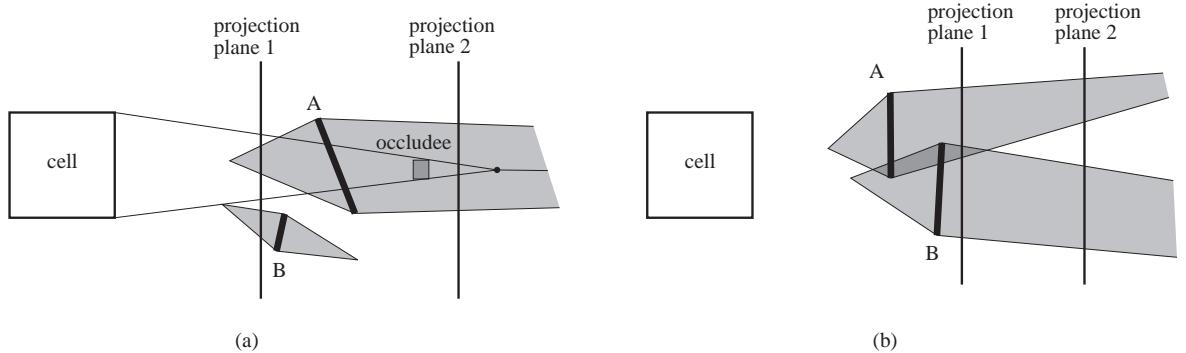


Figure 5.13: Choice of the projection plane. We do not show the supporting and separating lines for clarity. In grey we show the Projection volume, which is the locus of all the Projections of an occluder for all the possible placements of the projection plane. (a) We also show the volume of Projection of an occludee. (b) The intersection of the volumes of Projection of A and B (in dark grey) delimits the projection planes which handle the fusion of A and B.

We have to understand what the set of Projections of an occluder is, depending on the chosen projection plane. We define the *volume of Projection* as the set Projection onto all possible projection planes. Fig. 5.13(a) illustrates this for two occluders A and B. Recall that the Projection of an occluder in 2D is defined by its separating and supporting lines with the cell. The part of the volume of Projection of an occluder behind it is its umbra cone. The volume of Projection can be finite or infinite, just like an umbra volume. Similarly, Fig. 5.13(a) shows the volume of (improved) Projection of an occludee. It is a cone with its apex at the vanishing

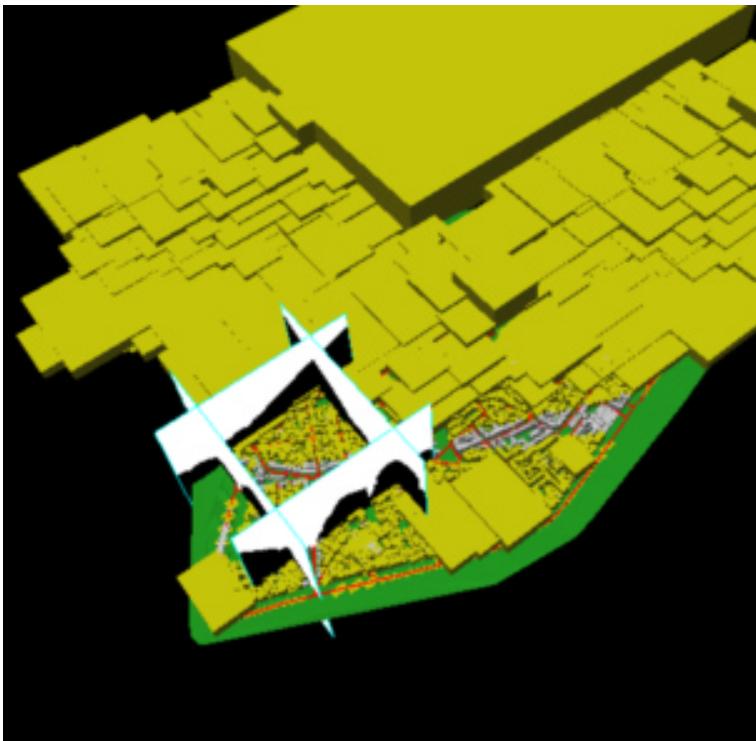


Figure 5.14: An example of projection planes for a cell of the city model. The yellow bounding boxes correspond to the nodes of the hierarchy of occludees which are culled.

point, prolonged by a line. The intersection of this volume of Projection with a plane gives the Projection onto this projection plane.

Consider the influence of the choice of the projection plane on the occlusion test of an occludee by a single occluder A . In Fig. 5.13, projection plane 1 will not detect occlusion, while plane 2 will. This is general: If the projection plane is behind the occluder, the test will always be correct in 2D. The test is less effective if the plane lies in front of the occluder, because of the use of the separating lines as shown in Fig 5.13(a).

We also want to take into account *occluder fusion*. Consider Fig. 5.13(b). A and B will aggregate only in the region defined by the intersection of their volume of Projection. Projection plane 1 will handle occluder fusion, plane 2 will not.

The heuristic we use is very simple, based on the optimization of a simple function. We want to maximize the number of pixels filled in our Depth Map. We place a candidate plane just behind each occluder. We evaluate the size of the Projection on each such plane for each occluder. This method is brute force, but remains very fast due to the low number of occluders considered. It however does not directly tries to maximize the handling of occluder fusion. A heuristic based on the observation of Fig. 5.13(b) should be developed.

Six projection planes are used to cover all directions. Unlike the hemicube [CG85] or light buffer [HG86] methods, our six planes do not define a box. The planes are extended (*e.g.*, by a factor 1.5 as we use in our tests) to avoid problems with severely oblique Projections, as illustrated in Fig. 5.14. This improves the detection of occlusion in gazing directions.

5.3 Adaptive preprocess

We organise the viewing cells into a spatial hierarchical data-structure. Note that it is separate from the hierarchy of bounding boxes used to enclose the occludees. Any hierarchical spatial subdivision, can be used, potentially related to the specific application. Our algorithm starts with an initial subdivision of the space of possible observer viewpoint positions. It can be the bounding box of the entire scene if no *a priori* knowledge is assumed, but any “natural” subdivision can be exploited (the bounding boxes of the streets of a city model for example).

For each viewing cell we perform an occlusion computation. We choose the appropriate occluders and projection planes. We then Project the occluders and build the Hierarchical Depth Map. Finally, the occludees are tested recursively. If a node is identified as hidden or fully visible, the recursion stops. By fully visible, we mean that its Projection intersects no occluder Projection, in which case no child of this node can be identified as hidden. The occludees declared visible are inserted in the Potentially Visible Set (PVS) of the cell.

If we are satisfied with the size of the PVS of a cell, we proceed to the next cell. Otherwise, the cell is subdivided and we recurse on the sub-cells. Nonetheless, occlusion culling is only performed on the remaining visible objects, *i.e.*, those contained in the PVS of the parent.

Performing computation on smaller viewing cells improves occlusion detection because the viewpoints are closer to each other. The views from all these viewpoints are thus more similar, resulting in larger occluder Projections, and smaller occludee Projections.

The termination criterion we have implemented is simply a polygon budget: if the PVS has more than a certain number of polygons, we subdivide the cell (up to a minimum size threshold). A more elaborate criterion would be to compare the PVS to sample views from within the cell.

PVS data can become overwhelmingly large in the memory required. To avoid this we use a *delta-PVS* storage mechanism. We store the entire PVS for a single arbitrary initial cell. Adjacencies are stored with each cell; a neighboring cell simply contains the *difference* with respect to the PVS of the previous cell. We have observed that storage requirements become reasonable as a result (see Section 6.2)

5.4 Dynamic scenes

Dynamic objects with static occluders can be treated using our extended projection approach. However, dynamic occluders cannot be handled.

A hierarchy of bounding boxes is constructed in the regions of space for which dynamic objects can move [SG96]. During preprocess, the bounding boxes for dynamic motion are tested for occlusion against the extended occlusion maps of each cell. During the viewing process, each dynamic object is displayed if its containing bounding box is visible.

In the urban driving simulator example, the roads are the dynamic object bounding box hierarchy. During the interactive walkthrough, a car is displayed only if the street in which it lies is visible from the current street of the observer.

5.5 On-line rendering

Once the preprocess has been completed and the PVS data stored to disk, a viewer process can use the data to rapidly display very complex models. We use a standard scene-graph structure [RH94] to represent the scene. A simple flag for each node determines whether it is active for display.

Each time the observer enters a new cell, the visibility status of the nodes of the scene graph are updated. This is very efficient thanks to our delta-PVS encoding. Nodes which were previously hidden are restored as visible, while newly hidden ones are marked as inactive.

The viewer process adds very low CPU overhead, since it only performs simple scene graph node updates. An extension handling disk or network prefetching will be discussed in section 7.3.

6 Implementation and results

6.1 Implementation

We have implemented two independent systems for the preprocessor and the viewer. The preprocessor uses graphics hardware acceleration wherever possible, notably for the Projection and convolution operations. The Depth Maps are read from graphics memory, and the occludee test is performed in software. This could be improved with modern graphics cards which permit efficient depth-test query [Sgi99]. The PVS's computed are stored to disk and made available to the interactive viewer. A typical projection plane configuration is shown in Figure 5.14.

Our current implementation of the viewer is based on SGI Performer [RH94]. Performer implements a powerful scene-graph and view-frustum culling, providing a fair basis for comparison.

6.2 Results

Extended projection onto single projection plane

The test scenes we have used for the single plane method are the following: (a) the model of a city district which contains a total number of about 150,000 polygons; (b) the city replicated 4 times, with 2,000 cars (of 1,000 polygons each) moving around the streets (2.6 million polygons total).

We use the convex occluder Projection using the stencil buffer and the improved Projection of occludees. All Projections were performed at 256x256 resolution. We did not notice artifacts in the tests we performed, but a comparison with higher resolution Depth Maps should be performed.

The preprocessing for a single city was about 1 hour, while the replicated 4 city model required around 4 hours on an Onyx 2 Infinite Reality using a single 195Mhz R10000 processor. 3479 final cells were used for the city replicated 4 times. Our preprocess thus spends 4.1 seconds per cell. In the latter scene, the storage for the PVS was around 25 Mbytes for a model size of 60 Mbytes (not including the cars). We removed around 95% of the geometry on average.

We have performed tests of the viewer with two hardware configurations. The first is a mid-range SGI O^2 R5000 graphics workstation and the second an Onyx 2 Infinite Reality 2xR10000 high-end graphics workstation. A speed-up of around 5-6 times was observed. This may seem small when compared to 95% culling, but our method is compared to Performer which performs a powerful view-frustum culling [GBW90] which eliminates a large part of the geometry (about 80%). Fig. 5.15 illustrates the results of our method.

As an informal comparison, we have implemented the algorithm of Cohen-Or *et al.* [COFH98, COZ98] (see Figure 5.16). For the city model, their algorithm declares four times more visible objects on average and the nominal computation time in our implementation is 150 times higher than for extended projection. A more optimized version may decrease this gap, but our 4.1seconds per cell seems hard to beat using ray-casting on such large models.

Occlusion sweep

We have ran some preliminary tests for the occlusion sweep. A model of a forest containing around 1200 trees with 1000 leaves each was used. The Projection of the leaves close to the projection plane were computed using the convex occluder Projection using the stencil buffer. The size of the convolution kernel was fixed to 5 pixels, and we used 15 planes for the sweep. The occlusion sweep took around 12 seconds per cell and about 75% of the trees were culled. Fig. 5.17 shows our sweeping process. Observe how the leaves aggregate on the Occlusion Map.

7 Discussion

7.1 Summary

We have presented *extended projection* operators which permit conservative occlusion tests with respect to volumetric viewing cells. The extended projection of an occluder is the intersection of its views with respect to any point within the cell, while for the occludee, it is the union of the views. We have also defined an *extended depth* permitting the definition of the *extended depth map* which is a generalization of the z-buffer for volumetric viewing cells.

Our operators yield conservative occlusion tests and can handle *occluder fusion*, that is, the occlusion caused by the cumulative effect of multiple occluders. We have proposed an efficient implementation of both operators, as well as an improvement in the case of convex or planar blockers.

We have defined a reprojection operator which allows us to reproject the Projections computed on a given projection plane onto another one, allowing us to define an *occlusion sweep*. This consists in sweeping the scene with a set of parallel planes leaving a viewing cell. The occlusion caused by small objects are aggregated as the planes are swept.

We have presented an efficient implementation which takes advantage of hardware acceleration wherever possible. Our preprocess adaptively subdivides the viewing space into cells for which *potentially visible sets* are computed using our extended projection method. The results we obtain show a significant speed-up of 5 to 6 times compared to a high-end interactive rendering library on a 2.6 million polygon model. Some initial

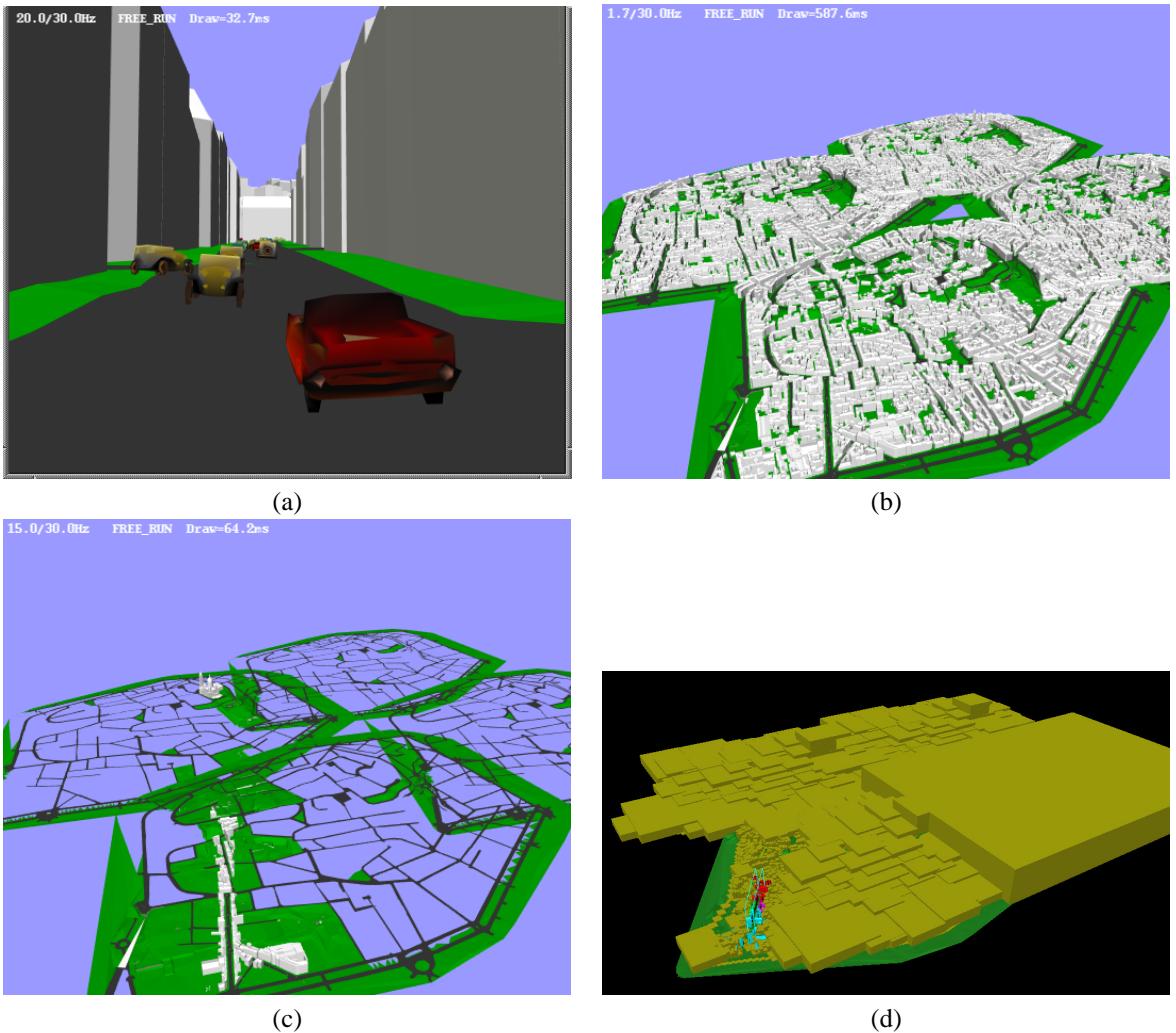


Figure 5.15: An illustration of the results of our algorithm (a) The scene viewed from the observer position. (b) The scene from a bird’s-eye view with no culling; the scene contains 600,000 building polygons and 2,000 moving cars containing 1,000 polygons each. (c) The same view using the result of our visibility culling algorithm. (d) Visualization of the occlusion culling approach, where yellow boxes represent the elements of the scene-graph hierarchy which have been occluded.

results have also been presented showing that our occlusion sweep makes it possible to compute occlusion caused by leaves in a forest, with respect to a volumetric viewing cell .

7.2 Discussion

We first have to note that the occlusions that our method identifies are a subset of the occlusions detected by a point-based online method such as the hierarchical z-buffer [GKM93] or the hierarchical occlusion maps [ZMH97, Zha98b]. Advantages of these methods also include their ability to treat dynamic occluders (since everything is recomputed for each frame) and the absence of preprocess or PVS storage.

However, our method incurs no cost at display time, while in the massive rendering framework implemented at UNC [ACW⁺99] two processors are sometimes used just to perform occlusion culling. Moreover, for some applications (games, network-based virtual tourism, etc.), the preprocess is not really a problem since it is performed once by the designer of the application. Our PVS data then permits an efficient predictive approach to database pre-fetching which is crucial when displaying scenes over a network or which cannot fit into main memory.

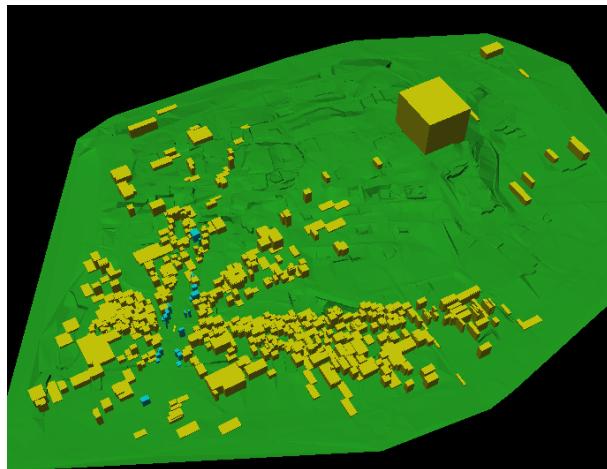


Figure 5.16: Comparative visualization of the occlusion culling of Cohen-Or et al. [COFHZ98, COZ98] and extended projection. For a cell at the lower left, we show all the buildings which their algorithm finds as visible which are culled by our extended projection approach.

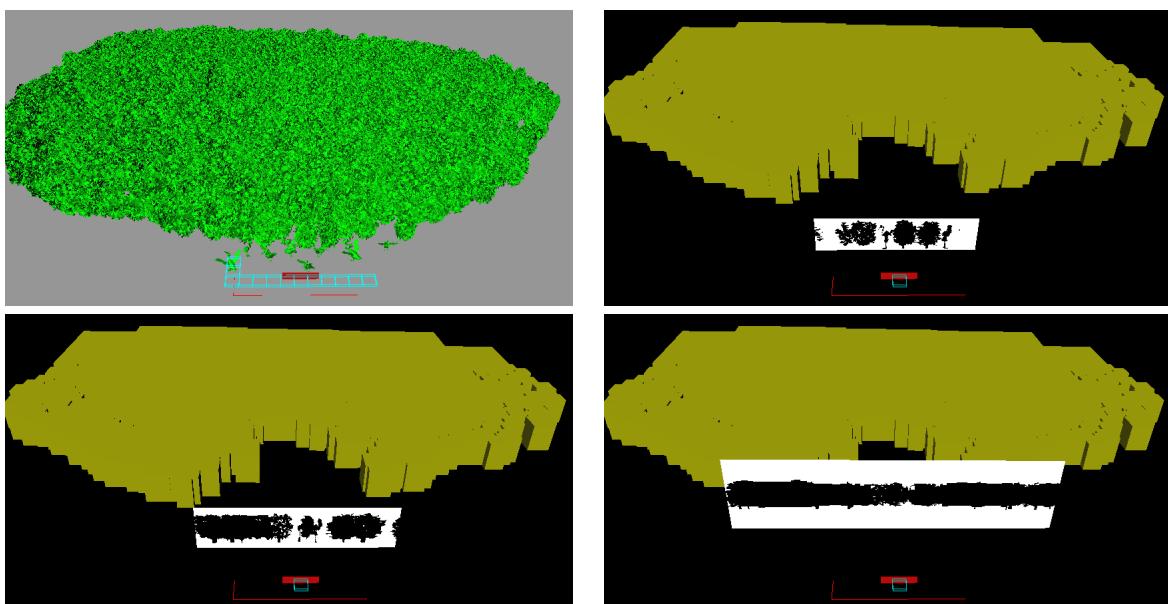


Figure 5.17: The sweeping process: (a) the forest model, (b)-(d) three positions for the sweep projection planes. The yellow bounding boxes are the culled occludees.

As opposed to exact visibility (as studied in the three previous chapter), extended projection operators may fail to identify some occlusions caused by the cumulative effect of multiple blockers. As we have seen in section 5.2, the handling of occluder fusion depends on the placement of the projection plane.

Our Projection reduces the 4D problem of occlusion to a 2D representation on a plane. The visibility information with respect to all rays going through one point of the plane and all the viewpoints inside the viewing cell is conservatively approximated by a single value (binary in the case of the Occlusion Map, real in the case of a Depth Map). This consists in projecting the visibility information inside the tangency volume of the viewing cell onto a 2D manifold.

Another important issue is the approximation induced by the rasterization phase. We believe that our method is less sensitive to artifact induced by low-resolution Depth Maps than the hierarchical z-buffer for which a low resolution depth map would result in artifacts at the silhouette of objects. Indeed, the depth map

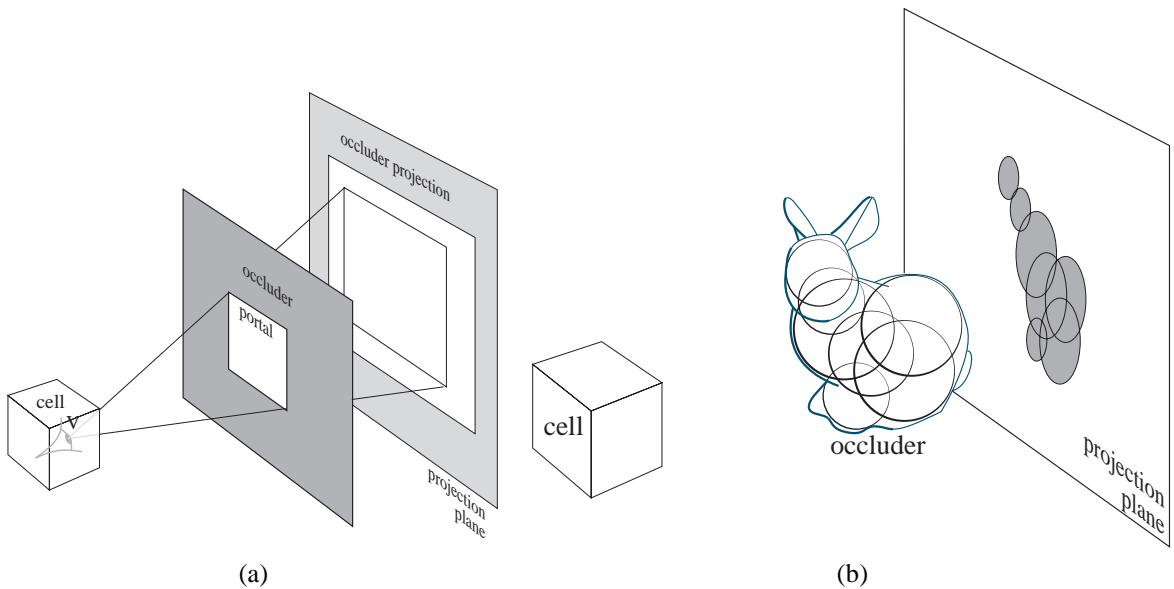


Figure 5.18: (a) The extended projection of a wall with a portal is the intersection of its views. A conservative estimate is the complement of the union of the views of the portal. (b) Concave occluder Projection using a union of convex shapes (spheres in our example).

used in the hierarchical z-buffer directly corresponds to the occlusions from the viewpoint, while in our case it is used together with the corresponding conservative Projections. Our estimate of occludee Projection is completely conservative, and slightly overestimates the exact occludee Projection (because of integer rounding). Moreover, occluder Projection is an underestimate of the view from any point inside the cell, reducing the probability to identify false occlusion because of rasterization error. The resolution of our Depth Maps is however rather low (256×256), and further analysis should be carried out, as well as a more conservative estimate.

There are situations in which even a perfect occlusion culling method (*i.e.*, an exact hidden-part removal) cannot cull enough geometry to achieve real-time rendering. For example, if the observer is at the top of a hill or on the roof of a building, the entire city may be visible. Other display acceleration techniques should thus be used together with our occlusion culling, as will be discussed in the next section.

7.3 Future work

Extended projection of concave occluders

The slicing process we have proposed to handle concave occluders is not completely satisfying. It is restricted to manifolds with closed intersection and which actually have an intersection with the projection plane. We propose here some techniques to handle certain classes of occluders.

The specificities of architectural environments have been exploited for visibility computations (*e.g.* [ARB90, Tel92b, TS91]). Different rooms are visible only through *portals* (doors, windows). Convex portals are in fact the complement of convex occluders. Their Projection can be computed by considering the complement of the union of the views of the portal (see Fig. 5.18(a)).

Some specific convex Projections can also be implemented, which will prove useful in the next paragraph for concave Projection. Consider the case of spheres. If a bounding sphere is placed around the viewing cell, the Projection of a sphere is an ellipse which can be efficiently computed,

We propose the use of unions of convex shapes (for example unions of spheres, *e.g.*, [RF95]) to approximate concave occluders (see Fig. 5.18(b)). The overlap of these convex shapes is crucial, since occluder fusion should then permit the computation of an efficient estimate of the Projection of the concave occluder. This method would alleviate the flaw of the Projection of each individual triangle of a concave mesh, where gaps appear between the Projections of connected triangles.

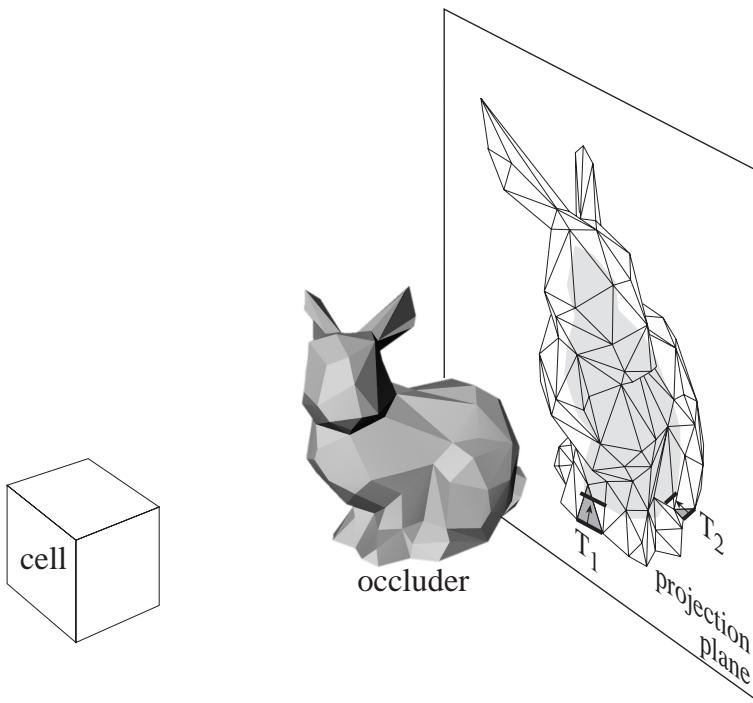


Figure 5.19: Concave occluder mesh Projection. The mesh is projected from the center of the cell, then shrunk. The silhouette edges (triangle T_1) are translated towards the center of the mesh. If a triangle is too small (T_2), the edge translation must be propagated to the neighbouring triangles.

Finally, we propose another way to cope with the connectivity problem of the Projection of concave polyhedral meshes. The connected mesh could be projected from the centerpoint of the viewing cell, then “shrunk” to compute the Projection. Our idea is illustrated in Fig. 5.19. The silhouette edges are translated towards the center of the object. Computing the translation vector is in a way similar to the computation of the convolution kernel for reprojection, the technical details are beyond the scope of this future work section. *A priori*, triangles in the center of the object need not be shrunk. However, if silhouette triangles are too small (for example triangle T_2), the translation of their edges should be propagated to the neighbouring triangles.

Real-time rendering

Our method dramatically decreases rendering time, but it does not guarantee a given framerate. We now propose its use in existing real-time rendering frameworks. They are all based on levels of details (LOD) [Cla76]: simplified representations are used for distant objects.

The *Performer* library [RH94] uses a simple framerate regulation scheme. A global *stress* value is used to scale a simple distance criterion for LOD switching. A target framerate is set, and if the display of the scene takes more than a given value, say 95%, of the time devoted to one frame, the stress value is increased, coarser LODs are used, and the framerate is regulated. However, if the number of displayed primitives varies too quickly, a couple of frames will be necessary to adapt the stress value.

Our preprocess can be used to predict these sudden increases, since the number of primitives potentially visible from the current cell and its neighbours had been precomputed. We can predict the variation and adapt more smoothly.

Funkhouser *et al.* [FS93] achieve real-time rendering by solving a knapsack problem for each frame: a budget of polygons has to be optimally used, by choosing the right level of detail for each objects. Maciel and Shirley [MS95] have extended this approach to hierarchies of level of details. Funkhouser *et al.* [FS93] have shown that a visibility preprocess can focus the optimization on objects which are actually visible.

For better LOD selection, we propose the use of a semi-quantitative visibility information. By counting the number of occluded pixels in the extended Projection of an occludee, we can estimate a *percentage of visibility*

of the occludee from the viewing cell. This can be used as a weight during the LOD optimization, using coarser versions for highly occluded objects. This scheme can also be used for stress-based regulation.

Consider however the case of the same occludee half hidden by a house, or half hidden by a bush. In the first case, a large connected part of the object is visible, it thus requires a high quality display. In the latter, the leaves of the bush cause the occlusion of small regions spread all over the occludee. This induces high frequencies which lower the sensitivity of the human eye to object simplification [FPSG97, PFFG98]. The frequent distribution of the occluded pixels in the Projections must be taken into account, and not only their mean value.

Disk and network pre-fetching

The main advantage of our preprocess is that it can be used for scenes which are too big to fit into main memory, or to be completely loaded from the network. The techniques developed by Funkhouser *et al.* [Fun96c] can easily be adapted. A separate process is in charge of the database management. Using the PVS of the neighbouring viewing cells, the priority of the objects which are not yet loaded is evaluated. The PVS information for all cells itself should not reside in main memory and should be loaded on-demand because of its memory cost. Similarly, a priority order is computed to delete invisible objects from memory. The prediction offered by our method cannot be achieved by previous online occlusion culling methods.

Online occlusion culling

The approach we have presented can be adapted for online or on-demand visibility computation. Online image-based occlusion culling [GKM93, ZMH97, Zha98b] could be extended to include some prediction. Convolution (or erosion [SD95]) can be used to approximate occluder Projection with respect to the neighbourhood of the current viewpoint. Similarly, occludee Projection can be approximated by simply growing the screen bounding rectangle of their projection from the current viewpoint.

This would permit the online computation of visibility information which is valid for more than one frame. It would thus be more suited to stress management or database pre-fetching.

Our method could also be used in an on-demand fashion. The computation time for one cell is around 4 seconds for the city model. If the observer moves slowly, this permits the computation of the visibility information for the neighbouring cells on-demand, especially in a parallel-processing context.

Other issues

The resolution of the Depth or Occlusion Map remains an important issue. The edge-translation method of Wonka *et al.* [WS99] should be implemented to obtain truly conservative computations.

Our occlusion sweep could be adapted for soft shadow computation. However, performing two convolutions is not equivalent to a convolution with a doubled kernel. The size of the zones of penumbra and umbra are preserved, but the values in the penumbra region are modified. A remapping should thus be performed after each convolution (Fig. 5.20). However, remapping is not sufficient in the case of more than two successive convolutions.

Our preprocess could also be used in the context of lighting simulation, following the work of Teller *et al.* [TH93, TFFH94, Fun96b].

Visibility information is also very useful for levels of details in animation [CH97, CF97] which are still in their beginning. The precision of physically based simulation or generative animation can be coarsened in occluded regions.

Finally, the issue of PVS compression remains critical. Possible approaches include entropy coding, hierarchical encoding or conservative vector quantization.

