PHD

Localisation for virtual environments

Law, Robin Ren-Pei

Award date:
2002

Awarding institution:
University of Bath

Localisation for Virtual Environments

Submitted by
Robin Ren-Pei Law BEng MSc
for the degree of Doctor of Philosophy
of the University of Bath
2002

Robin Law

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Abstract

The demand for high quality computer graphics images is ever-increasing and hence, the demand for realism now extends throughout the various user communities and embraces such diverse applications as architectural representations, computer games, CAD, medical imaging and the film and entertainment industry. Unfortunately, the cost of realism is high in terms of computational complexity and real time construction of such images is still beyond the power of available systems.

The highest quality realism is obtained from radiosity images, but, computing the radiosity solution for a scene is a time consuming process, requiring $O(n^2)$ space and time complexity to solve the radiosity matrix. Many variations of the radiosity algorithm have been investigated but hierarchical radiosity has been without a doubt, the most valuable method for the accurate and rapid radiosity solution for scenes of modest sizes. The $O(n)$ running time complexity of this algorithm represents the most optimal efficiency for arriving at a complete solution. However, the $O(k^2)$ linking time limits the overall size of scenes that hierarchical radiosity can manage. Clustering techniques have since been incorporated into hierarchical radiosity algorithms and have significantly improved the running time complexity from $O(k^2 + n)$ to $O(k\log k + n)$. Thus, clustered hierarchical radiosity algorithms represent the most efficient methods for solving the radiosity of a scene.

In this thesis, we have presented a new approach to clustering for hierarchical radiosity. The objective for this new approach was to improve the running time complexity hierarchical radiosity and thus its scalability, so that large, complex scenes can be rendered in a tractable amount of time. We have shown that the use of clustering to localise clusters of surfaces, reduces the size of the input surface geometry that needs to be solved by hierarchical radiosity. As a result, significant increases in performance have been achieved. The results of our clustering strategy show that our localisation technique scales linearly with increasing complexity of the input scene.

Acknowledgements

I would like to say a million thank yous and express my most heart-felt appreciation to my supervisors, Dr. Claire Willis and Dr. Derek Paddon. Without their invaluable guidance and encouragement, this PhD would not be possible.

A special big thank you to my mum and brother for all their love and support.

Thanks to Nam for being a great colleague and a good friend.

A big hug to my Kim.

Many thanks to Dr. Martin Oliver.

I have no doubt forgotten to say thanks to lots of people. So, I would also like to say 'cheers!' to all those who should appear here, but didn't.

Contents

Chapter 1 Introduction

The quality of images in all forms of modern media is expected to be of the highest possible standard. In computer graphics, this implies that they must represent what the eye sees in nature, or even the esoteric mind images of an artist. Such images must therefore portray three-dimensionality, colour and shadow subtlety, and articulation of forms. Given infinite computing power, we could satisfy all these parameters. Unfortunately, this is obviously not possible and so we must concentrate efforts on developing efficient algorithms and new image theories to attain the level of quality expected, within the bounds of current computing technology.

Photo-realistic image rendering is particularly difficult to compute because of the complexity of the physical nature of light. However, global illumination algorithms such as radiosity and Monte Carlo ray-tracing based methods (see Slater *et al.* [92]), provide the necessary foundation for extremely high quality rendered photo-realistic images.

Monte Carlo path tracing algorithms are stochastic sampling methods that have recently become very popular in solving the global illumination problem. These methods are relatively quick to generate images but they tend to be noisy due to stochastic sampling. Nevertheless, Monte Carlo path tracing methods are only really useful for the efficient rendering of local illumination. These methods require substantial computation to generate global illumination solutions for photo-realistic image generation, as they have to cast vast numbers of rays to achieve sufficient sampling.

The main disadvantage of Monte Carlo path tracing based algorithms, which are image space methods, is their dependence on viewpoint. There have been attempts to solve this problem, usually by employing caching techniques to accelerate image generation from the new viewpoint. With this restriction, Monte Carlo path tracing algorithms are typically only suitable for rendering high quality snapshots of environments at a particular viewpoint.

The radiosity rendering algorithm has become established as the method for rendering the highest quality, view independent images for virtual environments. Radiosity is currently the only global illumination algorithm that correctly computes shadows due to area light sources. In combination with discontinuity meshing, the most complete description of global illumination within a virtual environment is obtained, capturing all the subtle lighting effects such as colour bleeding and shadow penumbra and umbra.

However, due to the high computation cost of obtaining a global illumination solution and the limitation that all surfaces are assumed to be Lambertian (i.e. diffuse), radiosity algorithms have not gained wide use and have quite often been dismissed as being to expensive to compute.

There have been attempts to combine radiosity with Monte Carlo path tracing methods to generate hybrid algorithms, for example by Chen *et al.* [38]. These algorithms typically use radiosity to generate a coarse, diffuse global illumination solution and use ray-tracing techniques to generate the complete, high quality image with rendered specular effects, etc. However, these methods have yet to achieve significant popularity. Therefore, the thrust of this thesis will be to continue the quest for a highly scalable radiosity method.

In 1959 Eckbert and Drake [4] introduced a radiosity method to compute the radiant heat exchanges between surfaces. Since that time, radiosity has become a recognised method for realistic image synthesis. The first significant advancement to the radiosity algorithm was developed in 1984 by Goral *et al.* [15]. Contour integration was used to analytically solve the form factors between two surfaces and the radiosity equation was solved using Gaussian elimination.

Cohen *et al.* [19][22] made a substantial improvement on Goral *et al.'s* radiosity method. Form factors were solved using numerical integration techniques (the hemicube method) and the radiosity equation was solved iteratively using the Jocobi or Gauss-Seidel iterative methods.

In 1991 Hanrahan *et al.* [39] introduced the most advanced method for solving the radiosity equation. Hanrahan *et al.* introduced a hierarchical radiosity algorithm that adaptively and hierarchically refines surfaces according to the light transport between a surface and the surrounding geometry. Instead of refining a surface to create a single mesh, this method maintains a multi-resolution hierarchy of refined meshes. Thus, weakly interacting surfaces will only need to interact at the coarsest levels in the mesh hierarchy. Conversely, surfaces that require more accurate representations of light transport can interact with more accurate meshes. Therefore, the key part of the hierarchical radiosity method is the accurate building of the links between these interacting surfaces, which reflects the nature of light transport throughout a scene.

Unfortunately, the initial linking stage of hierarchical radiosity hinders the scalability of this algorithm, as it requires an $O(n^2)$ search of the scene geometry to establish a description of the light transport throughout the environment. There have been attempts to optimise the running time complexity consumed by initial linking. The current most popular method for optimising initial linking was presented by Holzschuch *et al.* [56], whereby the initial links are created lazily. In this method links are only created when the light transport between patches are sufficiently important.

The main drawback of hierarchical radiosity is that the coherence between the surfaces within a scene are not taken into account. In a typical scene, it is common for surfaces to group together and interact with the surrounding geometry as a single unit. Therefore, such surfaces can be approximated by a single entity and hence significant savings can be made on the numerous light interactions that would otherwise have to be computed.

Thus, the development of clustered hierarchical radiosity enables nearby surfaces to be grouped into clusters, such that clusters can now interact with the scene geometry. There are currently two main implementations of clustering for hierarchical radiosity. Sillion [57][66] represent the clusters as isotropic scattering volumes to approximate the light transported by the surfaces contained within the clusters, while Smits *et al.* [59] use the surfaces within the clusters to estimate the error bounds on the light transported between clusters. Clustered hierarchical radiosity is the most

significant advancement made to traditional hierarchical radiosity and has enabled the photo-realistic rendering for scenes of moderate size.

The principle aim of the research in this thesis is to improve the scalability of hierarchical radiosity, such that solving large complex scenes is possible in a tractable amount of time. Hence we aim to reduce the complexity of the input surface geometry supplied to hierarchical radiosity algorithm.

We specifically aim to utilise clustering to locate tight fitting clusters around distinct objects, since the surfaces within objects tend to have similar lighting properties. Thus good, robust approximations can be obtained from these clusters. However, the main purpose for using clustering is to obtain as many clusters available within a scene that contain as few surfaces as possible. Hence we aspire to:

- Locate the maximum number of object clusters within a scene.
- Optimise clusters such that they only contain a single object.

However even if optimal clustering is achieved, it is possible for clusters to contain objects that are composed of great number of finely tessellated surfaces. Thus we also present a prototype system that incorporates multi-resolution, level of detail (LOD) representation of these objects.

Thesis Overview

Chapter 2 introduces radiosity in a review that describes the basic definitions of radiosity and also the traditional techniques used to solve the radiosity problem. We commence by describing the main methods for computing form factors and then proceed with an explanation of progressive refinement radiosity. Also, we explain the two paradigms of shooting and gathering light and present the algorithm for progressive refinement radiosity. Finally, we discuss the benefits of sub-structuring and its application to progressive refinement radiosity.

Chapter 3 discusses in depth, the hierarchical radiosity algorithm and the most current techniques used to improve hierarchical radiosity. We begin by introducing a basic

algorithm for hierarchical radiosity and then proceed with a comprehensive analysis of patch refinement and linking, which is an essential component of this algorithm. Next the method of multigridding is reviewed and an algorithm given to demonstrate how this method would be applied to the current hierarchical radiosity algorithm. A detailed discussion on clustering techniques follows. Here, we discuss the various manual and automatic clustering algorithms that have been applied to classical radiosity and hierarchical radiosity. Finally, we examine the techniques of discontinuity meshing and dynamic discontinuity meshing. This technique produces the most complete definition of shadowing within an environment and is an important method for generating accurate shadows. However, computing discontinuity meshes is notoriously expensive and so we discuss extremal discontinuity meshing. We also provide an algorithm for computing extremal discontinuity lines. Dynamic discontinuity meshing is also discussed and we present a simple experiment showing the benefits of extremal meshing and dynamic discontinuity meshing.

Chapter 4 presents an improved clustering strategy for hierarchical radiosity. We establish that the only practical approach for improving the scalability of hierarchical radiosity such that large, complex scenes can be solved is to reduce the number of surfaces solved by hierarchical radiosity, by utilising clustering techniques. We discuss in depth our localisation technique, which utilises clustering in a unique way, such that weakly interacting, that is, the light transport between clusters are detached from the environment and rendered independently.

Chapter 5 presents a set of experiments to test the performance and scalability of our new clustering strategy presented in the previous chapter. A series of scenes that represents a wide range in scene complexity were used in our tests, and contained between 550 and 27000 surfaces. The results shown in this chapter suggest that our new algorithm is suitable for solving large complex scenes and importantly, is capable of delivering linear running time complexity for these environments.

Chapter 6 introduces multi-resolution modelling. The results shown in Chapter 5 demonstrate that reducing the number of input surfaces solved by hierarchical radiosity at any one time, is the only viable method for improving the scalability of hierarchical radiosity. However, as one should expect, our localisation technique performs poorly

when a scene is composed of a few objects that are composed of a very large number of tessellated surfaces. Thus, we describe level of detail representation in multi-resolution modelling and present a prototype system for incorporating level of detail representation to our localisation algorithm. Results are also given for the same set of scenes used in Chapter 5 and enables a comparison between the performance of the combined localisation and multi-resolution algorithm and the localisation algorithms in Chapter 4.

Chapter 7 presents the final conclusions of this thesis. We discuss the achievements of our new clustering strategy and suggest a possible direction for future work in extending the scalability of hierarchical radiosity.

Chapter 2 A Review of Radiosity

2.1 Introduction

Since its introduction to computer graphics, radiosity has matured into a well described and well implemented set of algorithms. It has evolved into a powerful technique that can render scenes to a high degree of accuracy and realism.

Radiosity or *radiant exitance*, began in 1936 when Moon [3] wrote "*The Scientific Basis of Illumination Engineering*". Interestingly, luminous exitance was called *luminosity* but there was no term for radiant exitance, so he coined the term *radiosity* to describe the radiant flux density leaving a surface [52]. It was not until Eckbert and Drake [4] introduced a radiosity method to compute the radiant heat exchanges between surfaces, that radiosity has become a recognised method for realistic image synthesis.

2.2 Basic Definitions

To tackle the radiosity problem, the following basic definitions describe fundamental properties of radiosity.

Radiant Energy (Q):

Measured in Joules (J), *radiant energy* is the amount of energy contained in a quantum of electromagnetic radiation. The energy can be calculated using Planck's equation:

$$Q = hf \tag{2.1}$$

where h is Planck's constant and f is the frequency of the quantum of electromagnetic energy.

Radiant Flux (Φ)

Measured in Watts (W) or Joules per second (Js^{-1}), *radiant flux* is the rate of change of radiant energy Q, with respect to time (Equation 2.2):

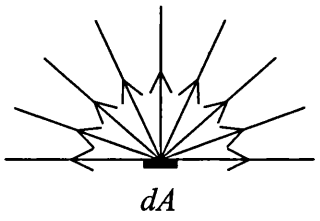
$$\Phi = \frac{dQ}{dt} \qquad 2.2$$

Radiant flux is also referred to as *radiant power* and can be conveniently thought of as the power contained in an infinitesimally thin geometric ray of light.

Radiant Flux Density ($d\Phi/dA$): Irradiance & Radiant Exitance

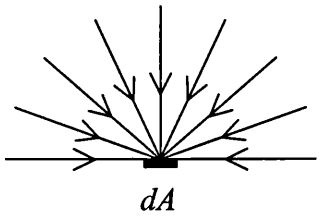
Measured in Watts per square metre (Wm^{-2}), *radiant flux density*, which is also known as *radiosity* (B), is the rate of change of radiant flux with respect to area (A). In other words, radiant flux density is the radiant energy flowing though a unit area per unit time.

When applied to a scene, radiant flux density can either arrive at, pass through or leave a surface. Radiant flux density that leaves a surface (see Figure 2.1) due to emission and/or reflection is referred to as *radiant exitance* (M). Radiant flux density that arrives at a surface (see Figure 2.2) is referred to as *irradiance* (E).



$$M = \frac{d\Phi}{dA}$$

Figure 2.1 Radiant exitance from a surface.



$$E = \frac{d\Phi}{dA}$$

Figure 2.2 Irradiance of a surface.

Radiance (L)

Measured in Watts per steradian per square metre ($Wsr^{-1}m^{-2}$), *radiance* is the amount of radiant flux (energy) that is contained in an infinitesimally thin cone of light (see Figure 2.3).

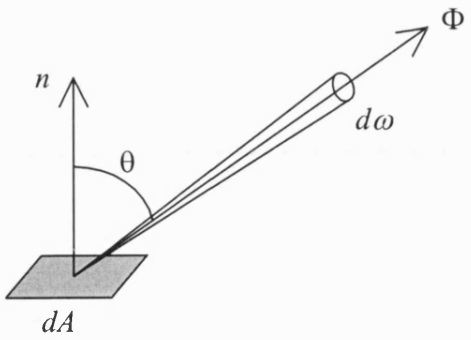


Figure 2.3 An illustration of radiance.

The definition of radiance is:

$$L = \frac{d^2\Phi}{dA d\omega \cos\theta} \qquad 2.3$$

where Φ is the radiant flux contained in an elemental cone $d\omega$, arriving or leaving a differential area dA at an angle θ from the normal n. It is important to note that radiance is independent of irradiance or radiant exitance. The ANSI/IES 1986 [20] definition of radiance states the radiant flux density can be "*leaving, passing through or arriving at*" a surface.

Bidirectional Reflectance Distribution Function, BRDF (f_r)

Measured in inverse steradians (sr^{-1}), the *bidirectional reflectance distribution function* describes the reflective properties of a surface and is the ratio of the reflected radiance to the incident radiosity.

Reflectivity (ρ)

Reflectivity is the fraction of incident radiant energy that is reflected and is related to the BRDF of a surface. Reflectivity is defined as:

$$\rho(x) = \int_{\Omega} f_r(x) \cos\theta_r \, d\omega \qquad 2.4$$

where:

$f_r(x)$	is the BRDF
θ_r	is the direction (angle) of the reflected radiance, with respect to the normal
Ω	is the surrounding hemisphere around point x

If all surfaces are restricted to being Lambertian, then the *diffuse* reflectivity ρ is:

$$\rho(x) = f_r(x)\ \pi \qquad 2.5$$

2.3 The Radiosity Function

The radiosity (B) at any point on a surface is the integral of the radiance at that point over the surrounding hemisphere Ω. This is also true for radiant exitance (M) as this quantity indicates the direction of the radiosity at a point on a surface. Radiosity (B) and radiant exitance (M) are commonly interchanged. The radiosity equation is as follows:

$$B(x) = \int_{\Omega} L(x,\theta)\cos\theta\, d\omega \qquad 2.6$$

In realistic scenes, the radiance across object surfaces is dependent upon θ (the direction of the ray of radiant flux, with respect to the normal of the surface). Therefore the radiance function can be extremely difficult to solve, if at all, analytically. This would appear to make radiosity rendering unfeasible, but for most cases, we can assume all surfaces are *Lambertian*. Lambertian surfaces have the important property of having a constant radiance function that is independent of viewing direction. These surfaces are often referred to as *ideal diffuse* emitters or reflectors.

Since Lambertian surfaces are completely diffuse, the distribution of light that leaves a surface is independent of the angle of incidence from the arriving light. Therefore, if

we consider the light that falls onto a differential area on our Lambertian surface, we can approximate that light as an infinite number of geometric rays. Consequently we can use Lambert's cosine law to calculate the intensity of any ray that leaves the surface. Thus the intensity of a ray I_θ that leaves a Lambertian surface at angle θ from the surface normal, can be calculated as follows:

$$I_\theta = I_n \cos\theta \tag{2.7}$$

I_n is the intensity of a ray leaving at the surface normal. Thus using Lambert's Cosine Law (Equation 2.7), we can remove the directional dependence of light θ from the radiosity equation, with the following relationship:

$$L(x,\theta) \equiv L(x) \tag{2.8}$$

Therefore the radiosity equation simplifies to:

$$\begin{aligned} B(x) &= L(x) \int_\Omega \cos\theta \, d\omega \\ &= L(x) \int_0^{2\pi} d\phi \int_0^{\frac{\pi}{2}} d\theta \sin\theta \cos\theta \\ &= L(x)\pi \end{aligned} \tag{2.9}$$

We can see from Equation 2.9 that radiosity algorithms aim to simulate the inter-reflecting photic field between diffuse surfaces in an environment.

As radiosity is the radiant flux flowing through a unit area per unit time, radiosity can be expressed as irradiance or radiant exitance depending on whether radiant flux is arriving or leaving a surface. However, by definition, *radiosity is radiant exitance* [48][52]. Therefore it is common for the radiosity equation to be described in terms of radiosity, B, or radiant exitance, M. This can be a source of confusion, but as long as the definition of radiosity is kept in mind, radiosity and radiant exitance can be used interchangeably. Thus radiosity algorithms calculate the *radiant exitance* of a scene:

$$M(x) = L(x)\pi \tag{2.10}$$

where $M(x)$ is the radiant exitance function and $L(x)$ is the radiance function of a surface at a point x.

2.4 The Rendering Equation

In 1986 Kajiya [23] introduced the *Rendering Equation*. Based upon Maxwell's equation for electromagnetics, the rendering equation is essentially a geometrical optics approximation [23] and is a compact formulation of the global illumination problem [36][48]. The rendering equation is stated as:

$$I(x,x') = g(x,x')\left(\varepsilon(x,x') + \int_{x'\in S} \rho(x,x',x'')\, I(x',x'')dx''\right) \qquad 2.11$$

where:

- $I(x, x')$ is related to the intensity of light passing from point x' to point x;
- $g(x, x')$ is a "geometry" term and encodes the occlusion of surface points by other surfaces points [23];
- $\varepsilon(x, x')$ is related to the intensity of the emitted light from point x' to point x;
- $\rho(x, x', x'')$ is related to the intensity of light scattered from x'' to x by a surface element at x';
- S is the union of all the surfaces in the environment.

Tampieri [48] expresses Kajiya's rendering equation [23] in a more convenient form (Equation 2.12):

$$L(x,x'',\lambda) = L_e(x,x'',\lambda) + \int_{x'\in S} f_r(x',x,x'',\lambda) L(x',x,\lambda) \frac{\cos\theta\cos\phi}{r^2} v(x,x')\, dAx' \qquad 2.12$$

where:

$L(x, x'', \lambda)$	is the radiance at a point x in the direction of point x'', at wavelength λ;
$L_e(x, x'', \lambda)$	is the radiant exitance per unit solid angle emitted from x towards x'', at wavelength λ;
S	is the union of all the surfaces in the environment;
$f_r(x, x, x'', \lambda)$	is a minor reformulation of the *bidirectional reflectance distribution function* (BRDF);
$v(x, x')$	is a visibility term; it is 1 if x and x' are visible to each other and 0 otherwise;
x, x', x''	are differential surface elements;
θ, ϕ	are the angles between the surface normals at x and x' and the line connecting the two points (see Figure 2.4).

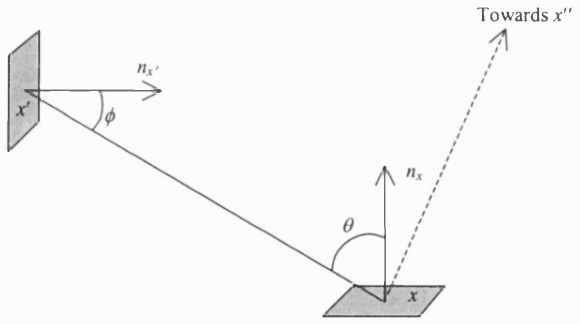


Figure 2.4 The geometry between two differential surface areas, used in Equation 2.1.

If we assume that all surfaces are Lambertian and omit the dependence on wavelength, we can further simplify the rendering equation in Equation 2.12 to:

$$L(x) = L_e(x) + f_r(x) \int_{x' \in S} L(x') \frac{\cos\theta \cos\phi}{r^2} v(x, x') dx' \qquad 2.13$$

where r is the distance between x and x'.

Equation 2.13 is known as the *Diffuse Rendering Equation* and is used by radiosity algorithms to solve the global illumination problem. From this equation, we can see

that the radiance functions $L(x)$ and $L(x')$ (the unknown quantities that need to be solved) occur both on the left hand side of the equation and inside the integral. These integral equations are known as *Fredholm integral equations of the second kind* [8][75] and are impossible to solve analytically in all but the simplest cases [83][84].

2.5 Solving the Radiosity Equation

For general scenes, it is impossible to solve the diffuse rendering equation analytically (Section 2.4). Therefore the only way to solve the (diffuse) rendering equation is numerically.

Firstly we need to rewrite the diffuse rendering equation in terms of radiosity B and radiant exitance M. The radiant exitance function $M(x)$ is explicitly stated here instead of as radiosity, because this term defines any light emitting surfaces that may exist in an environment.

From Equation 2.9, we can see that there is a relationship between radiosity B and radiance L. Therefore we have:

$$B(x) = M(x) + \rho_d(x) \int_{x' \in S} B(x') \frac{\cos\theta \cos\phi}{r^2} v(x, x')\, dx' \qquad 2.14$$

where $\rho_d(x)$ is the reflectivity at x.