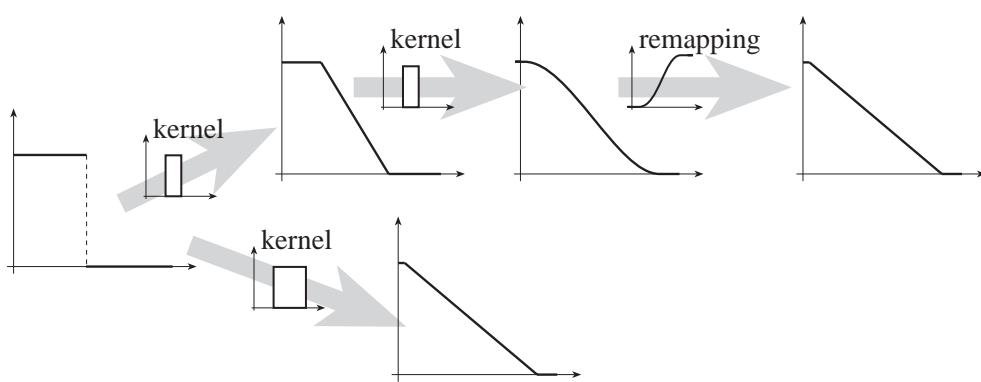


Figure 5.20: Successive convolutions and remapping. Note that convolving twice with the same kernel is not equivalent to the convolution by a double size kernel.

Part II

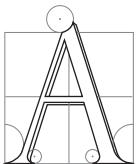
A Multidisciplinary Survey of Visibility

CHAPTER 6

Introduction

Il déduit 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*



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.

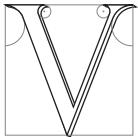
In appendix E we also give a short list of resources related to visibility which are available on the web. An index of the important terms used in this survey can be found at the end of this thesis. Finally, the references are annotated with the pages at which they are cited.

CHAPTER 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 objects¹. 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.

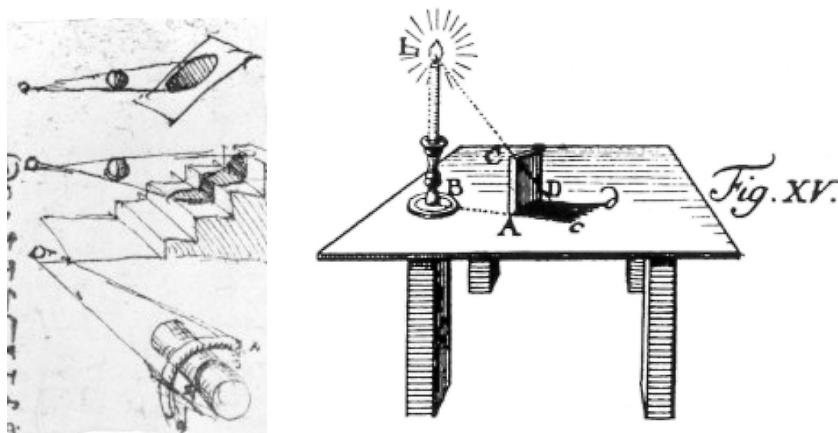


Figure 7.1: (a) Study of shadows by Leonardo da Vinci (Manuscript *Codex Urbinas*). (a) Shadow construction by Johann Heinrich Lambert (*Freye Perspective*).

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

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]

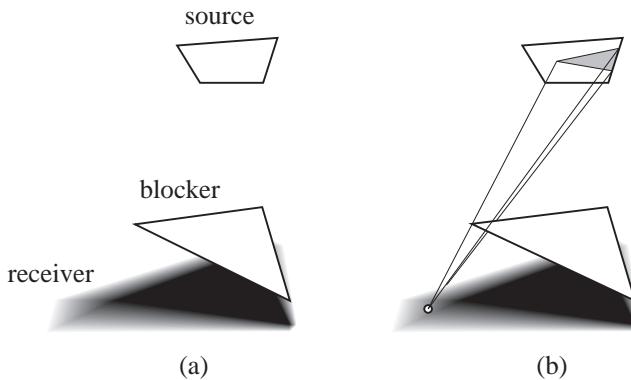


Figure 7.2: (a) Example of a soft shadow. Notice that the size of the zone of penumbra depends on the mutual distances (the penumbra is wider on the left). (b) Part of the source seen from a point in penumbra.

databases are often composed of millions of primitives, and also in driving/flight simulators, and in walk-throughs 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 hardware².

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 Kuttruff [Kut91] or [Dal96, FCE⁺98]. See section 4.3 of chapter 10. Sound however differs from light because

²Occlusion-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].

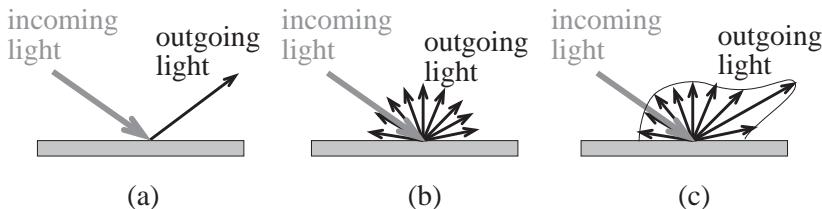


Figure 7.3: Light reflection for a given incidence angle. (a) Perfect mirror reflection. (b) Diffuse reflection. (c) General bidirectional reflectance distribution function (BRDF).

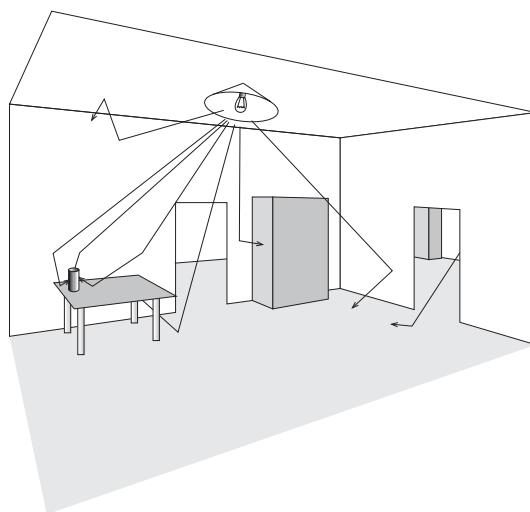


Figure 7.4: Global illumination. We show some paths of light: light emanating from light sources bounces on the surfaces of the scene (We show only one outgoing ray at each bounce, but light is generally reflected in all direction as modeled by a BRDF).

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

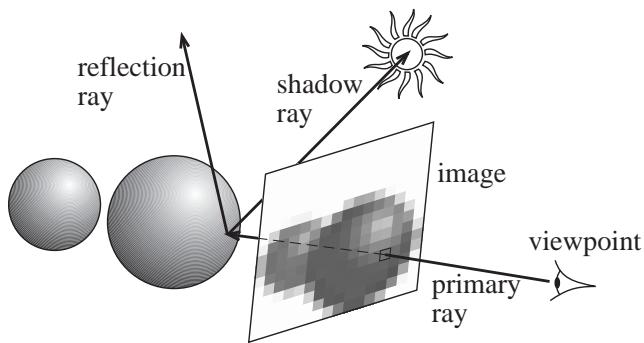


Figure 7.5: Principle of recursive ray-tracing. Primary rays are sent from the viewpoint to detect the visible object. Shadow rays are sent to the source to detect occlusion (shadow). Reflection rays can be sent in the mirror direction.

first object hit by a ray, or in the case of rays cast for shadows, to determine if the ray intersects an object. Many 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].

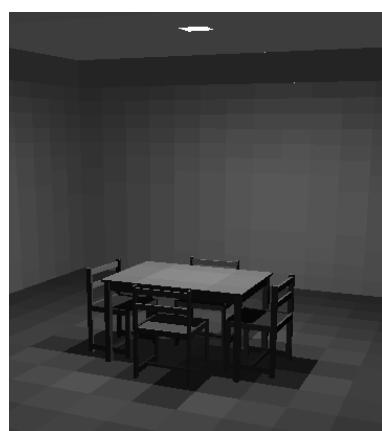


Figure 7.6: Radiosity methods simulate diffuse interreflexions. Note how the subdivision of the geometry is apparent. Smoothing is usually used to alleviate most of these artifacts.

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.

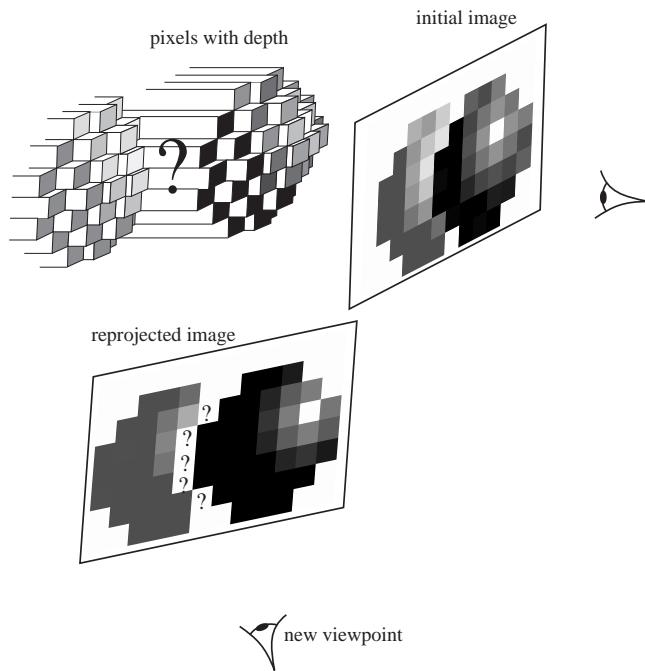


Figure 7.7: View warping. The pixels from the initial image are reprojected using the depth information. However, some gaps due to indeterminate visibility may appear (represented as “?” in the reprojected image)

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].

³Recent 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].

Others 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 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 other 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]

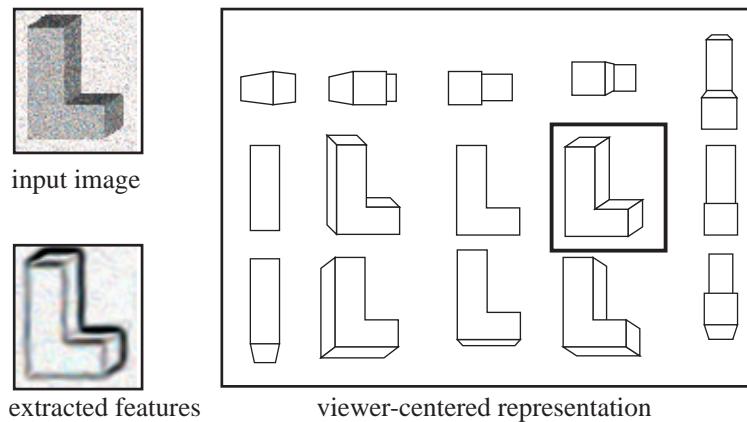


Figure 7.8: Model-based object recognition. Features are extracted from the input image and matched against the viewer-centered representation of an L-shaped object.

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.

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].

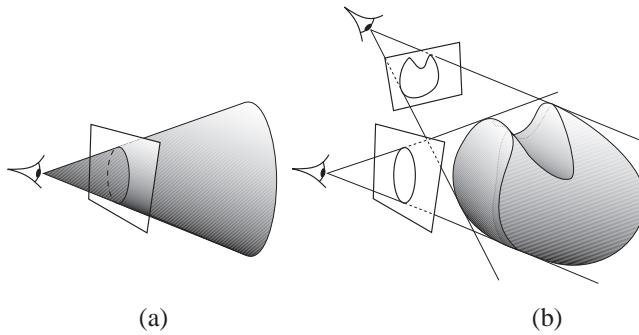


Figure 7.9: Object reconstruction by contour intersection. The contour in each view defines a general cone in which the object is constrained. A model of the object is built using the intersection of the cones. (a) Cone resulting from one image. (b) Intersection of cones from two images.

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.

⁴However 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].

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.

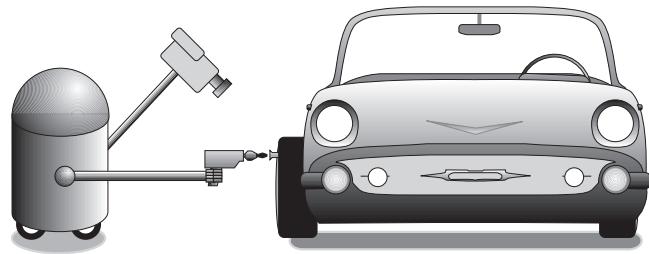


Figure 7.10: The screwdriver must be placed very precisely in front of the screw. The task is thus controlled by a camera.

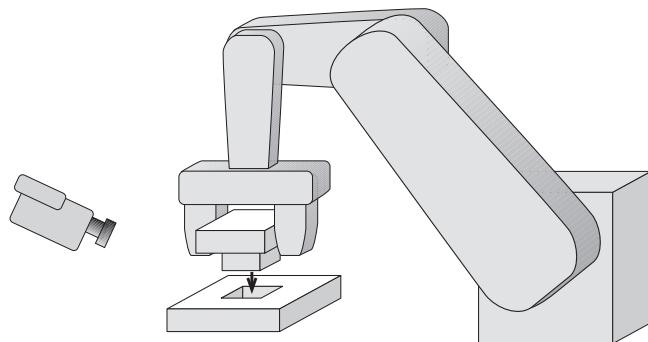


Figure 7.11: The insertion of this peg into the hole has to be performed very precisely, under the control of a sensor which has to be carefully placed.

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.

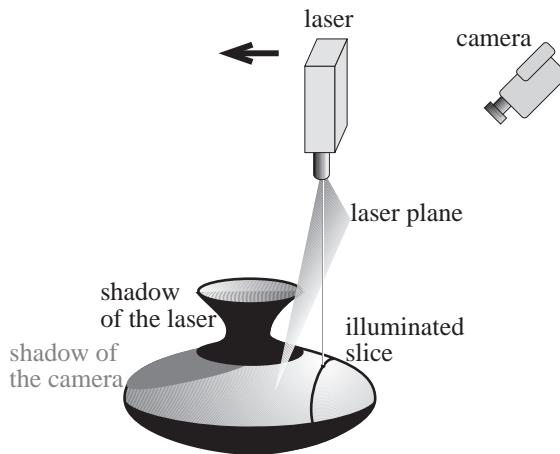


Figure 7.12: Object acquisition using a laser plane. The laser emits a plane, and the intersection between this plane and the object is acquired by a camera. The geometry of the slice can then be easily deduced. The laser and camera translate to acquire the whole object. Occlusion with respect to the 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.

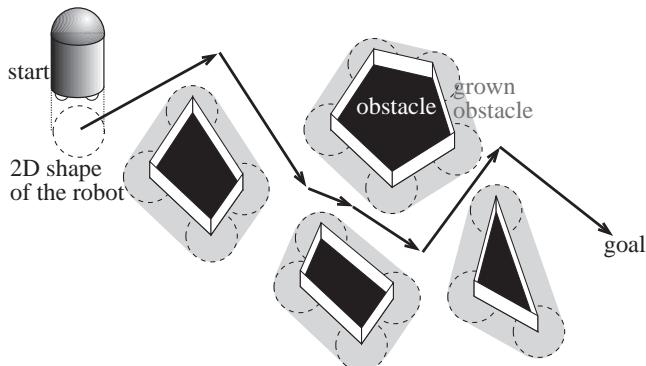


Figure 7.13: Motion planning on a floorplan. The obstacles are grown using the Minkovski sum with the shape of the robot. The motion planning of the robot in the non-grown scene resumes to that of its centerpoint in the grown scene.

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.

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.

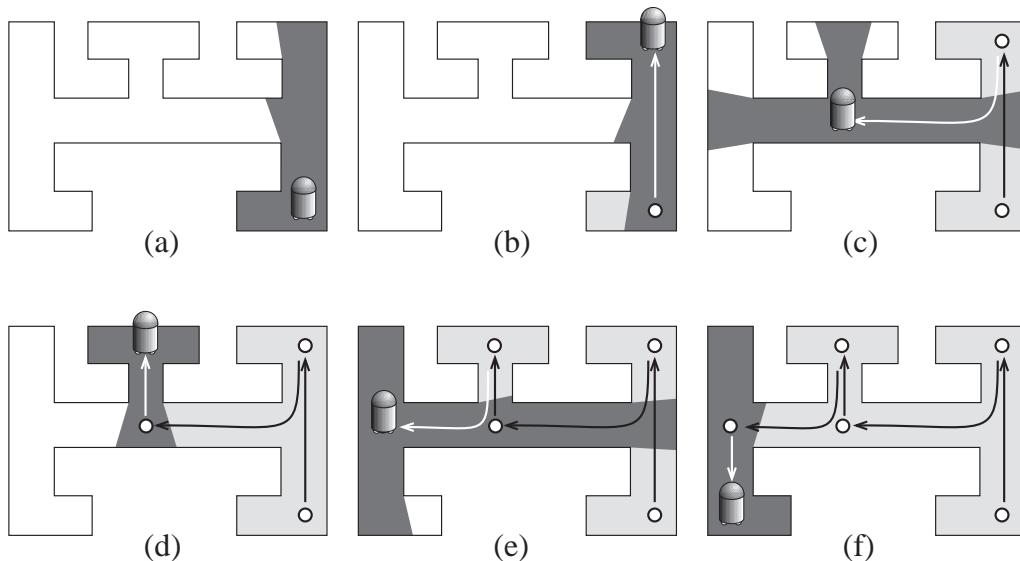


Figure 7.14: The robot has to search for an unknown intruder. The part of the scene visible from the robot is in dark grey, while the “cleaned” zone is in light grey. At no moment can an intruder go from the unknown region to the cleaned region without being seen by the robot.

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.

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.

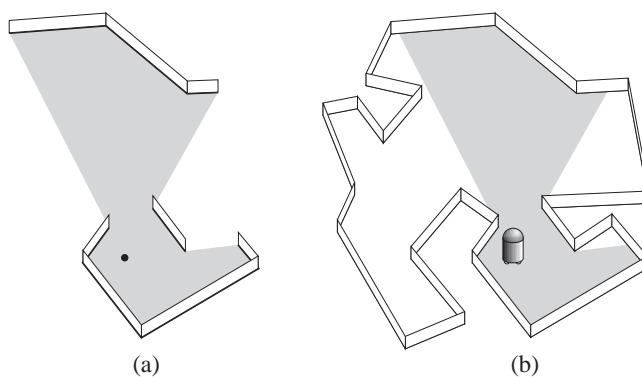


Figure 7.15: 2D Robot localisation. (a) View from the robot. (b) Deduced location of the robot.

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.

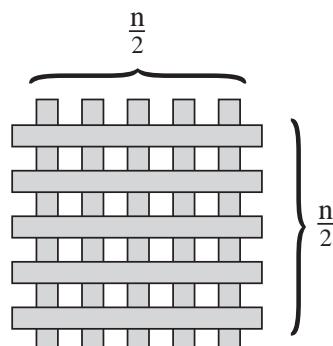


Figure 7.16: Scene composed of n rectangles which exhibits a view with complexity $O(n^2)$: the planar map describing the view has $O(n^2)$ segments because of the $O(n^2)$ visual intersections.

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.

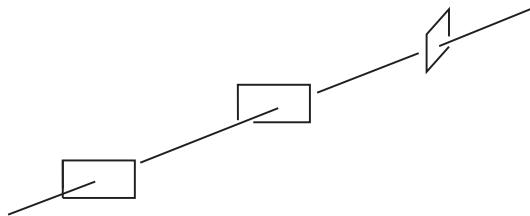


Figure 7.17: Line stabbing a set of convex polygons in 3D space

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 decide if a line of sight is possible through a sequence of doors⁶.

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].

⁶Stabbing 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.

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.

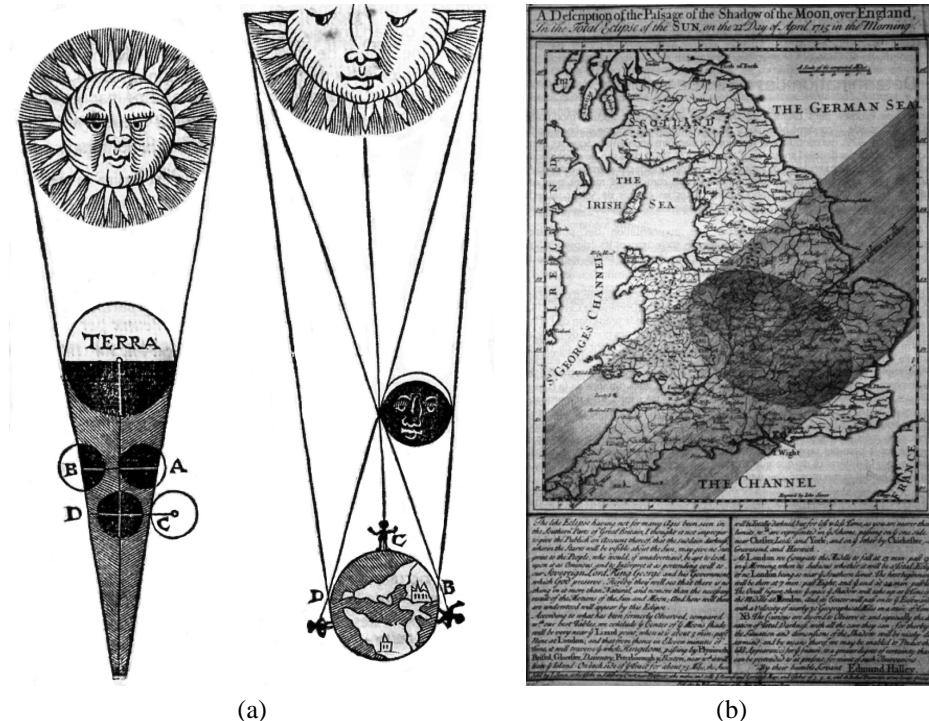


Figure 7.18: Eclipses. (a) Lunar and Solar eclipse by Purbach. (b) Prediction of the 1715 eclipse by Halley.



Figure 7.19: 1994 solar eclipse and 1993 lunar eclipse. Photograph Copyright 1998 by Fred Espenak (NASA/Goddard Space Flight Center).

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.

see e.g.

<http://www.astro.indiana.edu/personnel/rberring/sundial.html>
<http://www.sundials.co.uk/2sundial.htm>

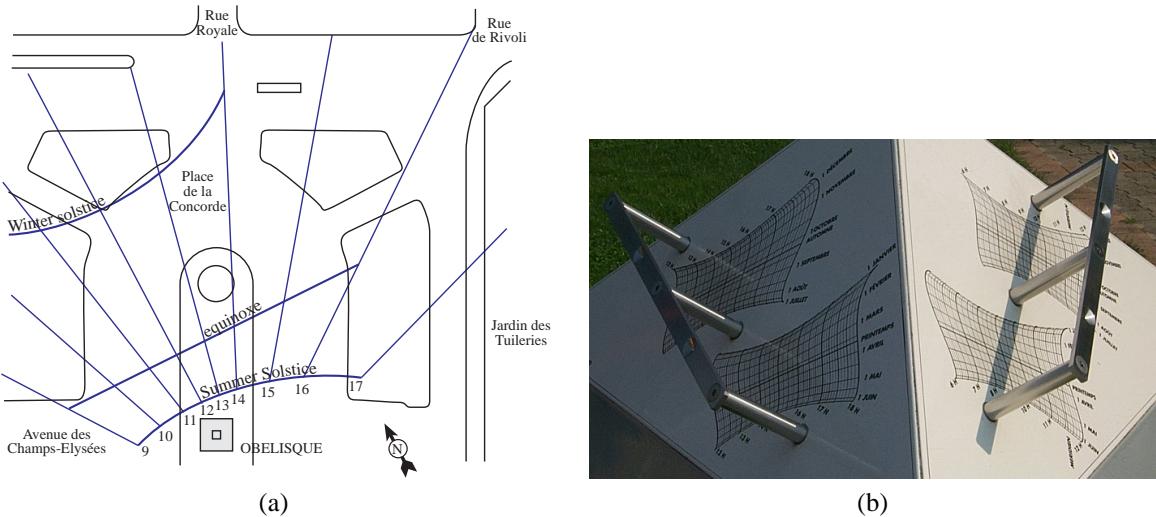


Figure 7.20: (a) Project of a sundial on the Place de la Concorde in Paris. (b) Complete sundial with analemmas in front of the CICG in Grenoble.

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 issues⁷. 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.

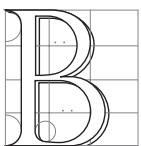
⁷Some author also define occlusion by other objects as global visibility effects as opposed to backface culling and silhouette computation.

CHAPTER 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*



BEFORE 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 follows, 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.

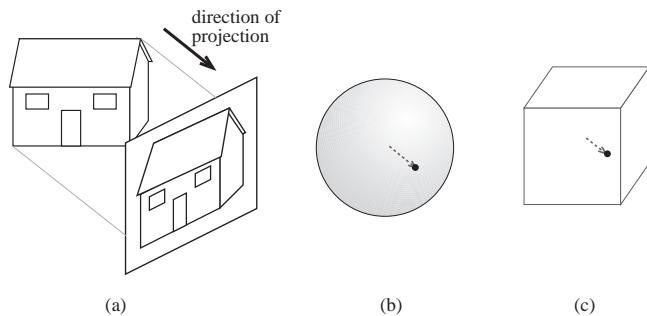


Figure 8.1: (a) Orthographic view. (b) Corresponding point on the viewing sphere and (c) on the viewing cube.

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.

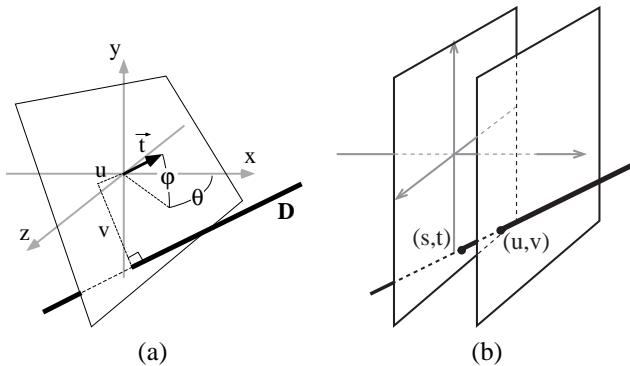


Figure 8.2: Line parameterisation. (a) Using two angles and the intersection on an orthogonal plane. (b) Using the intersection with two planes.

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*.

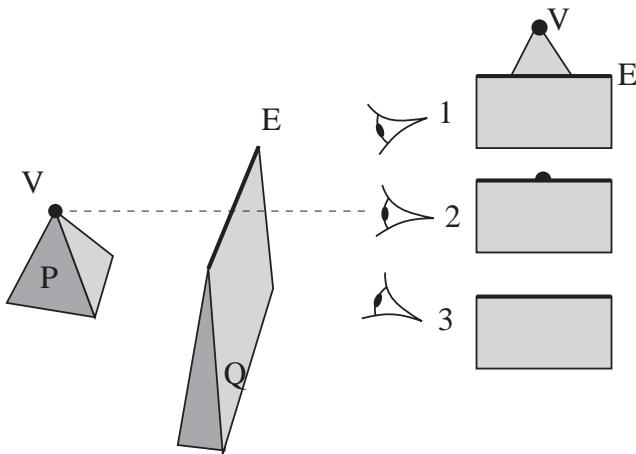


Figure 8.3: EV visual event. The views from the three eyepoints are represented on the right. As the eyepoint moves downwards, vertex V becomes hidden. Viewpoint 2 is the limit eyepoint, it lies on a *visual event*.

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.

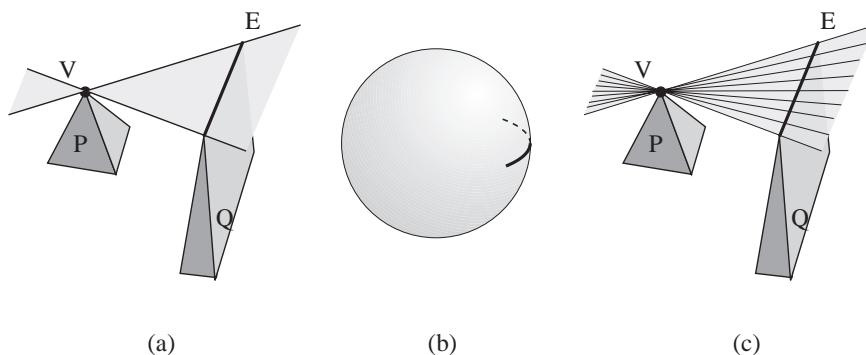


Figure 8.4: Locus of an EV visual event. (a) In object space or perspective viewpoint space it is a wedge. (b) In orthographic viewpoint space it is an arc of a great circle. (c) In line space it is the 1D set of lines going through V and E

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 placed at P .

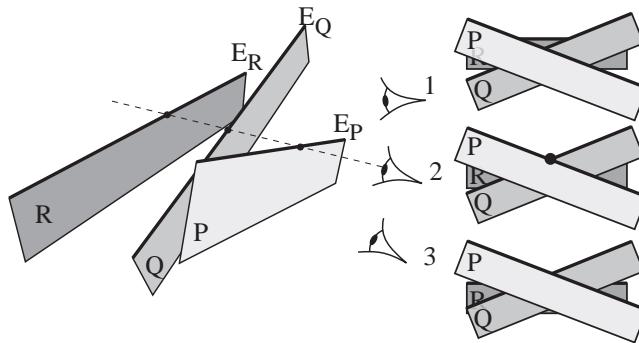


Figure 8.5: A EEE visual event. The views from the three eyepoints are represented on the right. As the eyepoint moves downwards, polygon R becomes hidden by the conjunction of polygon P and Q . From the limit viewpoint 2, the three edges have a visual intersection.

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.

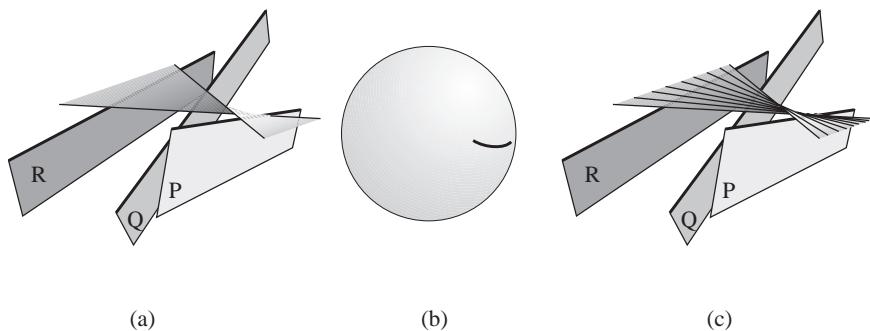


Figure 8.6: Locus of a EEE visual event. (a) In object-space or perspective viewpoint space it is a ruled quadrics. (b) In orthographic viewpoint space it is a quadric on the viewing sphere. (c) In line space it is the set of lines stabbing the three edges.

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 comic 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.

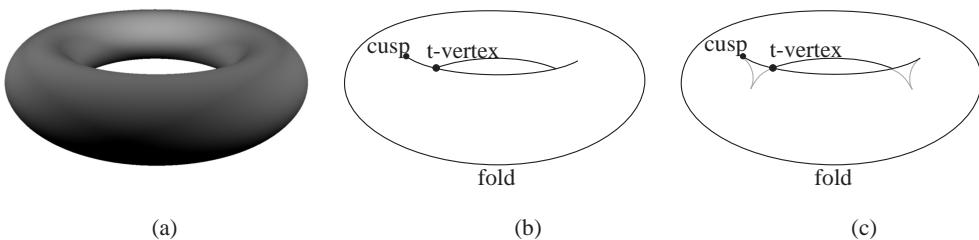


Figure 8.7: View of a torus. (a) Shaded view. (b) Line drawing with singularities indicated (b) Opaque and transparent contour.

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 direction¹.

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 inflection 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*.

¹What 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].

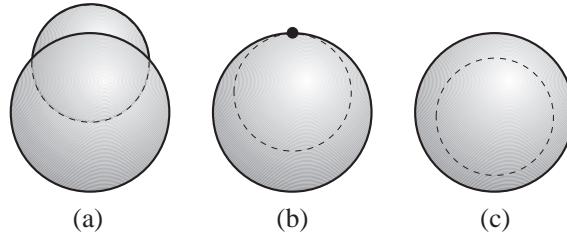


Figure 8.8: Tangent crossing singularity. As the viewpoint moves downwards, the back sphere becomes hidden by the frontmost one. At viewpoint (b) a singularity occurs (highlighted with a point): the two spheres are visually tangent.

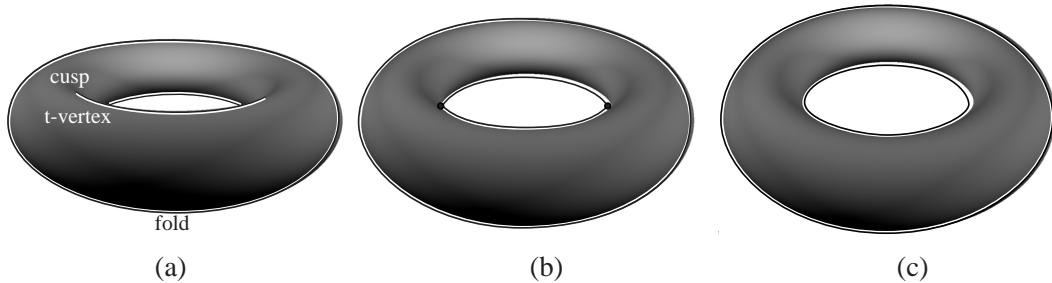


Figure 8.9: Disappearance of a cusp at a swallowtail singularity at viewpoint (b). (in fact two swallowtails occur because of the symmetry of the torus)

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 spheres 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 objects². 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?).

²Williams [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.

³Those 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.