Next we need to discretise Equation 2.14. Taking the finite element approach, each surface in a scene is approximated by dividing it into a fine mesh of *elements*. Each element is assumed to have constant radiosity. Hence for every element i, the area averaged radiosity B_i (Equation 2.15) and reflectance ρ_i (Equation 2.16) for a finite area surface S_i are given by:

$$B_i = \frac{1}{A_i} \int_{x \in S_i} \pi L_i(x) dA(x) \qquad 2.15$$

$$\rho_i = \frac{1}{A_i} \int_{x \in S_i} \pi f_{ri}(x) dA(x) \qquad 2.16$$

By integrating Equation 2.14 over the area of surface S_i, we derive the traditional radiosity equation (see Tampieri [48]):

$$B_i = M_i + \rho_i \sum_{j=1}^{N} B_j \frac{1}{A_i} \int_{x \in S_i} \int_{x' \in S_j} \frac{\cos\theta \cos\phi}{\pi r^2} v(x, x')\, dx' dx \qquad 2.17$$

Or more compactly:

$$B_i = M_i + \rho_i \sum_{j=1}^{n} B_j F_{ij} \qquad 2.18$$

where F_{ij} is the *form factor* between elements i and j, and is calculated as:

$$F_{ij} = \frac{1}{A_i} \int_{x \in S_i} \int_{x' \in S_j} \frac{\cos\theta \cos\phi}{\pi r^2} v(x, x')\, dx' dx \qquad 2.19$$

2.5.1 Form Factors

If a patch i is transmitting flux and a patch j is receiving flux, then the *form factor* F_{ij} is simply the ratio between the flux (Φ_{ij}) that reaches the receiving patch j and the flux that was transmitted (Φ_i) by patch i. See Figure 2.5 for an illustration.

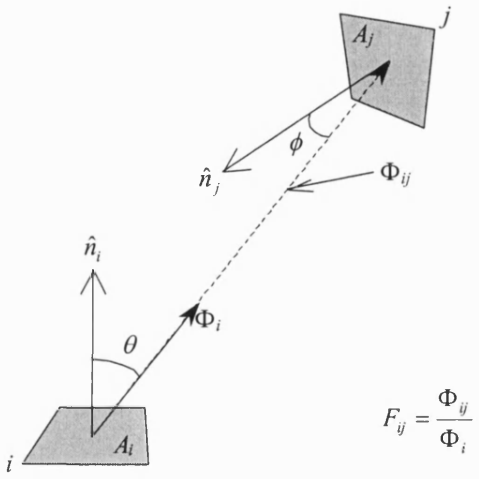


Figure 2.5 An illustration of the form factor between two patches *i* (emitter) and *j* (receiver).

From Equation 2.18 we can see that the most time consuming part of solving the radiosity equation, is computing the *form factor* (Equation 2.19) between two surface elements. Since form factors cannot be solved analytically, except for the most trivial cases, a numerical approach must be used.

To solve the form factor between two arbitrary patches i and j (as shown in Figure 2.5), we must first consider the form factor between two infinitesimally small differential areas. The solution for the form factor between two differential areas dA_i and dA_j is:

$$F_{dA_i \to dA_j} = \frac{\cos\theta_i \cos\theta_j}{\pi r^2} \qquad 2.20$$

From this point, it is now possible to consider the form factor between a differential area to a finite area, $F_{dA_i \to A_j}$. This is accomplished by integrating Equation 2.20 over the finite area A_j to give:

$$F_{dA_i \to A_j} = \int_{A_j} \frac{\cos\theta_i \cos\theta_j}{\pi r^2} dA_j \qquad 2.21$$

By integrating Equation 2.21 over the finite area A_i, the final 'full' form factor shown in Equation 2.22 can be solved for two finite areas:

$$F_{ij} = \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{\cos\theta_i \cos\theta_j}{\pi r^2} dA_j dA_i \qquad 2.22$$

Equation 2.22 is the same as Equation 2.19, but for *unoccluded* finite area patches.

When attempting to numerically solve the 'full' form factor equation, the first step is to consider Equation 2.21, the form factor between a differential area emitter dA_i and a receiver patch A_j. However care must be taken when using Equation 2.21, because we are in effect treating the differential area as a point source. Therefore it must be ensured that the distance between the emitter and the receiver must be greater than five times the maximum projected width of the emitter [52]. This is known as the *five-times rule* and was demonstrated by Murdoch [11] in a study in illumination engineering.

There is a surprisingly simple analytical solution for solving Equation 2.21. Providing that the patches are planar convex polygons, the form factor $F_{dA_i \to A_j}$ is:

$$F_{dA_i \to A_j} = \frac{1}{2\pi} \sum_{k=0}^{n-1} \beta_k \cos \alpha_k \qquad 2.23$$

If patch A_j has n vertices, then β_k is the angle (in radians) between the two vectors formed by each pair of consecutive vertices of A_j, and $\cos\alpha_k$ is the angle between the plane formed by the two vectors that were used to compute β_k and the plane containing dA_i. Figure 2.6(a) shows diagrammatically, the angles α and β.

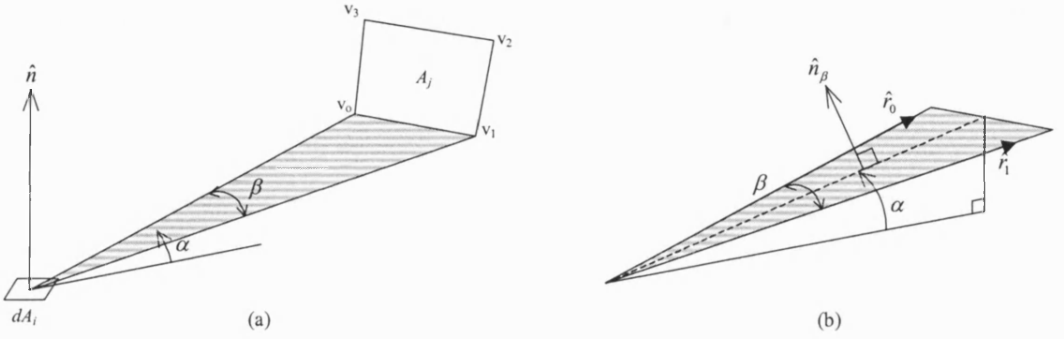


Figure 2.6 (a) A visual representation of how the angles α and β are constructed. (b) shows the vectors required to calculate α and β. Only calculations for *one* side of A_j are shown here.

Calculating β_k is a straightforward matter of computing the angle between two vectors, but it is not immediately obvious how to calculate cos α_k. Figure 2.6(b) shows in more detail how this would be accomplished for one side of A_i (i.e. $k=0$).

Thus for all sides, k=0 to n, by expressing cos α_k in terms unit vectors $\hat{r}_k$ and $\hat{r}_{k+1}$, and bearing in mind that A_j is a *closed* polygon, this enables Equation 2.23 to be expressed in a more convenient form. This is shown in Equation 2.24.

$$F_{dA_i \to A_j} = \frac{1}{2\pi} \sum_{k=0}^{n-1} \beta_k \hat{n}_i \cdot \left(\hat{r}_k \times \hat{r}_{(k+1)\%n} \right) \qquad 2.24$$

Equation 2.24 is in fact the contour integration solution to Equation 2.19 for the specialised case between an *unoccluded* (i.e. $v(x,x') = 1$) differential area and a finite area. This solution was introduced by Nishita and Nakamae [18], who generalised this method to account for partially occluded polygons. Nishita *et al.* recognised that since dA_i is infinitesimally small, it can be approximated as a point source. Thus, an area subdivision algorithm for hidden surface elimination, for example Warnock's algorithm, can be used to successively divide a polygon until all sub-polygons are either totally occluded, or totally unoccluded. The final form factor between dA_i and A_j is then, simply the sum of the form factors between dA_i and all the unoccluded sub-polygons of A_j.

Once the computation of the form factor $F_{dA_i \to A_j}$ is possible, F_{ij} can be computed by integrating over the finite area A_i.. For the method by Nishita *et al.* [18], providing that the patch dA_i is always sufficiently small that it can be approximated by a point source, this procedure is just a simple summing of all $F_{dA_i \to A_j}$ over the area A_i. That is:

$$F_{ij} = \frac{1}{A_i} \int_{A_i} F_{dA_i \to A_j} \to \frac{1}{A_i} \sum_{n} F_{dA_i \to A_j} \qquad 2.25$$

Nishita *et al.* [18] showed that their algorithm worked well for complex scenes with partially occluded polygons. However, the implementation and integration of Warnock's algorithm into form factor calculation is complex, thus many methods adopt numerical integration or stochastic techniques instead [52].

To date, the most popular methods are:

- The Hemicube method [19]
- The Cubic Tetrahedron method [35]
- The Ray casting method [33]

The first two methods are numerical integration techniques, known as *numerical quadrature*, that are based upon Nusselt's Analogy, which is discussed in Section 2.5.2.

The Ray casting method is another method for solving the form factor problem. Ray casting is a stochastic sampling method that combines Monte Carlo and ray-tracing techniques, to calculate the form factor between two patches. As a result, ray casting algorithms are much simpler and easier to implement than the numerical quadrature techniques i.e. the hemicube and cubic tetrahedral methods.

Ray casting has many attractive features: this technique can handle both planar and curved surfaces as well as non diffuse and transparent surfaces. More importantly, the ray casting method determines the form factors at the vertices of each patch. Hence the radiant exitance at each vertex of a patch can be calculated directly. This is required when shading the patches for final output to the display. In contrast, the hemicube and cubic tetrahedron methods compute the radiant exitance for the whole patch. Thus some form of interpolation is required to obtain the radiant exitances at the vertices.

Despite the advantages of ray casting, a large number of rays must be cast per form factor calculation, to approach the accuracy of the hemicube or cubic tetrahedron methods. Since casting rays is a costly procedure, potentially requiring $O(n^2)$ comparisons, this is perhaps the most major disadvantage of the ray casting technique. However, well proven ray-tracing acceleration techniques [16][27] can be applied to improve the performance of ray casting.

2.5.2 The Hemisphere Method: Nusselt's Analogy

Nusselt's analogy [1] is a purely geometric solution for computing the form factor between a differential area polygon dA_i and a finite area receiving polygon A_j (Equation 2.21). Nusselt's analogy works by projecting the outline of the receiver patch onto a hemisphere of unit radius, then directly projecting the outline onto the base of the hemisphere. See Figure 2.7.

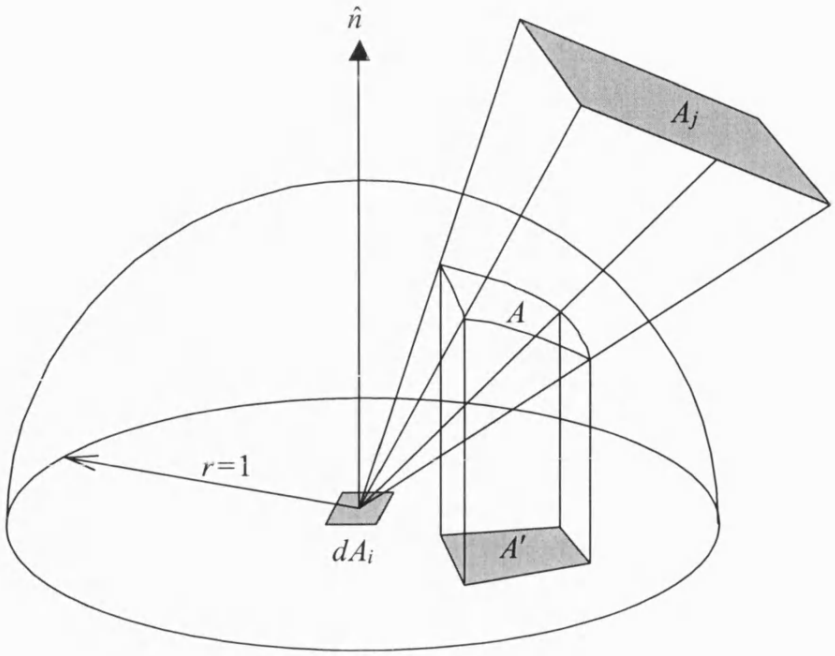


Figure 2.7 Nusselt's Analogy.

From Figure 2.7, the form factor $F_{dA_i \to A_j}$ is:

$$F_{dA_i \to A_j} = \frac{A'}{\pi} \qquad 2.26$$

Although Nusselt's analogy produces a very compact solution for calculating the form factor between dA_i and A_j, the major obstacle here is calculating A. Projecting A' onto the base of a unit hemisphere is anything but trivial!

From Nusselt's analogy, we obtain a very important relationship. The solid angle $d\omega$ that is subtended by dA_j as seen from dA_i, is equal to the projected area of dA_j onto the *surface* of a unit hemisphere. Hence the total projected area A' is calculated by integrating over the finite area A_j. From this relationship, computing the area A' can be accomplished numerically by breaking the hemisphere into small delta solid angles and summing all the covered delta solid angles that formed the projection onto the surface of the hemisphere.

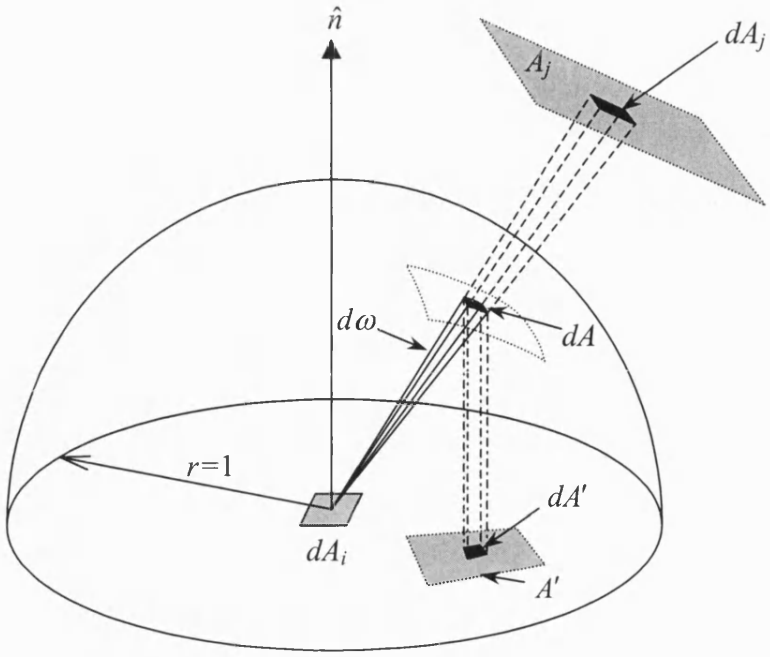


Figure 2.8 Computing the form factor using Nusselt's analogy.

Referring to Figure 2.8, if we consider a delta solid angle constructed by the centre of the differential area dA_i and the differential area dA_j of A_j, this will cause a projection dA onto the surface of the hemisphere. This projected area dA can be calculated as follows:

$$dA = \cos\theta_j d\omega \qquad 2.27$$

where $d\omega$ is the solid angle subtended by dA_j as seen from dA_i, and θ_j is the angle between the normal of dA_j and vector between dA_j and dA_i. See Figure 2.8.

Thus the projected area dA' onto the base circle of the hemisphere is:

$$dA' = \cos\theta_i d\omega \qquad 2.28$$

The final projected area A' (as shown in Figure 2.7) can then be calculated by integrating over the area A_j:

$$A' = \int_{A_j} \cos\theta_i d\omega \qquad 2.29$$

Since $d\omega = \frac{\cos\theta_j dA_j}{r^2}$, Equation 2.29 becomes:

$$A' = \int_{A_j} \frac{\cos\theta_i \cos\theta_j dA_j}{r^2} \qquad 2.30$$

Finally, Equation 2.30 can be substituted into Equation 2.26 to obtain the form factor $F_{dA_i \to A_j}$ as shown in Equation 2.21.

2.5.3 The Hemicube Method

If the hemisphere in Nusselt's analogy is replaced by a *hemicube*, we have Cohen and Greenberg's [19] *hemicube* method for solving form factors. Cohen and Greenberg realised that patches that have the same projected area on a hemisphere (see Figure 2.9(a)), will have the same solid angle as seen from the emitting patch. Hence these patches will have the *same* form factor. Also, if the same projected area on the surface of the hemisphere is projected onto any other surrounding surface, then that projected area, although different from the projected area on the hemisphere, will also have the *same* form factor. See Figure 2.9(b). Therefore, any geometric object can replace the hemisphere of Nusselt's analogy.

Cohen and Greenberg replaced the Nusselt's hemisphere by a geometrically similar object: the hemicube. The subdivision of all faces of the hemicube into a regular grid of cells is analogous to breaking up the hemisphere into smaller delta solid angles.

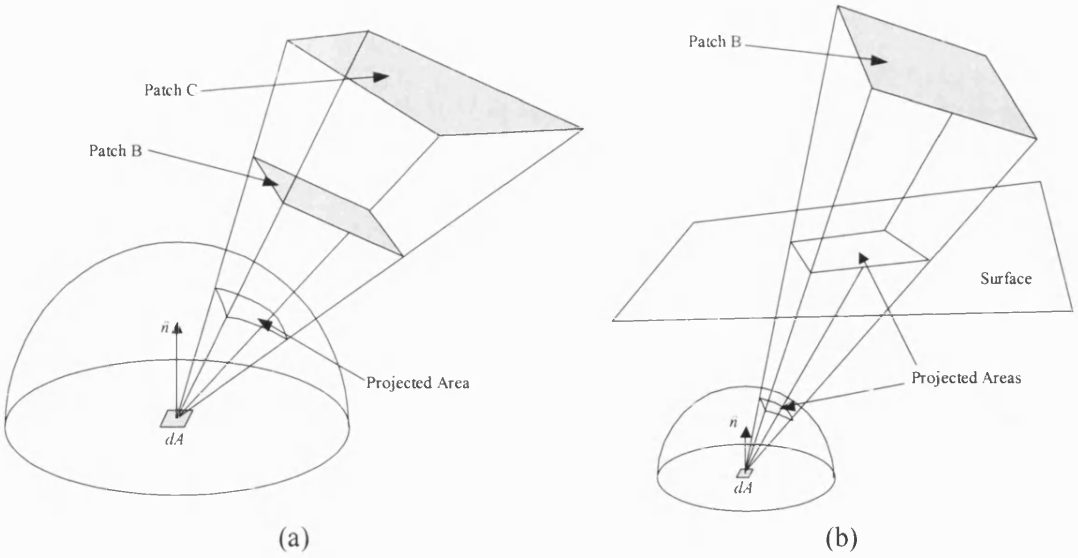


Figure 2.9 Two diagrams illustrating two patches that (a) have the same projected area and hence the same form factor and (b) the same projected area and hence the same form factor on any surrounding surface.

Thus, to calculate the form factor between two patches, the target patch is projected onto each face of the hemicube and the sum of all the delta form factors of the cells that were covered, is the final form factor. See Figure 2.10.

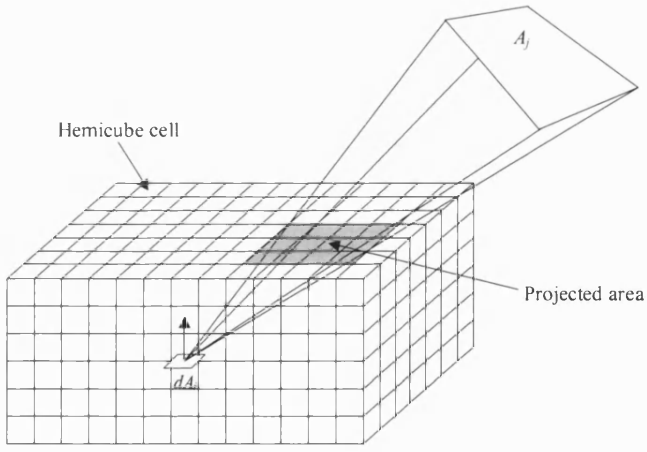


Figure 2.10 Calculating form factors using the Hemicube method.

The number of cells on the top face defines the resolution of the hemicube. Typically, hemicube resolutions range from 32x32 to 1024x1024 [52]. Cohen and Greenberg [19] used resolutions between 50x50 and 100x100.

The advantage of using the hemicube method, is that the delta form factors of each cell in the grid can be pre-calculated and stored in a convenient lookup table. Due to the eight-way symmetry of the top face and the two-way symmetry of the four half faces, for a hemicube with resolution n x n cells, only $3n^2/8$ values need to be stored in the lookup table [19][52]. Each delta form factor is calculated from the form factor between two differential areas (see Equation 2.20).

Also, since each face of a cube exactly covers a 90° frustum as viewed from the centre of a cube, existing clipping algorithms can be reused for calculating the projections of patches. In fact, the 90° frustum simplifies frustum clipping calculations, thus polygon clipping algorithms (e.g. Sutherland-Hodgman [5]) can be streamlined to make use of this advantage [19].

A detailed implementation of the hemicube method by Cohen and Greenberg [19], is given by Ashdown [52].

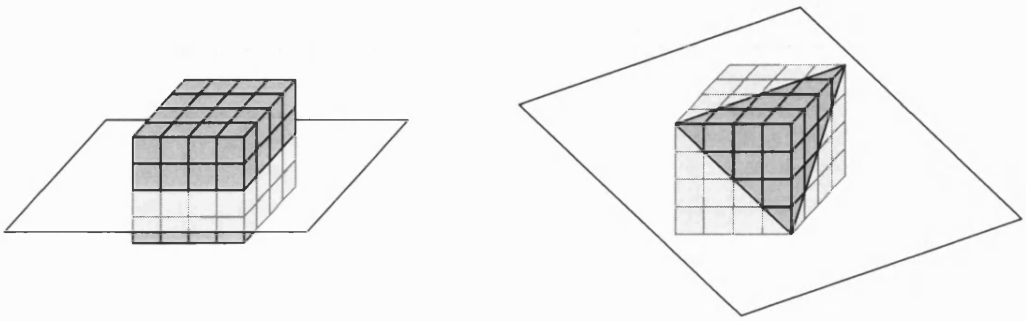
2.5.4 The Cubic Tetrahedron Method

The problem with Cohen and Greenberg's hemicube method [19] is that every patch within a scene has to be projected onto five separate planes; the top face and four half faces of the hemicube. See Figure 2.11(a). Ashdown [52] describes this as a 'nuisance', but for a scene with a large number of patches, this will hinder the performance of form factor calculation.

As mentioned in Section 2.5.3, Nusselt's hemisphere [1] can be replaced by *any* object that surrounds the differential area. This is because the projected area of a polygon onto the surface of a hemisphere will have *exactly* the same form factor if it was to be projected onto any surrounding surface [19]. See Figure 2.9 for an illustration. The actual size and type of volume object used for the projection medium, is arbitrary, since it is the *form factor* of the projected area that is required.

Beran-Koehn and Pavicic [35] improve upon Cohen and Greenberg's method by replacing the hemicube by a simpler geometric object known as the *cubic tetrahedron*.

The cubic tetrahedron object is formed when a cube that has been sliced by a plane, such that three of the cube's vertices lie in the slicing plane. Essentially the cubic tetrahedron object is nothing more than a tetrahedron that has been constructed from a cube. See Figure 2.11(b).



(a) (b)

Figure 2.11 Constructing (a) the hemicube and (b) the cubic tetrahedron.

The simple geometry of the cubic tetrahedron reduces the number of projection planes to three identical triangular faces. Considering that a scene may contain many hundreds or thousands of polygons, this is a significant improvement.

As with the hemicube method, when calculating the delta form factors for the cubic tetrahedron, the cells are square. However, at the base of each face of the cubic tetrahedron are a series of triangular cells, where the plane has sliced the cube. Beran-Koehn and Pavicic [43] suggest two solutions to this. If these cells are included in the delta form factor calculations, then the delta form factors must be adjusted to account for the area being only half that of the square cells. Otherwise, if the resolution of the cubic tetrahedron is suitably high, then these patches can be ignored.

Ashdown [52] has a highly detailed implementation of a progressive radiosity renderer that uses the Beran-Koehn and Pavicic's cubic tetrahedral method for calculating form factors [35][43].

The cubic tetrahedron method has two main advantages over Cohen and Greenberg's hemicube method. The first advantage is that each of the three faces of the cubic tetrahedron is identical. Therefore this simplifies the implementation of this method. Secondly, there are only three faces belonging to the cubic tetrahedron, which means that only three viewing frustums need to be used. As a result, fewer patches need to be clipped, compared to the hemicube method. This is an important result as the following benefits are gained:

- Reduced intersection testing
- Reduced polygon clipping
- Reduced hidden surface calculations

Intersection testing, polygon clipping and hidden surface removal are all expensive computations.

However due to the simpler geometry of the cubic tetrahedron, this method only samples a scene with three half-faces. This is less than the hemicube method, which samples a scene with one full face and four half-faces. This can be seen from Figure 2.11, which shows the construction of a hemicube and a cubic tetrahedron.

Thus, the resolution of the cubic tetrahedron must be doubled for the sampling to be the same as the hemicube method. As a result, the storage requirements for the delta form factor lookup table for the cubic tetrahedron will be slightly larger than for the hemicube.

Overall, the cubic tetrahedron method has many attractive benefits over the hemicube. In particular, the reduction in the number of projection planes that need to be considered will immediately improve the efficiency of form factor calculation. Ashdown [52] reports that the cubic tetrahedral method needs to perform on average, 2.75 clipping operations per polygon, compared to 3.83 operations for the hemicube method.

2.5.5 The Ray-casting Method

The form factor between two polygons can be calculated using ray-casting techniques. Ray-casting uses stochastic or Monte Carlo techniques to obtain a set of randomly distributed points across the surface of a polygon, from which rays are constructed and cast into the environment.

Early ray-casting techniques by Maxwell *et al.* [25] and Malley [30] used ray-tracing methods to calculate form factors, but it was Wallace *et al.* [33] that combined ray-tracing methods for calculating form factors with stochastic sampling. This led to the production a simple but effective algorithm for progressive refinement radiosity, that generates fewer visual artefacts and is algorithmically much less complex, than existing numerical integration methods (e.g. hemicube method, cubic tetrahedral method).

From Maxwell *et al.* [25], the radiative exchange process between surfaces in an environment are modelled by following the progress of discrete 'bundles' of energy, as they propagate and interact within the scene. Thus, the form factor (or 'configuration factor' as it is referred to by Maxwell *et al.* [25]) is equal to the fraction of the total energy bundles emitted from a surface that are incident upon a second surface.

The basic idea of Maxwell *et al.'s* method [25] for ray-casting is based upon Eckert's experimental method for determining form factors [2]. Eckert experimentally modelled Nusselt's analogy (see Figure 2.7) by using a hemispherical milk-glass[1] light fixture and a small electric light bulb (centred at the base of the light fixture) to represent the unit hemisphere (the milk-glass light fixture) and the differential surface element (the light bulb). This is shown in Figure 2.12.

An opaque model of the surface for which the form factor was to be determined, was oriented and positioned within the milk-glass light fixture. Then in a darkened room, the light bulb was illuminated and the whole assembly, photographed. Since the

[1] The 'milk-glass' definition of the light fixture describes the glass to be semi-opaque and is of a milky colour.

camera is located in the line of sight that is normal to the base of the hemispherical milk-glass light fixture, the shadow cast by the opaque object onto the milk-glass is approximately equivalent to the projected area onto the base of the milk-glass. Thus by measuring the area of the shadow and the base of the circle from the photograph, the form factor is simply the ratio between the two areas.

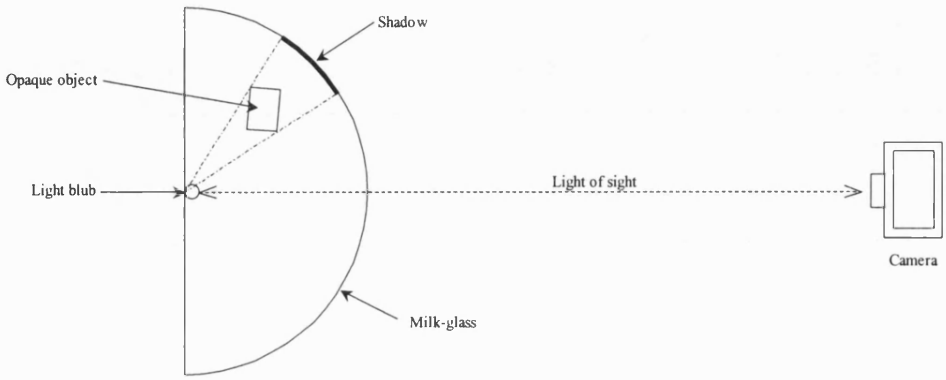


Figure 2.12 Eckert's experimental arrangement for calculating form factors.

Maxwell *et al.* [25] translated Eckert's experiment into a method for computer graphics by replacing the physical components of the experiment, by computer models. From Figure 2.12, the light bulb, milk-glass and opaque object were modelled as the differential area, the unit hemisphere and the finite area, respectively, of Nusselt's analogy. However, to calculate the area of the shadow that was projected onto the surface of the unit hemisphere, a ray-casting method was developed.

Maxwell *et al.* [25] replaced the photograph used by Eckert [2], by a plane that lies on top of, but is tangential to, the unit hemisphere (see Figure 2.13). This plane is then discretised into a regular grid of n by n cells. The projected area of a surface (shown as A_j in Figure 2.13) can be computed by casting rays from the centre of the differential area (which is centred about the base circle of the hemisphere) through the centre of each cell on the tangential plane. Every cell for which the ray shot was unobstructed, is marked as 'being in light'. Otherwise the ray was blocked and thus the cell was marked as 'being in shadow'.

As with Eckert's experiment, it is the shadow area that will be used to calculate the form factor. Since the area of each cell is known, by summing the areas for all cells that are in shadow, the total shadow area can be computed. The form factor is then simply the shadow area divided by the area of the base circle of the hemisphere. As a unit hemisphere was used, the area of the base circle is π. Maxwell *et al.* [25] used a raster graphics screen to represent the tangential plane, thus each cell of the plane corresponds to a pixel on the screen.

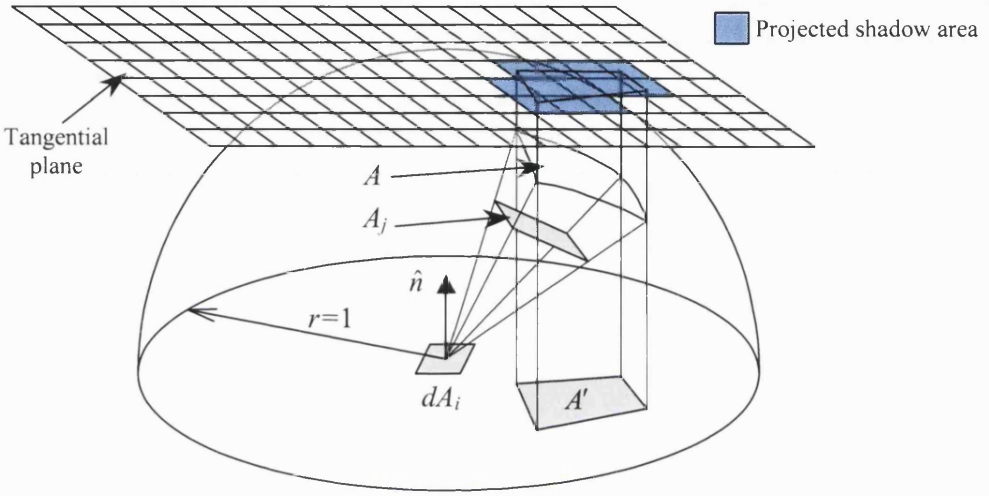
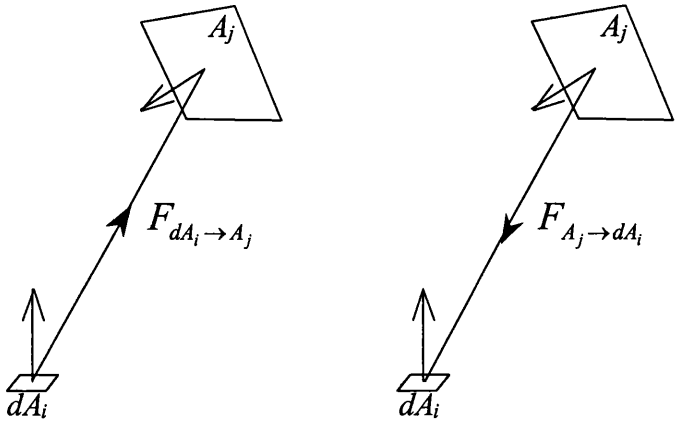


Figure 2.13 An illustration of Maxwell *et al's.* ray-casting method.

Wallace *et al.* [33] introduced an improved progressive refinement radiosity solution that eliminates the aliasing artefacts and implementation complexities of the hemicube method, by replacing the hemicube method with a ray-tracing algorithm for computing form factors. Wallace *et al.* [33] improved the ray-tracing techniques by Maxwell *et al.* [25] and Malley [30] to produce a ray-tracing method of calculating form factors that is not based upon Nusselt's analogy.

Based upon the shooting method of progressive radiosity, this method takes the novel approach of computing the form factors between every element vertex in the scene and every source patch that has energy to shoot. Therefore, the most significant difference between this method and all previous techniques including Maxwell *et al.* [25] and Malley [30], is that illumination can be computed directly at every vertex in the scene. All other methods compute the illumination at the centre of a patch, which is then assumed to be constant across the surface of the patch. Therefore, to compute the

illumination at the vertices, it is necessary to interpolate between neighbouring patches. This in turn is an approximation, and introduces inaccuracies and hence artefacts into the final solution.



(a) (b)

Figure 2.14 Two diagrams illustrating (a) the form factor between a differential area and a finite patch and (b) the form factor between a finite area and a differential area.

Calculating the form factors at the vertices is very similar to existing methods of calculating form factors. The only difference is that this method calculates the form factors between a finite area and a differential area (Figure 2.14(b)), that is, the reverse of the usual way (Figure 2.14(a)). Calculating the form factors between two areas (finite or differential) is made very simple, by making use of a very important relationship known as the *reciprocity principle* (Equation 2.31).

$$A_i F_{ij} = A_j F_{ji} \qquad 2.31$$

In essence, the reciprocity principle states that, given the form factor F_{ij} from patch i to patch j, the form factor in the reverse direction F_{ji}, is a simple ratio of the areas of the two patches.

Wallace *et al.* [33] solved Equation 2.21 by dividing the source polygon, A_j (see Figure 2.15) into delta areas, ΔA_j, such that the delta areas represent the differential area dA_j in

Equation 2.21. Thus Equation 2.21 can be expressed by the following summation (Equation 2.32):

$$F_{dA_i \to A_j} = \sum_{k=1}^{n} \frac{\cos\theta_{1k} \cos\theta_{2k}}{\pi r_k^2} \Delta A_j \qquad 2.32$$

However, it was reported [33] that Equation 2.32 would grow without bound if ΔA_j does not 'shrink' as the distance r becomes less that unity. The solution to the problem was to explicitly treat the delta areas as finite areas. This can be thought of as the calculation of a *delta form factor*.

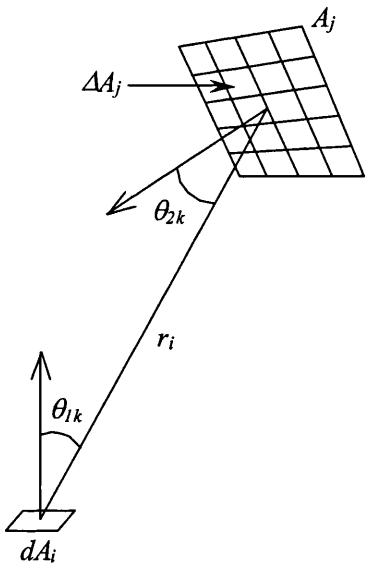


Figure 2.15 Numerical integration method of Equation 2.21 (from Wallace *et al.* [33]).

To keep the form factor calculations simple, Wallace *et al.* [33] used the analytical form factor (from Howell [12]) between a differential area and a finite disk that are parallel and face each other (see Figure 2.16), to represent dA_i and A_j respectively.

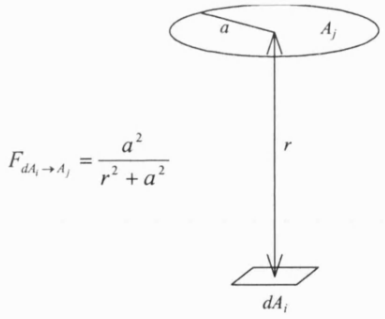


Figure 2.16 The analytical form factor between a differential area and a finite disk.

Thus the equation for the form factor becomes:

$$F_{dA_i \to A_j} = \frac{a^2}{r^2 + a^2} \quad , \quad A_j = \pi a^2 \therefore a^2 = \frac{A_j}{\pi}$$
$$\therefore F_{dA_i \to A_j} = \frac{A_j}{\pi r^2 + A_j} \qquad 2.33$$

Equation 2.33 can be generalised such that the differential area and finite disk can have different orientations, as shown in Figure 2.17. It must be noted that this equation (Equation 2.34), is an approximation.

$$F_{dA_i \to A_j} = \frac{A_j \cos\theta_i \cos\theta_j}{\pi r^2 + A_j} \qquad 2.34$$

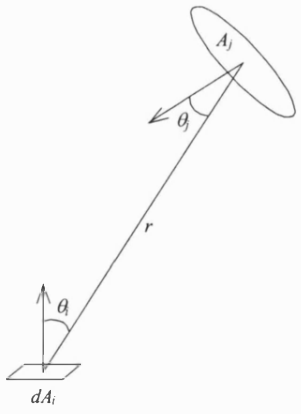


Figure 2.17 An approximate but generalised analytical form factor between a differential area and a finite disk.

Bearing in mind that it is the form factor between a finite area disk and a differential area i.e. $F_{A_j \to dA_i}$, that needs to be computed, the reciprocity principle (Equation 2.31) needs to be applied to Equation 2.34 as follows.

If i is the differential area dA_i and j is the finite area A_j, then we have the following reciprocity principle:

$$dA_i F_{dA_i \to A_j} = A_j F_{A_j \to dA_i} \qquad 2.35$$

Rearranging Equation 2.35 such that $F_{A_j \to dA_i}$ appears on the left hand side, we have:

$$F_{A_j \to dA_i} = \frac{dA_i}{A_j} F_{dA_i \to A_j} \qquad 2.36$$

Substituting Equation 2.36 into Equation 2.34, we finally have the form factor $F_{A_j \to dA_i}$, as shown in Equation 2.37.

$$F_{A_j \to dA_i} = \frac{dA_i}{A_j} \cdot \frac{A_j \cos\theta_i \cos\theta_j}{\pi r^2 + A_j} = \frac{dA_i \cos\theta_i \cos\theta_j}{\pi r^2 + A_j} \qquad 2.37$$

To compute the form factors for every element vertex in a scene, it is simply a case of evaluating Equation 2.37 as shown in Figure 2.18.

Wallace *et al.* [33] state that it is only necessary to determine the form factor between a source surface A_j to a vertex V on A_i (Figure 2.18(b)), rather than to the differential area dA_i (Figure 2.18(a)). Hence, this forms the basis of Wallace *et al's* ray-tracing method for calculating form factors.

The last step is to integrate over the source polygon A_j to obtain the total form factor. Performing this stochastically requires the source polygon to be sampled. If the sample points on the source polygon have been distributed evenly [33] (as shown in Figure 2.18), it can clearly be seen that the total form factor is simply the sum of each ray-traced *delta* form factor calculation, between each delta area (i.e. a sample point) on the source polygon A_j and every vertex on the receiving polygon A_i. However there are aliasing problems with this method, which are outlined by Wallace *et al.* [33].

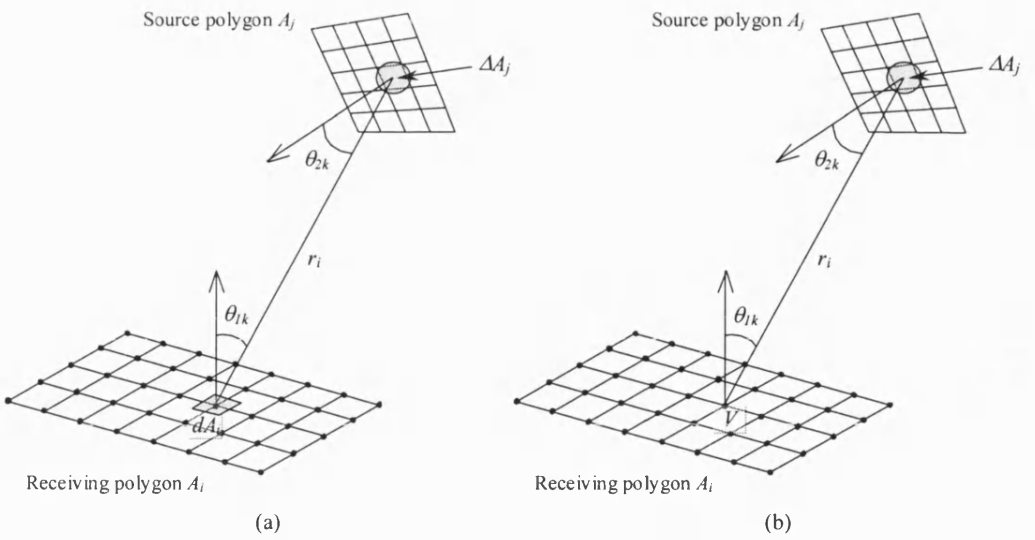


Figure 2.18 (a) Computing the form factors between a finite area and a differential area. (b) A simplified form factor calculation between a finite area and a vertex.

These aliasing artefacts typically manifest themselves along shadow boundaries, where insufficient sampling causes overlapping sharp edged shadows to appear, instead of a single soft edged shadow. One method for addressing this problem is to filter the radiosity at all vertices, by applying a weighted average of the radiosities of all neighbouring vertices [33]. However, the most successful method for addressing the sampling problem is to *jitter* the sample points on the source polygon (e.g. Cook [21]).

Jittering is a powerful method of random sampling and has the effect of changing the aliasing into noise.

All calculations so far are only valid for *unoccluded* source and receiver polygons. In general scenes, polygons are very rarely unoccluded, therefore to cater for polygon occlusion, a visibility δ is added to the form factor calculation. This is a simple boolean value that represents whether or not a ray was obstructed when it was cast from a vertex on the receiver polygon to a sample point on the source polygon. If the ray was obstructed, then δ would be zero and thus the delta form factor would be zero.

This finally leads to a general equation (Equation 2.38) for evaluating the form factor between a general source polygon A_j to a vertex on A_i:

$$F_{A_j \to dA_i} = \frac{dA_i}{n} \sum_{k=0}^{n} \delta_i \frac{\cos\theta_{1k} \cos\theta_{2k}}{\pi r_i^2 + A_j/n} \qquad 2.38$$

2.6 Computing the Radiosity Equation

As the radiosity equation (Equation 2.18) is simply a set of linear simultaneous equations, it is commonly expressed in matrix form (Equation 2.39).

$$\begin{bmatrix} M_1 \\ M_2 \\ \cdots \\ M_n \end{bmatrix} = \begin{bmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \cdots & -\rho_1 F_{1n} \\ -\rho_2 F_{21} & 1-\rho_2 F_{22} & \cdots & -\rho_2 F_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ -\rho_n F_{n1} & -\rho_n F_{n2} & \cdots & 1-\rho_{1n} F_{nn} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \cdots \\ B_n \end{bmatrix} \qquad 2.39$$

A variety of mathematical techniques can be used to solve the radiosity equation. The two most common methods of solving Equation 2.39 are:

- direct methods
- iterative methods

The main disadvantage of radiosity is that a typical scene will need to be decomposed to a very large number of surface elements. Since the overall size of the matrix is proportional to the number of surface elements *squared*, the computational cost for solving the radiosity equation (Equation 2.39) is therefore very high. Solving this equation using direct methods such as Gaussian elimination would require $O(n^2)$ space and $O(n^3)$ time [52] to compute. Hence, we can see that with an increasing number of surface elements the computational cost can very rapidly escalate and thus become unusable for solving scenes of any significance. For example, Goral *et al.* [15] used the Gaussian elimination approach with partial pivoting.

The classic alternative to direct methods like Gaussian elimination is to use iterative techniques such as the popular Jacobi and Gauss-Seidel methods for solving simultaneous equations. Nishita and Nakamae [18], Cohen and Greenberg [19], and Cohen *et al.* [22] used these methods for solving the radiosity equation. These iterative methods improve the time complexity for solving the radiosity equation to

$O(n^2)$ which is better than Gaussian elimination, but is still insufficient for large n. Nishita and Nakamae [18] note that setting up the n by n form factor matrix of Equation 2.39 for a large n, would require excessive amounts of memory. Thus to gain any significant performance increase, an improved approach to handling the radiosity equation needs to be considered.

2.7 Progressive Refinement Radiosity

Cohen *et al.* [31] developed a progressive refinement radiosity algorithm that can generate an image in $O(n)$ time and space. This algorithm is based upon the idea of *shooting flux*, whereby each light source in the scene *shoots* its flux to all the other elements (see Figure 2.19). Each element effectively becomes a secondary light source and hence shoots the flux they receive, back into the environment.

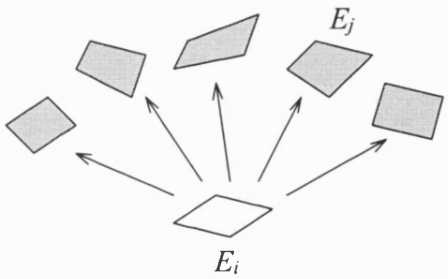


Figure 2.19 An illustration of shooting flux into the environment.