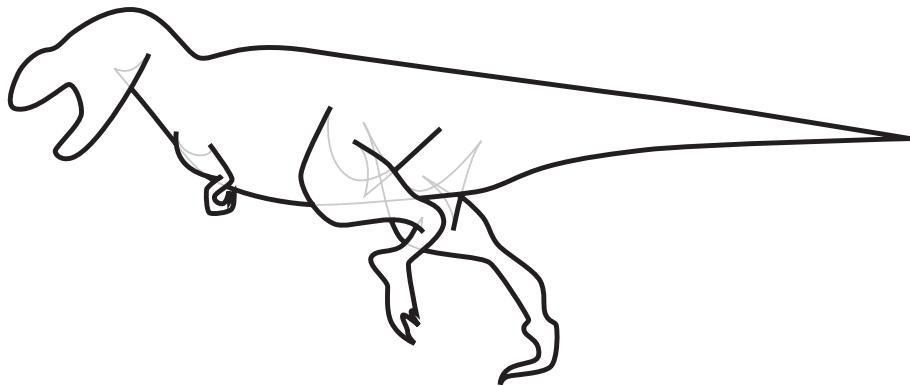


Figure 8.10: Opaque (bold lines) and semi-transparent (grey) singularities. After [Wil96].

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.

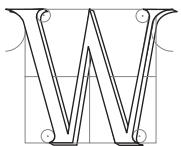
Visual events are simple in 2D: bitangent lines or tangent to inflection points. In 3D their locus are surfaces which are rarely planar (*EEE* or visual events for curved objects).

All these arguments make the sentence “the generalization to 3D is straightforward” a doubtful statement in any visibility paper.

The classics of hidden part removal

Il convient encore de noter que c'est parce que quelque chose des objets extérieurs pénètre 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).

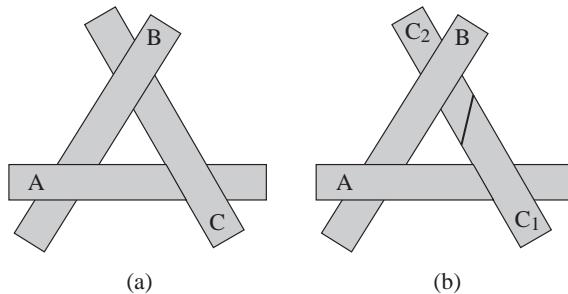


Figure 9.1: (a) Classic example of a cycle in depth order. (b) Newell, Newell and Sancha split one of the polygons to break the cycle.

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.

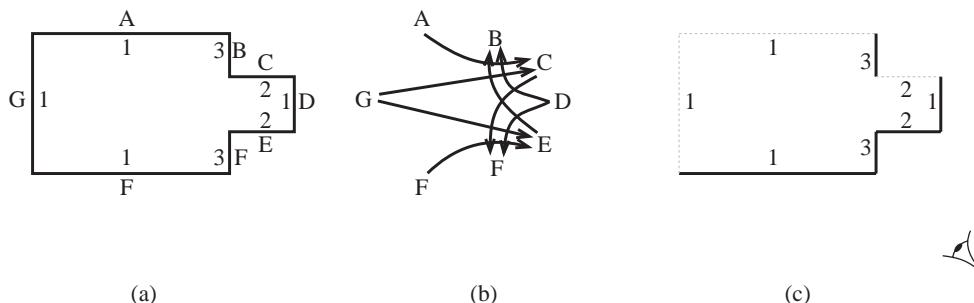


Figure 9.2: *A priori* depth order. (a) Lower number indicate higher priorities. (b) Graph of possible occlusions from any viewpoint. An arrow means that a face can occlude another one from a viewpoint. (c) Example of a view. Backfacing polygons are eliminated and other faces are drawn in the *a priori* 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 *et 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.

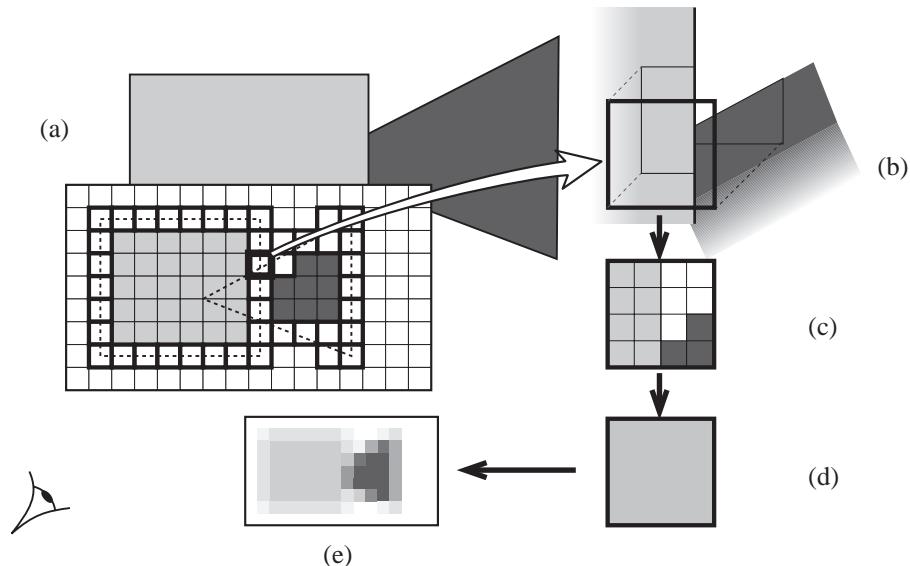


Figure 9.3: A buffer. (a) The objects are scan-converted. The projection of the objects is dashed and non-simple pixels are represented in bold. (b) Close-up of a non-simple pixel with the depth sorted fragments (*i.e.*, the polygons clipped to the pixel boundary). (c) The pixel is subsampled. (d) The 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 Aristotle 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.

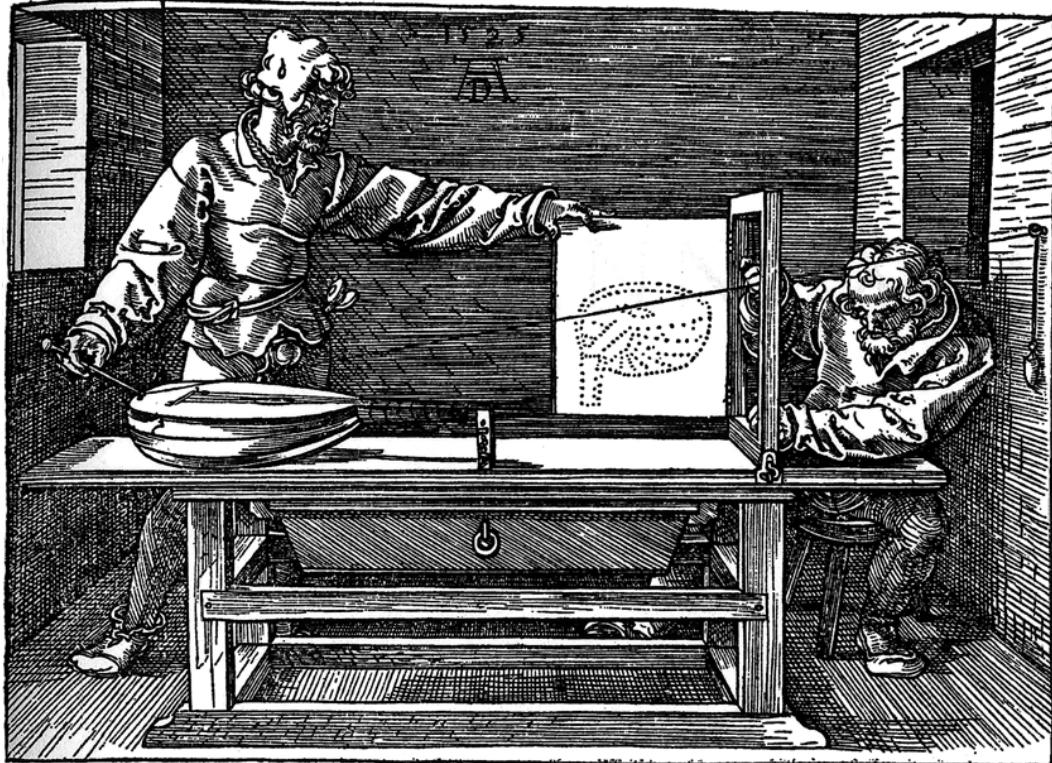


Figure 9.4: Drawing by Dürer in 1538 to illustrate his setting to compute perspective. It can be thought of as an ancestor of ray-casting. The artist's assistant is holding a stick linked to a string fixed at an eyebolt in the wall which represents the viewpoint. He points to part of the object. The 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).

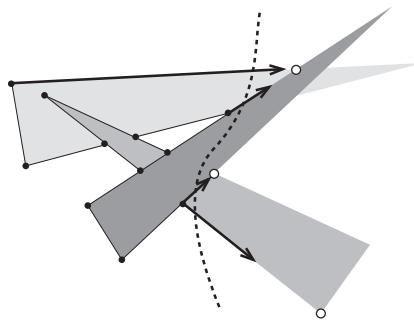


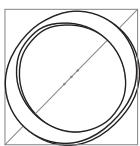
Figure 9.5: Sweep of a visibility map. Active edges are in bold. Already processed events are black points, while white points indicate the event queue.

CHAPTER 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*



OBJECT-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.

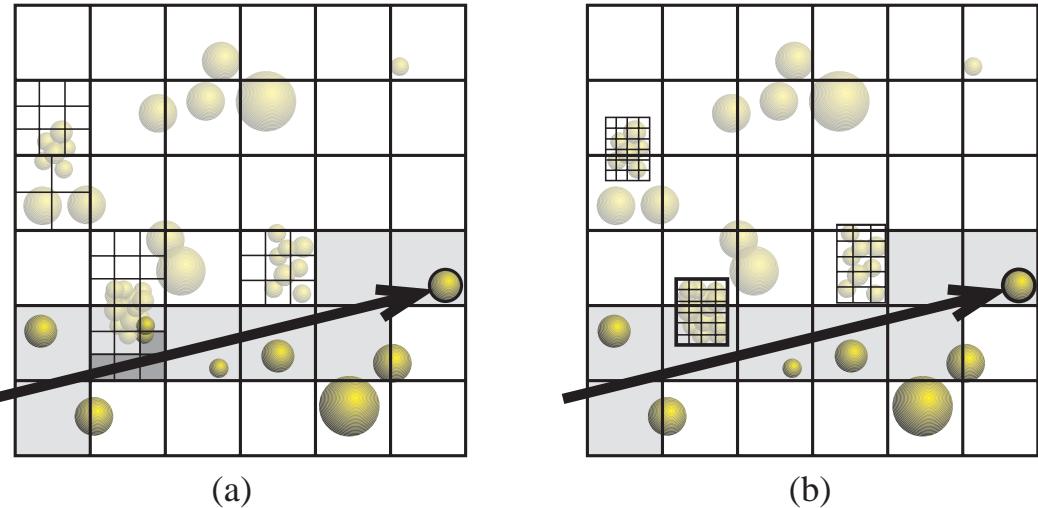


Figure 10.1: A 2D analogy of ray-tracing acceleration. An intersection test is performed for objects which are in bold type. (a) Recursive grid. (b) Hierarchy of uniform grids. Note that fewer intersections are computed with the latter because the grids fit more tightly to the geometry.

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 example¹.

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].

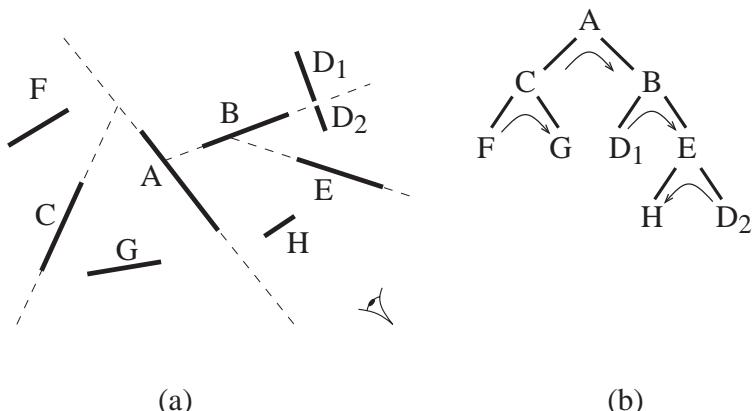


Figure 10.2: 2D BSP tree. (a) The scene is recursively subdivided along the polygons. Note that polygon D has to be split. (b) Corresponding binary tree. The traversal order for the viewpoint in (a) is depicted using arrows. The order is thus, from back to front: $FCGAD_1BHED_2$

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 of 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 BSPPs 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].

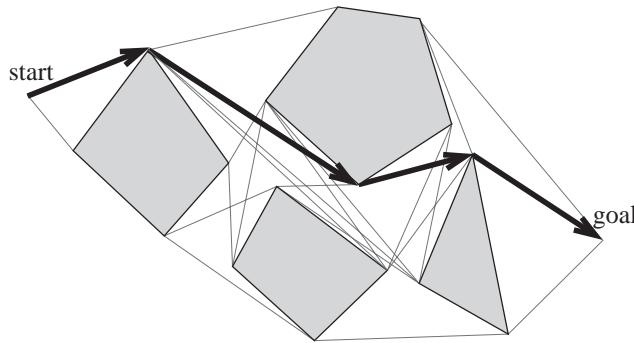


Figure 10.3: Path planning using the visibility graph.

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 $O(n^2)$ 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 segments².

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

²What we describe here are in fact shadows in a Riemannian geometry. Our curved lines of sight are in fact *geodesics*, *i.e.* the shortest path from one point to another.

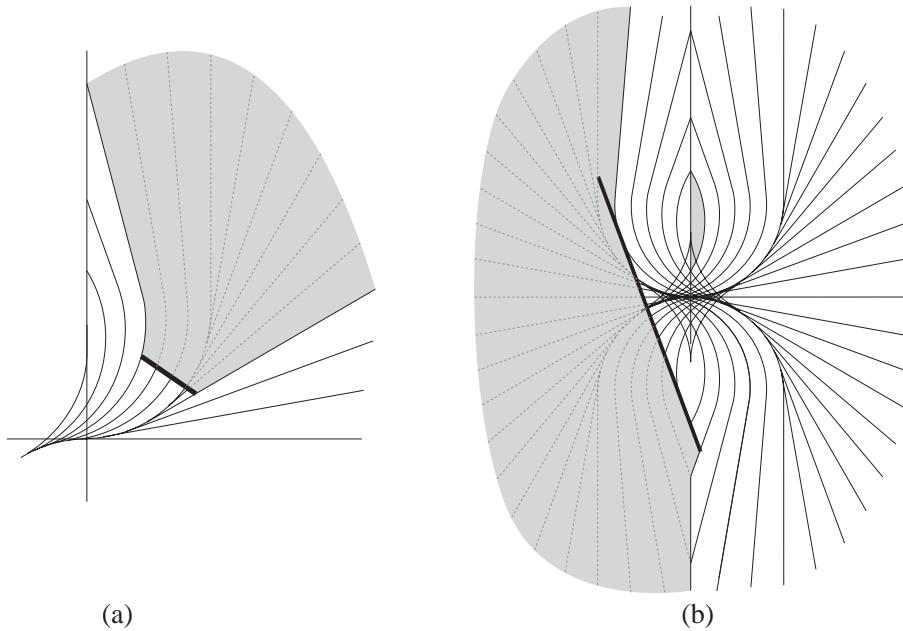


Figure 10.4: Shadow for non-holonomic path-planning (adapted from [VLN96]). (a) Simple (yet curved) shadow. (b) Complex shadows. Some parts of the convex blocker do not lie on the shadow boundary. The small disconnected shadow is caused by the impossibility to perform an initial backward circle arc.

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.

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.

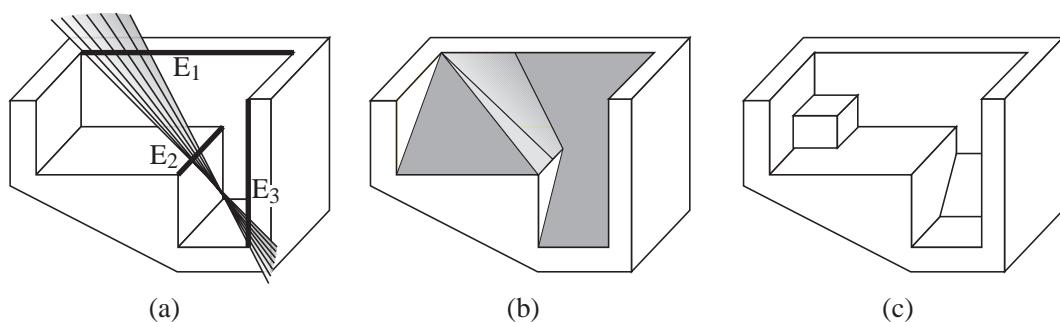


Figure 10.5: Visual hull (adapted from [Lau94]). (a) Initial object. A *EEE* event is shown. (b) Visual hull of the object (the viewer is not allowed inside the convex hull of the object). It is delimited by polygons and a portion of the ruled quadric of the $E_1E_2E_3$ event. (c) A different object 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.

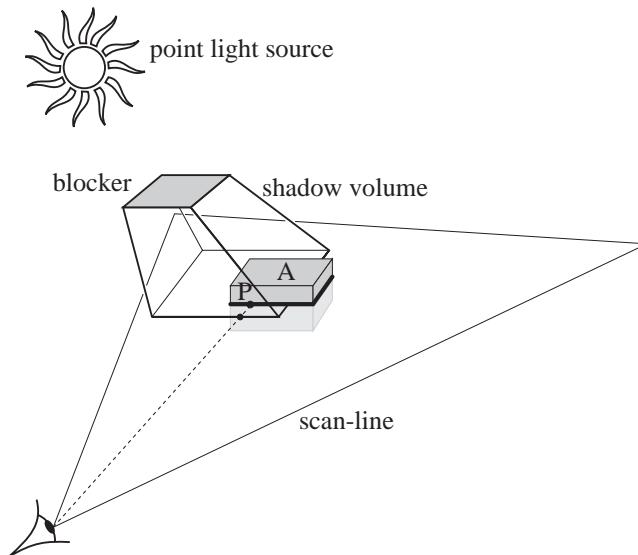


Figure 10.6: Shadow volume. As object *A* is scan converted on the current scan-line, the shadowing of each pixel is computed by counting the number of back-facing and front-facing shadow volume polygons on the line joining it to the viewpoint. For point *P*, there is one front-facing intersection, it 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

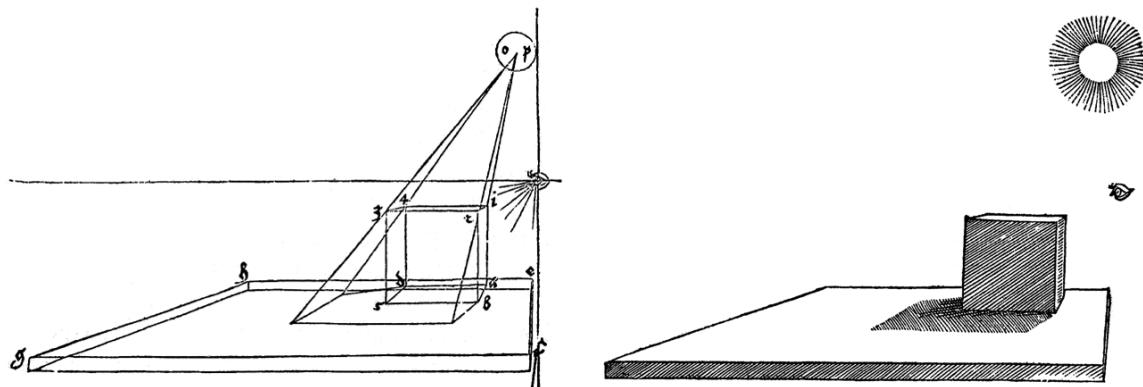


Figure 10.7: Construction of the shadow of a cube by Dürer.

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)).

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.

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.

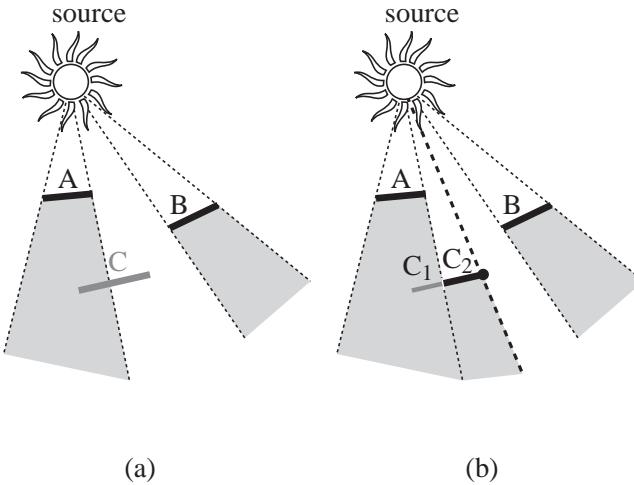


Figure 10.8: 2D equivalent of shadow BSP. The splitting planes of the shadow BSP are represented with dashed lines. (a) Polygon C is tested against the current shadow BSP. (b) It is split into a part in shadow C_1 and a lit part C_2 . The boundary of the lit part generates a new splitting plane.

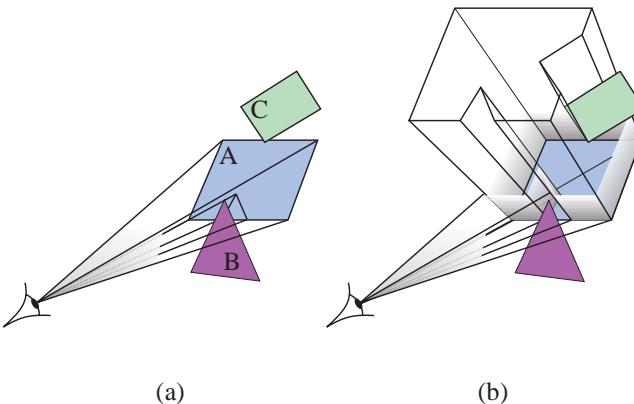


Figure 10.9: Beam tracing. (a) A beam is traced from the eye to the scene polygons. It is clipped against the other polygons. (b) Since polygon A is a mirror, a reflected beam is recursively traced and clipped.

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.

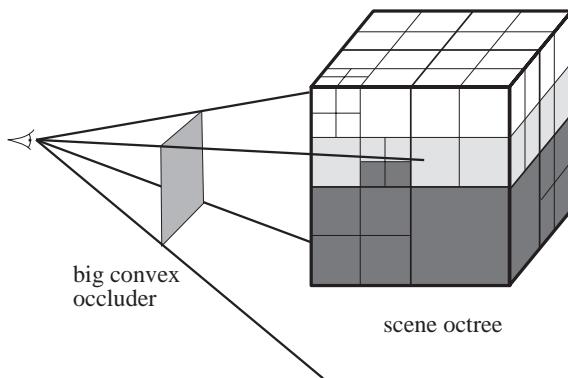


Figure 10.10: Occlusion culling with large occluders. The cells of the scene octree are classified against the shadow volumes. In dark grey we show the hidden cells, while those partially occluded are in light grey.

The occlusion by a conjunction of occluders is 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].

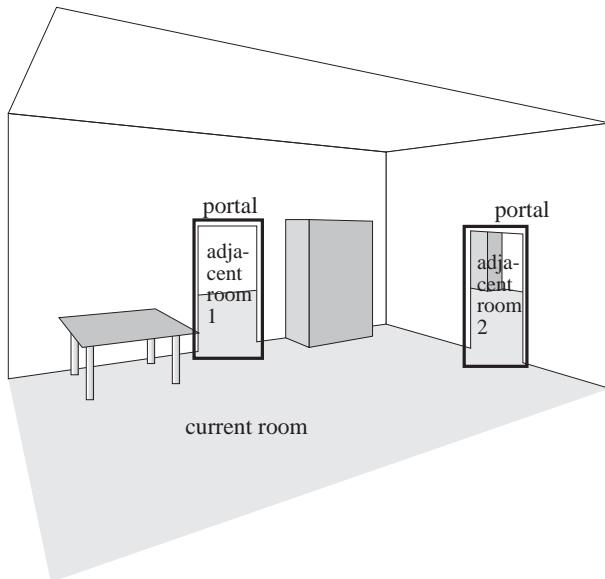


Figure 10.11: Occlusion culling using image-space portals. The geometry of the adjacent rooms is tested against the screen bounding boxes of the portals

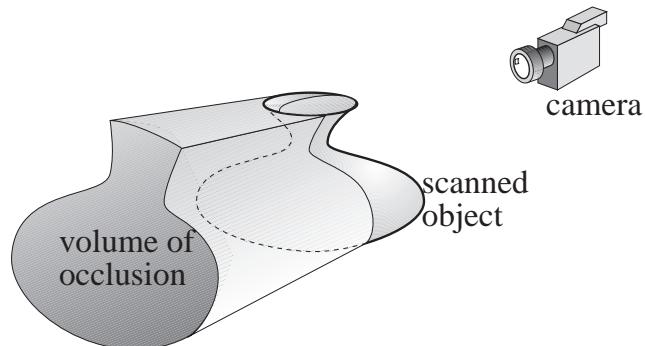


Figure 10.12: Acquisition of the model of a 3D object using a range image. The volume of occlusion is the unknown part of space.

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 146). When a view is reprojected, the regions of indeterminate visibility lie on the boundary of the volumes of occlusion.

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 inter-

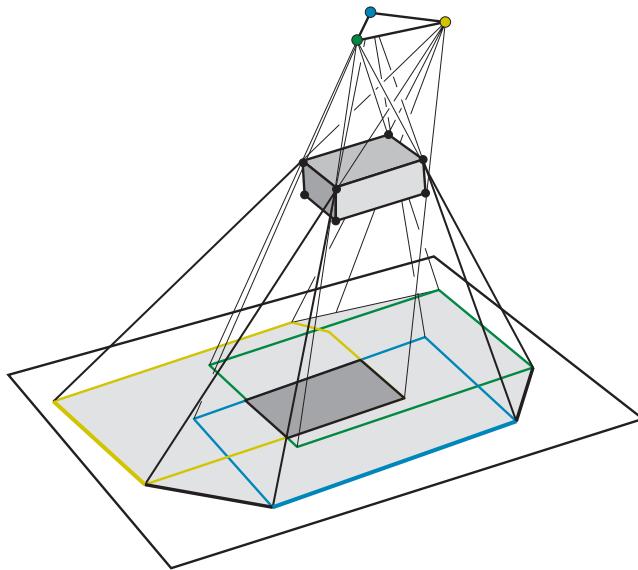


Figure 10.13: Umbra (dark grey) and penumbra (light grey) of a convex blocker (adapted from [NN85]).

section 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 161 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).

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.

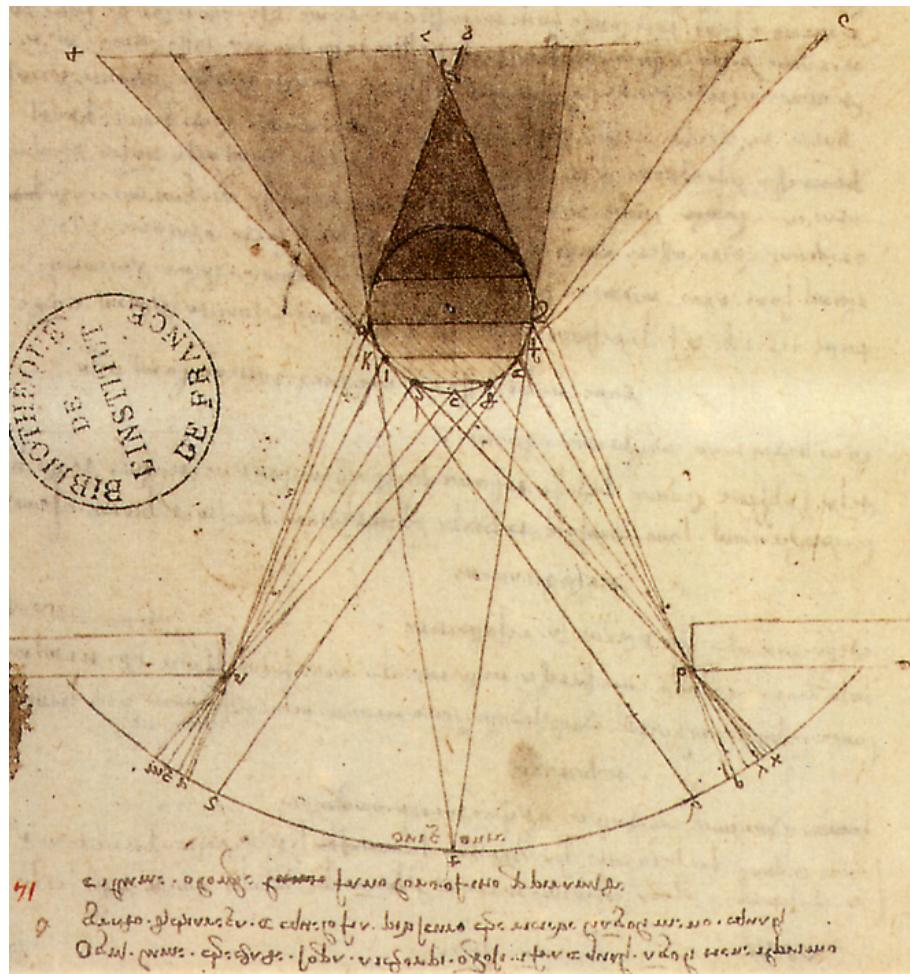


Figure 10.14: Penumbra by Leonardo da Vinci (Manuscript). Light is coming from the lower window, and the sphere causes soft shadows.

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].

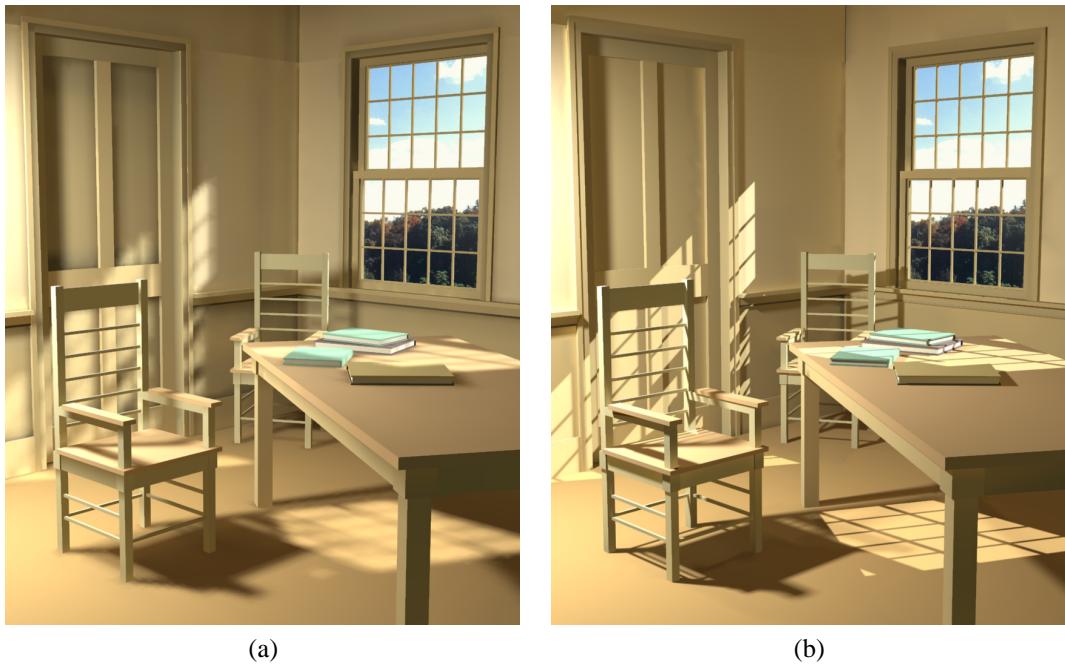
5.5 Viewpoint constraints

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.

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



(a)

(b)

Figure 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).

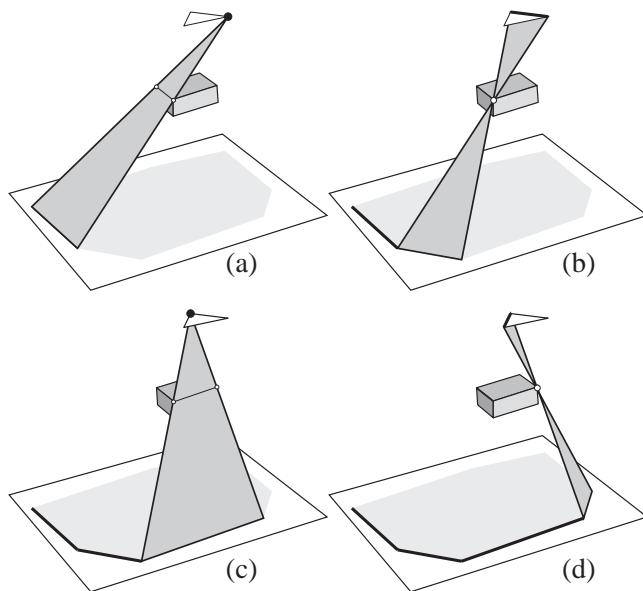


Figure 10.16: Linear time construction of a penumbra volume.

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 intersection of shadow volumes (see Fig. 10.17).

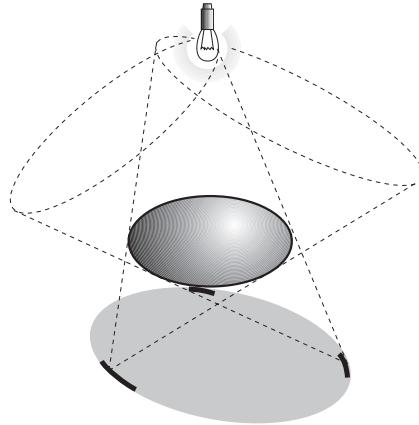


Figure 10.17: Sketching shadows. The user specifies the shadows of the ellipsoid on the floor with the thick strokes. This generates constraint cones (dashed). The position of the light source is then deduced (adapted from [PRJ97]).

Their method can also handle soft shadows, and additional constraints such as the position of highlights.

6 Shafts

Shaft methods 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 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 the union of their bounding boxes (see Fig. 10.18).

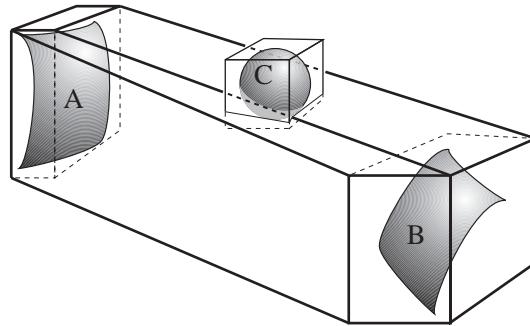


Figure 10.18: Shaft culling. The shaft between A and B is defined as the convex hull of the union of their bounding boxes. Object C intersects the shaft, it may thus cause occlusion between A and B .

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]. Bri  re 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.

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

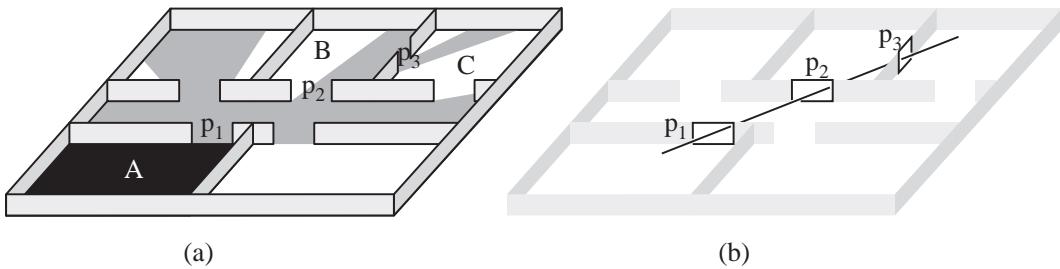


Figure 10.19: Visibility computations in architectural environments. (a) In grey: part of the scene visible from the black cell. (b) A stabbing line (or sightline) through a sequence of portals.

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.

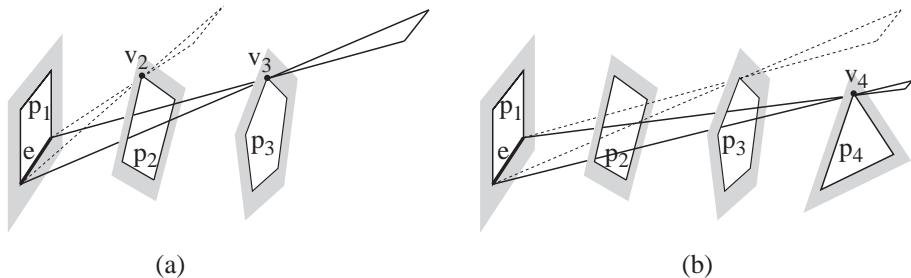


Figure 10.20: Conservative visibility propagation through arbitrary portals. (a) The separating plane considered for e is generated by v_3 because it lies below the one generated by v_2 . (b) As a new portal is added to the sequence, the separating plane is updated with the same criterion.

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.

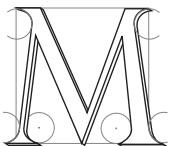
Funkhouser *et al.* [FS93] have extended Teller's system to use other rendering acceleration techniques such as mesh simplification in a real time context to obtain a constant framerate. He and his co-authors [FST92, Fun96c] have also used the information provided by the potentially visible sets to efficiently 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].

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 schemes¹. 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.



Figure 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 (gallon 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

¹Unexpected 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].

if a point is in shadow or not consists in projecting it back to the shadow map and comparing its distance to 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. Hard-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.

²A 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.

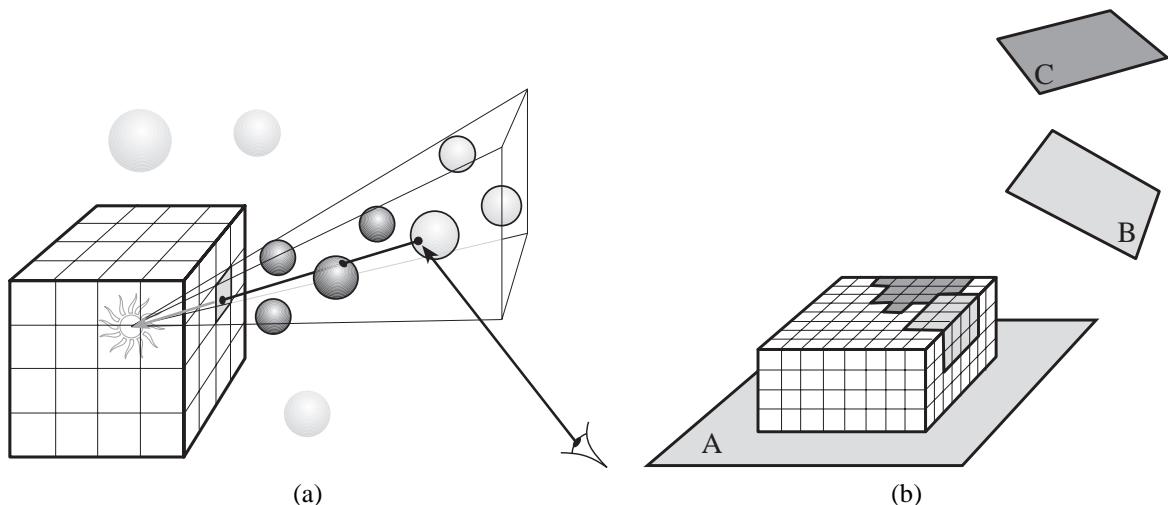


Figure 11.2: (a) Light buffer. (b) Form factor computation using the hemicube. Five z-buffers are placed around the center of patch A. All form factors between A and the other patches are evaluated simultaneously, and occlusion of C by B is taken into account.

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 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 to 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, an 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).

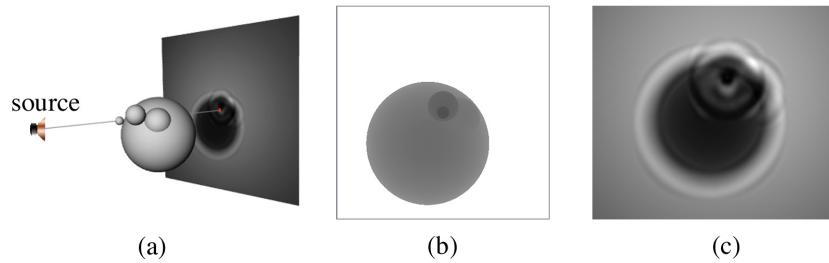


Figure 11.3: Non binary visibility for sound propagation. The diffraction by the spheres of the sound emitted by the source causes the diffraction pattern on the plane. (a) Geometry of the scene. (b) z-buffer from the source. (c) Close up of the diffraction pattern of the plane. (Courtesy of Nicolas Tsingos, iMAGIS-GRAVIR).

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.

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.

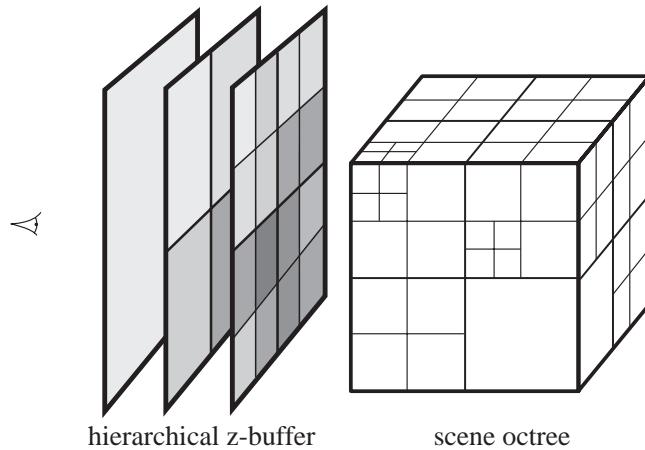


Figure 11.4: Hierarchical z-buffer.

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.* [ZMH97, 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 [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.

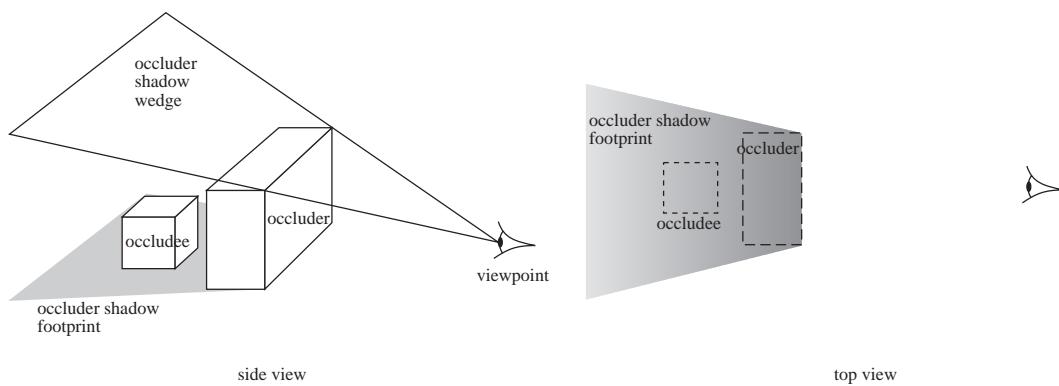


Figure 11.5: Occluder shadow footprints. A projection from above is used to detect occlusion. Objects are hidden if they are below the occluder shadows. The footprints (with height) of the occluded regions are rasterized using a z-buffer. Depth is represented as grey levels. Note the gradient in the 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*.

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.

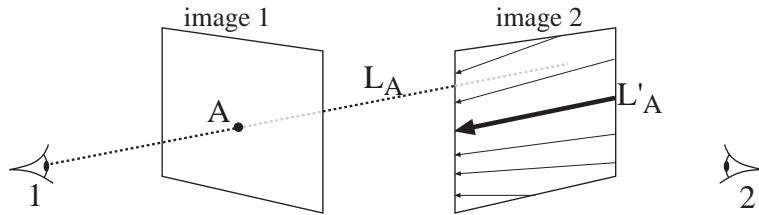


Figure 11.6: Epipolar geometry. L_A is the set of all points of the scene possibly projecting on A . L'_A is the projection on image 2. For a warping from image 2 to image 1, points of image 2 have to be reprojected to image 1 in the order depicted by the arrows for correct occlusion.

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.

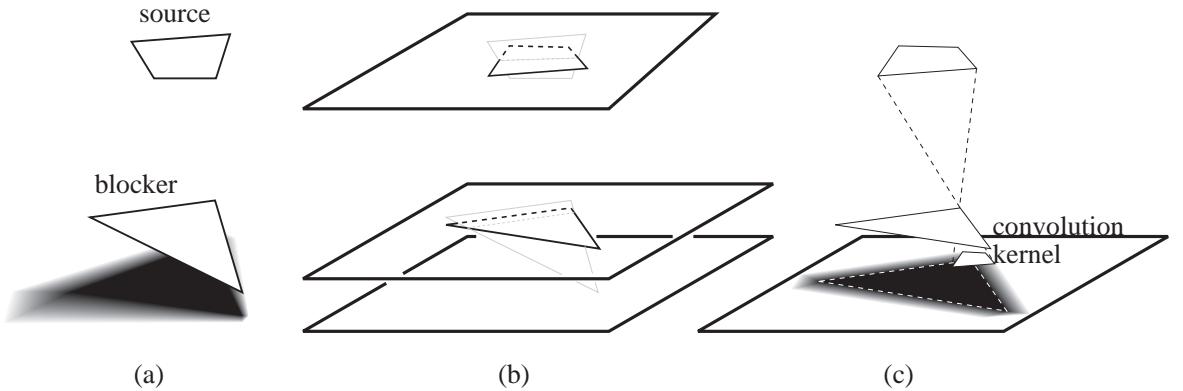


Figure 11.7: Soft shadows computation using convolution. (a) Geometry of the scene. (b) Projection on a parallel approximate geometry. (c) The shadow is the convolution of the projection of the blockers with the inverse image of the source.

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).

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

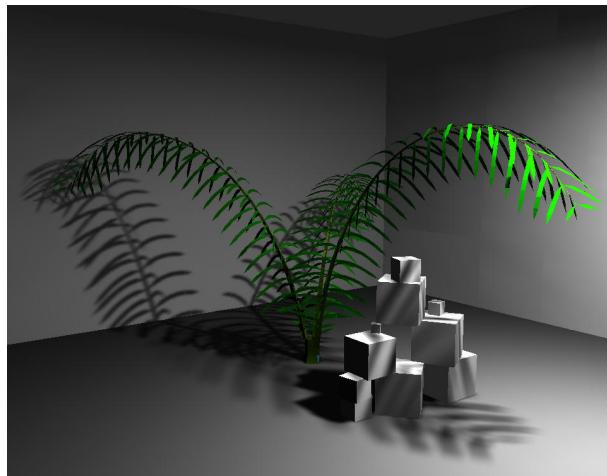


Figure 11.8: Soft shadows computed using convolutions (image courtesy of Cyril Soler, iMAGIS-GRAVIR)

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.

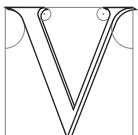
Hart *et al* [HDG99] perform a similar image-space floodfill to compute a blocker map: for each pixel, the objects casting shadows on the visible point are stored. They are determined using a low number of rays per pixel, but due to the image-space flood-fill the probability to miss blockers is very low. They then use an analytic clipping of the source by the blockers to compute the illumination of each pixel.

CHAPTER 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 148, 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 from 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.

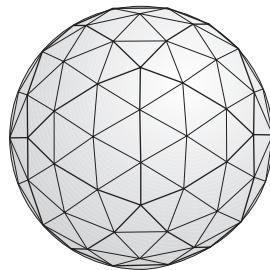


Figure 12.1: Quasi uniform subdivision of the viewing sphere starting with an icosahedron.

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.

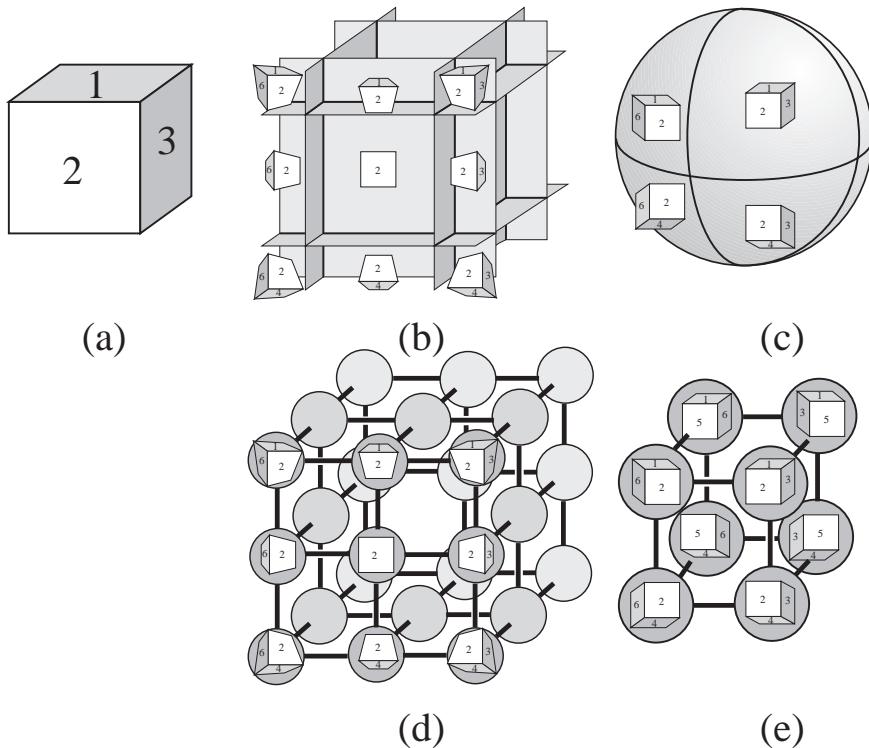


Figure 12.2: Aspect graph of a convex cube. (a) Initial cube with numbered faces. (b) and (c) Partition of the viewpoint space for perspective and orthographic projection with some representative aspects. (d) and (e) Corresponding aspect graphs. Some aspects are present in perspective projection but 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 Gigu 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.

Gigu 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.

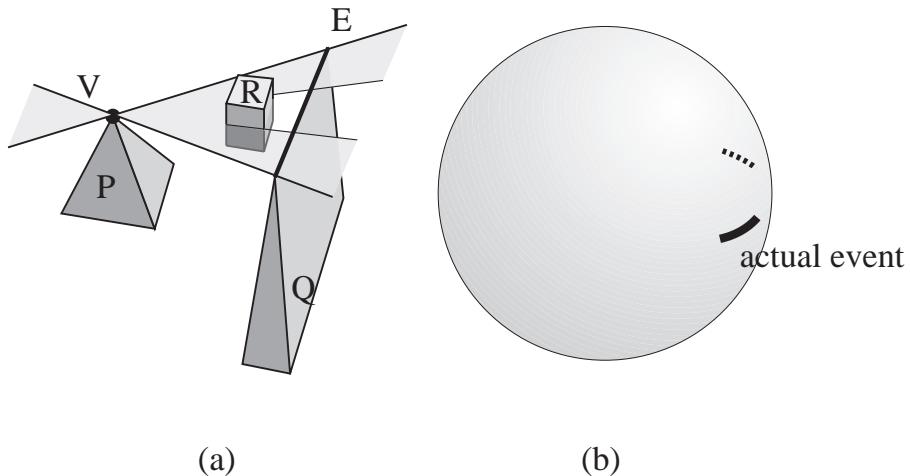


Figure 12.3: False event (“transparent” event). Object R occludes vertex V from edge E , thus only a portion of the potential visual event corresponds to an actual visual event. (a) In object space. (b) In orthographic viewpoint space.

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.

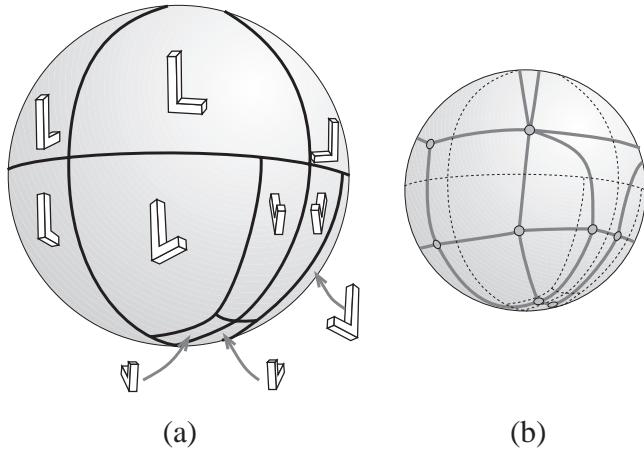


Figure 12.4: Aspect graph of a L-shaped polyhedron under orthographic projection (adapted from [GM90]). (a) Partition of the viewing sphere and representative views. (b) Aspect graph.

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 160). 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]. Their 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.

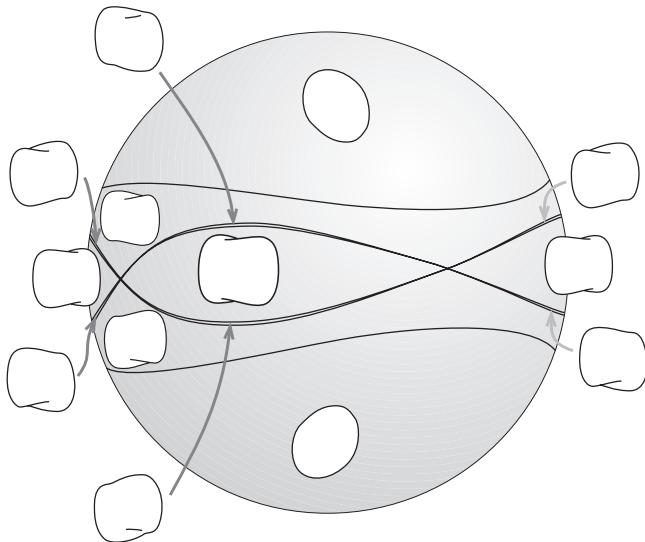


Figure 12.5: Partition of orthographic viewpoint space for a dimple object with representative aspects. (adapted from [PPK92]).

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.

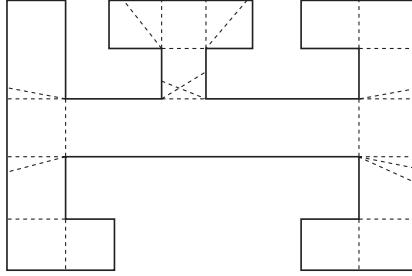


Figure 12.6: Robot self-localization. Partition of a scene into cells of structurally invariant views by visual events (dashed).

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 152 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].

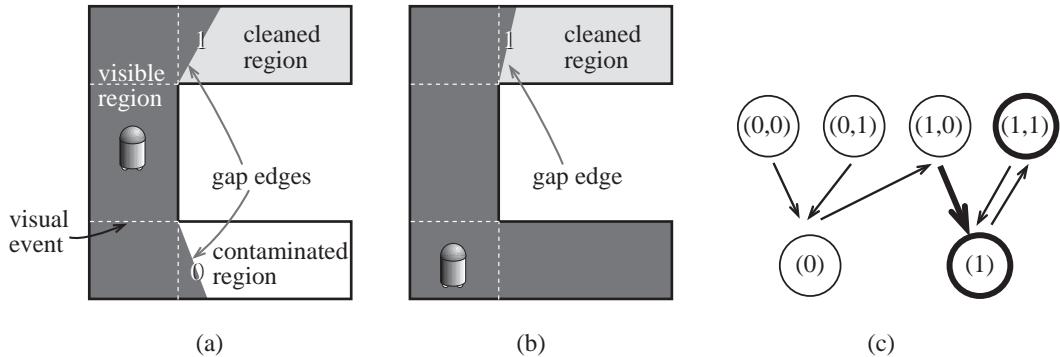


Figure 12.7: Pursuit-Evasion strategy. (a) The contaminated region can be cleaned only if the visual event is crossed. The status of the neighbouring regions is coded on the gap edges. (b) The robot has moved to a second cell, cleaning a region. (c) Part of the graph of possible states (upper node correspond 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 152 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.

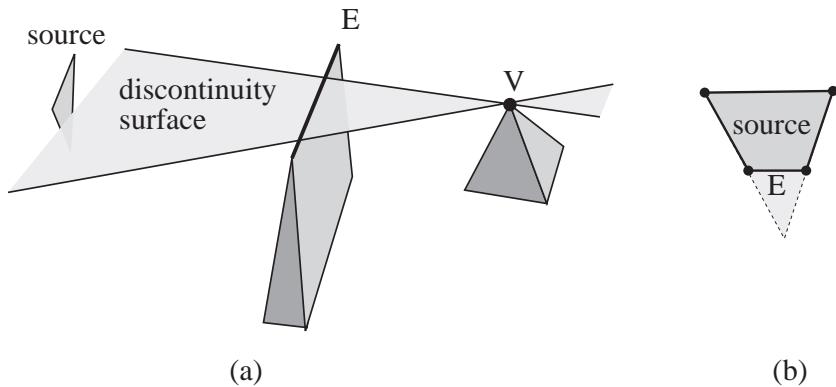


Figure 12.8: Complete discontinuity meshing with backprojections. (a) Example of an EV event intersecting the source. (b) In thick backprojection from V (structure of the visible part of the source)

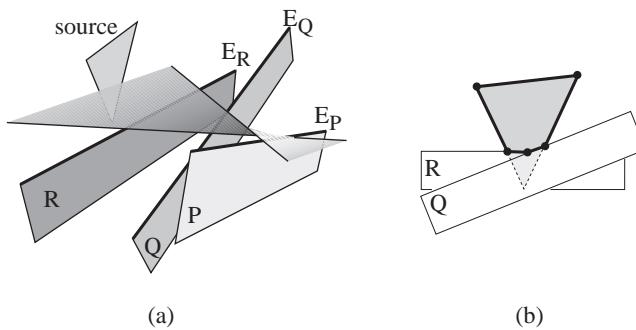


Figure 12.9: Discontinuity meshing. (a) Example of an EEE event intersecting the source. (b) In thick backprojection from a point on E_P (structure of the visible part of the source)

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.

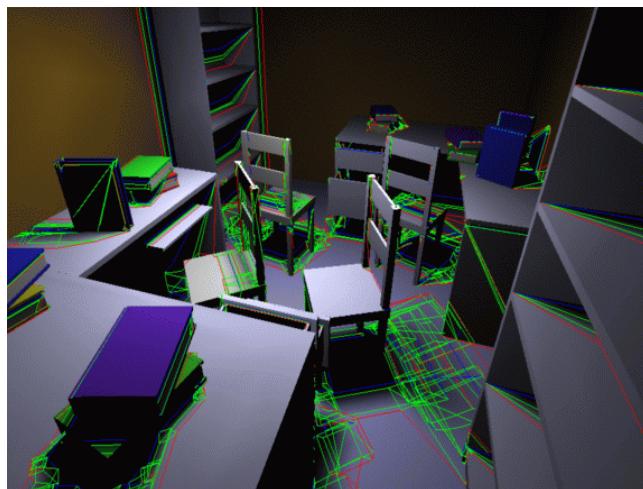


Figure 12.10: Complete discontinuity mesh of a 1000 polygons scene computed with Drettakis and Fiume's algorithm [DF94].

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 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.

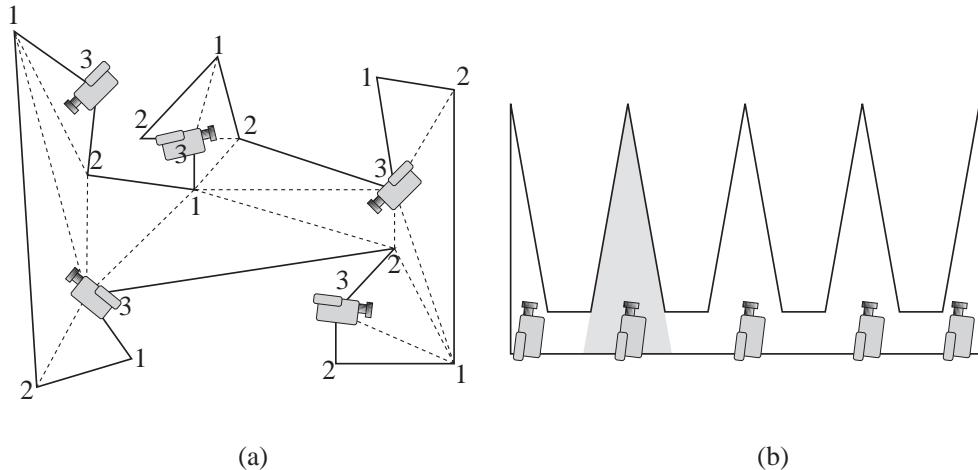


Figure 12.11: Art gallery. (a) The triangulation of a simple polygon is 3-colored with colors 1, 2 and 3. Color 3 is the less frequent color. Placing a guard at each vertex with color 3 permits to guard the polygon with less than $\lfloor \frac{n}{3} \rfloor$ guards. (b) Worst-case scene. To guard the 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 150) 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 200). 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.



Figure 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.

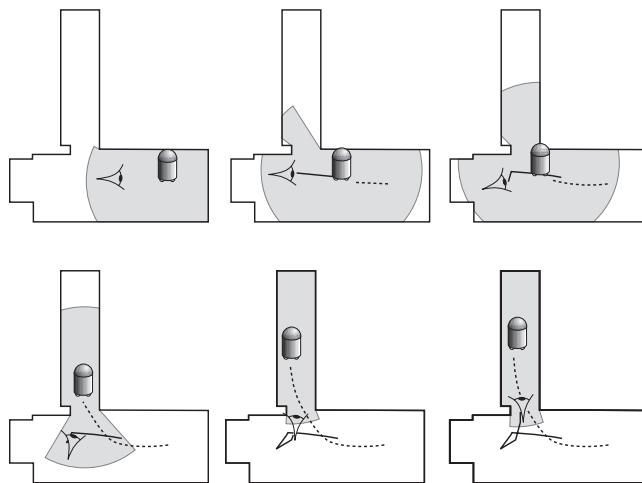


Figure 12.13: Tracking of a mobile target by an observer. The region in which the target is visible is in light grey (adapted from [LGBL97]).

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).

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.

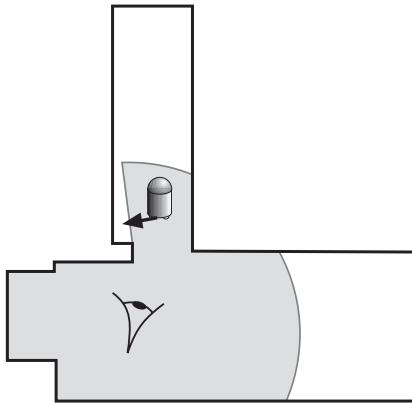


Figure 12.14: Tracking of a mobile target by an observer. The region in light grey is the region in which the target is visible from the observer. The thick arrow is the shortest path for the target to escape.

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.

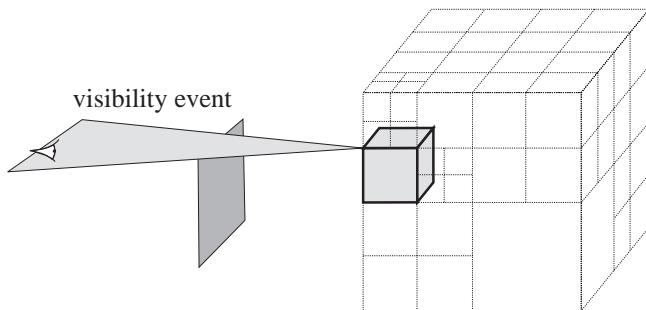


Figure 12.15: Occlusion culling and visual events

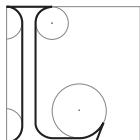
They explain that this can be seen as a local linearized version of the aspect graph. Indeed they maintain a superset of the *EV* boundaries of the current cell of the perspective aspect graph of the scene.

CHAPTER 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 159) 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

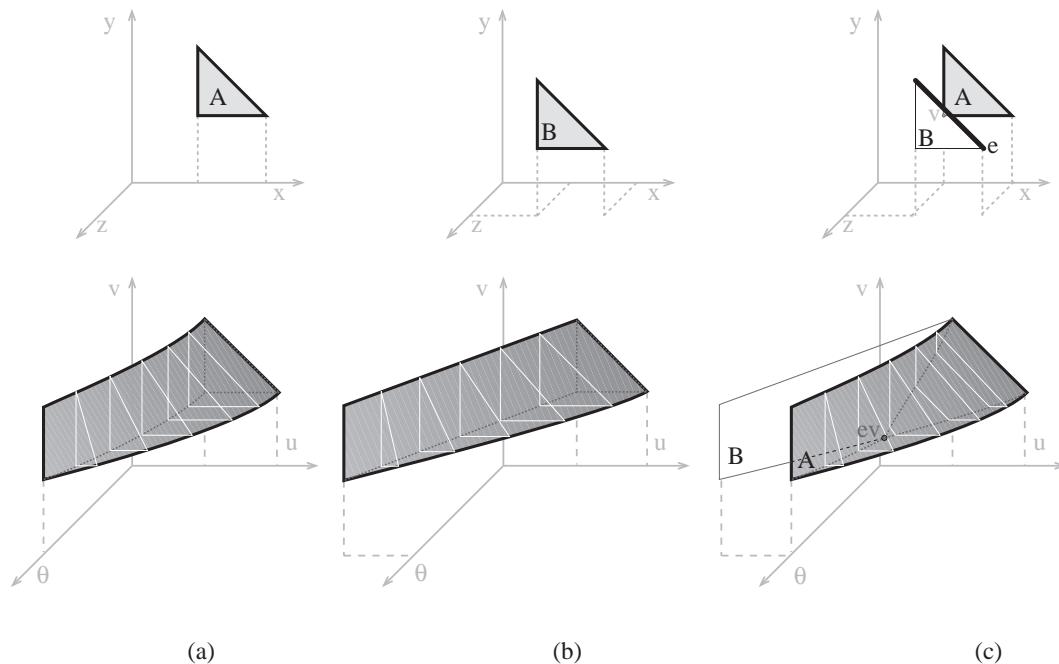


Figure 13.1: Slice of the *asp* for $\phi = 0$ (adapted from [PD90]). (a) and (b) *Asp* for one triangle. The θ slices in white correspond to orthographic views of the triangle. When $\theta = 90^\circ$ the view of the triangle is a segment. (c) Occlusion corresponds to subtraction in *asp* space. We 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* space.

is present under the form of a point. Since occlusion has been taken into account with subtraction, the *asp* 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

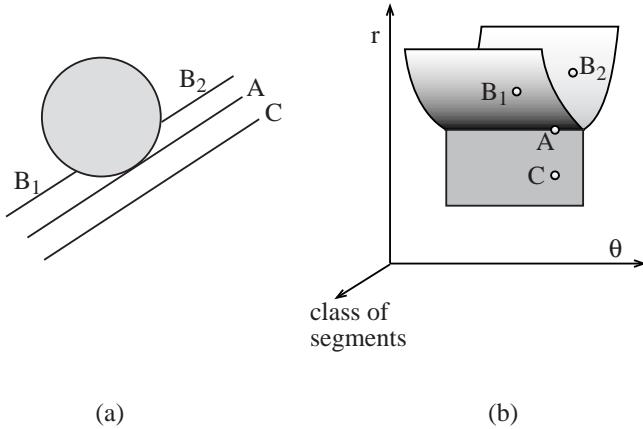


Figure 13.2: Topology of maximal free segments. (a) In the scene. (b) In a dual space where lines are mapped into points (the polar parameterization of line is used).