The progressive refinement algorithm is an iterative process that is still based upon the Gauss-Seidel approach. Gortler *et al.* [47] establish that the progressive refinement radiosity method is in fact a variant of the *Southwell* Relaxation iterative method.

Each iteration step takes $O(n)$ time to compute. However it must be noted that it still takes this algorithm $O(n^2)$ time to *completely* solve the radiosity equation. As with the Jacobi and Gauss-Siedel iterative methods, the progressive refinement algorithm terminates when the solution at the end of each iteration, falls within a specified error margin.

The main criterion for the progressive refinement radiosity method is to generate a *useful* image rather than a complete solution, in the most efficient manner, that is, in the quickest possible time. Therefore, Cohen *et al.* [31] restructured the radiosity algorithm in two important ways.

Firstly, in order for a valid image to be available after each iteration, the radiosity of all patches in the scene must be updated simultaneously. Secondly, to optimise the speed of convergence to which a solution is obtained, patches are processed by always selecting the patch that has the greatest amount of flux to shoot.

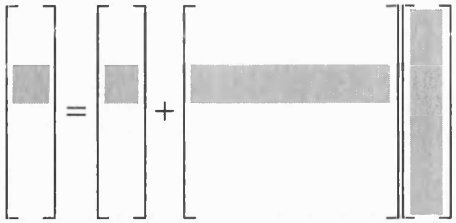
2.7.1 Gathering Light

In conventional radiosity algorithms, the Gauss-Seidel or Jacobi iterative methods solve the radiosity matrix of Equation 2.39 one row at a time. Referring to Equation 2.18, this is can be visualised by Figure 2.20(a).

The radiosity of each patch B_i in the scene is solved in turn, by summing the radiosity estimates of every patch B_j in the present iteration. This process can be intuitively thought of as *gathering* the radiosity from all patches B_j in the scene onto a single patch B_i. See Figure 2.20(b).

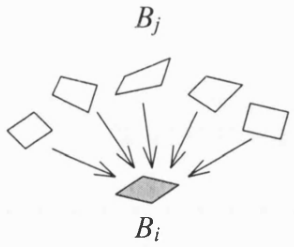
The 'gathering' of radiosity is a colloquial term used to describe the process by which the Gauss-Seidel or Jacobi iterative methods, that are used by traditional radiosity algorithms, to solve the radiosity matrix (Equation 2.39).

As there are n patches in a scene, the radiosity must be gathered onto every patch in the scene. Therefore for n patches within a scene, Equation 2.18 must be computed n times for one iteration step to be completed. As a result, computing one iteration step has a time complexity cost of $O(n^2)$. Unfortunately the $O(n^2)$ running time complexity does not scale well. Since a typical scene may contain a large number of patches, this method quickly becomes unfeasible.



$$B_i^{(k+1)} = M_i + \rho_i \sum_{j=1}^{n} F_{ij} B_j^{(k)}$$

(a)



(b)

Figure 2.20 (a) An illustration of Equation 2.18 for the gathering of radiosity and (b) a pictorial illustration.

2.7.2 Shooting Light

It is possible to reverse the process of Equation 2.18 such that the radiosity of a patch i is shot into the environment of all other patches j. This is achieved by applying the reciprocity principle shown in Equation 2.31 to Equation 2.18 as follows.

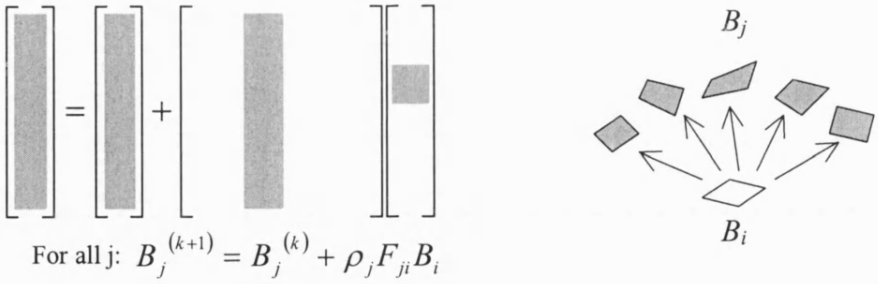
From Cohen *et al.* [31], the contribution of radiosity made by patch i *from* patch j is:

$$B_{i \to j} = \rho_i B_j F_{ij} \qquad 2.40$$

Therefore the contribution of radiosity made by a single patch i *to* a single patch j, using the reciprocity principle is:

$$B_{j \to i} = \rho_j B_i \frac{F_{ij} A_i}{A_j}, \quad where \frac{F_{ij} A_i}{A_j} = F_{ji} \qquad 2.41$$

This is true for all patches j, so Equation 2.41 must be applied to all patches j. This can be seen from the illustrations in Figure 2.21. Unlike the gathering method (Section 2.7.1), Equation 2.41 can be solved in constant time, as the form factors F_{ij} are pre-computed (for run time efficiency) and stored in lookup tables. Thus for n patches in a scene, the running time complexity to solve Equation 2.41 for all patches is O(n).



(a) (b)

Figure 2.21 (a) An illustration of Equation 2.18 for the shooting of radiosity and (b) a pictorial illustration.

2.7.3 Shooting vs. Gathering

To meet the criterion for progressive refinement radiosity, an approximate image rather than a complete image, must be available at interactive speeds.

Considering the gathering method in Section 2.7.1, a single iteration of the gathering method only solves the radiosity for a single patch. Therefore, to obtain an image would require Equation 2.18 to be solved for all patches *i*. Unfortunately, this will take $O(n^2)$ time and so this method does not fulfil the criterion for a progressive refinement algorithm. Although the physical interpretation of the gathering method can be easily identified from the radiosity matrix of Equation 2.39, as shown in Figure 2.20, it is too expensive to compute at interactive speeds. However, the gathering method does have the advantage of being able to solve the radiosity of specifically selected patches, after every iteration.

The shooting method is an entirely different paradigm, whereby the patch with the greatest amount of flux (i.e. the product of the radiosity and the area of a patch) to shoot, distributes its radiosity to *all* other patches within the environment. This is because there is no point distributing the radiosity of a patch that has little or no energy to shoot.

Shooting the flux for a single patch to another single patch within a scene can be accomplished in constant time, as the form factors F_{ij} have already been pre-computed and the form factors F_{ji} can be calculated on the fly using the reciprocity principle shown in Equation 2.31. Therefore to shoot flux to n other patches in the scene, can be computed in O(n) time and hence is suitable for generating approximate images after every iteration.

Initially, only light sources will shoot flux. However as the iterative process proceeds, patches within the environment will accumulate energy. In turn, these patches may gather enough energy to become 'secondary light sources' and shoot flux back into the environment.

2.7.4 The Algorithm

The progressive refinement algorithm basically consists of the following steps [34]:

1. Initialisation
2. Find shooting patch
3. Compute form factors
4. Distribute radiosity
5. Update and display results
6. Goto 2

The initialisation step initialises the radiosity values of every patch to its initial exitance value. Initially, only light sources will have a non zero value.

The iterative procedure commences by finding the patch with the greatest unshot energy (flux) to shoot into the scene. However if the remaining unshot energy within the entire scene is less than a predetermined value, then the radiosity equation has converged to a solution. From Ashdown [52], this is calculated as follows:

$$\sum_{i=1}^{n} \Delta\Phi_i \leq \varepsilon \quad where\ \Delta\Phi_i = \Delta B_i A_i$$

where ΔB_i is the unshot radiosity (radiant exitance) of patch i, A_i is the area of patch i, and ε is a predetermined threshold.

The form factors using the hemicube method can either be computed on the fly, or pre-computed before hand and stored in a lookup table. Cohen *et al.* [31] compute the form factors on the fly to avoid the $O(n^2)$ start up costs, as does Chen [34]. Thus for every patch, the hemicube is placed on the centre of the shooting patch and project all other patches onto all five hemicube faces. By summing all the delta form factors the form factor is obtained for two finite area patches.

The next step is to distribute the radiosity of the shooting patch to every other patch in the scene. This is done by applying Equation 2.41 to all patches in the scene, but not including the shooting patch. Once the shooting patch has distributed its energy into the scene, an approximate image is available for viewing. However this stage maybe postponed until a solution has been fully converged. The pseudo code for the progressive refinement radiosity method is as follows:

```
// Initialise the radiosities of all patches:
FOR every patch i DO
   ΔBi = initial radiosity of patch i
ENDFOR

WHILE Σ ΔΦk for every patch k > ε DO
   Select patch i with the greatest unshot flux ΔΦi
   Compute patch-patch form factors Fij
   FOR every patch j DO
      Δrad = ρjΔBiFji
      ΔBj += Δrad  // update unshot radiosity
      Bj += Δrad   // update total radiosity of patch j
   ENDFOR
   ΔBi = 0  // reset unshot radiosity for patch i
ENDWHILE
```

where:

ΔB is the unshot radiosity of a patch;

$\Delta \Phi$ is the unshot flux of a patch;

Δrad is the amount of flux contained by a shooting patch;

ρ is the reflectance of a patch.

During the early stages of progressive refinement, the initial images will most often appear quite dark. This is because only radiosity from the primary light sources would have been distributed into the scene. Most of the illumination within a scene actually comes from indirect or secondary illumination between the surfaces (patches) within a scene. As a result, secondary illumination has yet to contribute to energy to the current solution iteration, thus the global illumination of the scene has not yet been represented accurately.

In order to compensate for this, Cohen *et al.* [31] incorporate an *ambient correction* term after every iteration. However, this term does not play any part in the solution process. It is merely added for display purposes *only*. The ambient term itself diminishes as the solution progresses, therefore useful images will be available during the initial stages of progressive refinement.

Based upon the current iteration's estimate of the radiosity of all patches, the ambient term is essentially an average radiosity term, or in this case a radiant exitance term, that is computed from the *average reflectivity* of the scene. When added to all patches within a scene, this ambient term essentially simulates the effect of adding a completely diffuse light source that evenly illuminates every patch within the scene [52].

The radiant exitance of the ambient term is derived from the average reflectivity, ρ_{avg}, and the interreflectance factor, R, of a scene, as shown in Equation 2.42:

$$B_{ambient} = R \sum_{i=1}^{n} \Delta B_i A_i \Big/ \sum_{j=1}^{n} A_j \qquad 2.42$$

where the interreflectance factor R is:

$$R = \frac{1}{1 - \rho_{avg}} \qquad 2.43$$

and the average reflectivity ρ_{ave} of a scene is:

$$\rho_{avg} = \frac{\sum_{i=1}^{n} \rho_i A_i}{\sum_{i=1}^{n} A_i} \qquad 2.44$$

Detailed explanations and derivations of ρ_{ave}, R and the ambient term have been documented by Cohen *et al.* [34] and Ashdown [52].

2.8 Sub-structuring

When solving the radiosity matrix using popular methods such as the Jacobi or Gauss-Seidel iterative methods as described above (Section 2.6), the arrangement of the surfaces in the scene is not considered. Hence every element contributes a coefficient into the radiosity matrix regardless of what its impact is to other elements. What is needed is a more intelligent approach to solving the radiosity equation.

Cohen *et al.* [22] introduced a two level sub-structuring method, which creates one level of finely meshed *elements* and one level of coarsely meshed *patches.* The concept of this method is that an accurate solution for the radiosity at a point on a surface, can be obtained with the global radiosities generated by the coarse patch-to-patch interactions. Since the radiosity across a surface may vary significantly, for example at shadow boundaries, the coarse representation (i.e. the patch representation) may be insufficient to capture this detail. Hence each surface also has a fine mesh of elements.

Using this method [22], the time complexity for solving the radiosity equation can be reduced to O(nm), where n is the number of finely meshed elements and m is the number of coarse patches. The benefit of this method of sub-structuring becomes apparent when the number of patches is much less than the number of elements, that is, $m<<n$. In general, it has been proven [22] that $m<<n$ which implies that overall time complexity could theoretically reach O(n). Combining this sub-structuring method with their progressive refinement technique [31], this combined technique allows rendered images to be displayed after each iteration during the radiosity solving phase. This is very useful, as the progressive refinement technique tends to converge quite quickly, therefore enabling a reasonably good (but approximate) rendered image to be

rapidly available to the user. This method, however, still requires $O(n^2)$ time to completely solve the radiosity equation.

The pseudo code for the progressive refinement radiosity algorithm with sub-structuring is as follows:

```
// Initialise the radiosities of all patches:
FOR every patch i DO
   ΔB_i = initial radiosity of patch i
ENDFOR

WHILE Σ ΔΦ_k for all patches k > ε DO
   Select patch i with the greatest unshot flux ΔΦ_i
   Compute patch-element form factors F_ie
   FOR each patch j DO
      FOR each element k of patch j DO
         Δrad = ρ_e ΔB_i F_ei
         // update area weighted unshot radiosity of
         // element e belonging to patch j
         ΔB_e += Δrad A_e/A_j
         // update total radiosity of element e
         B_e += Δrad
      ENDFOR
   ENDFOR
   ΔB_i = 0  // reset unshot radiosity for patch i
ENDWHILE
```

where:

ΔB is the unshot radiosity of a patch;

$\Delta\Phi$ is the unshot flux of a patch;

Δrad is the amount of flux contained by a shooting patch;

ρ is the reflectance of a patch.

2.9 Summary

In this chapter, we have shown that the Gaussian elimination method for solving the radiosity matrix, which was originally used by Goral *et al.* [15], is too expensive for scenes of any practical size. As a result, Cohen *et al.* [19] and Cohen *et al.* [22] utilised iterative techniques such as the Jocobi and Gauss-Seidel methods for solving the radiosity equation.

These iterative methods improve the upon the $O(n^3)$ running time complexity of Gaussian elimination, to $O(n^2)$. However, the quadratic time and space complexity means that scenes are still limited to scenes of small size.

The progressive refinement radiosity algorithm introduced by Cohen *et al.* [31] is a viable, alternative method for solving the radiosity equation. The progressive refinement radiosity algorithm combines the hemi-cube technique from Cohen *et al.* [19] and the two-level sub-structuring and adaptive refinement techniques from Cohen *et al.* [22], with a *shooting* method to distribute energy throughout a scene. This energy shooting method enables quick convergence to a solution, however this algorithm still takes $O(n^2)$ to arrive at *complete* radiosity solution.

The main advantage of the progressive refinement radiosity is that it only requires $O(n)$ memory requirements per iteration and a partial image is available in $O(n)$ time (i.e. after every iteration). Thus progressive refinement radiosity is a very attractive algorithm for generating rendered images at interactive rates.

The main disadvantage of progressive refinement radiosity is that it is still bounded by the fundamental $O(n^2)$ complexity for completely solving the radiosity matrix [83].

In a scene, there will generally be surfaces that transport negligible quantities of energy to other surfaces and hence can be either ignored, or represented coarsely. This can be further extended to include groups of surfaces that weakly interact with the surrounding geometry.

Therefore in the following chapter, we will discuss algorithms that explore the effects of light transport within a scene. These algorithms will implement various techniques, that will surpass the $O(n^2)$ time complexity bound of current radiosity algorithms.

Chapter 3 Hierarchical Radiosity

3.1 Introduction

To date the most advanced radiosity engines incorporate a whole host of complex techniques that make computing the radiosity equation more efficient. The most important breakthrough since Cohen *et al's* methods [22][31] is the *hierarchical radiosity* technique [39].

Hanrahan *et al.* [39] introduced a hierarchical approach to 'patch to patch' interactions that was inspired by the technique for solving the n-body problem [17]. This technique enables the computation of all forces on a particle in less than quadratic time. When radiositising a scene, a very similar calculation is performed between the surfaces in a scene. Therefore, incorporating this technique can make a significant improvement to the radiosity algorithm

The hierarchical radiosity algorithm is a method that explicitly represents the transport of light throughout the environment of a scene, by representing each light interaction between each and every adjacent patch by a *link.* Thus each patch will have a collection of links that represents the light transported by every adjacent patch that interacted with it (see Figure 3.2(c)). To ensure that the accuracy of the radiosity across a patch is maintained, the hierarchical radiosity algorithm keeps a multi-resolution patch hierarchy for every patch in the scene. This extends Cohen *et al's* two level patch-element hierarchy, by adaptively storing representations of the original patch at various levels of detail. Since the hierarchical patch refinement process is integral to the hierarchical radiosity algorithm, this method has the advantage over previous radiosity algorithms of not requiring any initial meshing.

Hanrahan *et al's* hierarchical radiosity approach [39] improves upon Cohen *et al's* $O(nm)$ time complexity to $O(k^2+n)$, where k is the number of initial surfaces and thus corresponds to the maximum number of link interactions that have to be considered, and n is the total number of patches that are created after patch refinement. Hence

theoretically, using the hierarchical radiosity algorithm, a scene can be computed in linear time. Unfortunately we must also take into account the $O(k^2)$ linking time when considering the overall performance of hierarchical radiosity.

3.2 A Basic Hierarchical Radiosity Algorithm

The basic components of hierarchical radiosity can be summarised in Figure 3.1.

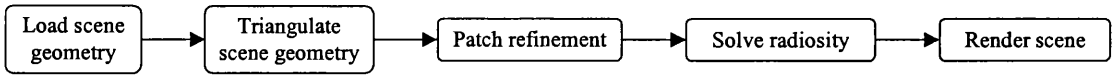


Figure 3.1 The basic components of hierarchical radiosity.

Figure 3.2 illustrates the main steps of the hierarchical radiosity method for a simple scene, consisting of a small rectangular light source and a larger rectangular floor surface (Figure 3.2(a)). The triangulation of the input geometry has been omitted from this illustration.

Once all the surfaces of the input scene have been loaded, they must be decomposed into simple polygons such as quadrilaterals and/or triangles. There are two main reasons for restricting all scene polygons into quadrilaterals or triangles.

Firstly, these two types of polygon will enable the reliable generation of convex mesh elements. This is important since most of our computations will only produce robust solutions with a mesh constructed from convex polygons. This can always be achieved if all polygons are reduced to triangles.

Secondly, the patch refinement stage oversees the construction of the hierarchical form factor matrix approximation. During this stage, it is often very convenient to utilise Siegel and Howell's analytical form factor estimate for a differential area and a finite circular disk [10]. However this form factor estimate only holds true for square quadrilaterals or equilateral triangles, as these polygons can be approximated to a circular disk. In other words, care must be taken to ensure that the quadrilateral or triangular polygons are not long and thin.

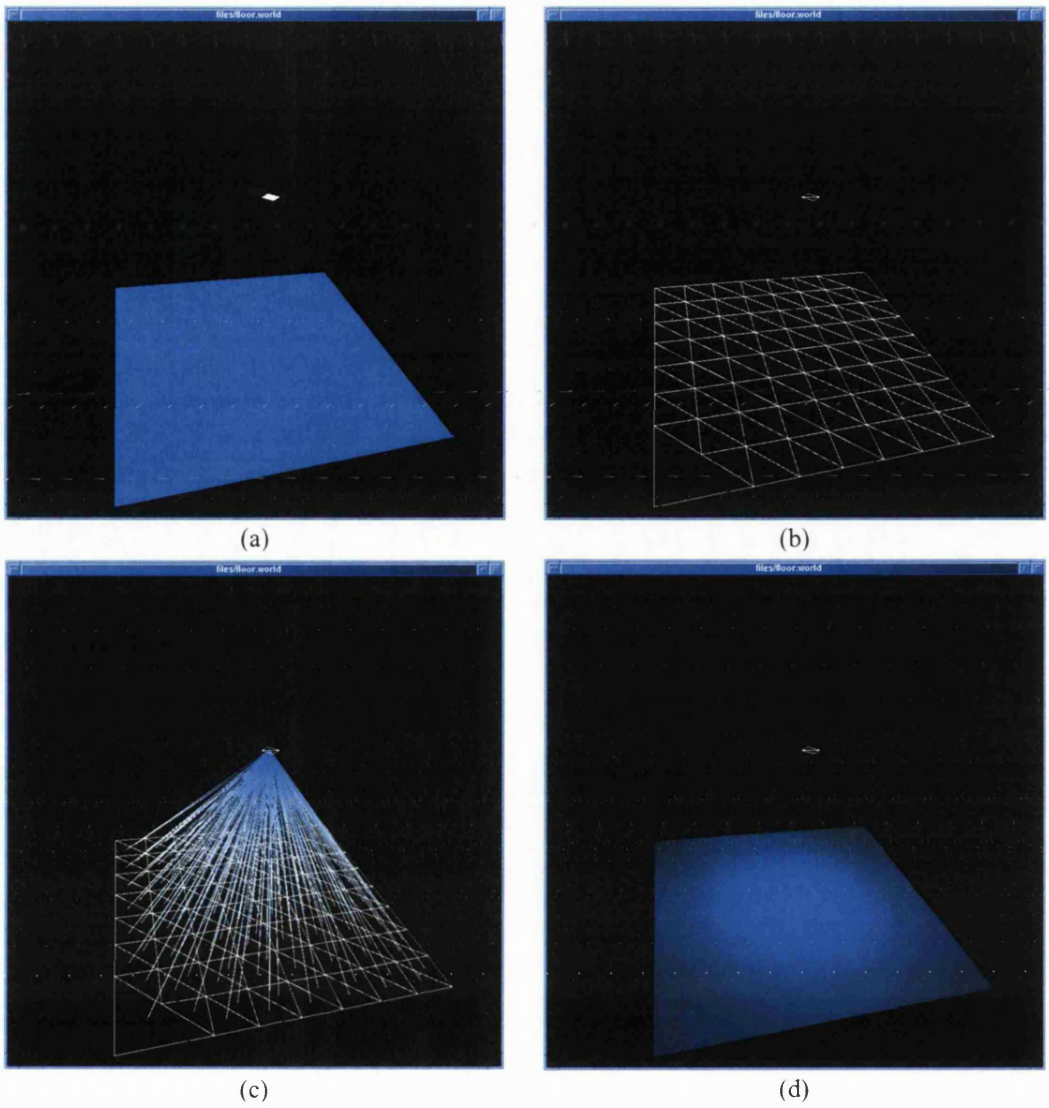


Figure 3.2 The hierarchical patch refinement and linking process of a simple scene. (a) shows initial geometry and (b) after patch refinement. (c) shows the link interactions between patches and (d) shows the final radiositised solution.

Generally, a scene can be efficiently decomposed into a mesh of roughly equilateral triangles by using a high quality *Constrained Delaunay Triangulation* (CDT) algorithm. Unfortunately CDT algorithms can be very complex, especially those that generate very high quality meshes.

There are a number of CDT implementations available, of varying sophistication, but there are robust CDT implementations by Shewchuk [70], Stuerzlinger [46] and

Lischinski [60] that are freely available. Both Worrall [82] and Hedley [83] used Shewchuk's CDT implementation with great success.