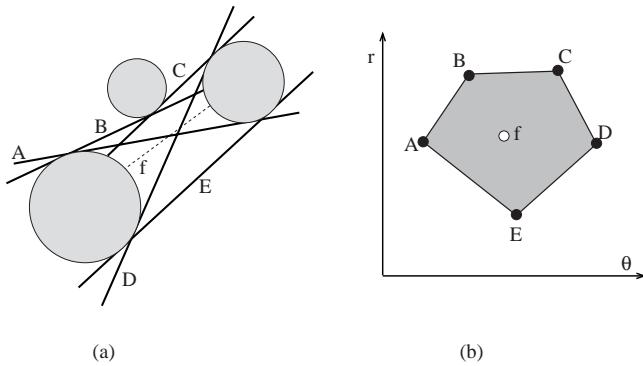


Figure 13.3: A face of the visibility complex. (a) In the scene. (b) In a dual space.

vertices are the bitangents forming the visibility graph.

The 2D visibility complex has been applied to the 2D equivalent of lighting simulation by Ortí *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).

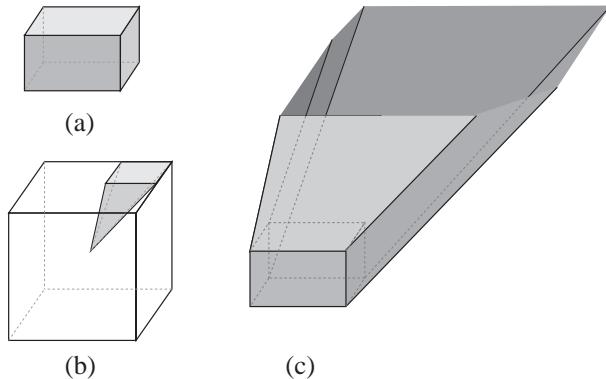


Figure 13.4: Ray classification. (a) interval in origin space. (b) interval in direction space. (c) Corresponding beam of rays.

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.

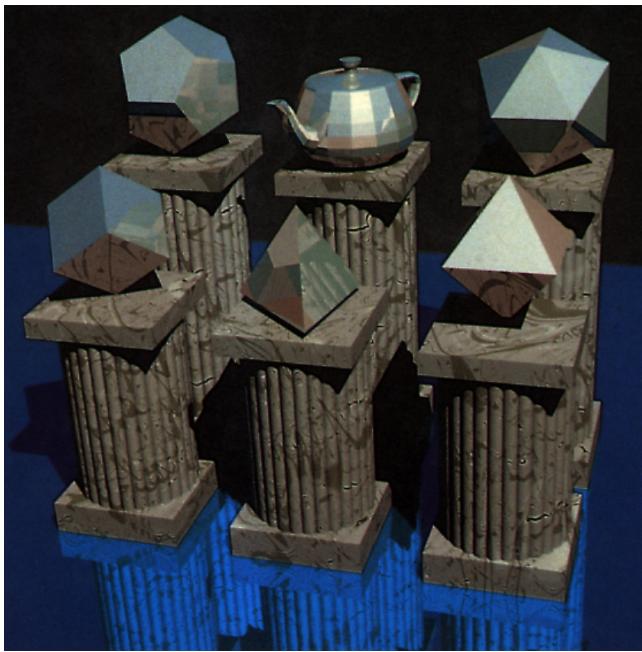


Figure 13.5: Image computed using ray classification (courtesy of Jim Arvo and David Kirk, Apollo Computer Inc.)

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 than 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.

Lamparameter *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 159). 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.* [WPB98] 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

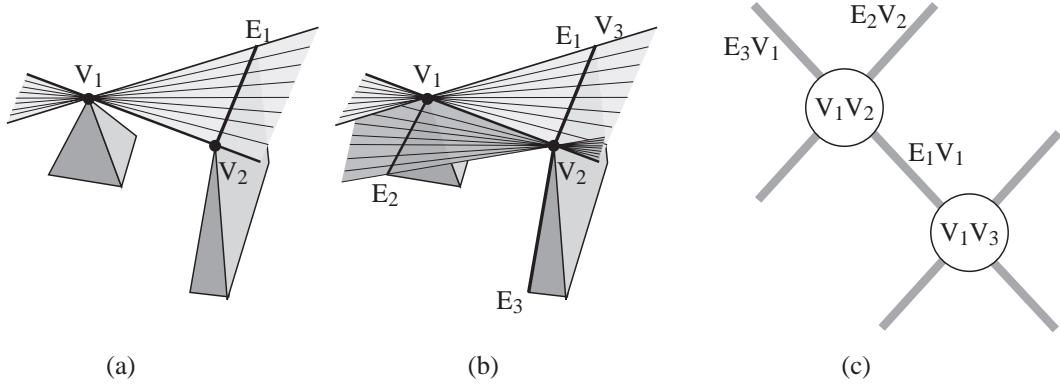


Figure 13.6: (a) An *EV* critical line set. It is bounded by two extremal stabbing lines V_1V_2 and V_1V_3 . (b) Other *EV* critical line sets are adjacent to V_1V_2 . (c) Corresponding graph structure in line space.

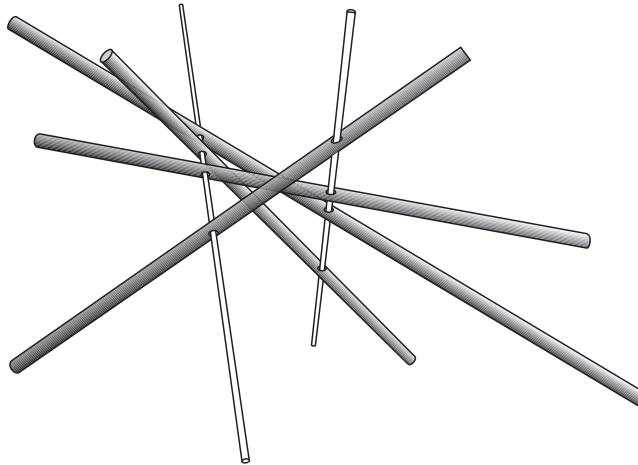


Figure 13.7: Four lines in general position are stabbed by two lines (adapted from [Tel92b])

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 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

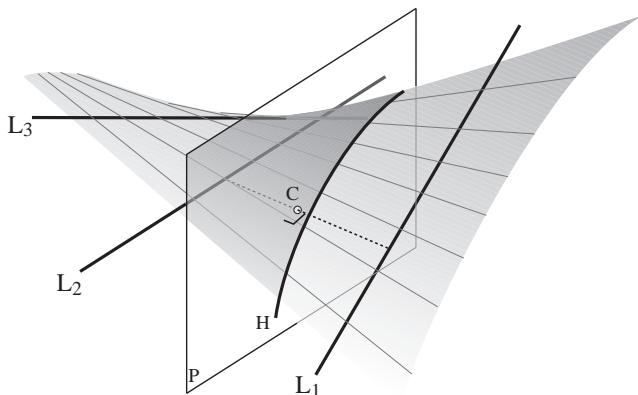


Figure 13.8: Skewed projection.

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).

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).

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 *Grassmann 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.

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.

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.

¹Plücker coordinates can also be extended to use the 6 coordinates to describe forces and motion. See *e.g.* [MS85, PPR99]

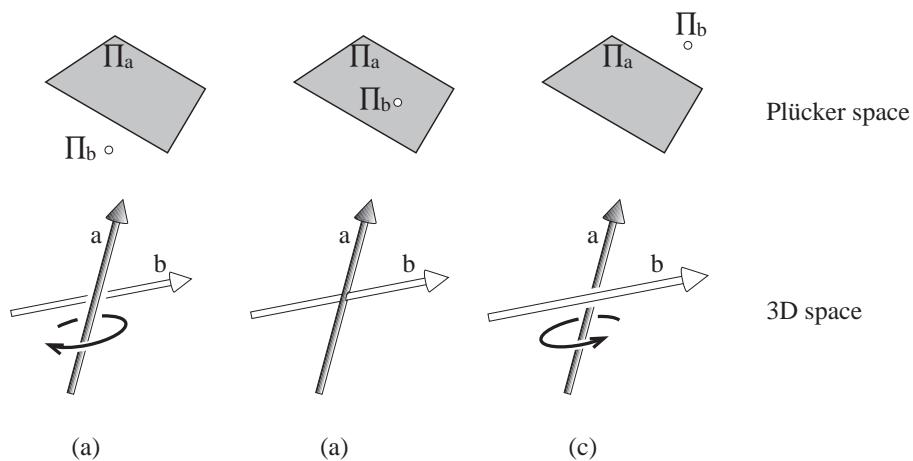


Figure 13.9: In Plücker space the hyperplane corresponding to a line a separates lines which turn clockwise and counterclockwise around a . (The hyperplane is represented as a plane because a five-dimensional space is hard to illustrate, but note that the hyperplane is actually 4D).

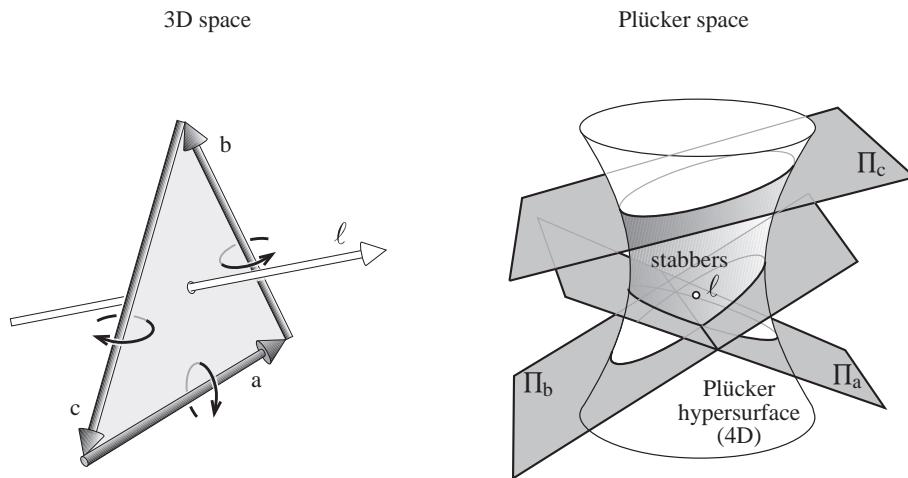


Figure 13.10: Lines stabbing a triangle. In 3D space, if the edges are well oriented, all stabbers rotate around the edges counterclockwise. In Plücker space this corresponds to the intersection of half spaces. To obtain real lines, the intersection with the Plücker hypersurface must be considered. (In fact the hyperplanes are tangent to the Plücker hypersurface)

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).

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.

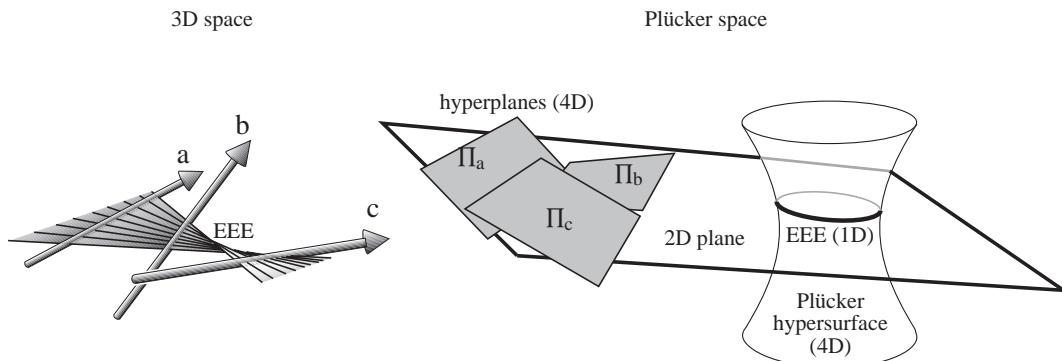


Figure 13.11: Three lines define a *EEE* critical line set in 3D space. This corresponds to the intersection of hyperplanes (not halfspaces) in Plücker space. Note that hyperplanes are 4D while their intersection is 2D. Unfortunately they are represented similarly because of the lack of dimensions of this sheet of paper.(adapted from [Tel92b]).

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 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).

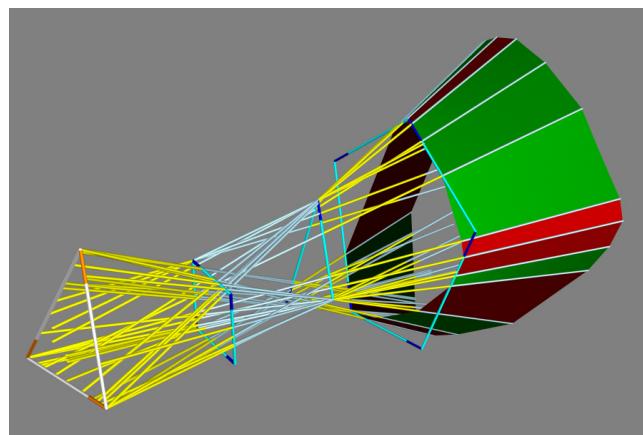


Figure 13.12: Antipenumbra cast by a triangular light source through a sequence of three polygonal openings. *EEE* boundaries are in red (image courtesy of Seth J. Teller, University of Berkeley).

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 159) has density $\frac{da db dp dq}{(1+a^2+b^2)^2}$. 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].

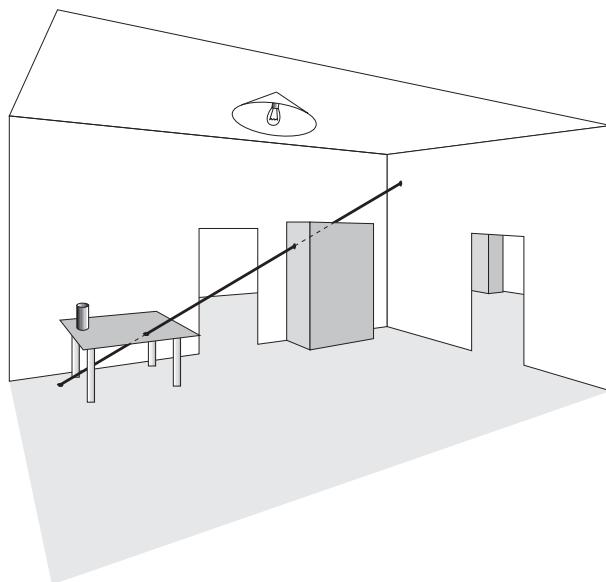


Figure 13.13: Global Monte-Carlo radiosity. The intersection of the line in bold with the scene allows the simulation of light exchanges between the floor and the table, between the table and the cupboard and between the cupboard and the ceiling.

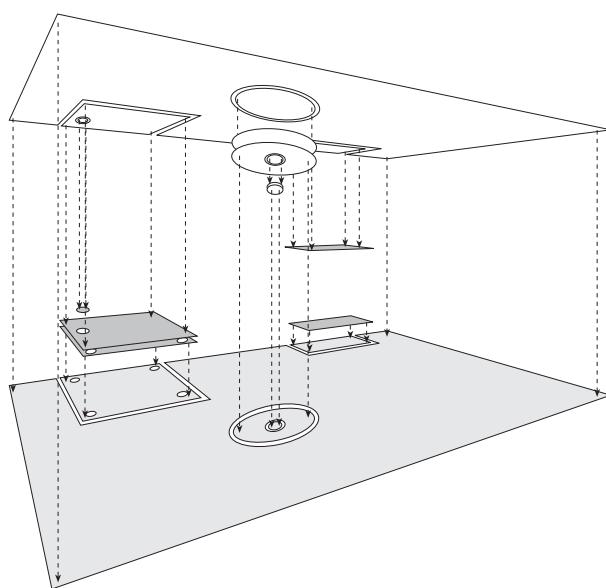


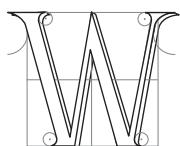
Figure 13.14: Transillumination plane. The exchanges for one direction (here vertical) are all evaluated simultaneously using an extended hidden surface removal algorithm.

CHAPTER 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].

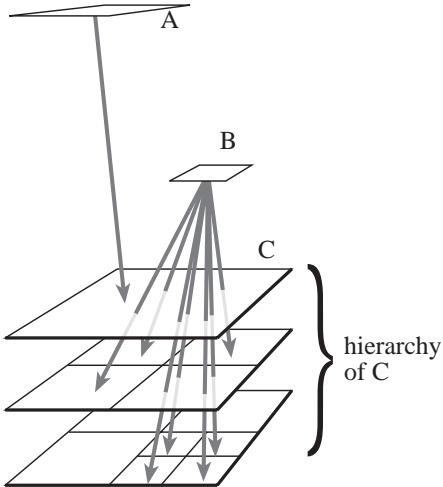


Figure 14.1: Hierarchical radiosity. The hierarchy and the exchanges arriving at *C* are represented. Exchanges with *A* are simulated at a coarser level, while those with *B* are refined.

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 objects 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 Hubbard [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].

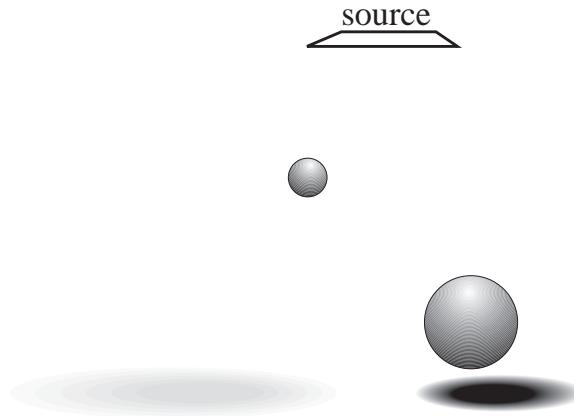


Figure 14.2: Objects which are larger and closer to the receiver cast more significant shadows. Note that the left hand sphere casts no umbra, only penumbra.

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.

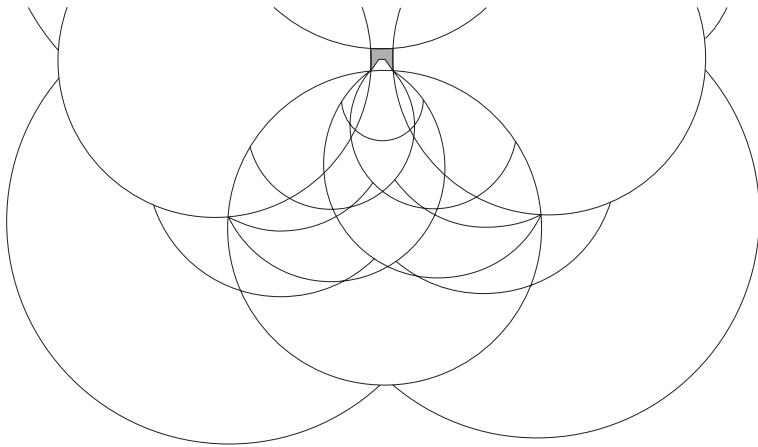


Figure 14.3: Scale-space aspect graph in 2D using perspective projection for the small object in grey. Features which subtend an angle of less than 4° are considered indistinguishable. The circles which subdivide the plane are the visual events where features of the object visually merge.

Scale (the angle threshold) defines a new dimension of the viewpoint space. Fig. 14.3 in fact represents a slice $scale = 4^\circ$ 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 increased 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.

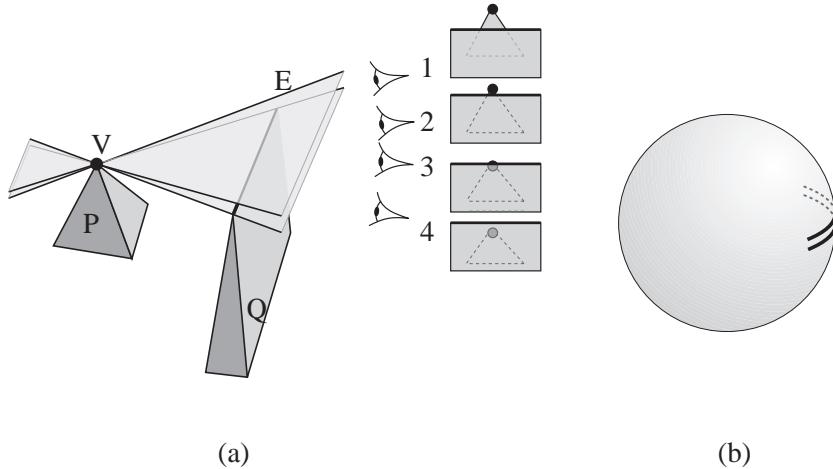


Figure 14.4: Finite resolution aspect graph. (a) The EV event is doubled. Between the two events (viewpoint 2 and 3), E and V are visually superimposed. (b) The doubled event on the viewing sphere.

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.* [BWC86] 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, 2 for 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 Besuevsky *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.

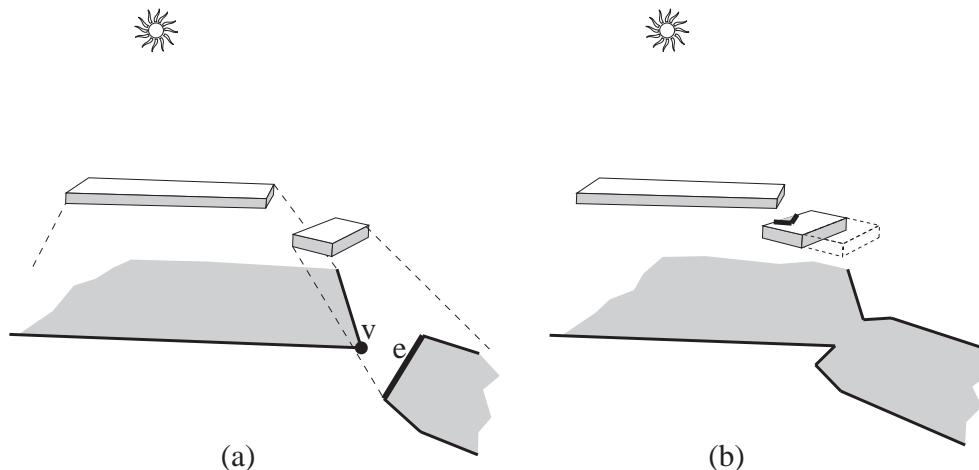


Figure 14.5: Dynamic update of limits of shadow. The situation where shadows appear on the moving object can be determined by checking the shadows on the floor. This can be generalized to discontinuity meshes (after [LD97]).

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.

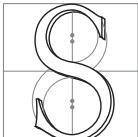
Similarly, a catalogue of singularities has been developed for moving objects, defining a *temporal visibility complex*.

CHAPTER 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 proposed 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 .

Image-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 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.

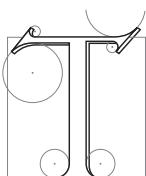
Table 15.1: Recapitulation of the techniques presented by field and by space.

	Object space	Image space	Viewpoint space	Line space
view computation				
Hard shadow	BSP 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	
Contour intersection	visual hull		viewpoint optimization	
Sensor placement	viewpoint constraint best-next view		art gallery	
Motion planning	visibility graph	use of the z-buffer	pursuit-evasion self localisation target tracking	

Conclusions

Étudie d'abord la science, puis apprends la pratique née de cette science.

Léonard DE VINCI, *Codex Urbinas*



HIS DOCUMENT describes theoretical and practical work on visibility. Thanks to a study in line-space, we have been able to better understand the visibility properties of a 3D scene and their coherence. This allowed us to develop a data-structure which has proved useful for lighting simulation, an application which is very demanding in visibility calculation. We have presented an efficient preprocess for the display of complex scenes, which reduces 3D visibility properties to simple tests on planes. Finally, we have proposed a vast survey of work related to visibility. In what follows we first summarize our contributions, we then present issues of future work, and conclude.

Contributions

The 3D visibility complex

The *3D visibility complex* presents new insights into visibility issues. Since it is based on the atomic notion of ray, any visibility problem finds a natural expression in the visibility complex. The notion of maximal free segments permits the simplification of ray-space into a 4-manifold, which facilitates the description of visibility properties and better accounts for the dimensionality of problems. The visibility complex makes the notion of *coherence* explicit; each cell contains a set of segments which see the same objects, and the adjacencies between the boundary faces represent the structure of the limits of visibility of a scene (limits of umbra and penumbra, appearance or disappearance of objects as seen from a moving observer, etc.) The visibility complex describes in a unified framework all the visibility properties of a scene composed of polygons and curved objects, as well as their modification when objects move.

The theoretical complexity of the visibility complex is between linear and $O(n^4)$ in the number of input objects, which provides little information on its practical behaviour. We have described a simplified model of “normal” scenes, and a probabilistic calculation has allowed us to show that under these assumptions, the size of the complex is $O(n^{2.67})$.

The visibility skeleton

To avoid the treatment of a four dimensional cellular complex, we have simplified the visibility complex into a graph in line-space, the *visibility skeleton*. Our construction algorithm is topological and local: compute the nodes of the graph and deduce the arcs using a catalogue of adjacencies. Previous methods based on visual events required the intricate treatment of (non-planar) surfaces in 3D space or curves on the sphere of directions. We only consider single lines, which makes our construction more robust and flexible. Visual events, which are geometrically complex, are deduced using simple topological adjacencies.

Thanks to this algorithm, we have been able to show that it is possible to implement a global visibility structure which is generic. Once the structure is built, very efficient queries are possible (*e.g.*, milliseconds for the retrieval of umbra and penumbra boundaries). The visibility skeleton is simpler and more robust than previous approaches such as discontinuity meshing, even though those methods are more restrictive, since they handle visibility with respect to primary light sources only.

The potential applications of the skeleton are beyond lighting simulation. The flexibility offered by the “on-demand” construction should permit its adaptation to several problems, including aspect graphs and occlusion-free viewpoints in computer vision, or visibility-based motion planning in robotics.

Visibility driven hierarchical radiosity

The application of the visibility skeleton to lighting simulation permits higher quality images together with improved computation time. Previous methods compute visibility by sampling, which is costly and inaccurate, while the visibility skeleton affords an efficient exact computation of the amount of light leaving one polygon arriving at a point (form factor).

It also allows us to subdivide the mesh used to represent the lighting function along the limits of umbra and penumbra, with respect to any polygon while previous methods were limited to a small number of primary light sources. In particular, we handle the discontinuities of the lighting function caused by the limits of visibility of indirect lighting. This technique together with the introduction of lazy wavelets defined on hierarchical triangulations provide a high-quality representation of illumination which is well suited to walkthroughs.

Our refinement criteria are based on the visibility information encoded in the skeleton and on a perceptual metric. This allows us to refine the simulation only if the additional precision may be noticeable by a human observer. We avoid the tedious setting of unpredictable and arbitrary criteria which have impeded the usability of hierarchical simulations.

The tests we have performed show that our method is efficient. It handles efficiently traditionally difficult configurations such as multiple light sources or scenes lit mainly by indirect lighting. The comparison with a recent method has shown that our approach provides higher quality with shorter computation time.

General occlusion-culling using extended projections

Our occlusion-culling preprocess is the first method to compute visibility with respect to a volumetric cell which handles the cumulative occlusion due to multiple occluders. The visibility tests are performed in planes using *extended projection* operators which underestimate the projection of the occluders with respect to any viewpoint of the scene, while the projection of an occludee is overestimated. Our technique is conservative (no visible object is identified as invisible), simple to implement and efficient thanks to the use of graphics hardware.

We have also developed an *occlusion sweep* which permits the handling of particularly difficult configurations, such as visibility from a cell within a forest. During the sweep by a set of parallel planes leaving the cell, occlusion due to leaves of the trees aggregates using a reprojection operator.

To prove the validity of the approach, we have demonstrated some original properties of shadows, in particular on the possible implications of the inclusion of the shadows of two objects.

A multidisciplinary survey of visibility

In the second part of this document, we propose an overview of the visibility techniques developed in different domains. We have proposed a classification according to the space in which computations are performed. This has allowed us to review a large number of methods without proposing a catalogue per field.

This part may be useful to anyone starting to work on visibility, as well as to anyone who searches inspiration and solutions in other communities. It can also be useful to rapidly find a reference to a given technique. Even though it was not organized with this purpose in mind, we hope that it will help those looking for practical answers to concrete problems.

Future work

Direct extensions of our work have been proposed throughout this document. We refer the interested reader to the conclusions sections of each chapter. In what follows, we describe more general issues for future research.

The visibility complex and the visibility skeleton do not scale well, as opposed to approximate approaches. Even though the theoretical $O(n^4)$ does not correspond to the practical behaviour, the quadratic cost predicted by our probabilistic study and confirmed by our experiments is still not acceptable for a practical use on large scenes. The utility of such information is moreover questionable. Is it useful to determine that the edge of a tile and the gap between two bricks of a second distant house generate a visual event? The amount of information is not only costly to compute and store, it is also intricate and unusable.

Multiscale approaches have to be explored. The simplification of the information encoded in the visibility skeleton or in the complex would be a first step to cope with this. Two major issues must be addressed. First, the inherently global nature of visibility makes it difficult to perform a simplification based purely on proximity in line space, since the corresponding spatial influence may be infinite. A first way to address this problem is to study the simplification of visibility information for use outside a group of objects.

The second problem is the definition of approximate or simplified visibility information. Interesting work has been published which address this issue in the context of discontinuity meshing [SK98] or aspect graphs [EBD⁺93, SP97]. The latter approaches define the notion of scale by the visual size of a feature and offers good insights on the topic, even though their applications have at the moment only led to a more complex visibility information.

Robustness issues require efficient treatment, which can be studied together with the issue of scalability using the definition of relevant thresholds. We repeat that we believe that degenerate configurations should not be reduced to general configuration using exact arithmetic. We want to avoid the treatment of infinitely small events which may then generate many problems. The scene encountered in practice contain many degeneracies; these should be exploited and not only avoided.

Another possibility to cope both with scalability and robustness consists in the use of purely approximate computation, based on volumetric [SD95] or rasterized [SK98] approaches. Difficulties include the error control and the possibility to refine calculations [SS96b, Sol98].

As was said before, theoretical bounds sometimes tell little about the practical complexity of methods or problems with “normal” scenes. Probabilistic approaches and a theoretical model of “normal scenes” (in the spirit of de Berg *et al.* [dBKvdSV97] or our probabilistic approach for triple-tangency events in chapter 2) should permit a better understanding of the actual complexity of visibility properties and methods. Such an approach should be accompanied by statistical experiments. This issue is not only relevant to visibility, but also to any geometric problem.

Finally, the treatment of scenes with moving objects deserves more attention. Temporal visual events permit better insights of the underlying phenomena. However, practical and efficient algorithms have to be developed, to avoid the redundancy of computation, since most of the methods recompute everything from scratch for each frame.

To conclude

This thesis has proposed theoretical and practical work on visibility. It is certainly exposed to the criticisms of theoreticians who will regret that it is not rigorous enough and too applied, while applied computer graphics researchers may disapprove of the abstract and multidimensional parts.

We however hope that it shows that practical problems raise difficult and fascinating theoretical issues. 3D visibility is interesting, both from the geometrical and algorithmic point of view. Beyond visibility itself, the

analysis of problems and algorithms in “normal” scenes requires careful and appropriate attention and provides an exciting theoretical field.

We also hope that it shows that an analytical study can have concrete contributions to the efficient resolution of practical problems. Our theoretical work has allowed us to develop new efficient solutions, which are rather simpler than existing methods, even though they are based on a complex multidimensional framework. We believe that the understanding offered by a theoretical study permits the development of solutions under a different angle.