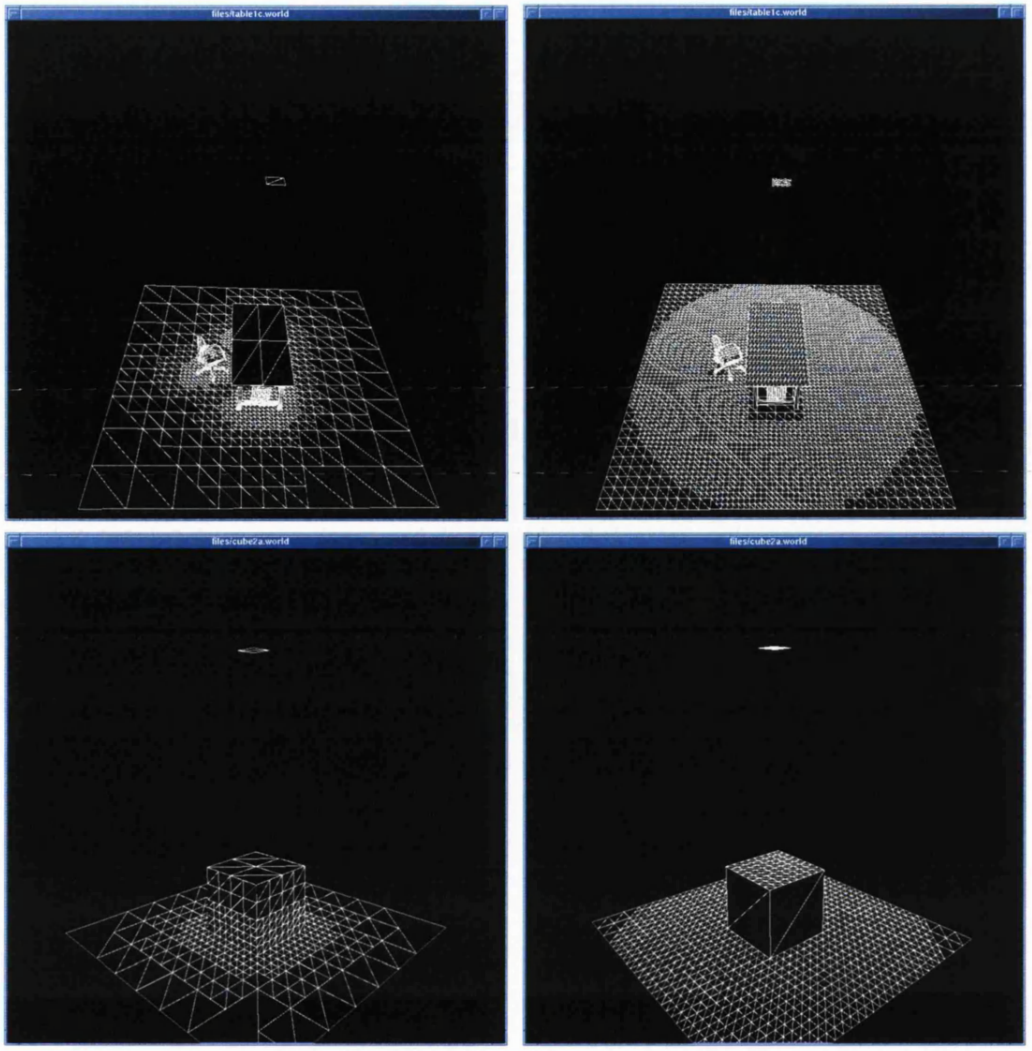
From Figure 3.2(b) we can see the results of hierarchical refinement. It is quite logical that the floor surface has been subdivided into a number of smaller patches, since it is usually not possible to capture the radiosity across a surface with only a single patch. Thus each time a patch is subdivided, the new sub-patches are entered into a new level in the patch hierarchy. Figure 3.5 illustrates patch sub-division and the creation of the patch hierarchy for the floor surface in Figure 3.2.

Figure 3.2(c) shows the interactions of the links between all patches in the patch hierarchy. The light transport is depicted by the colour shading at the ends of the link. Finally, Figure 3.2(d) shows the final radiositised and rendered scene.

3.3 Patch Refinement

One of the integral components of the hierarchical radiosity algorithm is patch refinement. The patch refinement process between two patches is guided by an *oracle* function that determines whether the accuracy of the refinement criterion between the two patches has been met. Depending upon the result returned by the refinement oracle, two patches may be considered accurate enough that refinement is not necessary, or, one or both patches may require further refinement by sub-division.

The two most typical refinement oracles are the F-only and BF oracles that were used by Hanrahan *et al.* [39]. The F-only refinement oracle is based entirely on the form factor between two patches. Thus sub-division is based entirely upon the geometry between two patches. The BF refinement oracle is a *brightness weighted* estimate for patch sub-division that uses the product between the form factor and the radiosity shot between two patches. Both types of refinement produce different final meshes as shown in Figure 3.3.



(a) (b)

Figure 3.3 Patch subdivision using (a) F only refinement and (b) BF refinement.

The refinement oracles that were originally used by Hanrahan *et al.* [39], represent only one type of refinement method. Stamminger *et al.* [80] published a comparison of three different types of hierarchical radiosity 'refiners':

- form factor based;
- bounding;
- sampling;

Form factor based refinement uses the form factor calculations themselves (as used by Hanrahan *et al.* [39]) as an approximation of the overall error, to drive the refinement process. Stamminger *et al.* [80] reported that form factor based refinement methods

are by far the easiest of the three methods to implement. However, the refinement process is only as accurate as the form factor approximations used. Thus artefacts will manifest themselves where the form factor approximations lead to inaccurate values, as will be shown later in this section.

Stamminger *et al.* [79] presented the bounding based refinement method, which computed the bounds on the light transport between two arbitrary objects. However, Stamminger *et al.* [80] state that the bounds from this method are conservative and tend to be too pessimistic, but the distance between the bounds can be used to guide the refinement process.

The main advantages of this refinement method are:

- It can be applied to any arbitrary type of object.
- It does not suffer from sampling artefacts.

The disadvantages are:

- It is quite expensive to compute.
- Visibility calculations are more difficult to compute and are less accurate.

The final refinement method is the sampling refinement method. This method was used by Gibson *et al.* [73] to compute the form factor between a point and a finite area. This method for calculating form factors is similar to the raycasting method by Wallace *et al.* [33]. It was found that sampling based refinement methods are quite versatile and can inherently handle partial occlusion between polygons.

The conclusion by Stamminger *et al.* [80] for the outcomes of the comparisons between the three types of hierarchical radiosity refinement methods is quite surprising. The results of their experimental error measurements [80] showed that there was no great difference in error and thus accuracy, between the three methods. Willmott [90] notes that most of the difference in accuracy between the methods comes from the accuracy of visibility calculation, and the robustness and simplicity of the implementation methods themselves.

Impressive results can be obtained from hierarchical radiosity, using form factor based refinement. However, due to the approximations used for form factor calculation, it is possible for the refinement process to produce erroneous results. By using sampling based refinement techniques, a viable hybrid 'form factor-sampling' solution can be created that maintains the simplicity and robustness of the form factor based refinement method.

This section proceeds with the form factor based method of refinement as first introduced by Hanrahan *et al.* [39], followed by a simple solution for overcoming the accuracy problems produced by approximate form factor calculations.

At the heart of the hierarchical radiosity algorithm is the construction of the hierarchical form factor matrix and the multi-resolution patch-element hierarchy. Hanrahan *et al.* [39] developed a recursive refinement procedure that decomposed two input polygons into a hierarchy of patches and elements, and built simultaneously a hierarchical representation of the form factor matrix.

During this process, the interactions between two adjacent patches are considered. Using an *oracle* function, if the interaction between the two patches is considered to be accurate enough, then a link is created between the two patches. If, on the other hand, the error estimate from the oracle function is too large, then the larger patch is subdivided.

The most common strategy for sub-dividing polygons is to use *quad-tree* subdivision. Quad-tree subdivision subdivides a polygon by bisecting each of the original polygon sides at the midpoint, to produce four smaller polygons of equal area. This is shown in Figure 3.4.

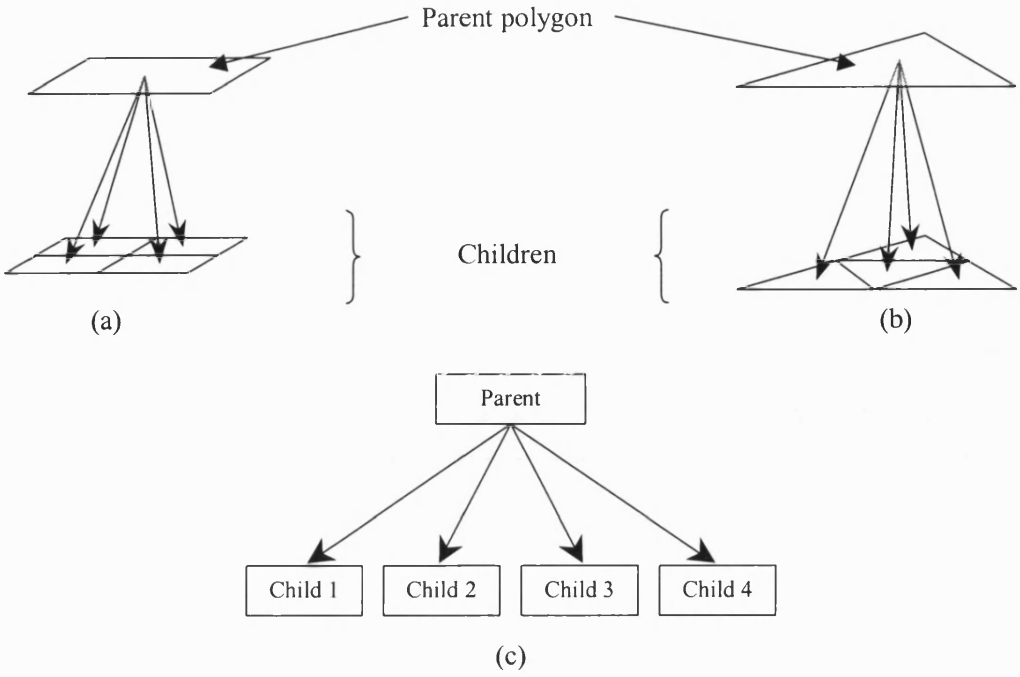


Figure 3.4 Quad-tree subdivision for (a) a quadrilateral and (b) a triangle. (c) Tree structure representation.

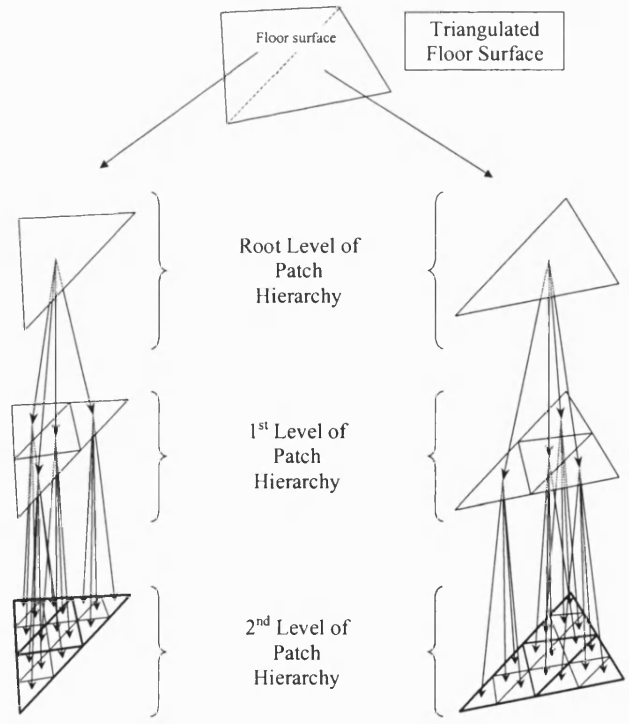


Figure 3.5 An illustration showing the construction of patch hierarchy for the floor surface in Figure 3.2.

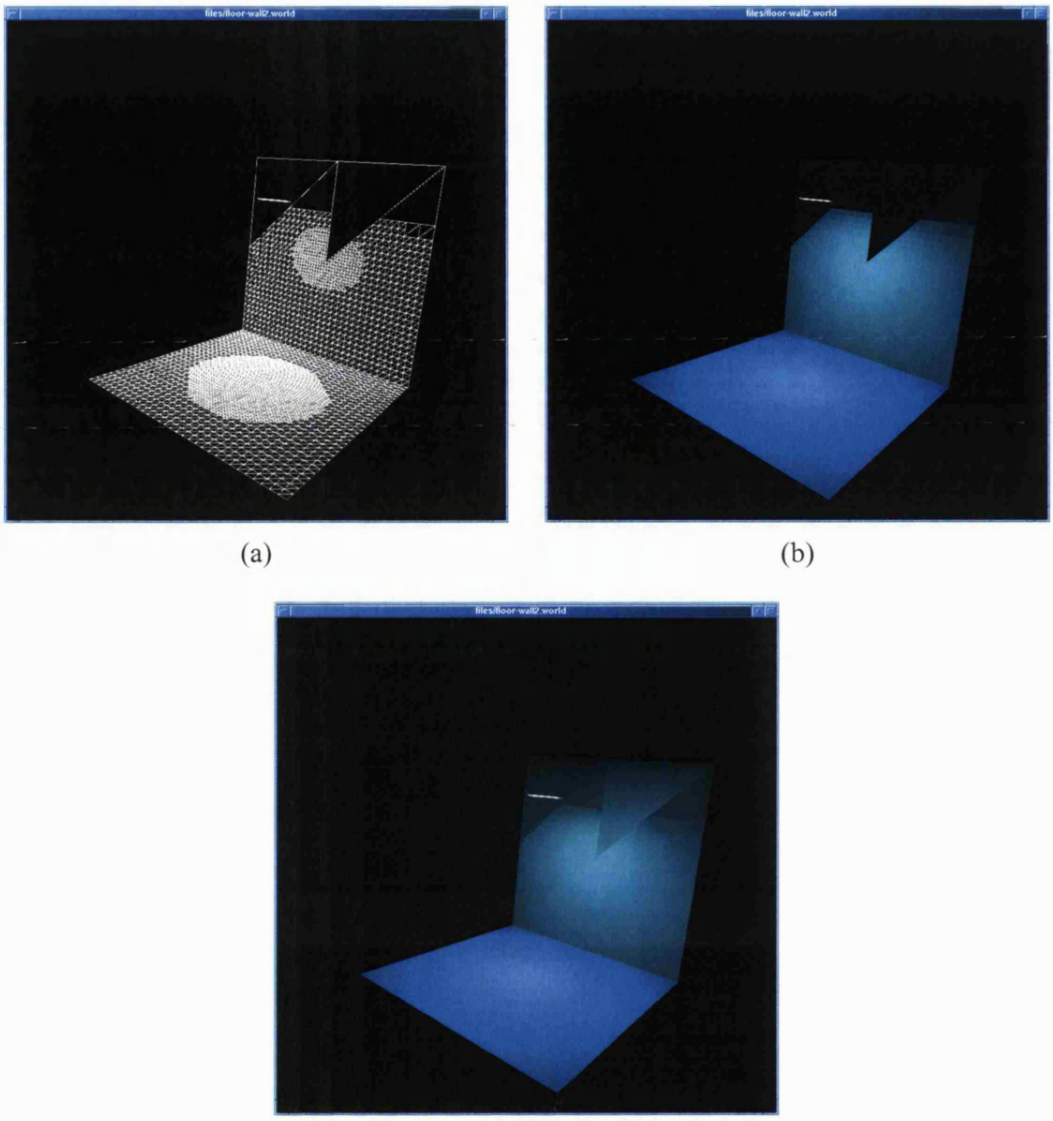
Each of the four subdivided polygons becomes a child polygon of the original parent polygon. In turn, each child polygon can become a parent. As both parent and children polygons are refined, a hierarchical quad-tree structure of polygons is built. An illustration of this process for the scene in Figure 3.2 is shown in Figure 3.5. Thus to build the entire hierarchical structure of the whole scene, each polygon must be refined against every other polygon.

The following pseudo-code shows Hanrahan *et al's* [39] original patch refinement procedure:

```
PROCEDURE Refine(patch p, patch q)
   Compute the form factors F_pq and F_qp
   IF (F_pq < F_ε) AND (F_qp < F_ε) THEN
      Link(p, q)
   ELSE
      IF F_pq > F_qp THEN
         IF A_q > A_ε THEN
            subdivide patch q
            refine all child patches of q
         ELSE
            Link(p, q)
         ENDIF
      ELSE
         IF A_p > A_ε THEN
            subdivide patch p
            refine all child patches of p
         ELSE
            Link(p, q)
         ENDIF
      ENDIF
   ENDIF
ENDPROCEDURE
```

F_ε is a predetermined threshold value that determines when a form factor estimate is accurate enough. A_ε is a predetermined threshold value that determines the minimum area a patch can have. This prevents infinite, recursive patch subdivision of the above refinement procedure.

During patch refinement, care must be taken when computing form factors. Unfortunately errors in form factor calculation do not always manifest themselves directly in the radiosity solution. There is a particular situation where polygon geometry can lead to erroneous form factor calculation and hence incorrect patch refinement. Figure 3.6 shows how the placement of the light source within a scene, can cause incorrect patch sub-division.



(a) (b)

(c)

Figure 3.6 A simple scene showing how the input scene geometry can lead to erroneous form factor calculation and hence incorrect patch sub-division. (a) wireframe view (b) flat shaded view (c) smooth shaded view.

The wireframe view in Figure 3.6(a) shows that there are a number of polygons on the top half of the back wall that appear not to have been subdivided correctly. Of course, it may be that this is a correct solution, however, the rendered image in Figure 3.6(b) shows that this is not the case. Occasionally, applying linear interpolation may solve the problem, but it can be seen in Figure 3.6(c) that there is clearly an error during form factor calculation (including visibility computation) and/or patch refinement.

The actual cause of the problem in Figure 3.6 is due to the way the form factors are calculated between a differential area polygon and a finite area polygon. The two most common ways for solving form factors are to use the contour integration approach that was used by Nishita *et al.* [18] (as expressed in Equation 2.24), or to use the form factor approximation between a differential area and a finite area disk [10][12]. Figure 3.7 shows what has happened.

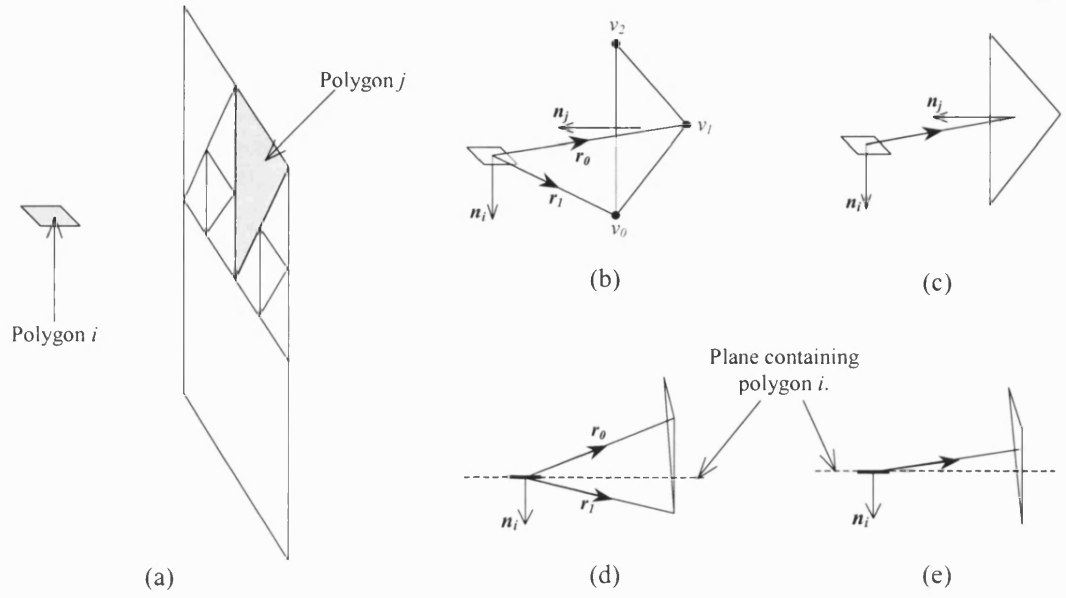


Figure 3.7 (a) A selected pair of erroneous patches from Figure 3.6. (b) Form factor calculation using the contour integration approach. (c) Form factor calculation between a differential area and a finite disk. (d) A view of (b) perpendicular to polygon *i*. (e) A view of (c) perpendicular to polygon *i*.

Considering the two shaded polygons *i* and *j* in Figure 3.7(a), Figure 3.7(b) shows diagrammatically, the process of computing the form factor shown in Equation 2.24, and Figure 3.7(c) the computation of the form factor shown in Equation 2.34.

From Figure 3.7(b), it can be seen that the vector r_0 lies *behind* the plane containing polygon *i* and the vector r_1 lies in *front* of this plane. For clarity, Figure 3.7(d) shows Figure 3.7(b) from a view point that is perpendicular to plane containing polygon *i*.

Form factor calculations between two polygons are only valid when they face each other, thus any calculations involving *backfaces*, have the undesirable effect of changing the sign of the calculation. Since form factors always obey $0 \leq FormFactor \leq 1$, a negative form factor is invalid and is usually set to zero, if detected.

In the case of Figure 3.6 and hence Figure 3.7(b) (i.e. using Equation 2.24 for form factor calculation), two out of three vertices that belong to polygon j, lie completely *behind* the plane of polygon i. In this particular case, it is also very unfortunate that the plane constructed by the origin of polygon i and the vertices v_1 and v_2 on polygon j, contributes a significantly large delta form factor such that the total form factor (i.e. after the summation) is negative. Hence if the refinement oracle function uses the form factor to determine patch sub-division, as is the case in Figure 3.6, even though it can be seen that the form factor is non-zero, due to the arrangement of the two polygons the form factor will be zero. Therefore the refinement oracle would indicate that no refinement should take place and hence the scene is left with an incorrect patch.

The same problem can occur if the differential area to circular disk (see Howell [10][12]) approach to form factor calculation is used. This can be seen in Figure 3.7(c).

The cause of the problem here is that only the centres of the two polygons i and j are considered when calculating form factors. Thus in the scenario shown in Figure 3.6, the vector between the centres of the two polygons i and j, lies behind the plane of polygon i. Again for clarity, Figure 3.7(e) shows Figure 3.7(c) from a view point that is perpendicular to the plane containing polygon i.

In the same way as the contour integration approach, the form factor value that is computed will also be negative and so invalid. As a result, this form factor value will cause the polygon refinement procedure to indicate that no refinement should take place and so produce the incorrectly shaded polygons shown in Figure 3.6.

To solve this problem, it is necessary to overcome the limitations imposed by the form factor approximations of Equation 2.24 and Equation 2.34. These approximations only

work reliably when the differential area polygons and the finite area polygons either directly face, or face away from each other. Polygons that 'partially face' each other (i.e. either polygon straddles the plane of the other) as shown in Figure 3.7, will cause unreliable results depending upon the geometry of the finite area polygon.

Since the key cause of these approximations is effectively the 'undersampling' of the finite area polygon, the solution would be to ensure that the finite area polygon is adequately sampled. In particular the sampling should increase, the larger the finite area polygon is, with respect to the differential area.

To maintain the benefits gained by using the form factor approximation methods, increased sampling should only be applied when the differential area polygons and the finite area polygons 'partially face' each other. This situation can be detected if the form factor result is negative or in the case of the differential area to finite disk approach, if $\cos\theta_i$ or $\cos\theta_j$ is less than zero.

There are a variety of sampling methods that can be applied. A simple uniform sampling method would be to subdivide the finite area polygon into a regular grid of delta areas. The total form factor would be the sum of all delta form factors that were positive. However, if the finite area polygon is large, it maybe necessary to finely subdivide this polygon and thus generate many unnecessary, small sub-polygons, that actually are positioned behind the plane of the differential area polygon.

A proposal for a more efficient method would be to use a stochastic variation on the uniform sampling method, such as the raycasting method used by Wallace *et al.* [33] (see Section 2.5.5). A fixed number of rays could be cast from the centre of the differential area to the finite area, or a more dynamic approach could be taken by using a heuristic based upon the size of the finite area polygon and possibly the distance between the differential area polygon and finite area polygon, to scale the number of rays shot. However, care must be taken to ensure that there is an even distribution of sample points across the surface of the finite area polygon.

The following C++ code shows the algorithm for incorporating the raycasting method to the form factor calculation between a differential area and a finite area disk.

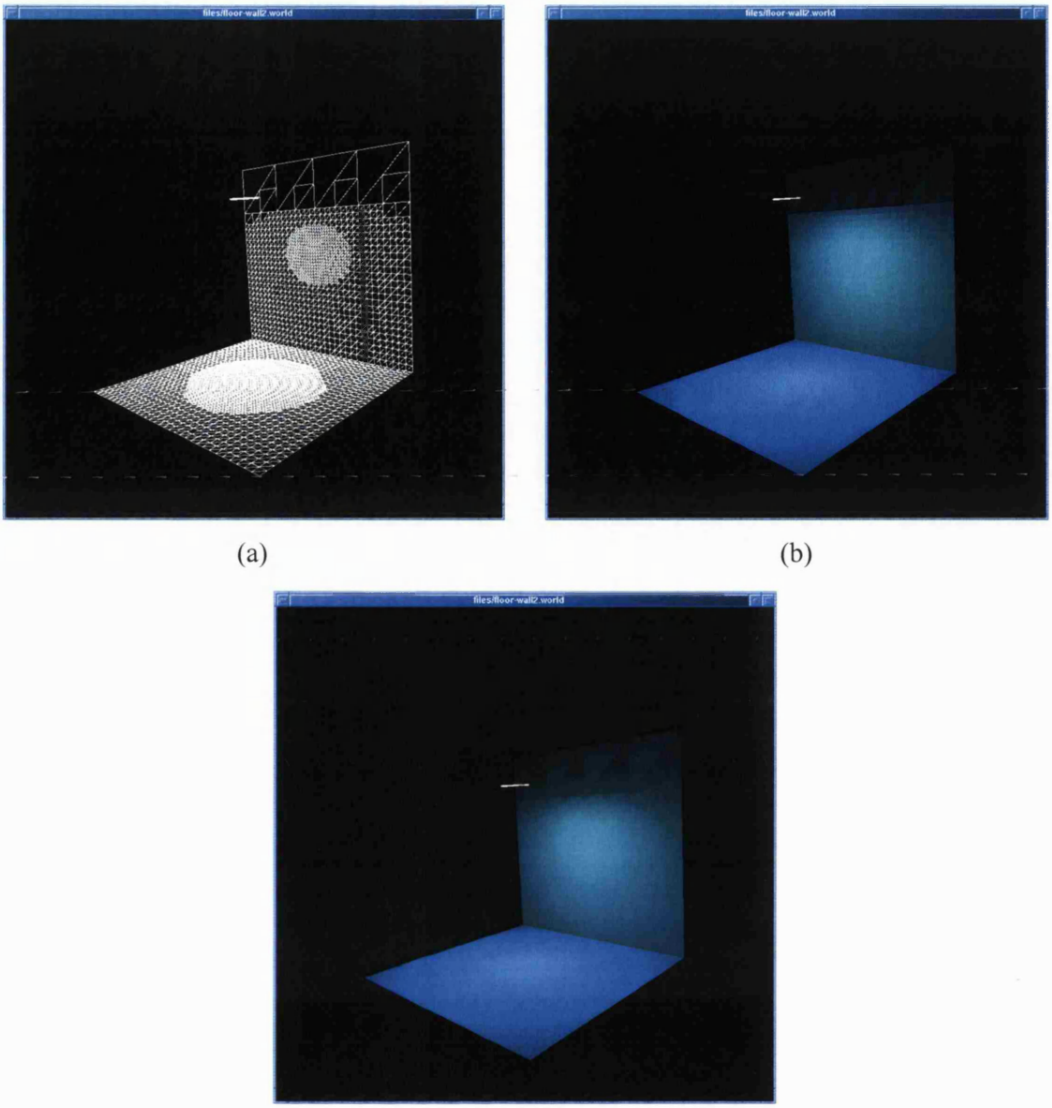
```
double FormFactor(Patch *diff, Patch *fin) {
  Vector R  = Subtract(&fin->_centre, &diff->_centre);
  double RR = R.x*R.x + R.y*R.y + R.z*R.z;
  R.normalise();
  double cos1 =  Dot(&R, &diff->_normal);
  double cos2 = -Dot(&R, &fin->_normal);
  if (cos1 >= 0.0 && cos2 >= 0.0)
    return (fin->_area * cos1 * cos2) / (M_PI * RR + fin->_area);
  return RaycastFF(diff, fin);
}

double RaycastFF(Patch* diff, Patch* fin) {
  const int numRays = 128;
  double ff = 0.0;
  double area = fin->_area / (double)numRays;

  for (int i=0; i<numRays; i++) {
    Point A = diff->select(); // select a random point on diff
    Point B = fin->select();  // select a random point on fin
    Vector dir = Subtract(&B, &A);
    double RR = Dot(&dir, &dir);
    dir.normalise();
    double cos1 =  Dot(&dir, &diff->_normal);
    double cos2 = -Dot(&dir, &fin->_normal);
    if (cos1 > 0.0 && cos2 > 0.0)
      ff += cos1*cos2/(M_PI*RR + area);
  }
  ff *= area;
  return ff;
}
```

The `RaycastFF` function can be used independently to calculate the form factor between a differential area and a finite area disk.

Figure 3.8 shows the results of the scene in Figure 3.6, using the above, more reliable method for form factor calculation. Figure 3.8(a) shows the wireframe mesh generated by hierarchical patch refinement. It can be seen that only patches below the light source were significantly refined, leaving the patches on the wall above the light source at a relatively coarse level of refinement. Figure 3.8(b) shows the scene rendered with flat shading and Figure 3.8(c) shows the scene rendered with linear (smooth) shading.



(a) (b)

(c)

Figure 3.8 The simple scene from Figure 3.6 with correct patch subdivision. (a) wireframe view, (b) flat shaded view and (c) smooth shaded view.

3.4 Improving the Linking Time for Hierarchical Radiosity

It is the linking stage of hierarchical radiosity that hinders the scalability of this algorithm. There have been attempts to minimise the time complexity of the linking stage. Holzschuch *et al.* [56] presented a *lazy linking* method whereby the whole initial linking procedure is avoided by deferring the creation of a link until the interactions between patches are sufficiently important. Lischinski [62] used an

ad hoc method of linking. In the initial linking stage, this method only allows non-primary light source surfaces whose form factor were below a preset threshold, to be linked.

The lazy linking method by Holzschuch *et al.* [56] has been used extensively by a number of authors (e.g. Sillion [56], Worrall [82], Hedley [83], Gibson [67][84], Willmott [90]) to reduce the lengthy initial linking stage of hierarchical radiosity.

The algorithm for Holzschuch *et al's* [56] lazy linking method (adapted from Gibson [67]) is as follows:

```
PROCEDURE LazyLink
   FOR each pair of unclassified polygons p, q DO
      IF p and q face each other THEN
         IF (Bp > εlink) OR (Bq > εlink) THEN
            Compute the form factor Fpq and Fqp
            IF (BpFqp > εlink) OR (BqFpq > εlink) THEN
               Link p and q
               Mark p, q as classified
               Compute the visibility factor between p, q
               IF p, q is not totally occluded THEN
                  Link p, q
               ENDIF
            ENDIF
         ENDIF
      ELSE
         Mark p, q as classified
      ENDIF
   ENDFOR
ENDPROCEDURE
```

B_p is the product between the reflectance and emittance of patch *p* and B_q is the product between the reflectance and emittance of patch *q*.

ε_{link} is the threshold used to establish top-level patch interactions. Only patches that are 'bright enough' i.e. greater than ε_{link}, will be considered for linking. To prevent energy being lost due to a BF product less than ε_{link}, the value for ε_{link} should be less than F_ε (or ε_{refine} as it is labelled in Holzschuch *et al's* algorithm [56]). A suitable value for ε_{link} is: $\varepsilon_{link} = F_\varepsilon / 5$ [56].

The main problem with hierarchical radiosity algorithms that use the 'gather' technique for solving the radiosity equation, is that all links created in the linking stage

must be kept. Fortunately Stamminger *et al.* [85] describe a method for getting rid of links in hierarchical radiosity.

Stamminger *et al.* [85] realised that if the 'shooting' method as used in progressive refinement radiosity, is used to distribute energy to all other patches in the environment, the amount of unshot energy decreases exponentially. Thus, instead of propagating the full amount of radiosity after every iteration, only the exponentially decreasing amount of unshot power is transported. As a result, there is a decreased probability is reusing an existing link, which enables links that will not be used again to be deleted.

All links that are reused are held in a cache. This is because the cost of recomputing these links will result in a very noticeable performance hit. A suitable caching strategy is given by Stamminger [85].

3.5 Solving the Radiosity Equation

Solving the radiosity matrix for hierarchical radiosity is performed for every surface in the scene, by repeatedly gathering the radiosity across all links connected to each sub-patch in the surface's multi-resolution patch hierarchy, until some convergence criterion is met.

For every patch in the patch hierarchy, this is accomplished by summing the product between the form factor, the diffuse colour and the average radiosity of every linked patch.

The following C++ code is an algorithm that shows the gather procedure for Hanrahan *et al's* [39] original hierarchical radiosity method.

```
void Gather(Patch* p) {
  p->_Bg.setRGB(0.0, 0.0, 0.0);
  for (Link* l=p->_links; l!=NULL; l=l->next)
    p->_Bg += l->q->_Bs * l->FF;

  p->_Bg *= p->_surface->refl();
  p->_Bg.clamp();

  if (p->_child[0] != NULL) {
    Gather(p->_child[0]);
    Gather(p->_child[1]);
    Gather(p->_child[2]);
    Gather(p->_child[3]);
  }
}
```

Once all energy has been gathered onto a surface, it must be then distributed evenly throughout the multi-resolution patch hierarchy. This must be done because the linking between patches can be at any level within the patch hierarchy. Hence this will cause the gathering of energy to be scattered throughout the patch hierarchy [67]. By performing a bi-directional *push-pull* sweep of the patch hierarchy after every gather step, the energy distributed within the patch hierarchy will be kept in a consistent state.

During the *push* phase of the sweep, all the gathered radiosity is pushed down the patch hierarchy to the leaf nodes. The total amount of energy received by a patch is the sum of the energy it received directly, plus the sum of all the energy received by its parent patches.

In the *pull* phase of the sweep, the energy at the leaf nodes is pulled back up the patch hierarchy and each of the patches at the current level of the patch hierarchy is set to the area weighted average of its children's radiosities [39].

The *push-pull* procedure is achieved by simply traversing recursively, the quadtree hierarchies of every patch. The *push* phase is achieved by recursively calling the children nodes until the leaf nodes are reached. Once at the leaf nodes, the *pull* phase will automatically execute, as the recursion process unwinds itself.

The algorithm for this procedure is best illustrated by the following code:

```
Spectra PushPull(Patch* p, Spectra Bdown, double* error) {
  Spectra Bup;
  if (p->_child[0] == NULL)
    Bup = p->_surface->emit() + p->_Bg + Bdown;
  else {
    double iArea = 1.0/p->_area;
    Bdown += p->_Bg;
    Bup    = PushPull(p->_child[0], Bdown, error) * p->_child[0]->_area;
    Bup   += PushPull(p->_child[1], Bdown, error) * p->_child[1]->_area;
    Bup   += PushPull(p->_child[2], Bdown, error) * p->_child[2]->_area;
    Bup   += PushPull(p->_child[3], Bdown, error) * p->_child[3]->_area;
    Bup   *= iArea;
  }
  Spectra e = p->_Bs - Bup;
  *error += e.red*e.red + e.green*e.green + e.blue*e.blue;

  p->_Bs = Bup;
  return Bup;
}
```

Thus to compute one iteration step, the following code would be:

```
double iterate(void) {
  double error = 0.0;
  for (long i=0; i<_numPatches; i++) {
    Gather(_patchList[i]);
    PushPull(_patchList[i], Spectra(0.0, 0.0, 0.0), &error);
  }
  return sqrt(error);
}
```

It can be seen that the error estimate can be calculated after each push-pull step. Thus a total error estimate can be obtained by summing over all patches. Hence when the total error estimate falls within a specified tolerance, the radiosity solution has converged.

The benefits from performing the initial linking and patch refinement become apparent when solving the radiosity matrix (Equation 2.39). As with progressive refinement radiosity (see Section 2.7), the radiosity system can be solved by using either the gathering or shooting methods.

The gathering method requires Equation 2.18 to be solved for each patch in the environment. However during the refinement and patch linking process, every patch in the scene accumulates a set of links to all patches within the environment that interacted with it. This, in fact, represents the summation of Equation 2.18 except that it has been optimised to only include patches of significance. Also, as patches can

interact between various levels in the multi-resolution patch hierarchy, this means many coefficients in the radiosity matrix will be represented by a single calculation. Hence the gathering procedure for hierarchical radiosity will be on average, far more efficient than for the progressive refinement radiosity method.

The shooting method can also be applied to hierarchical radiosity. As with Cohen *et al's* [31] progressive refinement algorithm, all patches are sorted into a priority queue based upon the brightness of every patch [39]. Each patch is then taken off the queue and its radiosity is shot to all patches that were linked to it. Unlike traditional progressive refinement radiosity, the shooting procedure is far more efficient when applied to hierarchical radiosity, as energy is only shot to linked patches, rather than to every patch within the entire scene.

3.6 Multigridding

An improvement to the hierarchical radiosity refinement process is to incorporate a method that refines the patch hierarchy as the iteration proceeds. This is an idea similar to a numerical technique known as multigridding.

To begin with, the multigridding technique solves a finite difference equation at a coarse resolution, then at successively finer resolutions. The advantage of this is that the lower resolution solutions are relatively cheap to compute and in turn, these solutions provide a better starting point for the more expensive solutions at finer resolutions.

A similar technique can be applied to the hierarchical radiosity refinement procedure. Hanrahan *et al.* [39] modify the *gather* procedure to allow patches to re-refine themselves against all patches that were linked to them. Consequently, the *refine* procedure has to be extended to delete existing links before re-refinement, to prevent existing energy from contributing to the solution again.

The following code shows how the *gather* procedure would be modified to incorporate multigridding.

```
void multigridGather(Patch* p, double eps) {
  Link* l = p->links, *L;
  while (l != NULL) {
    Patch* q = l->q;
    L = l;
    l = l->next;
    refinePatch(p, q, L, 0.0, eps);
  }

  p->Bg.setRGB(0.0, 0.0, 0.0);
  for (Link* l=p->links; l!=NULL; l=l->next)
    p->Bg += l->q->Bs * l->FF * l->vis;

  p->Bg *= p->surface->refl();
  p->Bg.clamp();

  if (p->child[0] != NULL) {
    multigridGather(p->child[0], eps);
    multigridGather(p->child[1], eps);
    multigridGather(p->child[2], eps);
    multigridGather(p->child[3], eps);
  }
}
```

The multigridding procedure is utilised by successively re-running the refinement process with smaller F_ε values. Effectively the radiosity solution is recomputed until the current solution is deemed to be accurate enough.

After every re-refinement step, a fully converged radiosity solution is obtained. This leads to the problem of when to terminate the re-refinement process. It is possible to use perception metrics (e.g. Hedley [83], Gibson [84]) to determine if there are any perceptible differences between two consecutive solutions. However these methods are expensive to compute [83]. A simpler solution that is most frequently adopted, is to run the re-refinement process for a fixed number of iterations [90].

3.7 Clustered Hierarchical Radiosity

On further inspection of the hierarchical radiosity method, we find that the main drawback of traditional hierarchical radiosity is that object coherence is not taken into account during the linking stage.

Although the hierarchical data structure in traditional hierarchical radiosity is a much more sophisticated extension of Cohen *et al's.* [22] two-level patch-element hierarchy, refinement is only performed on a *per surface* basis.

However, a typical scene is composed of many objects, of which each object will be represented by many surfaces that are mutually visible [59]. Therefore, not only do surfaces and elements interact with each other, but objects (i.e. groups of surfaces or elements) also interact with other elements, surfaces and objects. Thus, we must also consider objects as another level in the hierarchy. Unfortunately there is no mechanism in traditional hierarchal radiosity that will allow surfaces to be grouped together into larger entities.

With the development of the *clustered hierarchical radiosity* method, groups of surfaces or *clusters* can now be considered when calculating energy exchanges. This means that distant objects or objects that exchange very little energy into a scene can now be represented by a single energy interaction, saving many costly surface to surface calculations that would otherwise be required in the initial linking stage. Clustered hierarchical radiosity is the most significant advancement made to Hanrahan *et al's* [39] hierarchical radiosity algorithm.

Clustering is a pre-processing stage that sorts groups of surfaces into *clusters*, usually based upon how close each surface is to each other. Each cluster forms a small part in a general hierarchical data structure. There have been quite a few approaches to clustering surfaces. The most notable approaches to clustering have been made by Rushmeier *et al.* [49], Kok [50], Sillion [57][66] and Smits *et al.* [59].

Manual (non-automatic) Clustering Methods

Rushmeier *et al.* [49] describe a non-automatic approach for clustering groups of small surfaces. In their method, an optically equivalent box approximates groups of small clusters of surfaces, such that the energy reflection distribution approximates the overall BRDF of the surfaces. This was achieved by using Monte Carlo sampling.

Based upon the progressive multi-pass method for solving global illumination by Chen *et al.* [38], this method attempts to simplify the geometry of an environment for indirect illumination to accelerate global illumination calculations.

The progressive multi-pass method is a hybrid method that aims to combine the advantages of radiosity and ray-tracing. In particular, radiosity is used to generate a coarse, but view independent, global illumination solution, whilst ray-tracing techniques such as Monte Carlo Path Tracing techniques are used to generate high resolution, view dependent, final rendered images.

Thus, the aim of the Rushmeier *et al.* [49] method is to simplify the geometry of a scene, such that the $O(k^2)$ time complexity of hierarchical radiosity is reduced. However, the radiosity solution that is computed for the simplified scene is only used to calculate weakly directional, indirect illumination.

This method has two important disadvantages.

- Cluster selection must be done manually, and in advance.
- This method does not maintain a hierarchy, which therefore means clusters cannot interact.

Kok [50] introduces a method that extends traditional radiosity by grouping together surfaces that are small and close together, to create 'macro-patches'. The motivation for grouping these patches together is that individually, these patches are almost never selected to shoot energy into an environment. However, the *total* unshot energy from a number of these small, but closely packed surfaces, can be quite considerable.

There are two methods for calculating form factors, based on ray-tracing. The first method is by Wallace *et al.* [33] and uses *directed* rays between a source patch and all other patches in a scene, to calculate the energy received by the source patch. The second method, as used by Malley [30] and Sillion *et al.* [32], is an *undirected* method that casts rays in a cosine distribution, from the source patch into the environment. Thus, Kok [50] presents two methods for grouping patches.

Unfortunately the patch-grouping method presented by Kok [50] is non-automatic, but it is mentioned that the adaptive grouping of patches is possible.

Automatic Clustering Methods

Sillion [57][66] proposed a clustering method that works by creating a hierarchical set of abstract volumetric entities that exchange energy. Each volume is represented as an isotropic scattering medium and is expressed by an *extinction coefficient*, which measures the rate of attenuation per unit length due to absorption and scattering.

The aim of isotropic clustering is to eliminate the $O(n^2)$ initial linking time, that is present in traditional hierarchical radiosity [39]. Sillion uses a *k-d* tree to partition the environment into clusters of surfaces. A *k-d* tree is essentially identical to a BSP tree, except that the splitting planes are always axially aligned. However, other clustering methods such as octrees or hierarchical bounding boxes can be used in place of *k-d* trees [66].

Whichever clustering method is used, the entire scene is represented by a root cluster and the initial linking process simply consists of creating a *self link*, which is a single link from the root cluster to itself. Since a self link takes constant time to build, the $O(n^2)$ linking time is completely eliminated. Instead, linking is performed solely within the hierarchical refinement method, such that light transport is modelled between adjacent patches, patches and clusters, and adjacent clusters.

Smits *et al.* [59] presented a clustering algorithm that estimates the energy transfers between clusters, whilst maintaining reliable error bounds on each transfer. Two methods for bounding the energy transfer between adjacent clusters are presented [59]. The two methods are derived by systematically introducing approximations into the exact expression of energy transfer (see Equation 2.13).

The first method uses the maximum radiance value between pairs of surfaces in adjacent clusters, to bound the energy transfer between clusters. Thus, any cluster that

has been linked according to this criterion has established an *α-link*. *α-links* are efficient to compute, requiring O($n$log$n$) time and space complexity.

The second method introduces a further approximation. A much cruder bound that only uses the distance between surfaces in both clusters, is placed on the energy transfer equation. Hence this bound requires no knowledge of the surfaces and so is very efficient to compute. *β-links* only require O(n) time and space complexity to compute. Thus any clusters that were linked using this bound, have established a *β-link*.

The linking stage attempts to link clusters together via *β-links*. If the link between a pair of clusters is considered not to be accurate enough, then *α-linking* is attempted. If *α-linking* is still insufficient, then the child clusters of the larger cluster are recursively refined against the smaller cluster. However, if neither of the clusters have children, then normal hierarchical patch refinement is performed on the surfaces of each cluster.

Thus, this method accelerates the traditional hierarchical radiosity algorithm by replacing the very expensive initial linking procedure with clustering. The results [59] show that the overall complexity of this clustered hierarchical radiosity algorithm is O($k$log$k$ + n), where k and n represent the number of initial surfaces and the number of patches created after refinement, respectively.

Hasenfratz *et al.* [87] have compiled an excellent analysis of clustering strategies for hierarchical radiosity. The following clustering strategies were studied:

- Proximity clustering.
- Overlapping *k-d* trees.
- Overlapping *k-d* trees with limited branching.
- Tight fitting octrees.