APPENDIX A

The 3D Visibility Complex

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

Table 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,

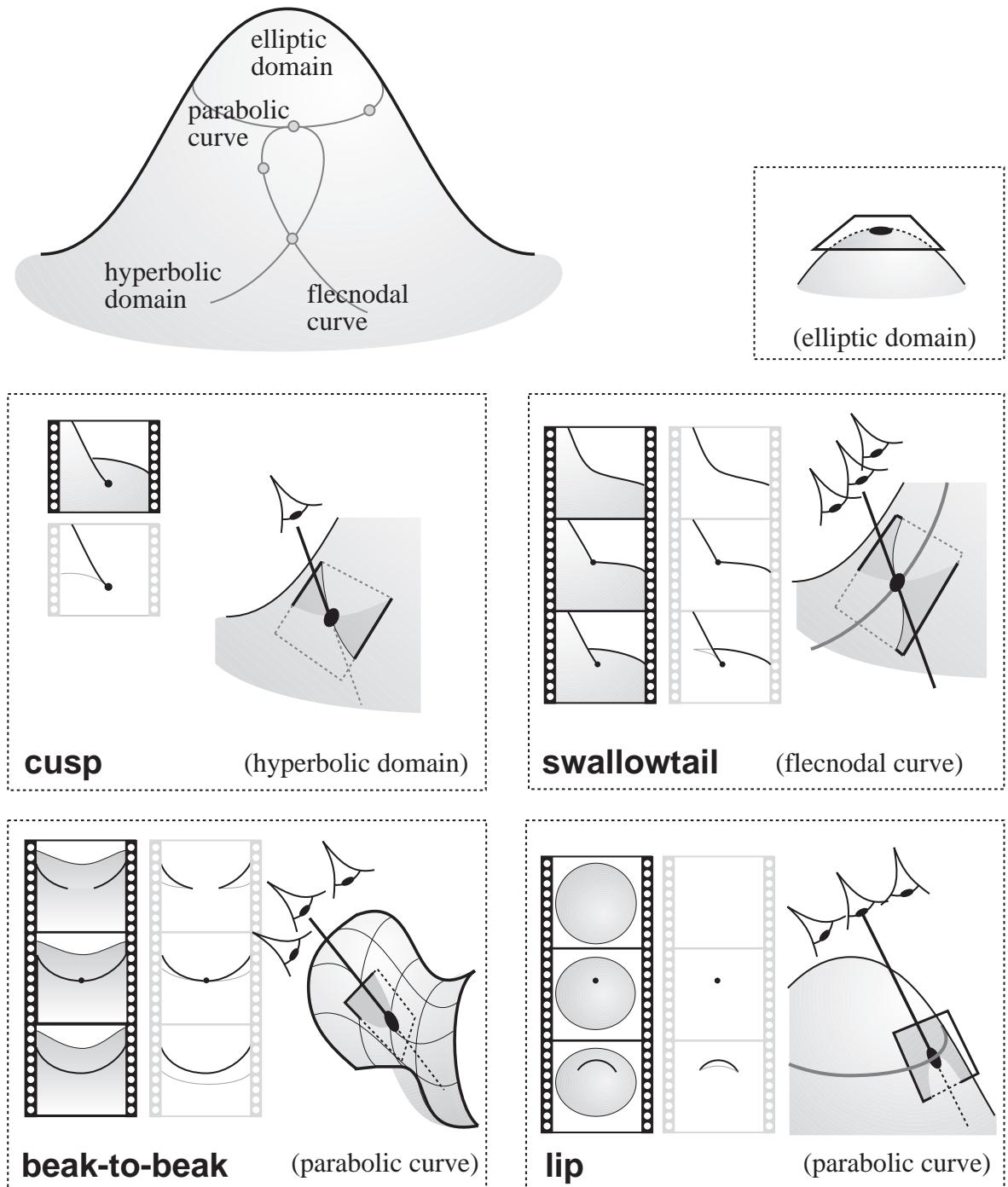


Figure A.1: Singularities of a general smooth surface (adapted from [Pet92]). We represent the views both for segment-visibility (black frames) and line visibility (grey-frames, hidden-lines are in light-grey).

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.

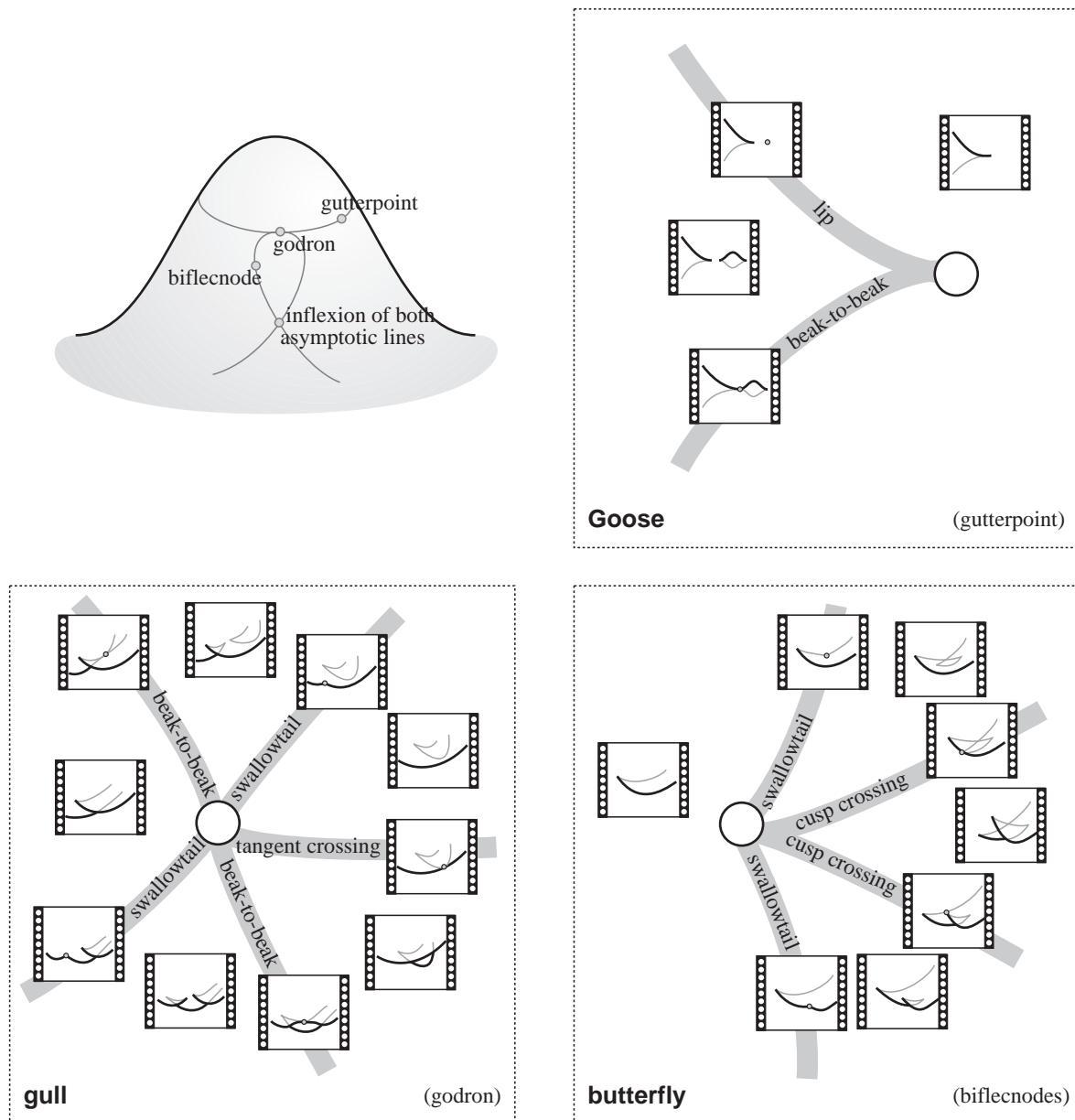


Figure A.2: Singularities of concave surfaces. Codimension 4 singularities (adapted from [Pet92]).

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:

- 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.

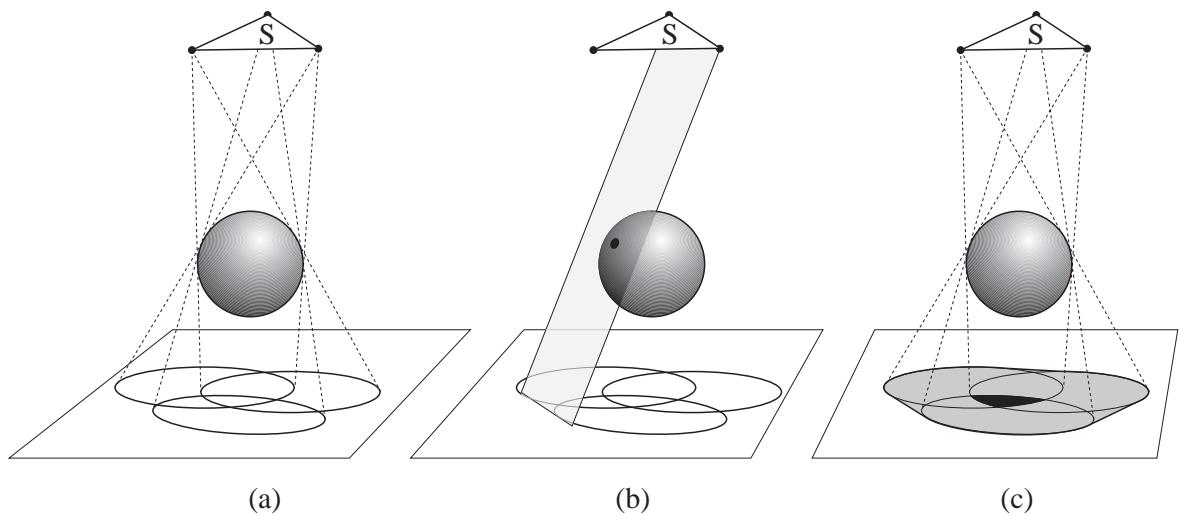


Figure A.3: Discontinuity mesh of a curved object. (a) $T + V$ events. (b) A $T + +T$ event. (c) Complete discontinuity mesh.

APPENDIX B

The Visibility Skeleton

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.

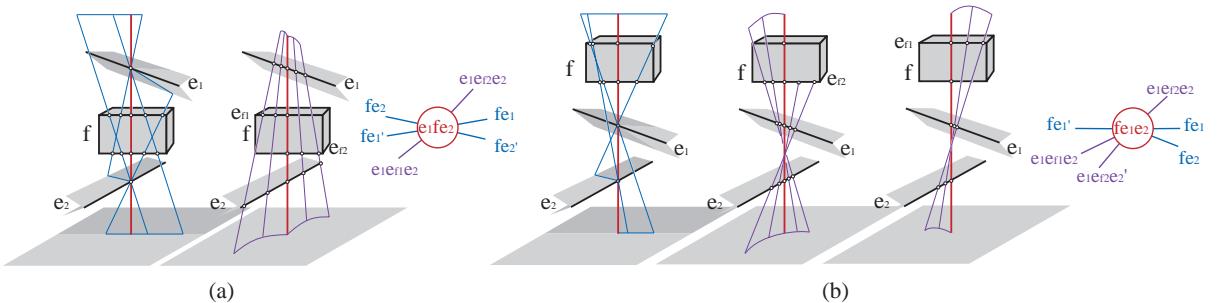


Figure B.1: An EFE node.

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.

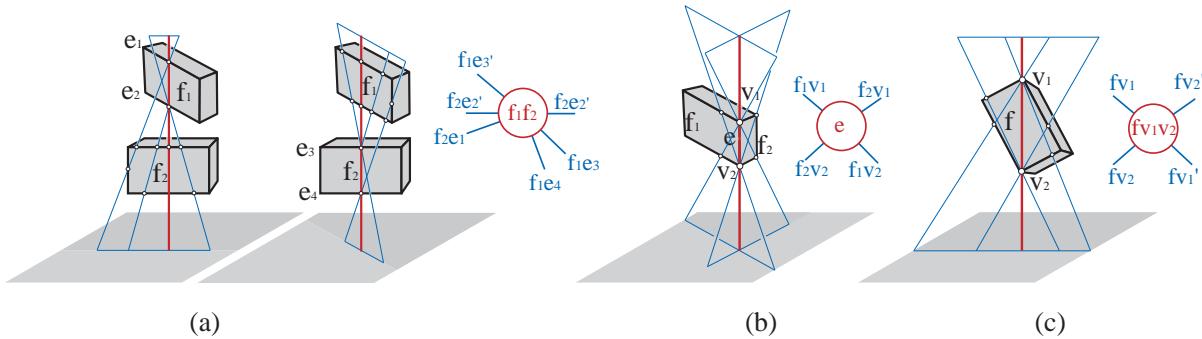


Figure B.2: (a) A FF node, (b) an Fe node and (c) and Fvv node.

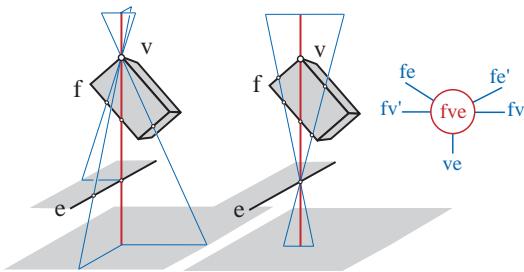


Figure B.3: A FvE node.

Consider the infinitesimal motion $d\vec{\epsilon}_1$ 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 $d\vec{\epsilon}_1$ on e_1 corresponds to a rotation of $\alpha = \frac{\vec{\epsilon}_1 \cdot \vec{n}}{d_1}$ of the plane around e_2 . Symmetrically, this rotation corresponds to the motion $d\vec{\epsilon}_3$ on e_3 and we have $\alpha = \frac{d\vec{\epsilon}_3 \cdot \vec{n}}{d_3}$, by angle equality. Thus, $d\vec{\epsilon}_3 = \vec{\epsilon}_3 \frac{d_3 d\vec{\epsilon}_1 \cdot \vec{n}}{d_1 d\vec{\epsilon}_3 \cdot \vec{n}}$.

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

The sign of $(\vec{e}_4 \times \vec{e}_4) \cdot \vec{\text{node}}$ 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).

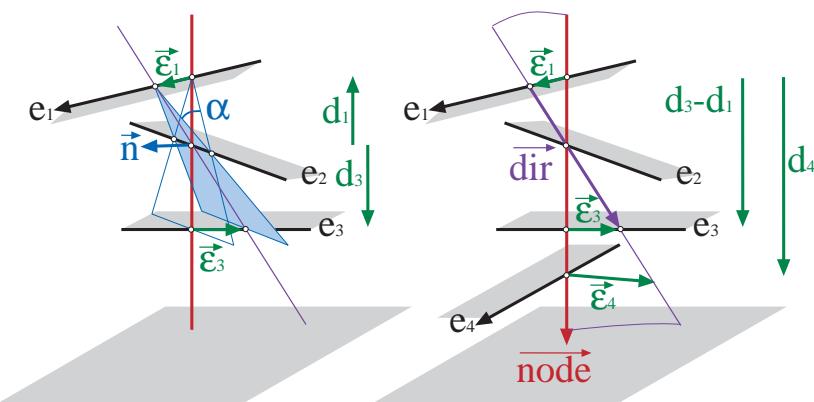


Figure B.4: Determining the direction of an E4 node insertion.

Node	Adjacent arc	Start or End Criterion
v_1v_2	v_1e_3	$v_1 == startV(e)$
ve_1e_2	ve_2 $e_3e_1e_2$ $e_2e_1e_3$	$(\vec{e}_2 \times \vec{e}_1).\vec{n}ode > 0$ $v == startV(e_3)$ $\vec{n} = \vec{normal}(v, \vec{e}_2)$ $\vec{e}_3.\vec{n} * \vec{e}_1.\vec{n} > 0$
e_1ve_2	ve_2 $e_1e_3e_2$	$(\vec{e}_1 \times \vec{e}_2).\vec{n}ode > 0$ $\vec{n} = \vec{normal}(v, \vec{e}_2)$ $\vec{e}_3.\vec{n} * \vec{e}_1.\vec{n} > 0$
$e_1e_2e_3e_4$	$e_1e_2e_3$	$\vec{n} = \vec{normal}(\vec{e}_2, \vec{n}ode);$ $\vec{e}_3 = \vec{e}_3 \frac{d_3 \vec{e}_1 \cdot \vec{n}}{d_1 \vec{e}_3 \cdot \vec{n}}$ $\vec{e}_4 = \vec{e}_1 + \frac{d_4}{d_3 - d_1} (\vec{e}_3 - \vec{e}_1)$ $(\vec{e}_4 \times \vec{e}_4).\vec{n}ode > 0$
e_1fe_2	fe_1 $e_1e_f1e_2$	$\vec{e}_2.\vec{normal}(f) > 0$ $\vec{e}_1.\vec{normal}(f) > 0$
fe_1e_2	fe_2 $e_2e_1e_{f1}$	$\vec{e}_1.\vec{normal}(f) > 0$ $\vec{n} = \vec{normal}(\vec{n}ode, \vec{e}_1)$ $\vec{n}.\vec{e}_2 * \vec{n}.\vec{e}_{f1} > 0$
fve	fv ve	$\vec{e}.\vec{normal}(f) > 0$ $\vec{e}.\vec{normal}(f) > 0$

Table 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.