The most important goal for a good clustering algorithm is a method that faithfully models the light transfer between groups of objects. Unfortunately, there is currently no precise definition that can be computed *a priori*, which will specify the creation of clusters that will faithfully model light transfer. Instead, a set of heuristics have been

devised by several researchers (e.g. Christensen *et al.* [61], Sillion *et al.* [63], Gibson *et al.* [67]) to determine how surfaces are to be clustered.

Hasenfratz *et al.* [87] outline four required properties that will yield a satisfactory simulation of light transfer:

- A proportional representation of light transfer precision, with respect to the level of cluster refinement.
- The avoidance of overlapping clusters.
- The preservation of object group shapes.
- Clusters should maintain surfaces of similar sizes.

Results show that the time taken to cluster a scene using proximity clustering is much slower than the overlapping *k-d* tree or tight fitting octree methods, and can be a problem for scenes that contain a very large number of surfaces. This is primarily due to the more complex bottom-up algorithm required to group surfaces together. This is in contrast to the *k-d* tree and octree methods, which use simple but very fast, top-down recursive subdivision methods.

The proximity clustering technique produces the most predictable behaviour and generates the best shadow quality results, which is due to the more tightly fitting clusters that are generated. However, the overlapping *k-d* tree with limited branching method appears to offer the best compromise in quality and speed.

An important observation gained from the analysis by Hasenfratz *et al.* [87] is that clustering generally generates reliable results, but exhibits less well behaved results for scenes that contain large numbers of small surfaces.

Regular Grid Data Structures

The most popular regular data structures are the uniform grid and the octree, which were commonly used in the past to accelerate ray-tracing. These two data structures

are simple to construct and offer very efficient access to the voxels contained within the data structure.

The *k-d* tree data structure was used in the first clustering algorithm by Sillion [57][66], for hierarchical radiosity. However, this data structure caused problems with surfaces that straddle the splitting planes of the *k-d* tree.

The inability of these data structures to adequately handle surfaces that straddle voxel boundaries, is a major drawback. Sillion [57] proposed that these surfaces should be placed at the lowest level in the cluster hierarchy, which entirely contains them. However, Hasenfratz [87] reports that for many models, this can have very negative consequences since a large number of objects can end up at very high levels in the hierarchy. This can adversely affect computation speed.

Christensen *et al.* [61] modify the octree method such that bounding box of the child octants can change. This solves the problem of surfaces that straddle voxel boundaries, but has the drawback of introducing overlapping clusters.

Despite the problems outlined above, regular grid data structures offer extremely reliable cluster creation and are very quick to build. These data structures have been used successfully by a number of authors (e.g. Sillion [57][66], Christensen *et al.* [61], Hedley [83], Willmott [90]) to construct clusters for clustered hierarchical radiosity.

Hierarchical Bounding Volumes

Hierarchical bounding volumes aim to provide tight fitting bounding shapes around groups of spatially adjacent surfaces, such that the entire scene is organised into distinct objects. In order to simplify bounding volume calculations, axially aligned boxes are usually used although any simple geometric shape could be used, for example, bounding spheres.

Constructing hierarchical bounding volumes is complex. Many bounding box generation techniques are bottom-up techniques that are based on Goldsmith and

Salmons' [26] automatic bounding box creation method. In order to construct the bounding boxes using this method [26], it is necessary to estimate the cost of adding a new object and thus the evaluation cost of the whole tree. It is therefore necessary to use a cost function based on the intended use of the tree.

In Goldsmith and Salmons' algorithm [26] the heuristics was tailored towards producing efficient ray-bounding box intersections. However, Gibson [73] reported that this method will generally group surfaces together, that are well suited for use in hierarchical radiosity clustering algorithms. A very useful explanation of Goldsmith and Salmon's algorithm [26] is given by Haines [29]. This document [29] also includes changes to the original algorithm that provides addition efficiency for ray tracing.

Muller *et al.* [88] present an automatic bounding box creation method that stores hierarchical bounding boxes in a binary tree. Thus, this method can efficiently cluster a scene in $O(n\log n)$ time.

There has been much research into the automatic construction of hierarchical bounding volumes. Although these data structures can be complex to build, they do create good quality, tight-fitting bounding volumes that are suitable for clustered hierarchical radiosity.

3.8 Discontinuity Meshing

Shadows are extremely important features in a scene. They provide important visual cues, hence if they are missing or misrepresented in a scene, they can cause glaring visual anomalies. Discontinuity meshing is a technique that computes the shadows cast by objects. When casting shadows due to area light sources, discontinuity meshing provides the most complete description of a shadow, as this technique is able to locate discontinuities *within* the penumbra. Discontinuity meshing is a purely geometric approach to computing shadows that does not make use of any object or image space coherence.

Discontinuity meshing for radiosity systems was introduced independently by Heckbert [36] and Lischinski *et al.* [41]. Both Heckbert and Lischinski *et al.* initially focused on a 2D 'flatland' world, before extending the work to three dimensions [44][45].

The main aim of discontinuity meshing is to generate high quality meshes that accurately encode the discontinuities in the radiosity function across a surface, before a radiosity solution is computed for the mesh. This is because the quality of a rendered image is heavily dependent on the size and shape of the initial mesh [45]. Otherwise, artefacts such as *light leaks* and *shadow leaks* will manifest themselves in the final solution. Discontinuity meshing avoids these artefacts by explicitly calculating all shadows (umbra and penumbra) in a scene.

Generating a discontinuity mesh requires a wedge tracing technique for casting shadow discontinuities from all light sources, through all occluding surfaces, onto the surrounding geometry. For a polygonal light sources and occluding surfaces, this accomplished by:

- Casting all wedges constructed from each vertex of a light source, through all edges of an occluding surface (Vertex-Edge events or VE events).
- Casting all wedges constructed from each edge of a light source, through each vertex of an occluding surface (Edge-Vertex events or EV events).

Each wedge tracing operation produces a discontinuity line, where a discontinuity mesh is composed of a complete set of discontinuity lines. It must be noted that this procedure will not always produce a *complete* discontinuity mesh. There is a particular visual event known as the edge-edge-edge (EEE) event, which must be included to produce a complete discontinuity mesh.

EEE events occur when three edges interact together in the environment. These events produce a 'ruled quadratic' discontinuity surface. There are two main types of EEE events: E_eEE and EEE events. E_eEE are similar to EEE except that one edge belongs to an edge of a light source. Unfortunately, EEE events are not easy to find and produce quadratic discontinuity surfaces, which in turn are awkward to triangulate.

Computing a complete discontinuity mesh is extremely expensive. For this reason, most discontinuity meshing algorithms omit EEE visual events. According to Worrall [82], EEE events have very little perceptible impact on the final solution, so can largely be ignored. Hedley [83] describes the three types of discontinuity meshes that can be created:

- A complete discontinuity mesh.
- A full discontinuity mesh.
- An extremal discontinuity mesh.

Complete discontinuity meshes contain discontinuities that represent all visual events (EV, VE and EEE events) involving light sources. These meshes have been used in 'back-projection' algorithms by Drettakis *et al.* [58]. Back-projections can be used to quickly compute exact radiance values from light sources, but Hedley [83] notes that it is not obvious whether the computational costs associated with calculating a complete discontinuity mesh is justified, because back-projections only accelerate visibility computations from primary light sources.

A full discontinuity mesh is a complete discontinuity mesh that excludes all visual events that require a search of the scene polygons [83]. In other words, only visual events that involve the vertices and edges of light sources are considered. EEE events are not computed for these meshes.

An extremal discontinuity mesh is a mesh that only contains discontinuity lines that form the minimal and maximal shadow boundaries. The minimal boundary is defined by the boundary that separates the umbra and penumbra, and the maximal boundary is defined by the maximum penumbra boundary.

It is important to note that Drettakis [53] states that the maximal shadow boundary is comprised totally of EVE events (EV and VE events), but the minimal shadow boundary can be comprised of both EVE and EEE events. Thus a full discontinuity mesh and an extremal discontinuity mesh may not always define the minimal boundary correctly.

A very useful summary of discontinuity meshing was given by Slater *et al.* [92]. This book [92] also contains information on other methods for generating shadows (e.g. Blinn [28]) and describes the most popular data structures on handling shadows, most notably BSP trees, the 'shadow volumes' method by Crow [9] and DM-trees (discontinuity meshing trees).

A DM-tree combines a 2D BSP tree and a 'winged edge data structure' (Baumgart [6][7]) to enable polygons to be combined by its discontinuity edges. The winged edge data structure maintains efficient adjacency information.

The shadow volumes method by Crow [9] has since been revised by Bergeron [24] and is still one of the most popular methods for shadow computation [92].

3.9 Extremal Discontinuity Meshing

Discontinuity meshing is currently the only algorithm that can locate discontinuities within a shadow, in particular, within the shadow's penumbra. Thus together with radiosity, the most complete representation of the radiance function across a surface can be obtained.

However, computing a full discontinuity mesh is computationally very expensive. It was found by Hedley [83] that the radiosity gradients within the penumbra of a shadow, are very subtle. Using perception metrics, Hedley [83] found that the penumbral boundary could be approximated by linear interpolation, without any perceptible difference. Thus by only casting an extremal discontinuity mesh, the total number of discontinuity lines cast can be greatly reduced.

Nishita and Nakamae [13] were the first to compute the exact extremal shadow boundaries. Campbell and Fussell [40] have since optimised this algorithm and have incorporated shadow volume BSP trees and object space acceleration techniques, for efficient shadow boundary computation.

Campbell [42] gave an algorithm for computing extremal discontinuity lines. The algorithm for computing extremal discontinuity lines is:

```
FOR every edge belonging to an occluder DO
   FOR every light source vertex DO
      Construct the wedge plane formed by the current light source vertex
         AND the current occluder edge
      Compute the angle between the normal of the wedge plane
         AND the normal of the occluder
   ENDFOR
   The plane with the smallest angle will form a maximal extremal boundary
   The plane with the largest angle will form a minimal extremal boundary
ENDFOR
```

Figure 3.9 shows an illustration of the wedge tracing process for casting discontinuity lines. The two highlighted wedges show an illustration of the wedges that will form the maximal and minimal extremal discontinuity lines.

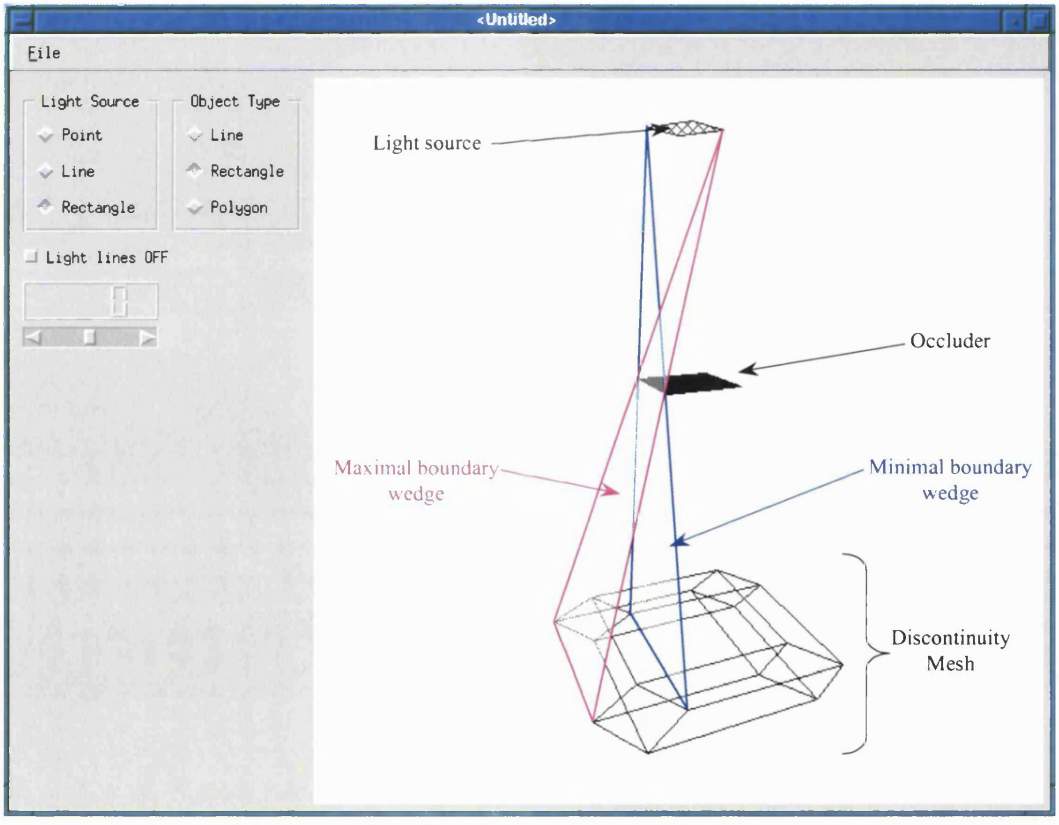


Figure 3.9 Computing extremal discontinuity lines.

From the previous algorithm, the main task for computing discontinuity lines is the calculation of the angle between the plane of the wedge formed, and the plane of the occluding surface.

Figure 3.10(a) shows a diagram of a single wedge, formed between one light source vertex and one edge belonging to an occluding polygon.

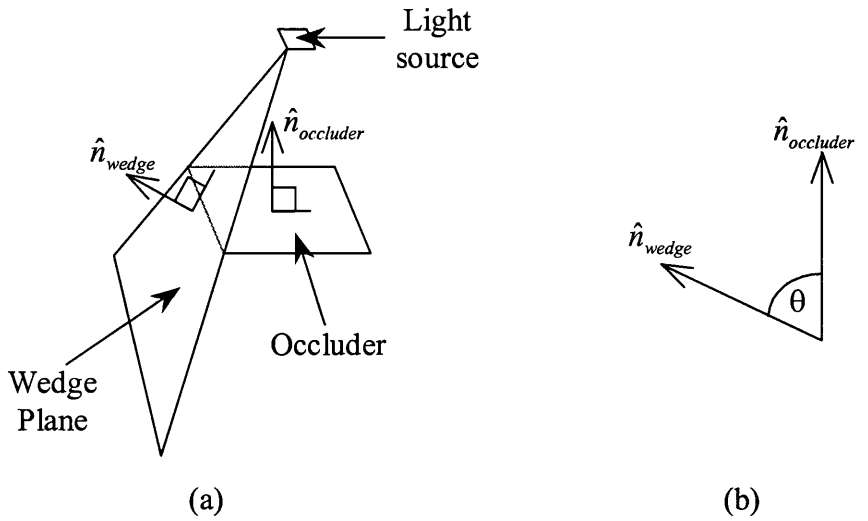


Figure 3.10 Calculating the angle between a wedge and an occluder.

By including the normals of the plane of the wedge and the plane of the occluding polygon in this diagram, we can see that the angle θ between the two planes is simply the angle between the normals (see Figure 3.10(b)).

However, it is not necessary to calculate the angle θ explicitly. Since the goal is to find out which wedges form smallest and largest angles between the light source vertices and the occluding polygon edges, this can be accomplished very efficiently by computing the dot product between the normals of the two planes.

Hence, if both normal vectors $\hat{n}_{wedge}$ and $\hat{n}_{occluder}$ are *unit vectors*, then $\cos\theta$ is simply:

$$\cos\theta = \hat{n}_{wedge} \cdot \hat{n}_{occluder}$$

Thus, this expression for calculating $\cos\theta$ can be substituted in place of calculating θ in the above algorithm for computing an extremal discontinuity lines.

The Scope of Discontinuity Meshing

In general, radiosity rendering methods only consider static scenes. This is mainly due to the rigidity of the meshes that are required to compute a radiosity solution for a scene. A moving object would require parts of the mesh to change dynamically and hence the re-computation of the radiosity solution for the sections of the scene that have changed. The main complexity of dynamic scenes is the problem of computing the *dynamic shadowing* for moving objects, due to area light sources.

Unfortunately, there has been very little progress made towards the goal of dynamic shadowing. This topic is extremely complex and only a few attempts at solving this problem have been made. The most significant research into the area of dynamic shadowing has been carried out by Worrall *et al.* [68][82] and Loscos and Drettakis [78]. Chrysanthou [71] also proposed a method of updating shadow discontinuities due to area light sources, which use dynamic BSP trees. Unfortunately this method requires frequent insertion and deletion of items in the BSP trees. Due to the nature of BSP trees, it is inevitable that these items will become fragmented and thus will increase the total size of the BSP trees. Chrysanthou [71] acknowledges the problem of fragmentation, but Worrall [82] concludes that BSP trees are unsuitable for dynamic shadowing.

Extremal Meshing for Dynamic Discontinuity Meshing

The main problem with dynamic discontinuity meshing is the heavy cost involved in recomputing the polygon meshes that belong to every surface in a scene, every time a surface is moved.

Every mesh has been constructed such that the radiosity across all surfaces is accurately modelled. Hence any moving objects will change the light transport within a scene and thus the shadows cast by occluding surfaces will change accordingly.

Since shadows interact with the surrounding geometry, it is a very costly procedure to locate and update any such changes.

There are two approaches to dynamic shadowing and dynamic discontinuity meshing [78][82]. Both methods are based upon the original method by Worrall *et al.* [68]. The approach by Worrall [82] is a 'mesh-based' approach, whereas Loscos *et al.* [78] used a 'wedge-based' approach.

The principle differences between the two approaches, as summarised by Worrall [82], are:

Data: The wedge-based method requires all wedges in the scene to be explicitly stored; the mesh-based method does not.

Time: The mesh-based method only re-casts wedges when a migration is either starting or ending; the wedge-based method re-casts them every frame.

It can be seen that Worrall's [82] mesh-based approach for dynamic discontinuity meshing is far more scalable than the wedge-based approach by Loscos *et al.* [78].

Regardless of whichever approach is taken, a significant amount of work has to be done when updating the polygon meshes, after the movement of surfaces.

The Benefits of Extremal Discontinuity Meshing

Neither Worrall [68][82] nor Loscos *et al.* [78] make use of extremal discontinuity meshing. Instead, Worrall [82] generates a *full discontinuity mesh* (i.e. no EEE events) and Loscos *et al.* [78] generates a *complete discontinuity mesh.*

Casting a discontinuity mesh is expensive [83], thus reducing the number of discontinuity lines that need to be generated will significantly reduce the time taken for discontinuity meshing. Also, reducing the number of discontinuity lines cast will also reduce size of the mesh generated after a constrained Delaunay triangulation.

Figure 3.11 presents an illustration showing the differences between an extremal discontinuity mesh and a full discontinuity mesh. The resulting polygon meshes are also displayed, to show how discontinuity meshes are integrated into a scene.

Figure 3.11(a) shows the extremal discontinuity mesh generated by a rectangular area light source and a rectangular occluding surface. Once the discontinuity lines have been cast, they need to be incorporated into the surfaces that they have interacted with. For the scene shown in Figure 3.11(a), all discontinuity lines interact with a single floor surface and thus after a constrained Delaunay triangulation, Figure 3.11(b) shows the final polygon mesh.

Figure 3.11(c) shows a full discontinuity mesh generated by the same scene geometry as shown in Figure 3.11(a). The definition of a 'full discontinuity mesh' as stated by Hedley [83], is a discontinuity mesh that contains discontinuity lines generated by vertex-edge and edge-vertex visual events.

It can be clearly seen in Figure 3.11(c) that a full discontinuity mesh is more complex than an extremal mesh. As a result, the triangulated mesh generated from a full discontinuity mesh (Figure 3.11(d)) is significantly larger than the mesh produced by an extremal discontinuity mesh (Figure 3.11(b)).

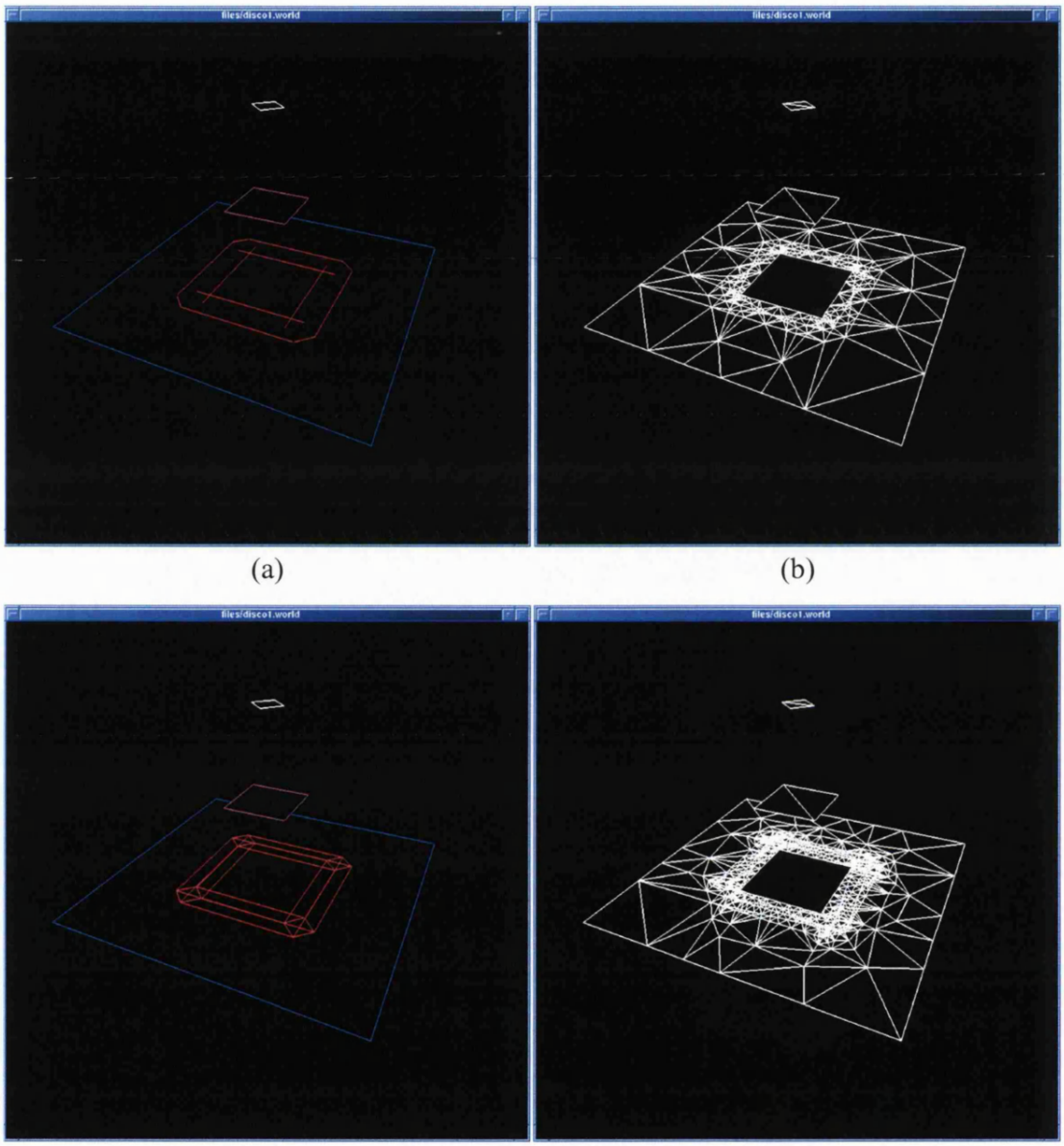


Figure 3.11 (a) An extremal discontinuity mesh and (b) its resulting triangulated mesh. (c) A full discontinuity mesh and (d) its resulting triangulated mesh.