APPENDIX C

Efficient extended projection using OpenGL

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. Unfortunately, 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
```

Figure C.1: Efficient OpenGL implementation of blocker projection.

APPENDIX 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 *Grassmann manifold* or *Klein quadric* is defined by

$$\Pi_\ell \odot \Pi_\ell = 0$$

APPENDIX 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 library
<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/>

CONTENTS

Introduction	7
Context	7
Motivation	8
Contributions	12
Thesis structure	13
 I Contributions	15
 1 Previous work	17
1 Visibility problems	18
1.1 Computer graphics	18
1.2 Computer Vision	18
1.3 Robotics	18
2 On-line occlusion culling	19
2.1 Hierarchical z-buffer and Hierarchical occlusion maps	19
2.2 Convex occluders and visual events	19
2.3 Occluder shadow footprints	20
3 Shafts and portal sequences	20
3.1 Shaft culling	20
3.2 Architectural environments and portals	21
3.3 Off-line occlusion culling for convex occluders	22
4 Umbra and penumbra boundaries	22
4.1 Umbra and penumbra caused by a convex blocker	23
4.2 Discontinuity meshing	23
4.3 Complete discontinuity meshing with backprojections	23
5 The Aspect graph	24
6 Line space	25
6.1 Discrete approaches	25
6.2 Plücker space	25
6.3 The asp	26
6.4 The 2D visibility Complex	26
7 Dynamic scenes	27
7.1 Shafts and motion volumes	27

7.2	BSP trees	27
7.3	4D ray-tracing	27
7.4	Aspect graph and discontinuity meshing	27
8	Discussion	27
2	The 3D Visibility Complex	29
1	Introduction to the 3D Visibility Complex	30
1.1	Duality	30
1.2	Tangency volumes	31
1.3	Bitangents	33
1.4	Tritangents	34
1.5	Tangent crossing	35
1.6	The 3D Visibility Complex	36
2	A definition for scenes of polygons and smooth objects	37
2.1	Critical segments	37
2.2	The 3D Visibility Complex for scenes of polygons and smooth objects	38
2.3	Probabilistic complexity	41
3	Extension to temporal visibility	44
3.1	Temporal visual event	44
3.2	The Temporal Visibility Complex	44
4	Output-sensitive sweep	46
4.1	Sweeping the initial slice	46
4.2	Principle of the φ sweep	47
4.3	Regular 0-faces	48
4.4	Irregular 0-faces	49
4.5	Non monotonic 1-faces	49
4.6	Complexity of the algorithm	50
5	Applications of the approach	50
5.1	View computation	50
5.2	Form-factors	51
5.3	Computer vision and robotics	51
5.4	Umbra and penumbra	52
6	Related approaches	52
6.1	The aspect graph	52
6.2	The asp	53
6.3	Plenoptic function, lumigraph and light-field	53
6.4	Plücker space	53
6.5	The 2D visibility complex	54
7	Conclusions	54
7.1	Summary	54
7.2	Discussion	54
7.3	Future work	55
3	The Visibility Skeleton	57
1	Motivation	57
2	The Visibility Skeleton	58
2.1	Visual events	58
2.2	The 3D visibility complex, the asp and the visibility skeleton	60
2.3	Catalogue of visual events and their adjacencies	60
3	Data structure	62
3.1	Initial data structure	62
3.2	Reducing memory requirements	63
4	Construction algorithm	64
4.1	Finding nodes	64
4.2	Creating the arcs	67

4.3	Treating degenerate configurations	68
5	Implementation and first applications	69
5.1	Implementation and construction statistics	69
5.2	Point-to-area form-factor for vertices	69
5.3	Global and on-the-fly discontinuity meshing	71
5.4	Exact blocker lists, occlusion detection	73
6	Dealing with spatial complexity: on-demand construction	73
7	Dynamic scenes	74
7.1	Temporal visual events	74
7.2	Temporal visual event detection	75
7.3	Potential applications: form factor maintenance	75
8	Conclusions	75
8.1	Summary	75
8.2	Discussion and future work	76
4	Visibility Driven Hierarchical Radiosity	79
1	Motivation and previous work	81
1.1	Hierarchical radiosity	81
1.2	Accurate visibility and image quality	81
1.3	Visibility-based refinement strategies for radiosity	81
1.4	Perceptually based refinement	82
1.5	Discussion	83
2	Irregular hierarchical triangulations and lazy wavelets	83
2.1	Hierarchical triangulation	84
2.2	Piecewise linear multiresolution representation	85
3	Visibility-driven hierarchical radiosity: Algorithm and data structures	87
3.1	Algorithm outline	87
3.2	Link data structures and form-factors	88
4	Link refinement	91
4.1	Refinement overview	91
4.2	Source and receiver refinement	91
4.3	Visibility updates	92
4.4	Alternative visibility updates	94
4.5	Treating degeneracies	94
5	Polygon subdivision	94
5.1	Perceptual just noticeable difference	94
5.2	Polygon subdivision	95
5.3	Maxima insertion	96
5.4	Refinement criterion	96
6	Implementation and results	98
6.1	Implementation	98
6.2	Results	98
7	Improvements	108
7.1	Taking advantage of invariant backprojections	108
7.2	Time-memory tradeoff	108
8	Discussion	110
8.1	Summary	110
8.2	Limitations	110
8.3	Advantages	110
8.4	Future work	111

5 General Occlusion Culling using Extended Projections	113
1 Extended projections	114
1.1 Principle	114
1.2 Extended projections	114
1.3 Depth	115
1.4 Overview	117
1.5 Implementation choices	117
2 Computation of the extended projections	118
2.1 Occludee Projection	118
2.2 Convex occluders Projection using intersections	119
2.3 Concave occluder slicing	119
3 Improved Projection of occludees	120
3.1 Some properties of umbra	120
3.2 2D improved Projection	122
3.3 3D improved Projection	122
4 Occluder reprojection and occlusion sweep	123
4.1 Validity	124
4.2 Reprojection	125
4.3 Occlusion sweep	125
5 System	126
5.1 Occluder selection	127
5.2 Choice of the projection plane	127
5.3 Adaptive preprocess	128
5.4 Dynamic scenes	129
5.5 On-line rendering	129
6 Implementation and results	129
6.1 Implementation	129
6.2 Results	130
7 Discussion	130
7.1 Summary	130
7.2 Discussion	131
7.3 Future work	133

II A Multidisciplinary Survey of Visibility 137

6 Introduction	139
1 Spirit of the survey	139
2 Flaws and bias	140
3 Structure	140
7 Visibility problems	141
1 Computer Graphics	141
1.1 Hidden surface removal	141
1.2 Shadow computation	142
1.3 Occlusion culling	142
1.4 Global Illumination	143
1.5 Ray-tracing and Monte-Carlo techniques	144
1.6 Radiosity	145
1.7 Image-based modeling and rendering	146
1.8 Good viewpoint selection	147
2 Computer Vision	147
2.1 Model-based object recognition	147
2.2 Object reconstruction by contour intersection	148
2.3 Sensor placement for known geometry	149

2.4	Good viewpoints for object exploration	150
3	Robotics	150
3.1	Motion planning	151
3.2	Visibility based pursuit-evasion	151
3.3	Self-localisation	152
4	Computational Geometry	153
4.1	Hidden surface removal	153
4.2	Ray-shooting and lines in space	153
4.3	Art galleries	154
4.4	2D visibility graphs	154
5	Astronomy	155
5.1	Eclipses	155
5.2	Sundials	155
6	Summary	156
8	Preliminaries	157
1	Spaces and algorithm classification	157
1.1	Image-space	157
1.2	Object-space	158
1.3	Viewpoint-space	158
1.4	Line-space	158
1.5	Discussion	159
2	Visual events, singularities	160
2.1	Visual events	160
2.2	Singularity theory	162
3	2D <i>versus</i> 3D Visibility	163
9	The classics of hidden part removal	165
1	Hidden-Line Removal	166
1.1	Robert	166
1.2	Appel	166
1.3	Curved objects	166
2	Exact area-subdivision	166
2.1	Weiler-Atherton	166
2.2	Application to form factors	166
2.3	Mulmuley	167
2.4	Curved objects	167
3	Adaptive subdivision	167
4	Depth order and the painter's algorithm	167
4.1	Newell Newell and Sancha	167
4.2	Priority list preprocessing	168
4.3	Layer ordering for image-based rendering	168
5	The z-buffer	169
5.1	Z-buffer	169
5.2	A-buffer	169
6	Scan-line	170
6.1	Scan-line rendering	170
6.2	Shadows	170
7	Ray-casting	170
8	Sweep of the visibility map	171

10 Object-Space	173
1 Space partitioning	173
1.1 Ray-tracing acceleration using a hierarchy of bounding volumes	174
1.2 Ray-tracing acceleration using a spatial subdivision	174
1.3 Volumetric visibility	174
1.4 BSP trees	175
1.5 Cylindrical decomposition	176
2 Path planning using the visibility graph	176
2.1 Path planning	176
2.2 Visibility graph construction	177
2.3 Extensions to non-holonomic visibility	177
3 The Visual Hull	177
4 Shadows volumes and beams	179
4.1 Shadow volumes	179
4.2 Shadow volume BSP	180
4.3 Beam-tracing and bundles of rays	180
4.4 Occlusion culling from a point	182
4.5 Best-next-view	182
5 Area light sources	183
5.1 Limits of umbra and penumbra	183
5.2 BSP shadow volumes for area light sources	184
5.3 Discontinuity meshing	184
5.4 Linear time construction of umbra volumes	184
5.5 Viewpoint constraints	185
5.6 Light from shadows	187
6 Shafts	187
6.1 Shaft culling	187
6.2 Use of a dual space	188
6.3 Occlusion culling from a volume	188
7 Visibility propagation through portals	188
7.1 Visibility computation	188
7.2 Applications	189
11 Image-Space	191
1 Projection methods	191
1.1 Shadow projection on a sphere	191
1.2 Area light sources	192
1.3 Extended projections	192
2 Advanced z-buffer techniques	192
2.1 Shadow maps	192
2.2 Ray-tracing optimization using item buffers	193
2.3 The hemicube	194
2.4 Sound occlusion and non-binary visibility	194
3 Hierarchical z-buffer	195
4 Occluder shadow footprints	196
5 Epipolar rendering	196
6 Soft shadows using convolution	197
7 Shadow coherence in image-space	197
12 Viewpoint-Space	199
1 Aspect graph	199
1.1 Approximate aspect graph	200
1.2 Convex polyhedra	200
1.3 General polyhedra	201
1.4 Curved objects	203

1.5	Use of the aspect graph	204
2	Other viewpoint-space partitioning methods	204
2.1	Robot Localisation	204
2.2	Visibility based pursuit-evasion	205
2.3	Discontinuity meshing with backprojections	206
2.4	Output-sensitive discontinuity meshing	207
3	Viewpoint optimization	208
3.1	Art galleries	208
3.2	Viewpoint optimization	208
3.3	Local optimization and target tracking	209
4	Frame-to-frame coherence	210
4.1	Coherence constraints	210
4.2	Occlusion culling with visual events	211
13	Line-Space	213
1	Line-space partition	213
1.1	The Asp	213
1.2	The 2D Visibility Complex	214
1.3	The 3D Visibility Complex	215
1.4	Ray-classification	215
1.5	Multidimensional image-based approaches	217
2	Graphs in line-space	217
2.1	The Visibility Skeleton	217
2.2	Skewed projection	218
3	Plücker coordinates	219
3.1	Introduction to Plücker coordinates	219
3.2	Use in computational geometry	220
3.3	Implementations in computer graphics	221
4	Stochastic approaches	222
4.1	Integral geometry	222
4.2	Computation of form factors using ray-casting	222
4.3	Global Monte-Carlo radiosity	222
4.4	Transillumination plane	222
14	Advanced issues	225
1	Scale and precision	225
1.1	Hierarchical radiosity: a paradigm for refinement	225
1.2	Other shadow refinement approaches	226
1.3	Perception	226
1.4	Explicitly modeling scale	227
1.5	Image-space simplification for discontinuity meshing	228
2	Dynamic scenes	228
2.1	Swept and motion volumes	229
2.2	4D methods	229
2.3	BSP	229
2.4	Aspect graph for objects with moving parts	229
2.5	Discontinuity mesh update	230
2.6	Temporal visual events and the visibility skeleton	230
15	Conclusions of the survey	231
1	Summary	231
2	Discussion	232

Conclusions	235
Contributions	235
Future work	237
To conclude	237
A The 3D Visibility Complex	239
1 Catalogue of adjacencies	239
2 Concave objects	239
3 Discontinuity meshing for curved objects	242
B The Visibility Skeleton	245
1 Complete Catalogue of Face Adjacencies	245
2 Details of the Construction to find the Orientation of Arcs	245
C Efficient extended projection using Open GL	249
D Some Notions in Line Space	251
E Online Ressources	253
1 General ressources	253
2 Available code.	253
Contents	255
List of Figures	263
Index	269
References	275

LIST OF FIGURES

1.1	Occlusion culling using occlusion maps	19
1.2	Occlusion culling and visual events	20
1.3	Occlusion culling using shadow footprints	21
1.4	Shaft culling	21
1.5	Visibility propagation through portals	22
1.6	Off-line occlusion culling for convex occluders	22
1.7	Umbra and penumbra boundaries	23
1.8	EEE shadow boundary	24
1.9	EV visual event	25
1.10	2D equivalent of ray-classification	25
1.11	2D visibility complex	26
 2.1	 Maximal free segment	 30
2.2	Duality and tangency volume	31
2.3	Degrees of freedom of a tangent line	31
2.4	Line and segment visibility	32
2.5	Dual arrangement for two spheres.	33
2.6	Auxiliary Complex for two spheres.	34
2.7	φ -slice of the faces of the visibility complex	35
2.8	Visibility Complex of a scene of three spheres.	36
2.9	Zoomed view of the φ -slice $\varphi = 0$	36
2.10	Tritangency visual event.	37
2.11	Critical segments: local and multilocal events	38
2.12	Bitangents adjacent to a tritangent face	39
2.13	Construction of a segment of a $T + T$ 2-face adjacent to a T 3-face.	40
2.14	Lower and upper bound scenes for the visibility complex	40
2.15	Probabilistic complexity of $T + T + T$ events	41
2.16	Probability distribution of distances of pairs of points inside a sphere	43
2.17	Temporal visual event and Temporal Visibility Complex	45
2.18	Sketch of sweep direction	46
2.19	Sweep of the first vertex of a polyhedron	47
2.20	Fusion-restriction of a view around edges when a vertex is swept	48
2.21	$T + T + T$ critical line sets adjacent to a $T + T + T + T$ critical line.	48
2.22	Irregular $V + V$ critical segment	49
2.23	Different sweep-events represented in the dual space	50

2.24 View around a point with the visibility complex	51
2.25 Traversal of a φ -slice to compute the view around a point	52
3.1 Example of queries using the skeleton	58
3.2 <i>EV</i> visual event and graph structure	59
3.3 adjacencies of a <i>VEE</i> extremal stabbing line	60
3.4 Visual events	61
3.5 <i>VV</i> and <i>EVE</i> extremal stabbing lines and their adjacencies	61
3.6 <i>E4</i> extremal stabbing lines and its adjacencies	62
3.7 Basic Visibility Skeleton Structure.	63
3.8 Summary of the visibility skeleton structures	63
3.9 <i>VEE</i> and <i>E4</i> computation	65
3.10 Enumeration of Potential <i>VEE</i> and <i>E4</i> Nodes.	66
3.11 Acceleration using a regular grid and hourglasses	66
3.12 Finding Face Nodes.	67
3.13 Node Creation.	67
3.14 Example of node insertions	68
3.15 Graphic of the growth in memory and computation time	70
3.16 View queries	71
3.17 Discontinuity mesh queries	71
3.18 Occluder queries	72
3.19 Dynamic update of the skeleton	74
3.20 General definition of extremal stabbing lines	76
4.1 Images computed using our new hierarchical radiosity algorithm	80
4.2 Hierarchical Triangulation Construction	84
4.3 The “matching” constraint for the Hierarchical Triangulation	84
4.4 Principle of our multiresolution representation	85
4.5 Consistent multiresolution representation with lazy wavelets	86
4.6 Visibility driven hierarchical radiosity	87
4.7 Point-polygon link	88
4.8 Geometry for the calculation of a form factor	89
4.9 Example of Form-Factor computation	89
4.10 Negative links	90
4.11 Gather and push-pull	90
4.12 Polygon-polygon link	91
4.13 Receiver refinement	92
4.14 Source visibility updates	92
4.15 Receiver visibility updates	93
4.16 Receiver refinement with visibility events	93
4.17 Degeneracy due to discontinuity meshing	94
4.18 Effect of Ward’s tone-mapping on the same scene with different light source intensities	95
4.19 Polygon subdivision	96
4.20 Refinement criterion geometry	97
4.21 Initial Desk Scene	102
4.22 Many Lights scene	103
4.23 Hierarchical Radiosity comparative results for Many Lights scene	103
4.24 Indirect lighting scene	104
4.25 Indirect lighting scene	105
4.26 Hierarchical Radiosity comparative results for Indirect Lighting scene	106
4.27 Village scene	107
4.28 Taking advantage of invariant backprojections	108
4.29 Time-memory tradeoff	109
4.30 Optimizing a triangulation	112

5.1	Extended projection	115
5.2	Occluder fusion with extended projections	116
5.3	Inverse of a point in an extended projection and conservative depth definition	117
5.4	Occludee extended projection	118
5.5	Convex occluder extended projection	119
5.6	Configuration where initial extended projections is too restrictive	120
5.7	Case of a concave planar occluder	121
5.8	Improved extended projection in 2D	122
5.9	Improved extended projection in 3D	123
5.10	Extended occlusion map reprojection	124
5.11	Umbra of the subset of an umbra	124
5.12	Occlusion sweep	126
5.13	Choice of the projection plane	127
5.14	Example of projection planes for a cell of the city model	128
5.15	Results of our algorithm for a city scene	131
5.16	Comparative results	132
5.17	The sweeping process	132
5.18	Portal extended projection and concave occluder Projection using a union of convex	133
5.19	Concave occluder mesh Projection	134
5.20	Successive convolutions and remapping	136
7.1	Study of shadows by Leonardo da Vinci and Johann Heinrich Lambert	142
7.2	Soft shadow	143
7.3	BRDF	144
7.4	Global illumination	144
7.5	Principle of recursive ray-tracing	145
7.6	Radiosity	145
7.7	View warping	146
7.8	Model-based object recognition	148
7.9	Object reconstruction by contour intersection	148
7.10	Viewpoint optimization for a screwdriver	149
7.11	Viewpoint optimization for a peg insertion	149
7.12	Object acquisition using a laser plane	150
7.13	Motion planning on a floorplan	151
7.14	Visibility based pursuit-evasion	152
7.15	2D Robot localisation	152
7.16	View with size $O(n^2)$	153
7.17	Line stabbing a set of convex polygons in 3D space	154
7.18	Eclipse by Purbach and Halley	155
7.19	Solar and lunar eclipse	155
7.20	Sundials	156
8.1	orthographic viewpoint space	158
8.2	Line parameterisations	159
8.3	EV visual event	160
8.4	Locus an EV visual event	160
8.5	A EEE visual event	161
8.6	Locus of a EEE visual event	161
8.7	Line drawing of the view of a torus	162
8.8	Tangent crossing singularity	163
8.9	Disappearance of a cusp at a swallowtail singularity	163
8.10	Opaque and semi-transparent visual singularities	164
9.1	Classic example of a cycle in depth order	168
9.2	<i>A priori</i> depth order	168

9.3 A buffer	169
9.4 Ray-casting by Dürer	171
9.5 Sweep of a visibility map	172
10.1 A 2D analogy of ray-tracing acceleration	175
10.2 2D BSP tree	176
10.3 Path planning using the visibility graph	177
10.4 Shadow for non-holonomic path-planning	178
10.5 Visual hull	178
10.6 Shadow volume	179
10.7 Construction of a shadow by Dürer	180
10.8 2D equivalent of shadow BSP	181
10.9 Beam tracing	181
10.10 Occlusion culling with large occluders	182
10.11 Occlusion culling using image-space portals	183
10.12 Volume of occlusion for model acquisition	183
10.13 Umbra and penumbra of a convex blocker	184
10.14 Penumbra by da Vinci	185
10.15 Global illumination simulation with discontinuity meshing	186
10.16 Linear time construction of a penumbra volume	186
10.17 Sketching shadows	187
10.18 Shaft culling	187
10.19 Visibility computations in architectural environments	189
10.20 Conservative visibility propagation through arbitrary portals	189
11.1 Shadow map principle	192
11.2 Light buffer and hemicube	194
11.3 Non binary visibility for sound propagation	195
11.4 Hierarchical z-buffer	195
11.5 Occluder shadow footprints	196
11.6 Epipolar geometry	197
11.7 Soft shadows computation using convolution	197
11.8 Soft shadows computed using convolutions	198
12.1 Quasi uniform subdivision of the viewing sphere starting with an icosahedron	200
12.2 Aspect graph of a convex cube	201
12.3 False event	202
12.4 Aspect graph of a L-shaped polyhedron under orthographic projection	202
12.5 Partition of orthographic viewpoint space for a dimple object	203
12.6 Scene partition for robot self-localisation	205
12.7 Pursuit-Evasion strategy	205
12.8 Complete discontinuity meshing: example of an <i>EV</i>	206
12.9 Complete discontinuity meshing: example of a <i>EEE</i>	206
12.10 Complete discontinuity mesh of a scene	207
12.11 Art gallery	208
12.12 Planning of a stereo-sensor	209
12.13 Tracking of a mobile target by an observer	210
12.14 Shortest path for the target to escape	211
12.15 Occlusion culling and visual events	211
13.1 Slice of the <i>asp</i> for $\phi = 0$	214
13.2 Topology of maximal free segments	215
13.3 A face of the visibility complex	215
13.4 Ray classification	216
13.5 Image computed using ray classification	216

13.6 Adjacencies of critical lines and extremal stabbing lines	218
13.7 Four lines in general position are stabbed by two lines	218
13.8 Skewed projection.	219
13.9 Plücker space and line orientation	220
13.10 Lines stabbing a triangle in Plücker space	220
13.11 <i>EEE</i> in Plücker spaces	221
13.12 Antipenumbra	221
13.13 Global Monte-Carlo radiosity	223
13.14 Transillumination plane	223
14.1 Hierarchical radiosity	226
14.2 Shadow refinement	227
14.3 Scale-space aspect graph	227
14.4 Finite resolution aspect graph	228
14.5 Dynamic update of limits of shadow	230
A.1 Singularities of a general smooth surface	241
A.2 Singularities of concave surfaces	242
A.3 Discontinuity mesh of a curved object	243
B.1 An <i>EFE</i> node.	245
B.2 <i>FF</i> , <i>Fe</i> and <i>Fvv</i> nodes	246
B.3 A <i>FvE</i> node.	246
B.4 Determining the direction of an E4 node insertion.	246
C.1 Efficient OpenGL implementation of blocker projection.	250

LIST OF TABLES

2.1	Elements of the visibility complex	37
2.2	Faces of the visibility complex of polygons and smooth objects.	39
2.3	Vertices of the temporal visibility complex	46
3.1	Construction statistics	70
4.1	Importance of the form-factor accuracy	99
4.2	Timing and memory results for the test scenes	99
4.3	Comparative Timing and memory results	100
15.1	Recapitulation of the techniques presented by field and by space.	233
A.1	Adjacencies of the faces of the visibility complex of polygons and smooth objects.	240
B.1	Start/end criteria for adjacent arcs	247

INDEX

2D vs. 3D 163

A

a priori order 168
 A-buffer 169, 181, 193
 accidental configuration 229
 accidental viewpoint 162, 228
 active vision (*see sensor placement*), 149
 adaptive subdivision 167, 194
 Alias Wavefront Maya 170, 193
 aliasing 142, 146, 169, 180, 181, 191, 193, 194,
 200
 amortized beam-tracing 181
 antialiasing 168–170, 193
 antipenumbra 221
 architectural scene (*see portal*), 188
 arrangement 200
 art gallery 151, (*see sensor placement*), 154, 208
 asp 213, 215
 aspect 147, 199
 aspect graph 148, 199–204
 approximate aspect graph 200, 208
 asp 213
 convex polyhedra 200
 curved objects 203
 finite resolution 227
 general polyhedra 201
 local and linearized 211
 moving parts 229
 scale-space 227
 use 204
 assembly planning 151

B

backprojection 206–207

beam 179–183
 beam-tracing 180
 amortized 181
 antialiasing 181
 bidirectional 181
 BSP 180
 caustics 181
 real-time 181
 sound propagation 181
 best-next-view 150, 182
 bifurcation (*see singularity*), 162
 binary space partitioning tree (*see BSP*), 175
 bitangent 161, 164, 176, 185, 215
 bounding volumes 174
 BRDF 143
 BSP 168, 175
 beam tracing 180
 cylindrical BSP 175, 229
 dynamic 229
 feudal priority 175
 modeling 175, 182
 occlusion culling 175, 182
 Quake 176
 shadow 180, 191
 soft shadow 184
 bundle of rays 180

C

camera placement (*see sensor placement*), 147
 catastrophe (*see singularity*), 162
 caustics 181
 characteristic view (*see aspect graph*), 199
 classification of lines 154
 cluster 174, 226
 computational geometry 153–154
 computer graphics 141–147
 computer vision 147–150

- cone-tracing 180
 configuration space 151
 conflict 167
 conservative 143, 188
 constructive geometry 175
 contour intersection (*see silhouette intersection*), 148
 convolution 197, 226
 correlation 197, 226
 critical line 161, 215, 217, 221
 cross scan-line 170
 curved object
 adaptive subdivision 169, 170
 aspect graph 203
 BSP 175
 pursuit-evasion 206
 shadow 193
 singularity 166
 soft shadow 166
 view computation 167, 169, 170
 visibility complex 214
 visibility graph 176, 214
 visual hull 179
 cylindrical BSP 175, 229
 cylindrical partition 176
- D**
- da Vinci (Leonardo) 142, 184
 degeneracy 163, 166
 depth order 147, 167, 175, 221
 differential geometry 163
 diffraction 144, 194
 diffuse 143, 145
 discontinuity mesh
 moving object 229, 230
 discontinuity meshing (*see soft shadow*), 146, 180, 184, 207, 215, 218, 226
 backprojection 206
 complete 206
 finite resolution 228
 indirect lighting 184, 218
 output-sensitive 207
 Dürer 171, 180
 dynamic scene 228–230
- E**
- epipolar geometry 196, 217
 extended projection 192
 extremal stabbing line 217, 221
- F**
- feature of the shadow 226
- feature-based visibility 226
 feudal priority 175
 finite resolution aspect graph 227
 finite-element 145
 flight simulator 141, 143, 168, 170
 footprint 196
 form factor 145, 166, 194, 204, 207, 215, 218
 analytical 145, 166
 frame-to-frame coherence (*see temporal coherence*), 210
 Fresnel ellipsoid 194
 from factor 222
- G**
- game 147, 176, 182
 gap edge 205
 gasp 182
 general view (*see aspect*), 199
 generator 161
 generic viewpoint 162
 geodesic 177
 geographical information system 140
 geon 204
 GIS 140
 global illumination 143, 217
 global Monte-Carlo radiosity 222, 229
 global visibility 156
 good viewpoint (*see sensor placement*), 147
 graph drawing 147, 204
 graph in line space 217
 Grassman manifold 219
- H**
- hard shadow 142, 157, 158, 179–180
 added to the scene 166, 180
 BSP 180, 191
 ray-tracing 144, 170, 182, 193, 197
 scan-line 170, 191
 shadow volume 179
 shadow-map 192
 hardware 143, 169, 170, 180, 191–195
 heat-transfer 145
 height field 140, 196
 hemicube 194
 hidden-line removal 166
 hidden-part removal 141, 153, 154, 165–171, 175
 hierarchical occlusion map 195
 hierarchical z-buffer 176, 195
 human perception 142, 147, 226
- I**
- image precision 165

image structure graph 200
 image-based modeling and rendering 146, 168, 182,
 196, 209, 217
 importance 222
 inflexion point 161
 integral geometry 222
 interpretation tree 204
 interval arithmetic 166
 invariant 148
 item buffer 193

K

Klein quadric 219

L

Lambert 142
lambertian (*see diffuse*), 143
 laser plane 150
 layer 147, 168
 level of detail 189, 195
 light buffer 193, 216
 light-field 217
 lighting simulation 143
 limb (*see silhouette*), 162
 line classification 154, 218
 line drawing 161, 162, 166
 line parameterization 159, 164, 217–219, 222
 linear light source 193
 localisation (*see self-localisation*), 152
 lumigraph 217

M

matching 147
 maximal free segment 159, 214
 Maya 170, 193
 measure of a set of lines 222
model-based recognition (*see recognition*), 147
 Monte-Carlo 144, 222
 motion planning 151, 154, 176
 NP-complete 176
 pursuit-evasion 151, 205
 target tracking 152, 210
 use of the z-buffer 192
 motion volume 229
 motion-planning 154
 non-holonomic 177

N

network 143, 189
 non-binary visibility 144, 194
 non-holonomic 177

non-photorealistic rendering 141, 142, 166
 NP-complete
 art-gallery 208
 motion-planning 176
 NURBS 167

O

object precision 165
 object recognition (*see recognition*), 147
 object-centered representation 147
 object-precision 153, 166, 170
 occluder fusion 182, 183, 188, 192, 195, 211
 occluder shadow 196
 occluding contour (*see silhouette*), 162
 occlusion culling 142
 BSP 175, 182
 extended projection 192
 from a volume 188
 hierarchical occlusion map 195
 hierarchical z-buffer 195
 line-space 217
 moving object 229
 occluder simplification 179
 offline 188, 192, 217
 online 175, 182, 195, 211
 portal 182, 188
 shadow footprint 196
 shadow volume 182
 temporal coherence 211
 occlusion-free viewpoint 185
 opaque *vs.* transparent 162, 163, 201, 211
 Open GL Optimizer 196
 oracle 226
 output-sensitive 153, 177

P

painter's algorithm 167, 175
 path planning (*see motion planning*), 176
 peak vertex 207
 pencil tracing 181
 penumbra (*see soft shadow*), 142, 183, 185, 221
 perception 226
 Pixar Renderman 170
 pixel-plane 180
 Plücker 219–221
 plenoptic function 217
 portal 182, 188, 221
 potentially visible set 143, 188
 predicate 221
 probability 144, 174, 222
 production rendering 169, 179, 193
 purposive viewpoint adjustment 150

pursuit-evasion 151, 205

Q

Quake 176

quantitative invisibility 166, 167

R

radiosity 145

2D 215

architectural scenes 189

convolution 197

discontinuity mesh 146, 180, 184, 206, 215, 218

form factor 145, 166, 194, 204, 207, 215, 218, 222

global Monte-Carlo 222, 229

hemicube 194

hierarchical radiosity 225

moving object 229

mutual visibility 187, 189, 221

perceptual metric 226

shadow improvement 226

visibility preprocessing 189, 215, 218

volumetric visibility 174

randomized 167

randomized algorithm 167

range image 150

raster 158, 165, 169

ray classification 215, 220, 229

ray-shooting 220

ray-tracing 144, 170, 222

4D 229

antialiasing 170, 180, 181, 193

beam-tracing 180

bounding volume 174

computational geometry 153

cone-tracing 180

depth of field 180, 193

item buffer 193

light-buffer 193

octree 174

pencil tracing 180

production rendering 193

ray-classification 215, 229

ray-surface intersection 170

shadow cache 197

shadow ray 144, 170, 182, 193, 197

shadow volume 179

soft shadow 180, 193, 198

space-time 229

spatial subdivision 174

update 188

ZZ-buffer 193

reachability 177

recognition 152, 199, 204

refinement 197, 225–227

reflection 143, 144, 170, 180, 182

refraction 144, 170, 180

rendering equation 143

Renderman 170

Riemannian geometry 177

robotics 150–153

robustness (*see degeneracy*), 163, 166, 184, 204, 221

S

scale 204, 225–228

scale space 227

scale-space aspect graph 227

scan-line 170, 179, 191, 216

self-localisation 152, 204

sensor placement 147

known geometry 149, 154, 185, 208–210

object exploration 150, 182

separating plane 175, 184, 185, 211

shadow 142, 158

importance in perception 142

shape from shadow 140

sketching 142, 187

shadow BSP 180, 184, 191

shadow cache 197

shadow floodfill 198

shadow footprint 196

shadow map 179, 192, 193

shadow ray 144, 170, 182, 193, 197

shadow volume 158, 179–183

area light source 183

BSP 180, 191

Dürer 180

hardware implementation 180

light scattering 180

occlusion culling 182

ray-tracing 179

scan-line 179

sketching shadow 187

tiling 191

volume of occlusion 182

z-buffer 179

shaft culling 187–189

dual space 188

hierarchical 188

motion volume 229

shape from shadow 140

silhouette 162, 166, 167, 204

non-photorealistic rendering 142, 166

- silhouette intersection 148, 177
 silhouette tree 204
 singularity 162–163, 203
 catalogue 163
 cusp 162
 fold (*see silhouette*), 162, 166
 stability 162
 swallowtail 163
 t-vertex 162, 166, 171
 tangent crossing 162
 view update 162, 166
 sketching shadow 142, 187
 skewed projection 218
 soft shadow 142, 166, 216, 226
 boundary 183–185
 convolution 197, 226
 radiosity 146, 194
 ray-tracing 181, 193, 198
 shadow-map 193, 197
 supersampling 179
 visible part of the source 142, 158, 165, 166, 198
 sound propagation 143, 194
 beam-tracing 181
 space partition 174, 182, 195
 space-time 229
 span 170
 spatial subdivision (*see space partition*), 174
 stab-tree 188
 stabbing line 154, 188, 217, 219
 stable views (*see aspect graph*), 199
 strategy tree 204
 sub-sampling 169, 170, 193
 supporting plane 184
 swallowtail 163
 swath 161
 swept volume 229
- T**
- target tracking 152, 210
 temporal bounding volume 229
 temporal coherence 210–211, 219, 228
 temporal visibility complex 230
 temporal visual event 230
 terrain 140, 196
 texel 175
 tiling 191
 tracking (*see target tracking*), 152
 transillumination plane 222
 transmittance 174
 transparency 169, 170
 transparent *vs.* opaque (*see opaque vs. transparent*), 162
- transversal (*see stabbing line*), 154
 trapezoidal decomposition 167
- U**
- umbra (*see shadow*), 142
- V**
- view classes (*see aspect graph*), 199
 view computation 141, 153
 view graph (*see aspect graph*), 199
 view warping 146, 182, 193, 196
 view-dependent 144
 view-independent 145
 viewability matrix 209
 viewer-centered representation 147, 199
 viewing data 199, (*see aspect graph*), 199
 viewing sphere 158
 viewing sphere tessellation 200, 209, 217
 viewpoint constraint 185
 viewpoint planning (*see sensor placement*), 149
 viewpoint set 209
 viewpoint space partition 199
 visibility complex 214, 215
 dynamic update 230
 visibility culling 143
 visibility event (*see visual event*), 160
 visibility graph 151, 154, 176, 179
 construction 177, 214
 visibility map 165, 171
 visibility skeleton 217
 dynamic update 230
 vision 147–150
 visual event 160–162
 2D 161, 164
 aspect graph 199
 discontinuity meshing 184, 206, 207
 EEE 161, 178, 184, 201, 206, 218, 219
 EV 160, 183–185, 206, 210, 217
 face 161
 generator 161
 line-space 217, 221
 moving object 229
 occlusion-free viewpoint 160, 185
 shadow boundary 160, 179, 184, 206, 207
 temporal 229, 230
 view update 160, 210, 211
 visual hull 178
 visual hull 148, 178
 visual potential (*see aspect graph*), 199
 volume of occlusion 182
 volumetric visibility 174, 226
 voxel 174

W

walkthrough 143, 145, 189
 volumetric visibility 175
warping 146, 182, 193, 196
window (Warnock) 167

Z

z-buffer 169, 192–196
ZZ-buffer 193

REFERENCES

- [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/>. (cited on pages 27, 229)
- [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. (cited on pages 18, 150)
- [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. (cited on pages 53, 217)
- [Abr96] Michael Abrash. Inside quake: Visible-surface determination. *Dr. Dobb's Journal of Software Tools*, 21(??):41–??, January/February 1996. (cited on page 176)
- [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>. (cited on pages 131, 195)
- [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. (cited on pages 175, 176, 229)
- [Aga84] P. K. Agarwal. *The art gallery problem: Its Variations, applications, and algorithmic aspects*. PhD thesis, Johns Hopkins University, Baltimore, 1984. (cited on page 208)
- [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. (cited on pages 175, 176)
- [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>. (cited on page 194)
- [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. (cited on pages 21, 188, 189)

- [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. (cited on pages 11, 25, 73, 74, 215)
- [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. (cited on pages 145, 174)
- [Ama84] John Amanatides. Ray tracing with cones. *Computer Graphics*, 18(3):129–135, July 1984. (cited on page 181)
- [App67] Arthur Appel. The notion of quantitative invisibility and the machine rendering of solids. *Proc. ACM Natl. Mtg.*, page 387, 1967. (cited on page 166)
- [App68] Arthur Appel. Some techniques for shading machine renderings of solids. In *AFIPS 1968 Spring Joint Computer Conf.*, volume 32, pages 37–45, 1968. (cited on pages 144, 170)
- [ARB90] John M. Airey, John H. Rohlff, 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. (cited on pages 21, 111, 133, 189)
- [Arn69] V.I. Arnold. Singularities of smooth mappings. *Russian Mathematical Surveys*, pages 3–44, 1969. (cited on page 162)
- [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. (cited on pages 111, 166, 184)
- [Ase92] Frederic Asensio. A hierarchical ray-casting algorithm for radiosity shadows. *Third Eurographics Workshop on Rendering*, pages 179–188, May 1992. (cited on page 226)
- [Ash94] Ian Ashdown. *Radiosity: A Programmer's Perspective*. John Wiley & Sons, New York, NY, 1994. (cited on page 145)
- [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. (cited on pages 111, 226)
- [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. (cited on page 174)
- [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. (cited on pages 166, 180)
- [Bax95] Michael Baxandall. *Shadows and enlightenment*. Yale University Press, London, UK, 1995. (Ombres et Lumières, Gallimard). (cited on page 142)
- [BB82] D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice Hall, Englewood Cliffs, 2 edition, 1982. (cited on page 147)
- [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. (cited on page 179)
- [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>. (cited on page 185)
- [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. (cited on page 219)
- [Ber86] P. Bergeron. A general version of Crow's shadow volumes. *IEEE Computer Graphics and Applications*, 6(9):17–28, 1986. (cited on page 179)

- [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. (cited on pages 18, 141, 153, 168, 171)
- [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. (cited on page 182)
- [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. (cited on page 222)
- [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. (cited on page 204)
- [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. (cited on pages 20, 182)
- [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. (cited on page 191)
- [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. (cited on page 170)
- [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. (cited on pages 111, 226)
- [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*. (cited on page 196)
- [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. (cited on pages 81, 222)
- [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. (cited on page 170)
- [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. (cited on page 174)
- [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. (cited on page 82)
- [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. (cited on page 188)
- [Bre92] M. Quinn Brewster. *Thermal Radiative Transfer & Properties*. John Wiley & Sons, New York, 1992. (cited on page 145)
- [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. (cited on pages 70, 88, 96, 166, 194)
- [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. (cited on pages 27, 229)

- [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. (cited on pages 27, 229)
- [BY98] Jean-Daniel Boissonnat and Mariette Yvinec. *Algorithmic geometry*. Cambridge University Press, 1998. (cited on pages 38, 153, 200)
- [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. (cited on page 182)
- [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/>. (cited on pages 23, 180, 184, 185, 188)
- [Can88] John F. Canny. *The Complexity of Robot Motion Planning*. MIT Press, Cambridge, Massachusetts, 1988. (cited on page 176)
- [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. (cited on pages 169, 170)
- [Cat74] Edwin E. Catmull. *A Subdivision Algorithm for Computer Display of Curved Surfaces*. Ph.d. thesis, University of Utah, December 1974. (cited on page 169)
- [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. (cited on page 169)
- [Cat78] E. Catmull. A hidden-surface algorithm with anti-aliasing. In *Computer Graphics (SIGGRAPH ’78 Proceedings)*, volume 12(3), pages 6–11, August 1978. (cited on page 169)
- [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. (cited on page 170)
- [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. (cited on page 229)
- [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/>. (cited on pages 25, 217)
- [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. (cited on page 175)
- [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. (cited on page 174)
- [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. (cited on page 221)
- [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. (cited on pages 11, 26, 221)
- [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. (cited on page 180)

- [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/>. (cited on pages 23, 180)
- [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/>. (cited on page 184)
- [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. (cited on page 203)
- [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. (cited on page 184)
- [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. (cited on page 135)
- [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. (cited on pages 70, 128, 194)
- [CH97] Deborah A. Carlson and Jessica K. Hodgins. Simulation levels of detail for real-time animation. In *Graphics Interface*, pages 1–8, May 1997. (cited on page 135)
- [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/>. (cited on page 153)
- [Chv75] V. Chvátal. A combinatorial theorem in plane geometry. *J. Combin. Theory Ser. B*, 18:39–41, 1975. (cited on page 208)
- [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. (cited on page 185)
- [CL97] D. Crevier and R. Lepage. Knowledge-based image understanding systems: A survey. *Computer Vision and Image Understanding: CVIU*, 67(2):161–??, ??? 1997. (cited on page 147)
- [Cla76] James H. Clark. Hierarchical geometric models for visible surface algorithms. *Communications of the ACM*, 19(10):547–554, October 1976. (cited on pages 18, 19, 134, 143, 182)
- [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. (cited on page 170)
- [Cla87] K. L. Clarkson. New applications of random sampling to computational geometry. *Discrete and Computational Geometry*, 2:195–222, 1987. (cited on pages 55, 220)
- [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. (cited on page 217)
- [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. (cited on page 169)
- [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. (cited on page 175)
- [COFHZ98] Daniel Cohen-Or, Gadi Fibich, Dan Halperin, and Eyal Zadicario. Conservative visibility and strong occlusion for visibility partitioning of densely occluded scenes. In *Proceedings of Eurographics*, September 1998. <http://www.math.tau.ac.il/~daniel/>. (cited on pages 22, 130, 132, 188)

- [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. (cited on pages 55, 203)
- [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/>. (cited on pages 18, 22, 130, 132, 143, 188)
- [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. (cited on page 174)
- [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. (cited on pages 19, 179)
- [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. (cited on page 170)
- [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. (cited on pages 27, 229)
- [CS94] D. Cohen and Z. Sheffer. Proximity clouds: An acceleration technique for 3D grid traversal. *The Visual Computer*, 11?:27–38, 1994? (cited on page 174)
- [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. (cited on pages 27, 180, 229)
- [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. (cited on pages 27, 192, 229)
- [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. (cited on pages 79, 81)
- [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>. (cited on pages 19, 50, 182, 211)
- [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, June 18–20 1997. <http://graphics.lcs.mit.edu/~capps/>. (cited on page 143)
- [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, April 27–30 1997. ACM Press. <http://graphics.lcs.mit.edu/~satyan/pubs.html>. (cited on pages 19, 50, 127, 182)
- [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. (cited on page 203)
- [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. (cited on page 146)
- [CW93b] Michael F. Cohen and John R. Wallace. *Radiosity and Realistic Image Synthesis*. Academic Press Professional, Boston, MA, 1993. (cited on pages 9, 18, 143, 145)
- [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. (cited on page 175)

- [Dal96] Bengt-Inge L. Dalenbäck. Room acoustic prediction based on a unified treatment of diffuse and specular reflection. *Journal of the Acoustical Society of America*, 100(2):899–909, August 1996. (cited on page 143)
- [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. (cited on pages 41, 237)
- [dBvKOS97] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 1997. (cited on pages 153, 167, 200, 205, 208)
- [DDP96] Frédéric 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/>. (cited on pages 29, 215)
- [DDP97a] F. Durand, G. Drettakis, and C. Puech. 3d visibility made visibly simple. In *video 13th Annu. ACM Sympos. Comput. Geom.*, 1997. (cited on page 217)
- [DDP97b] Frédéric 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>. (cited on pages 29, 215)
- [DDP97c] Fredéric 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/>. (cited on pages 57, 217, 246)
- [DDP99] Frédéric 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>. (cited on pages 57, 80, 218, 226, 227)
- [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. (cited on page 96)
- [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>. (cited on pages 23, 24, 59, 71, 81, 82, 108, 206, 207, 242)
- [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. (cited on page 180)
- [Dor94] Susan Dorward. A survey of object-space hidden surface removal. *International Journal of Computational Geometry and Applications*, 4(3):325–362, 1994. (cited on pages 18, 141, 153, 171)
- [DORP96] Frédéric Durand, Rachel Orti, Stéphane Rivière, and Claude Puech. Radiosity in flatland made visibly simple. In *video 12th Annu. ACM Sympos. Comput. Geom.*, 1996. (cited on pages 27, 51, 215)
- [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. (cited on page 84)
- [DP95b] Frédéric Durand and Claude Puech. The visibility complex made visibly simple. In *video 11th Annu. ACM Sympos. Comput. Geom.*, 1995. (cited on pages 29, 46, 214)
- [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. (cited on page 204)
- [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 Using Sonar. (cited on page 152)

- [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>. (cited on pages 23, 24, 70, 71, 72, 79, 82, 83, 99, 207)
- [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>. (cited on pages 27, 188, 229)
- [DSSD99] Xavier Decoret, Gernot Schaufler, Francois Sillion, and Julie Dorsey. Improved image-based impostors for accelerated rendering. In *Eurographics'99*, August 1999. (cited on page 168)
- [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. (cited on page 146)
- [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. (cited on page 177)
- [Dür38] Albrecht Dürer. *Underweysung der Messung mit dem Zirckel und richtscheyd*. Nuremberg, 1538. (cited on pages 171, 180)
- [Dur95] Frédo Durand. étude du complexe de visibilité. Master's thesis, Institut National Polytechnique de Grenoble, June 1995. (cited on page 77)
- [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. (cited on page 170)
- [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. (cited on page 147)
- [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. (cited on page 203)
- [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>. (cited on page 147)
- [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. (cited on pages 27, 229)
- [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. (cited on pages 24, 51, 77, 81, 200)
- [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. (cited on pages 227, 237)
- [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. (cited on page 166)
- [Ede87] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer Verlag, 1987. (cited on pages 153, 200)
- [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. (cited on page 177)

- [Egg91] D.W. Eggert. Aspect graphs of solids of revolution. In *Ph. D.*, 1991. (cited on page 203)
- [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. (cited on page 204)
- [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. (cited on page 179)
- [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. (cited on page 174)
- [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. (cited on page 204)
- [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. (cited on page 154)
- [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. (cited on pages 175, 229)
- [Fau93] O. Faugeras. *Three-dimensional computer vision*. MIT Press, Cambridge, MA, 1993. (cited on pages 147, 196)
- [FCE⁺98] Thomas Funkhouser, Ingrid Carlstrom, 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. (cited on pages 143, 181)
- [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. (cited on page 200)
- [FF88] Alain Fournier and Donald Fussell. On the power of the frame buffer. *ACM Transactions on Graphics*, 7(2):103–128, 1988. (cited on page 192)
- [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. (cited on page 169)
- [FGH⁺85] Henry Fuchs, Jack Goldfeather, Jeff P. Hultquist, Susan Spach, John D. Austin, Frederick P. Brooks, Jr., John G. Eyles, and John Poultney. 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. (cited on page 180)
- [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>. (cited on pages 153, 232)
- [Fis78] S. Fisk. Short proof of chvatal's watchman theorem. *Journal of Combinatorial Theory*, 24:374, 1978. (cited on page 208)
- [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. (cited on page 175)
- [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/>. (cited on page 210)

- [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. (cited on pages 204, 227)
- [FMC99] Thomas A. Funkhouser, Patrick Min, and Ingrid Carlom. Real-time acoustic modeling for distributed virtual environments. In *Computer Graphics (SIGGRAPH '99 Proceedings)*, August 1999. (cited on page 181)
- [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/>. (cited on page 140)
- [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. (cited on page 95)
- [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. (cited on pages 111, 135)
- [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/>. (cited on pages 134, 189)
- [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/>. (cited on pages 18, 21, 189)
- [FTI86] Akira Fujimoto, Takayuki Tanaka, and Kansei Iwata. ARTS: Accelerated ray tracing system. *IEEE Computer Graphics and Applications*, 6(4):16–26, 1986. (cited on page 174)
- [Fun95] T. Funkhouser. RING - A client-server system for multi-user virtual environments. *SIGGRAPH Symposium on Interactive 3D Graphics*, pages 85–92, 1995. (cited on pages 143, 189)
- [Fun96a] T. Funkhouser. Network topologies for scalable multi-user virtual environments. *Proceedings of VRAIS'96, Santa Clara CA*, pages 222–229, 1996. (cited on page 143)
- [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. (cited on pages 21, 135, 189)
- [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>. (cited on pages 18, 21, 135, 143, 189)
- [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. (cited on pages 18, 141, 143, 158, 165, 174, 195)
- [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. (cited on pages 19, 130)
- [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. (cited on pages 24, 58, 201, 206, 217)

- [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/>. (cited on page 217)
- [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/. (cited on page 204)
- [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/>. (cited on pages 53, 217)
- [GH94] Neil Gatenby and W. T. Hewitt. Optimizing Discontinuity Meshing Radiosity. In *Fifth Eurographics Workshop on Rendering*, pages 249–258, Darmstadt, Germany, June 1994. (cited on page 184)
- [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. (cited on pages 79, 81, 98, 100, 101, 226)
- [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>. (cited on page 18)
- [GH97b] S. Gibson and R. J. Hubbold. Perceptually-driven radiosity. *Computer Graphics Forum*, 16(2):129–141, 1997. ISSN 0167-7055. (cited on pages 80, 82, 95, 226)
- [GH98] Djamchid Ghazanfarpour and Jean-Marc Hasenfratz. A beam tracing with precise antialiasing for polyhedral scenes. *Computer & Graphics*, 22(1):103–115, 1998. (cited on page 181)
- [Gho97] Ghosh. On recognizing and characterizing visibility graphs of simple polygons. *GEOMETRY: Discrete & Computational Geometry*, 17, 1997. (cited on page 154)
- [GI87] Keith D. Gremban and Katsushi Ikeuchi. Planning multiple observation for object recognition. *International Journal of Computer Vision*, 1(2):145–65, 1987. (cited on page 204)
- [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. (cited on pages 19, 114, 118, 131, 135, 195)
- [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. (cited on page 209)
- [Gla84] Andrew S. Glassner. Space sub-division for fast ray tracing. *IEEE Computer Graphics and Applications, October 1984*, 4:15–22, 1984. (cited on page 174)
- [Gla88] Andrew S. Glassner. Spacetime ray tracing for animation. *IEEE Computer Graphics and Applications*, 8(2):60–70, March 1988. (cited on pages 27, 229)
- [Gla89] Andrew Glassner. *An introduction to raytracing*. Academic Press, Reading, MA, 1989. (cited on page 144)
- [Gla95] Andrew S. Glassner. *Principles of Digital Image Synthesis*. Morgan Kaufmann, San Francisco, CA, 1995. (cited on pages 18, 143)
- [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/>. (cited on pages 24, 77, 205)
- [GLL98] 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. (cited on pages 18, 24, 52, 77, 205)

- [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. (cited on pages 24, 51, 58, 64, 81, 201, 202, 203, 214)
- [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. (cited on page 177)
- [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. (cited on pages 18, 24, 205)
- [GN71] Robert A. Goldstein and Roger Nagel. 3-D visual simulation. *Simulation*, 16(1):25–31, January 1971. (cited on page 170)
- [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. (cited on page 200)
- [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. (cited on pages 27, 229)
- [Gra92] Charles W. Grant. *Visibility Algorithms in Image Synthesis*. PhD thesis, U. of California, Davis, 1992. <http://www.hooked.net/~grant/>. (cited on pages 141, 156, 165, 193, 197)
- [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. (cited on pages 19, 195)
- [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. (cited on page 98)
- [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/>. (cited on page 207)
- [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. (cited on page 81)
- [GSHG98] Gene Greger, Peter Shirley, Philip M. Hubbard, and Donald P. Greenberg. The irradiance volume. *IEEE Computer Graphics & Applications*, 18(2):32–43, 1998. (cited on page 217)
- [Gue98] Concettina Guerra. Vision and image processing algorithms. In M. Atallah, editor, *CRC Handbook of Algorithms and Theory of Computation*. CRC Press, 1998. (cited on page 147)
- [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. (cited on page 193)
- [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. (cited on page 192)
- [Hai] Eric Haines. Ray tracing news. <http://www.povray.org/rtn/>. (cited on page 144)
- [Hai91] Eric A. Haines. Beams O' Light: Confessions of a Hacker. In *SIGGRAPH '91 Course Notes - Frontiers in Rendering*. ACM, July 1991. (cited on page 180)
- [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. (cited on page 217)
- [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. (cited on page 170)

- [Has98] Jean-Marc Hasenfratz. *Lancer de Faisceaux en Synthèse d'Image*. PhD thesis, Université de Limoges, 1998. (cited on page 181)
- [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. (cited on page 147)
- [HDG99] David Hart, Philip Dutré, and Donald P. Greenberg. Direct illumination with lazy visibility evaluation. In *Computer Graphics (SIGGRAPH '99 Proceedings)*, August 1999. (cited on page 198)
- [Hec87] Paul S. Heckbert. Ten unsolved problems in rendering. In *Workshop on Rendering Algorithms and Systems, CHI+GI '87*, Toronto, April 1987. (cited on page 18)
- [Hec92a] Paul Heckbert. Discontinuity meshing for radiosity. *Third Eurographics Workshop on Rendering*, pages 203–226, May 1992. <http://www.cs.cmu.edu/~ph/>. (cited on pages 23, 71, 81, 82, 83, 184)
- [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. (cited on pages 23, 184)
- [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. (cited on pages 128, 193, 197)
- [HG94] P. Heckbert and M. Garland. Multiresolution modelling for fast rendering. *Proceedings of Graphics Interface '94*, pages 43–50, 1994. (cited on pages 18, 143)
- [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/>. (cited on page 18)
- [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. (cited on page 180)
- [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. (cited on page 204)
- [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. (cited on page 149)
- [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/>. (cited on page 193)
- [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. (cited on page 200)
- [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. (cited on page 204)
- [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>. (cited on page 150)
- [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>. (cited on pages 149, 150, 151)
- [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. (cited on page 209)
- [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/>. (cited on page 217)

- [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, June 4–6 1997. ACM Press. <http://www.cs.unc.edu/~hudson/projects/occlusion/scg97.ps>. (cited on pages 20, 127, 182)
- [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. (cited on page 182)
- [Hof92] Georg Rainer Hofmann. Who invented ray tracing? a historical remark. *The Visual Computer*, 9(1):120–125, 1992. (cited on page 171)
- [Hop96] H. Hoppe. Progressive meshes. *Computer Graphics*, 30(Annual Conference Series):99–108, 1996. (cited on page 18)
- [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. (cited on page 111)
- [HS99] Nicolas Holzschuch and Francois X. Sillion. An exhaustive error-bounding algorithm for hierarchical radiosity. *Computer Graphics Forum*, 1999. to appear, <http://www.loria.fr/~holzschu>. (cited on pages 111, 226)
- [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. (cited on pages 9, 52, 77, 79, 81, 82, 83, 85, 91, 96, 97, 225, 226)
- [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>. (cited on pages 9, 18, 145)
- [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. (cited on pages 23, 72, 82, 96, 98, 110, 184)
- [HW91] Eric Haines and John Wallace. Shaft culling for efficient ray-traced radiosity. In *Eurographics Workshop on Rendering*, 1991. (cited on pages 20, 73, 187)
- [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. (cited on page 204)
- [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. (cited on pages 82, 227)
- [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. (cited on pages 20, 210)
- [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. (cited on pages 20, 210)
- [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. (cited on page 192)
- [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. (cited on page 204)

- [JF93] A. K. Jain and P. J. Flynn. *Three Dimensional Object Recognition Systems*. Elsevier, Amsterdam, 1993. (cited on page 147)
- [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. (cited on page 219)
- [Jon71] C. B. Jones. A new approach to the ‘hidden line’ problem. *The Computer Journal*, 14(3):232–237, August 1971. (cited on pages 19, 182)
- [JW89] David Jevans and Brian Wyvill. Adaptive voxel subdivision for ray tracing. In *Proceedings of Graphics Interface ’89*, pages 164–172, June 1989. (cited on page 174)
- [Kaj82] James T. Kajiya. Ray tracing parametric patches. In *Computer Graphics (SIGGRAPH ’82 Proceedings)*, volume 16(3), pages 245–254, July 1982. (cited on page 170)
- [Kaj86] J. T. Kajiya. The rendering equation. In David C. Evans and Russell J. Athay, editors, *Computer Graphics (SIGGRAPH ’86 Proceedings)*, volume 20(4), pages 143–150, August 1986. (cited on page 143)
- [KB98] D. Kriegman and P. Belhumeur. What shadows reveal about object structure. In *Proceedings European Conference on Computer Vision*, pages 399–414, 1998. <http://www-cvr.ai.uiuc.edu/~kriegman/>. (cited on page 140)
- [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. (cited on page 193)
- [Ker81] Y.L. Kergosien. La famille des projections orthogonales d’une surface et ses singularités. *C.R. Acad. Sc. Paris*, 292:929–932, 1981. (cited on pages 24, 38, 163, 203, 239)
- [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. (cited on page 170)
- [Kir87] D. B. Kirk. The simulation of natural features using cone tracing. *The Visual Computer*, 3(2):63–71, August 1987. (cited on page 181)
- [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. (cited on page 174)
- [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. (cited on page 216)
- [KKMB96] D. Kersten, D.C. Knill, P. Mamassian, and I. Bülthoff. Illusory motion from shadow. *Nature*, 379(31), 1996. (cited on page 142)
- [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>. (cited on page 167)
- [Kø84] J. J. Koenderink. What does the occluding contour tell us about solid shape? *Perception*, 13:321–330, 1984. (cited on page 162)
- [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. (cited on page 162)
- [Koe90] Jan J. Koenderink. *Solid Shape*. MIT Press, Cambridge, Massachusetts, 1990. (cited on page 162)
- [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. (cited on page 203)

- [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. (cited on page 174)
- [Kut91] H. Kutruff. *Room Acoustics (3rd edition)*. Elsevier Applied Science, 1991. (cited on page 143)
- [Kv76] J.J. Koenderink and A.J. vanDoorn. The singularities of the visual mapping. *BioCyber*, 24(1):51–59, 1976. (cited on pages 10, 24, 77, 162, 199)
- [Kv79] J.J. Koenderink and A.J. vanDoorn. The internal representation of solid shape with respect to vision. *BioCyber*, 32:211–216, 1979. (cited on pages 24, 77, 199)
- [KvD82] Jan J. Koenderink and Andrea J van Doorn. What does the occluding contour tell us about solid shape? *Perception*, 11:129–137, 1982. (cited on page 162)
- [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/>. (cited on page 174)
- [KYCS98] Kim, Yoo, Chwa, and Shin. Efficient algorithms for computing a complete visibility region in three-dimensional space. *Algorithmica*, 20, 1998. (cited on page 186)
- [Lat91] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer, Dordrecht, The Netherlands, 1991. (cited on pages 150, 151, 203)
- [Lau94] A. Laurentini. The visual hull concept for silhouette-based image understanding. *T-PAMI*, 16:150–162, 1994. (cited on pages 52, 77, 178, 179)
- [Lau95] A. Laurentini. How far 3d shapes can be understood from 2d silhouettes. *T-PAMI*, 17:188–195, 1995. (cited on pages 77, 178)
- [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. (cited on pages 77, 178)
- [Lau99] A. Laurentini. Computing the visual hull of solids of revolution. *Pattern Recognition*, 32(3), 1999. (cited on pages 77, 178, 179)
- [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. (cited on page 170)
- [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. (cited on page 170)
- [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-imagin.imag.fr/Publications/>. (cited on pages 27, 230)
- [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. (cited on page 147)
- [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. (cited on page 217)
- [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>. (cited on pages 21, 182)
- [LGLB97] 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. (cited on page 210)

- [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/>. (cited on pages 53, 217)
- [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>. (cited on page 206)
- [Lis94] Dani Lischinski. Incremental delaunay triangulation. In Paul S. Heckbert, editor, *Graphics Gems IV*, pages 47–59. Academic Press Professional, San Diego, CA, 1994. (cited on page 98)
- [LL86] D. T. Lee and A. K. Lin. Computational complexity of art gallery problems. *IEEE Transactions on Information Theory*, 32:276–282, 1986. (cited on page 208)
- [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>. (cited on pages 24, 77, 205)
- [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>. (cited on page 153)
- [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. (cited on pages 25, 217)
- [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. (cited on page 176)
- [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. (cited on page 192)
- [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. (cited on page 147)
- [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. (cited on pages 70, 79, 81, 98, 100, 101, 111, 226)
- [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. (cited on page 179)
- [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>. (cited on pages 23, 71, 81, 82, 83, 184)
- [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>. (cited on pages 23, 71, 79, 82, 83, 110, 184)
- [LW95] Eric P. Lafourture 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. (cited on page 217)
- [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. (cited on pages 53, 215)
- [MAG68] MAGI. 3-D simulated graphics offered by service bureau. *Datamation*, 14:69, February 1968. (cited on page 170)

- [Mal87] Jitendra Malik. Interpreting line drawings of curved objects. *International Journal of Computer Vision*, 1:73–103, 1987. (cited on page 162)
- [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. (cited on page 197)
- [May99] Using Maya, rendering. Alias Wavefront, (Maya user manual), 1999. (cited on page 193)
- [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. (cited on page 182)
- [MB95] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. *Computer Graphics*, 29(Accidental Conference Series):39–46, 1995. (cited on pages 147, 196)
- [MBGN98] Tom McReynolds, David Blythe, Brad Grantham, and Scott Nelson. Advanced graphics programming techniques using Open GL. Siggraph’1998 course notes, 1998. (cited on pages 119, 180, 193)
- [McK87] Michael McKenna. Worst-case optimal hidden-surface removal. *ACM Transactions on Graphics*, 6(1):19–28, January 1987. (cited on page 153)
- [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. (cited on pages 27, 229)
- [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. (cited on page 194)
- [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. (cited on page 209)
- [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. (cited on page 143)
- [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. (cited on page 149)
- [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. (cited on page 170)
- [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. (cited on page 89)
- [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. (cited on pages 142, 166)
- [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. (cited on pages 11, 218)
- [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. (cited on page 192)
- [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. (cited on page 82)

- [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. (cited on page 219)
- [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. (cited on page 134)
- [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>. (cited on pages 141, 170)
- [Mul89] Ketan Malmuley. 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. (cited on page 167)
- [Mul91] Malmuley. Hidden surface removal with respect to a moving view point (extended abstract). In *STOC: ACM Symposium on Theory of Computing (STOC)*, 1991. (cited on pages 55, 171, 176)
- [Mun95] Olaf Munkelt. Aspect-trees: Generation and interpretation. *Computer Vision and Image Understanding: CVIU*, 61(3):365–386, May 1995. (cited on page 204)
- [Mur87] G. Murch. Color displays and color science. In H. John Durret, editor, *Color and the Computer*, pages 1–25. Academic Press, Boston, 1987. (cited on page 95)
- [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. (cited on pages 167, 188)
- [Mys98] Karol Myszkowski. The visible differences predictor: applications to global illumination problems. In *Eurographics Workshop on Rendering*, Vienna, Austria, June 1998. (cited on pages 111, 226)
- [MZ92] J.L. Mundy and A. Zisserman. *Geometric Invariance in Computer Vision*. MIT press, Cambridge, MA, 1992. (cited on page 148)
- [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. (cited on page 175)
- [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. (cited on page 176)
- [Neu95] Laszlo Neumann. Monte Carlo Radiosity. *Computing*, 55(1):23–42, 1995. (cited on page 222)
- [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. (cited on page 175)
- [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. (cited on page 175)
- [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. (cited on page 176)
- [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. (cited on page 180)
- [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. (cited on pages 23, 183)
- [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. (cited on pages 23, 166, 183, 184)

- [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. (cited on page 222)
- [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. (cited on page 167)
- [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. (cited on pages 23, 183)
- [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. (cited on page 175)
- [NSL99] C. Nissoux, T. Simeon, and J.P. Laumond. Visibility based probabilistic roadmaps. Technical Report 99057, LAAS, February 1999. doc@laas.fr. (cited on page 177)
- [O'R87] J. O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, New York, NY, 1987. (cited on pages 154, 208)
- [O'R94] Joseph O'Rourke. *Computational geometry in C*. Cambridge University Press, 1994. (cited on pages 153, 200)
- [ORDP96] Rachel Ortí, Stéphane Rivière, Frédéric 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. (cited on pages 27, 51, 75, 215)
- [Ort97] Rachel Ortí. *Radiosité dynamique 2D et complexe de visibilité*. PhD thesis, Université Joseph Fourier (Grenoble), 1997. PhD Thesis. (cited on page 27)
- [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. (cited on page 154)
- [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. (cited on page 177)
- [PA91] Pierre Poulin and John Amanatides. Shading and shadowing with linear light sources. *Computers and Graphics*, 15(2):259–265, 1991. (cited on pages 193, 216)
- [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. (cited on page 147)
- [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. (cited on page 200)
- [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. (cited on page 213)
- [PD90] H. Plantinga and C. R. Dyer. Visibility, occlusion, and the aspect graph. *International Journal of Computer Vision*, 5(2):137–160, 1990. (cited on pages 24, 26, 40, 51, 53, 58, 60, 81, 202, 213, 214, 217)
- [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. (cited on pages 26, 204)
- [Pel90] M. Pellegrini. Stabbing and ray shooting in 3-dimensional space. In *Proc. 6th Annu. ACM Sympos. Comput. Geom.*, pages 177–186, 1990. (cited on pages 11, 55, 59, 220)

- [Pel93] Marco Pellegrini. Ray shooting on triangles in 3-space. *Algorithmica*, 9:471–494, 1993. (cited on pages 11, 26, 53, 55, 220)
- [Pel94] Pellegrini. On lines missing polyhedral sets in 3-space. *GEOMETRY: Discrete & Computational Geometry*, 12, 1994. (cited on pages 55, 220)
- [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>. (cited on page 171)
- [Pel97a] Pellegrini. Monte carlo approximation of form factors with error bounded a priori. *Discrete & Computational Geometry*, 17, 1997. (cited on page 222)
- [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/>. (cited on pages 11, 26, 154, 221)
- [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. (cited on pages 204, 222)
- [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. (cited on pages 24, 38, 162, 203, 239, 241, 242)
- [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. (cited on pages 55, 203)
- [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. (cited on pages 55, 203)
- [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/>. (cited on page 179)
- [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/>. (cited on page 187)
- [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. (cited on pages 111, 135)
- [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. (cited on page 194)
- [PJ91] Andrew Pearce and David Jevans. Exploiting shadow coherence in ray-tracing. In *Proceedings of Graphics Interface '91*, pages 109–116, June 1991. (cited on page 198)
- [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. (cited on page 203)
- [Pla88] William Harry Plantinga. *The asp: a continuous, viewer-centered object representation for computer vision*. PhD thesis, The University of Wisconsin, Madison, December 1988. (cited on page 213)
- [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. (cited on pages 26, 204)
- [Plü65] J. Plücker. On a new geometry of space. *Phil. Trans. Royal Soc. London*, 155:725–791, 1865. (cited on pages 219, 251)

- [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>. (cited on pages 18, 147)
- [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. (cited on page 148)
- [PPK92] S. Petitjean, J. Ponce, and D.J. Kriegman. Computing exact aspect graphs of curved objects: Algebraic surfaces. *IJCV*, 1992. (cited on pages 24, 203)
- [PPR99] Helmut Pottmann, Martin Peternell, and Bahram Ravani. An introduction to line geometry with application. *Computer-Aided Design*, 31:3–16, 1999. (cited on page 219)
- [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. (cited on page 187)
- [PS92] Pellegrini and Shor. Finding stabbing lines in 3-space. *GEOMETRY: Discrete & Computational Geometry*, 8, 1992. (cited on pages 26, 55, 220)
- [PV96a] Pocchiola and Vegter. Topologically sweeping visibility complexes via pseudotriangulations. *GEOMETRY: Discrete & Computational Geometry*, 16, 1996. (cited on pages 26, 27, 29, 46, 214)
- [PV96b] M. Pocchiola and G. Vegter. The visibility complex. *Internat. J. Comput. Geom. Appl.*, 1996. special issue devoted to ACM-SoCG'93. (cited on pages 9, 12, 26, 27, 29, 46, 214)
- [PY90] Paterson and Yao. Efficient binary space partitions for hidden-surface removal and solid modeling. *GEOMETRY: Discrete & Computational Geometry*, 5, 1990. (cited on page 175)
- [Qua96] Matthew Quail. Space time ray-tracing using ray classification. Master's thesis, Macquarie University, November 1996. (cited on pages 27, 229)
- [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/>. (cited on page 182)
- [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. (cited on page 51)
- [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. (cited on page 81)
- [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. (cited on page 133)
- [RH94] John Rohlff 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. (cited on pages 129, 134)
- [Rie87] J.H. Rieger. On the classification of views of piecewise smooth objects. *Image and Vision Computing*, 1987. (cited on pages 163, 239)
- [Rie90] J. H. Rieger. The geometry of view space of opaque objects bounded by smooth surfaces. *Artificial Intelligence(I-2)*, 44:1–40, 1990. (cited on pages 163, 239)
- [Rie92] J.H. Rieger. Global bifurcation sets and stable projections of non-singular algebraic surface. *IJCV*, 1992. (cited on pages 24, 203)
- [Rie93] J. H. Rieger. Computing view graphs of algebraic surfaces. *Journal of Symbolic Computation*, 16(3):259–272, September 1993. (cited on pages 24, 203)

- [Riv95] S. Rivière. Topologically sweeping the visibility complex of polygonal scenes. In *Comm. 11th Annu. ACM Sypos. Computat. Geom.*, 1995. (cited on pages 27, 214)
- [Riv97a] S. Rivière. *Calculs de visibilité dans un environnement polygonal 2D*. PhD thesis, Université Joseph Fourier (Grenoble), 1997. PhD Thesis. (cited on pages 27, 46, 214)
- [Riv97b] S. Rivière. Dynamic visibility in polygonal scenes with the visibility complex. In *Comm. 13th Annu. ACM Sypos. Computat. Geom.*, 1997. (cited on page 47)
- [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. (cited on page 27)
- [RM97] D.R. Roberts and A.D. Marshall. A review of viewpoint planning. Technical Report 97007, University of Wales, Cardiff, August 1997. <http://www.cs.cf.ac.uk/Research/Rrs/1997/detail007.html>. (cited on pages 18, 150)
- [Rob63] L.G. Roberts. Machine perception of three dimensional solids. Technical Report TR-315, Lincoln Laboratory, MIT, Cambridge, MA, May 1963. Also in Tippett, J.T *et al.*, eds., *Optical and Electro Optical Information Processing*, MIT Press, Cambridge, MA, 1964. (cited on page 166)
- [Rog97] David F. Rogers. *Procedural Elements for Computer Graphics*. Mc Graw-Hill, 2 edition, 1997. (cited on pages 18, 141, 165, 170, 174, 195)
- [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. (cited on pages 111, 226)
- [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. (cited on page 177)
- [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. (cited on page 193)
- [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. (cited on page 174)
- [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. (cited on pages 77, 111, 226)
- [San76] L. A. Santaló. *Integral Geometry and Geometric Probability*. Addison-Wesley, Reading, MA, 1976. (cited on pages 42, 51, 217, 222)
- [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. (cited on page 174)
- [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. (cited on page 202)
- [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. (cited on page 200)
- [Sbe93] M. Sbert. An integral geometry based method for fast form-factor computation. *Computer Graphics Forum*, 12(3):C409–C420, 1993. (cited on pages 51, 222)
- [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. (cited on page 168)

- [Sch93] L. L. Schumaker. Computing optimal triangulations using simulated annealing. *Computer Aided Geometric Design*, 10(3):329–346, August 1993. (cited on page 111)
- [SD92] W. Brent Seales and Charles R. Dyer. Viewpoint from occluding contour. *Computer Vision, Graphics and Image Processing: Image Understanding*, 55:198–211, 1992. (cited on page 203)
- [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/>. (cited on pages 28, 135, 174, 226, 237)
- [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. (cited on page 181)
- [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>. (cited on pages 18, 147)
- [SDS96] Eric J. Stollnitz, Tony D. DeRose, and David H. Salesin. *Wavelets for Computer Graphics: Theory and Applications*. Morgan Kaufmann, San Francisco, CA, 1996. (cited on pages 83, 85, 226)
- [SG82] S. Sechrest and D. P. Greenberg. A visible polygon reconstruction algorithm. *ACM Transactions on Graphics*, 1(1):25–42, January 1982. (cited on page 170)
- [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/>. (cited on pages 23, 24, 59, 71, 81, 108, 207)
- [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. (cited on pages 27, 129, 229)
- [Sgi99] Silicon Graphics 320 datasheet. <http://visual.sgi.com/research/data/whitepapers.html>, 1999. (cited on page 129)
- [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. (cited on pages 51, 145)
- [Sha97] Erin Shaw. Hierarchical radiosity for dynamic environments. *Computer Graphics Forum*, 16(2):107–118, 1997. ISSN 0167-7055. (cited on pages 27, 229)
- [She92] Thomas Shermer. Recent results in art galleries. In *Proc. IEEE*, pages 80:1384–1399, September 1992. (cited on pages 154, 208)
- [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. (cited on page 111)
- [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. (cited on pages 77, 111, 174, 226)
- [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>. (cited on page 146)
- [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. (cited on page 203)

- [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. (cited on page 174)
- [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/>. (cited on pages 228, 237)
- [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. (cited on page 222)
- [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. (cited on page 193)
- [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. (cited on page 168)
- [Sla92] M. Slater. A comparison of three shadow volume algorithms. *The Visual Computer*, 9(1):25–38, 1992. (cited on page 191)
- [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. (cited on page 148)
- [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. (cited on pages 18, 147)
- [Smi99] Brian Smits. Efficiency issues for ray tracing. *Journal of Graphics Tools*, 1999. to appear, <http://www2.cs.utah.edu/~bes/>. (cited on page 174)
- [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. (cited on page 166)
- [Sol78] Herbert Solomon. *Geometrue Probability*. SIAM CBMS-NSF Regional Conference Series in Applied Mathematics, Philadelphia, PA, 1978. (cited on page 222)
- [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-imagin.imag.fr/~Cyril.Soler>. (cited on pages 28, 77, 125, 197, 226, 237)
- [Som51] D. M. Y. Sommerville. *Analytical Geometry in Three Dimensions*. Cambridge University Press, Cambridge, 1951. (cited on page 219)
- [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. (cited on pages 18, 24, 205)
- [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. (cited on page 194)
- [SP94] Francois Sillion and Claude Puech. *Radiosity and Global Illumination*. Morgan Kaufmann, San Francisco, CA, 1994. (cited on pages 9, 18, 143, 145)
- [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. (cited on pages 228, 237)

- [Spe92a] L. Richard Speer. An updated cross-indexed guide to the ray-tracing literature. *Computer Graphics*, 26(1):41–72, January 1992. (cited on page 144)
- [Spe92b] R. Speer. A new data structure for high-speed, memory efficient ray shooting. In *Eurographics Workshop on Rendering*, 1992. (cited on page 216)
- [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. (cited on page 222)
- [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. (cited on page 193)
- [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. (cited on page 193)
- [SS96a] G. Schaufler and W. Stürzlinger. A three-dimensional image cache for virtual reality. In *Proceedings of EUROGRAPHICS'96*, 1996. (cited on pages 18, 147)
- [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>. (cited on pages 28, 77, 226, 237)
- [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>. (cited on pages 125, 197, 226)
- [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>. (cited on page 197)
- [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. (cited on pages 18, 141, 157, 165, 170)
- [Ste82] Ian Stewart. *Oh Catastrophe!* Belin, 1982. (cited on page 162)
- [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. (cited on page 202)
- [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. (cited on page 181)
- [Sto91] J. Stolfi. *Oriented Projective Geometry: A Framework for Geometric Computations*. Academic Press, 1991. (cited on page 219)
- [Stu94] W. Stürzlinger. 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/>. (cited on pages 23, 82, 96, 184)
- [Stu99] Wolfgang Stuerzlinger. Imaging all visible surfaces. In *Proceedings of Graphics Interface, Kingston, Ontario*, 1999. <http://www.dgp.toronto.edu/gi99/papers/programme.html>. (cited on page 209)
- [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. (cited on page 193)
- [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. (cited on page 170)

- [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. (cited on pages 18, 24, 205)
- [TA98] Seth Teller and John Alex. Frustum casting for progressive interactive rendering. Technical Report TR-740, MIT Laboratory for Computer Science, January 1998. (cited on page 181)
- [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. (cited on page 153)
- [Tam93] Filippo Tampieri. *Discontinuity Meshing for Radiosity Image Synthesis*. Ph.D. thesis, Cornell University, Ithaca, NY, 1993. (cited on pages 71, 82, 83)
- [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. (cited on pages 18, 150)
- [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>. (cited on pages 11, 23, 26, 28, 53, 58, 59, 81, 221)
- [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>. (cited on pages 21, 53, 133, 188, 189, 218, 219, 221)
- [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>. (cited on pages 21, 135, 189)
- [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. (cited on page 208)
- [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/>. (cited on page 194)
- [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/>. (cited on page 194)
- [TH91] Seth J. Teller and Michael E. Hohmeyer. Computing the lines piercing four lines. Technical report, CS Dpt. UC Berkeley, 1991. (cited on pages 65, 69)
- [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>. (cited on pages 20, 21, 26, 28, 73, 81, 135, 188, 189, 221)
- [Tho56] R. Thom. Les singularités des applications différentiables. *Annales Institut Fourier*, 6:43–87, 1956. (cited on page 162)
- [Tho72] R. Thom. *Structural Stability and Morphogenesis*. Benjamin, New-York, 1972. (cited on pages 12, 162)
- [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. (cited on page 229)

- [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. (cited on pages 82, 95)
- [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>. (cited on pages 21, 133, 188, 189)
- [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. (cited on page 170)
- [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. (cited on page 216)
- [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. (cited on pages 170, 216)
- [TTK96] Konstantinos Tarabanis, Roger Y. Tsai, and Anil Kaul. Computing occlusion-free viewpoints. *PAMI*, 18(3):279–292, March 1996. (cited on pages 77, 185)
- [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. (cited on page 209)
- [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. (cited on page 226)
- [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. (cited on page 222)
- [Ull89] Shimon Ullman. Aligning pictorial descriptions: An approach to object recognition. *Cognition*, 32(3):193–254, August 1989. (cited on page 147)
- [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. (cited on pages 154, 208)
- [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. (cited on pages 79, 81)
- [Vea97] Eric Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. Ph.d. thesis, Stanford University, December 1997. <http://www-graphics.stanford.edu/~ericv/>. (cited on page 144)
- [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. (cited on page 166)
- [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. (cited on page 112)
- [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. (cited on pages 177, 178)
- [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. (cited on pages 166, 180)

- [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. (cited on page 182)
- [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. (cited on page 140)
- [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. (cited on page 142)
- [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. (cited on page 167)
- [War94] Greg Ward. A contrast-based scalefactor for luminance display. In Paul Heckbert, editor, *Graphics Gems IV*, pages 415–421. Academic Press, Boston, 1994. (cited on pages 82, 95)
- [Wat70] G.S. Watkins. *A Real Time Visible Surface Algorithm*. Ph.d. thesis, University of Utah, June 1970. (cited on page 170)
- [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. (cited on page 200)
- [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. (cited on page 182)
- [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. (cited on pages 25, 217)
- [WC91] Andrew Woo and Steve Chall. An efficient scanline visibility implementation. In *Proceedings of Graphics Interface '91*, pages 135–142, June 1991. (cited on page 170)
- [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. (cited on pages 70, 88, 222)
- [Wei93] I. Weiss. Geometric invariant and object recognition. *Internat. J. Comput. Vision*, 10(3):207–231, 1993. (cited on page 148)
- [WF90] R. Wang and H. Freeman. Object recognition based on characteristic view classes. In *Proceedings 10th International Conference on Pattern Recognition*, Atlantic City, NJ, 17-21 June 1990. (cited on page 200)
- [WH96] L. R. Williams and A. R. Hanson. Perceptual completion of occluded surfaces. *Computer Vision and Image Understanding: CVIU*, 64(1), 1996. (cited on page 163)
- [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. (cited on pages 174, 193)
- [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. (cited on page 162)
- [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. (cited on page 170)
- [Whi80] Turner Whitted. An improved illumination model for shaded display. *CACM*, 1980, 23(6):343–349, 1980. (cited on pages 144, 170, 174, 180)
- [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>. (cited on pages 27, 230)

- [Wil78] Lance Williams. Casting curved shadows on curved surfaces. In *Computer Graphics (SIGGRAPH '78 Proceedings)*, volume 12(3), pages 270–274, August 1978. (cited on page 193)
- [Wil96] L. R. Williams. Topological reconstruction of a smooth manifold-solid from its occluding contour. *Computer Vision and Image Understanding: CVIU*, 64(2), 1996. (cited on pages 163, 164)
- [Woo92] Andrew Woo. The shadow depth map revisited. In David Kirk, editor, *Graphics Gems III*, pages 338–342, 582. Academic Press, Boston, MA, 1992. (cited on page 193)
- [Woo97] Andrew Woo. Recursive grids and ray bounding box comments and timings. *Ray Tracing News*, 10(3), December 1997. <http://www.povray.org/rtn/>. (cited on page 174)
- [Wor97] Steve Worley. Fast soft shadows. *Ray Tracing News*, 10(1), January 1997. <http://www.povray.org/rtn/>. (cited on page 198)
- [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/>. (cited on pages 18, 142, 158)
- [WREE67] C. Wylie, G.W. Romney, D.C. Evans, and A.C. Erdahl. Halftone perspective drawings by computer. In *FJCC*, pages 49–58, 1967. (cited on page 170)
- [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/>. (cited on page 142)
- [WS99] Peter Wonka and Dieter Schmalstieg. Occluder shadows for fast walkthroughs of urban environments. In *Eurographics'99*, August 1999. (cited on pages 20, 135, 196)
- [WW92] Alan Watt and Mark Watt. *Advanced Animation and Rendering Techniques: Theory and Practice*. Addison-Wesley, 1992. (cited on pages 141, 169, 174)
- [WW97] Daphna Weinshall and Michael Werman. On view likelihood and stability. *T-PAMI*, 19-2:97–108, 1997. (cited on page 228)
- [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>. (cited on pages 27, 230)
- [Yan85] Johnson K. Yan. Advances in computer-generated imagery for flight simulation. *IEEE Computer Graphics and Applications*, 5(8):37–51, August 1985. (cited on page 168)
- [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. (cited on page 209)
- [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. (cited on pages 23, 184)
- [YR96] R. Yagel and W. Ray. Visibility computation for efficient walkthrough complex environments. *PRES-ENCE*, 5(1):1–16, 1996. <http://www.cis.ohio-state.edu/volviz/Papers/1995/presence.ps.gz>. (cited on page 189)
- [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. (cited on pages 81, 193)
- [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. (cited on pages 20, 188)
- [Zha91] Ning Zhang. Two Methods for Speeding up Form-factor Calculation. In *Second Eurographics Workshop on Rendering*, Barcelona, Spain, May 1991. (cited on page 188)

- [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>. (cited on page 193)
- [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>. (cited on pages 19, 127, 131, 135, 195)
- [ZMH97] 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>. (cited on pages 19, 114, 118, 127, 131, 135, 195)