Type of Discontinuity Mesh	Number of discontinuity lines generated	Number of triangles in triangulated mesh
Extremal	12	584
Full	32	1528

Table 3.1 A comparison between extremal and full discontinuity meshing.

Table 3.1 shows a summary of the number of discontinuity lines and number of triangles generated by the scenes in Figure 3.11.

From the results shown in Table 3.1, it can be seen that dynamic discontinuity meshing can benefit greatly from extremal discontinuity meshing.

Research performed by Hedley [83] shows that there is very little perceptible difference between the results obtained by extremal discontinuity meshing or by full discontinuity meshing. Thus extremal discontinuity meshing should always be used in preference to full discontinuity meshing.

Further Experimental Justification for the Use of Extremal Meshing

Since Worrall [68][82] and Loscos *et al.* [78] ignored the use of extremal discontinuity meshes, we performed a simple experiment in using extremal discontinuity meshes for modelling a rotating object. We show that extremal meshes are sufficient to represent all shadow subtleties in this experiment.

Using a single, stationary rectangular area light source, we rotate a planar square occluder to show the progression of both umbra and penumbra of the shadow by using extremal meshing. A full mesh comparison is also given.

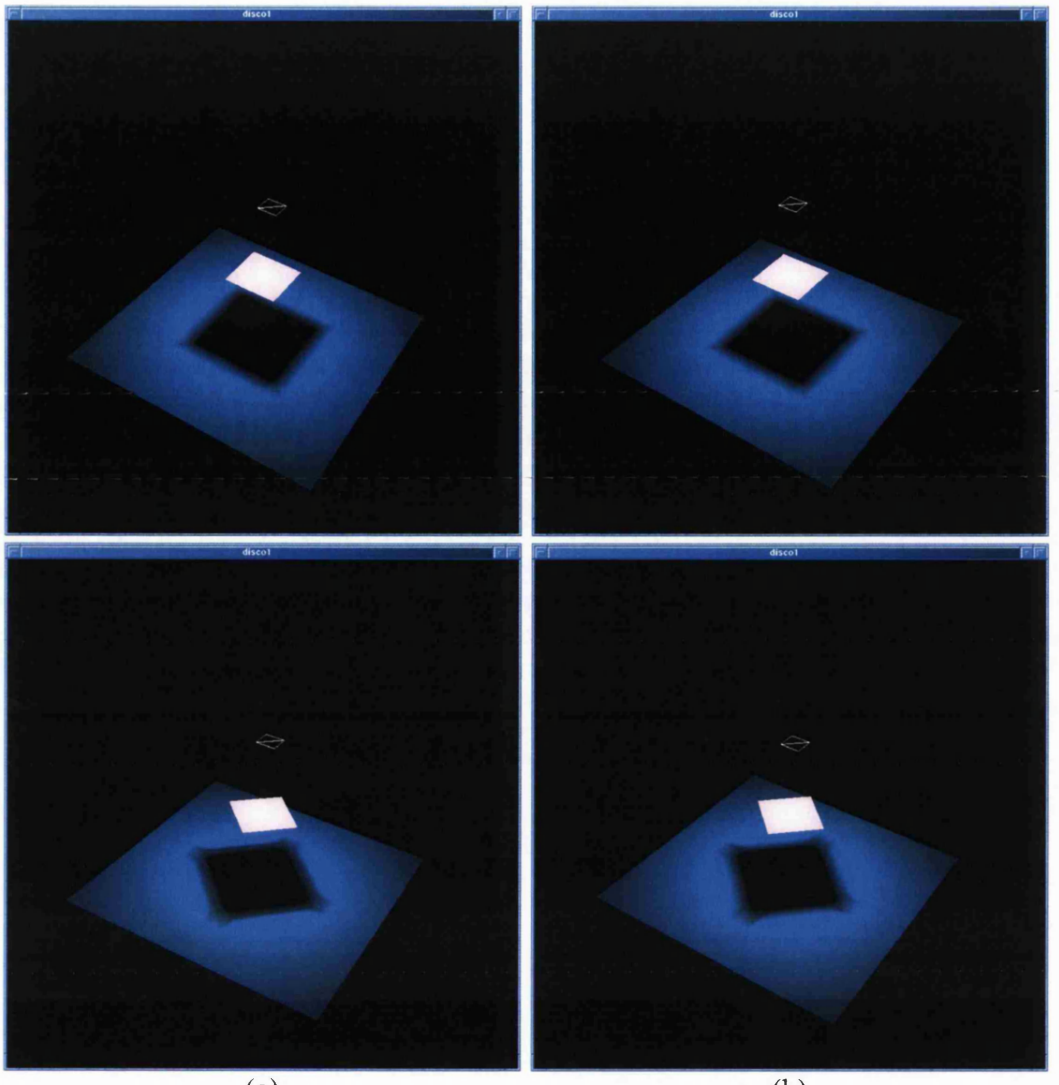


Figure 3.12 The images shown in column (a) show the results from a full discontinuity mesh and column (b) the results from an extremal discontinuity mesh.

Figure 3.12 shows two rendered snapshots of the experiment, with the occluder rotated at 0° and 45° respectively. The full discontinuity meshed solution is shown in the left column and the extremal discontinuity meshed solution, in the right column.

These images show that there are minimal perceptible differences in the quality of the shadows generated by extremal meshing. Even though there are radiosity gradients within the penumbra of the shadow, these are very subtle and can be approximated by linear shading. Thus, when considering the discontinuity mesh of dynamic objects,

extremal meshing will enable significant savings in shadow computation with no loss in quality.

The following table (Table 3.2) shows the amount of work done by full discontinuity meshing and extremal meshing for rotating the occluding polygon in Figure 3.12, 360° in 15° increments.

	Full discontinuity meshing	Extremal discontinuity meshing
Total no. of discontinuity lines cast	768	288
Total no. of triangles generated	36672	14016

Table 3.2 A comparison of the total work done by rotating the occluding polygon shown in Figure 3.12 for full and extremal discontinuity meshing.

3.10 Summary

In this chapter, it has been established that hierarchical radiosity [39] is the most advanced method for solving the radiosity equation. However, hierarchical radiosity has a running time complexity of $O(k^2 + n)$, where k is the number of initial surfaces and n is the total number of patches generated by hierarchical refinement. This means that hierarchical radiosity can *completely* solve the radiosity matrix (Equation 2.39) in linear time – a significant improvement on the $O(n^2)$ running time complexity of previous radiosity methods, for example progressive refinement radiosity [31].

The only downside to the hierarchical radiosity method, is the $O(k^2)$ initial linking time. This problem has since been optimised by incorporating various clustering techniques (e.g. Sillion [57][66], Smits *et al.* [59]) into the hierarchical radiosity algorithm. Hence the improved running time complexity for a clustered hierarchical radiosity algorithm is now $O(k\log k + n)$.

However, even the improved $O(k\log k + n)$ time complexity is still insufficient for a large k. Since very little more can be done to improve upon the clustered hierarchical radiosity algorithm, optimisation must be found elsewhere.

The most logical place to look for optimisation is within the input surfaces. Due to the general nature of global illumination within the scene, there will often be localised portions of illumination within a scene. Hence careful optimisation of a scene, can lead to a smaller, but essential set of input surfaces that need to be considered by a radiosity algorithm. Thus in the following chapter, we propose and implement a technique that utilises the illumination information within a scene to localise groups of surfaces, which can be gathered together into clusters. These clusters of surfaces can then be rendered independently.

We have also demonstrated in this chapter the benefits of extremal discontinuity meshing. The simple example presented in Figure 3.11 can be extended to complex scenes without a loss in perceptible detail in the rendered scene. This is consistent with the results by Hedley [83] and is proposed as a simple extension to the work by Worrall [82].

To demonstrate the effectiveness of extremal discontinuity meshing, the simple example shown in Figure 3.11 was extended to allow the occluder to rotate. Dynamic scenes are non-trivial problems to solve. By allowing the occluding polygon to rotate, the entire discontinuity mesh and radiosity solution must be recomputed. Recasting the full discontinuity mesh for Figure 3.11 would require 32 discontinuity lines to be wedge traced and the creation of 1528 triangles after triangulation, and represents a considerable amount of work.

In contrast, casting an extremal discontinuity mesh would only require 12 discontinuity lines to be wedge traced and only 584 triangles to the created after triangulation. As shown in Figure 3.12, the rendered images for extremal discontinuity meshing are practically indistinguishable from the fully discontinuity meshed solutions. Therefore, we will restrict attention to all further work on discontinuity meshing to extremal meshing.

In the following chapter, we develop a new clustering strategy for hierarchical radiosity that utilises extremal meshing, to enable efficient computation of large complex scenes.

Chapter 4 An Improved Clustering Strategy for Hierarchical Radiosity

4.1 Introduction

Radiosity is a powerful tool for rendering photo-realistic scenes. Once the radiosity of a scene has been calculated, a 'virtual reality' walkthrough of the scene is immediately available. However, this comes at a costly price as calculating the radiosity of a scene is anything but trivial.

With traditional radiosity techniques, rendering a scene meant solving the often very large radiosity matrix (Equation 2.39). As mentioned in the previous chapter, the sub-structuring method introduced by Cohen *et al.* [22] can be used to reduce the computation time, but even this method becomes impractical when solving scenes that contain a moderately large number of surfaces.

The hierarchical radiosity technique described in Chapter 3 has since superseded Cohen's sub-structuring method. Currently, larger scenes can now be rendered in a practical amount of time. However, hierarchical radiosity does have a very costly $O(k^2)$ initial linking stage, where k is the number of initial surfaces. As a result, for large to very large scenes this method will spend most of its time in the linking stage.

To overcome this problem, a clustering technique is applied to the input scene such that spatially adjacent surfaces are grouped together into *clusters*. A cluster is simply a bounding volume that encapsulates a number of nearby surfaces. Clustering in this fashion assumes that spatially adjacent surfaces have similar properties, such that the properties of the cluster realistically represent all the surfaces contained within the volume. In general scenes, groups of surfaces naturally cluster into distinct objects, so the clustering technique is normally very effective.

Clustering is an invaluable technique for optimising radiosity computation. Without clustering, the hierarchical radiosity algorithm would be too expensive to solve large complex scenes. However most clustering techniques do not make "intelligent" use of the scene geometry, particularly in the surrounding geometry. Clustering algorithms simply group together surfaces that fit within the dimensions of their data structures.

For example, the *octree* data structure is one of the most popular methods used in clustering. It is very easy to visualise and computationally very quick to construct - the $O(n\log n)$ construction time is very desirable. However this method is not without its problems [87]. The rules for refining the octree structure are quite rigid, only allowing eight fixed size, axially aligned children to be created. During octree construction, we quite often find that surfaces straddle adjacent child octants and hence a decision on whether the surface should be split or replicated, must be made. Neither of these options are really acceptable. To circumvent these problems, Christensen *et al.* [61] modified the octree such that variable sized octants are allowed, but unfortunately this introduces the problem of overlapping octants. Other methods involve using heuristics to determine where to place these surfaces within the data structure [63].

An alternative method for clustering scenes is to use *hierarchical bounding volumes*. This data structure is usually constructed in a bottom-up fashion, and was used by Smits *et al.* [59] in order to avoid the problem of eliminating surface duplication, that was associated with octrees and other rigid data structures. It is also possible to generate hierarchical bounding volumes using a multi-resolution hierarchy of regular grids [63]. This is also a bottom-up technique and has been used successfully by Sillion [63] and Gibson *et al.* [73].

Both these methods of clustering significantly improve the scalability of traditional hierarchical radiosity, but it must be noted there are hidden costs within the $O(k\log k+n)$ running time complexity of the clustered hierarchical radiosity algorithm. Within the patch/cluster refinement and linking phase there are two costly procedures: form factor calculation and visibility computation. Although we can approximate unoccluded form factor calculations by using Siegel and Howell's analytic form factor

for a finite differential area and a finite disk [10], we are still left with a very costly visibility computation.

Visibility computation can potentially take $O(n^2)$ time to compute, but acceleration techniques can reduce this to $O(n\log n)$ for tree base subdivision methods, or even $O(n)$ for uniform grid methods. Thus we must include these costs within the linking phase, which means that as the number of input surfaces increases, these costs will become more and more noticeable. Willmott [90] calculated that the time complexity of finite-element radiosity methods including visibility testing, is potentially $O((k\log k)^2+n\log n)$ where k is the number of initial surfaces in the scene and n is the total number of patches generated after hierarchical refinement.

Hence we find that the overall scalability of clustered hierarchical radiosity is inevitably bounded by the number of input surfaces. Therefore, the only way to improve scalability is to reduce the number of surfaces that have to be solved by the hierarchical radiosity algorithm. Currently, this can only be achieved by clustered hierarchical radiosity methods [56].

Improving the Performance of Hierarchical Radiosity

In order to reduce the time complexity of the hierarchical radiosity algorithm to below the existing time complexity of current clustered hierarchical radiosity algorithms, a new strategy for processing the input scene is required. Since we cannot change the time complexity with which the hierarchical radiosity algorithm solves n final solution elements, the only way to improve the algorithm is to reduce the amount of work done by the hierarchical radiosity solver. This invariably means the number of input polygons that are passed to the hierarchical radiosity solver has to be reduced, but at the same time, this reduction must have minimal affect on the final solution.

For scenes of moderate size, existing clustered hierarchical radiosity methods produce excellent results in a tractable amount of time. However, recently Willmott [90] stated that for scenes with a very large number of input polygons (e.g. in the order of millions of polygons), even with the $O(k\log k+n)$ running time complexity of these algorithms, such scenes are no longer feasible with current technology.

To date, there has only been one significant attempt at reducing the time complexity of the hierarchical radiosity algorithm, by directly minimising the number of input polygons. Willmott [90] designed a new *Face Cluster Radiosity* algorithm that is based upon Garland's [86] method for polygonal surface simplification. Essentially Willmott's [90] new algorithm reduces the number of input polygons by simplifying tessellated surfaces and applying vector based radiosity to remove the faceted appearance that would otherwise be left by the simplified surfaces. This method is very effective at reducing the number of input polygons and has been shown to render enormous scenes in remarkably little time.

Soler *et al.* [89] introduced an interesting method for reducing the time taken for solving very large hierarchical radiosity problems. In this method, a very large hierarchical radiosity problem is replaced by a collection of smaller hierarchical radiosity problems. Unfortunately, in order to gain the benefits of this method, scenes need to have a high degree of self-similarity. However, this method was proved to be very effective in solving botanical scenes [89].

Another approach to reducing the time complexity of hierarchical radiosity is to make more use of the input scene geometry. The topology of the input scene contains a wealth of information that can aid in large reductions in the amount of computation required to solve a scene. Unfortunately, to fully utilise the scene information for reducing the number of input polygons solved by hierarchical radiosity, would require a full radiosity solution in the first place! This means that this optimisation method would become an *a posteriori* algorithm and thus cannot be used as an optimisation pre-processing stage. However, there is still important information within the initial scene that can be utilised *a priori*. In particular, the *coherence* of the input scene geometry could lead to a better clustering strategy.

Generally, within a scene, groups of polygons will cluster together to form objects of varying complexity. Most importantly, the illumination of the scene is usually configured according to the layout of these objects. Therefore, we generally find that scenes will be comprised of distinctly illuminated and non-illuminated areas.

Current clustered hierarchical radiosity methods only make use of the object clusters during the radiosity link refinement stage. Thus, in the final radiosity solution, all clusters and surfaces contribute to the work that has to be performed by the hierarchical radiosity solver.

It is however, possible to improve upon this by making greater use of the object clusters that are created and how those clusters affect other clusters and the surrounding geometry. Thus, if an area of the scene is considered relatively self contained, why not simply remove that area from the scene and render it as a separate entity?

It is possible, in a pre-processing stage, to localise the initial scene geometry to form 'independent areas' of illumination that can be detached from the main scene. This can be achieved by estimating the energy interactions between the primary light sources and the object clusters, and the interactions between object clusters, to produce bounded areas that can be solved independently. These clusters effectively *localise* the scene into smaller sub-scenes.

Since each of these bounding areas can be solved independently, this means that a sub-portion of the entire scene can be removed. Hence there will be fewer surfaces that need to be processed by the hierarchical radiosity solver.

4.2 Reducing Scene Complexity

Consider the scene shown in Figure 4.1. It is composed of a single rectangular light source, a single rectangular floor surface and two chairs. Each chair is constructed from 548 triangles, hence this scene contains a total of 1098 polygons. After triangulation, the scene in Figure 4.1 will contain a grand total of 1100 triangles.

Using traditional hierarchical radiosity, potentially 1.21 million link interactions may need to be created in the initial linking stage. By ignoring pairs of polygons that do not face each other, the number of links can be greatly reduced. In this particular case,

approximately 191000 links would be generated. This leads to a significant saving in memory, but the linking stage is still has a running time complexity of $O(n^2)$.

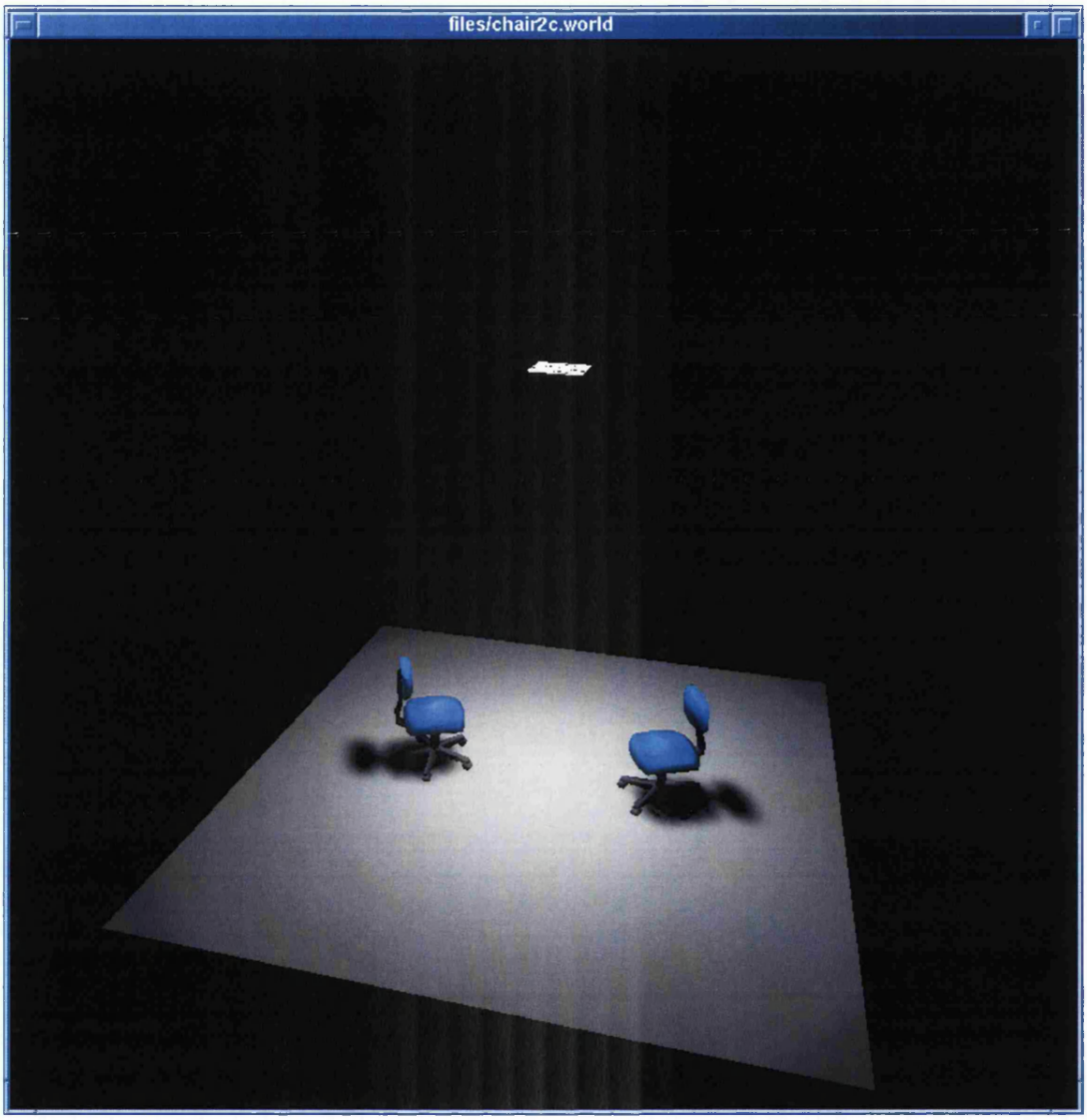


Figure 4.1 A simple scene.

The scene in Figure 4.1 is very simple, only containing two distinct objects (i.e. the two chairs), but a great deal of work has to be done to compute a radiosity solution. Adding more chairs to the scene will quickly make computing a radiosity solution intractable.

Upon closer inspection of Figure 4.1 it can be seen that due to the configuration of the scene, the two chairs do not significantly interact with each other. Thus each chair

could be treated independently. Due to the $O(n^2)$ time complexity of the linking procedure, solving each chair independently will yield significant savings.

Hence introducing a *scene localisation* technique has the potential of greatly reducing that total amount of computation that needs to be done by the hierarchical radiosity algorithm. In the case of Figure 4.1, solving each chair independently would potentially require 302500 link interactions each. For both chairs, this results in an immediate reduction by half, of the 1.21 million links that otherwise would be potentially need to be computed. By ignoring pairs of polygons that do not face each other, approximately 41000 links would actually need to be created, per chair. Again, this is yields a significant saving in memory.

Also, by considering each chair separately reduces the total number of surfaces by half, that need to be processed by the hierarchical radiosity algorithm. Since the time complexity for the hierarchical radiosity algorithm is $O(k^2+n)$, this corresponds to a theoretical four-fold gain in efficiency at the linking stage. Thus less links will be generated, which leads to a quicker linking time and a saving in memory. Since there are two chairs, a more realistic gain of two is likely.

4.3 Localising a Scene

To localise a scene, all surfaces within the scene must be grouped into suitable clusters. The criteria for generating clusters are:

- Surfaces must group together to form distinct clusters.
- Clusters must have tight fitting bounding boxes around their contents.
- Clusters must not overlap.

Grouping surfaces together to form distinct clusters is the most important step in scene localisation. Without well-formed clusters, it is very difficult to accurately judge where the boundary for localisation should begin. By ensuring that clusters have tight fitting bounding boxes, this will minimise any unnecessary overlapping between clusters.

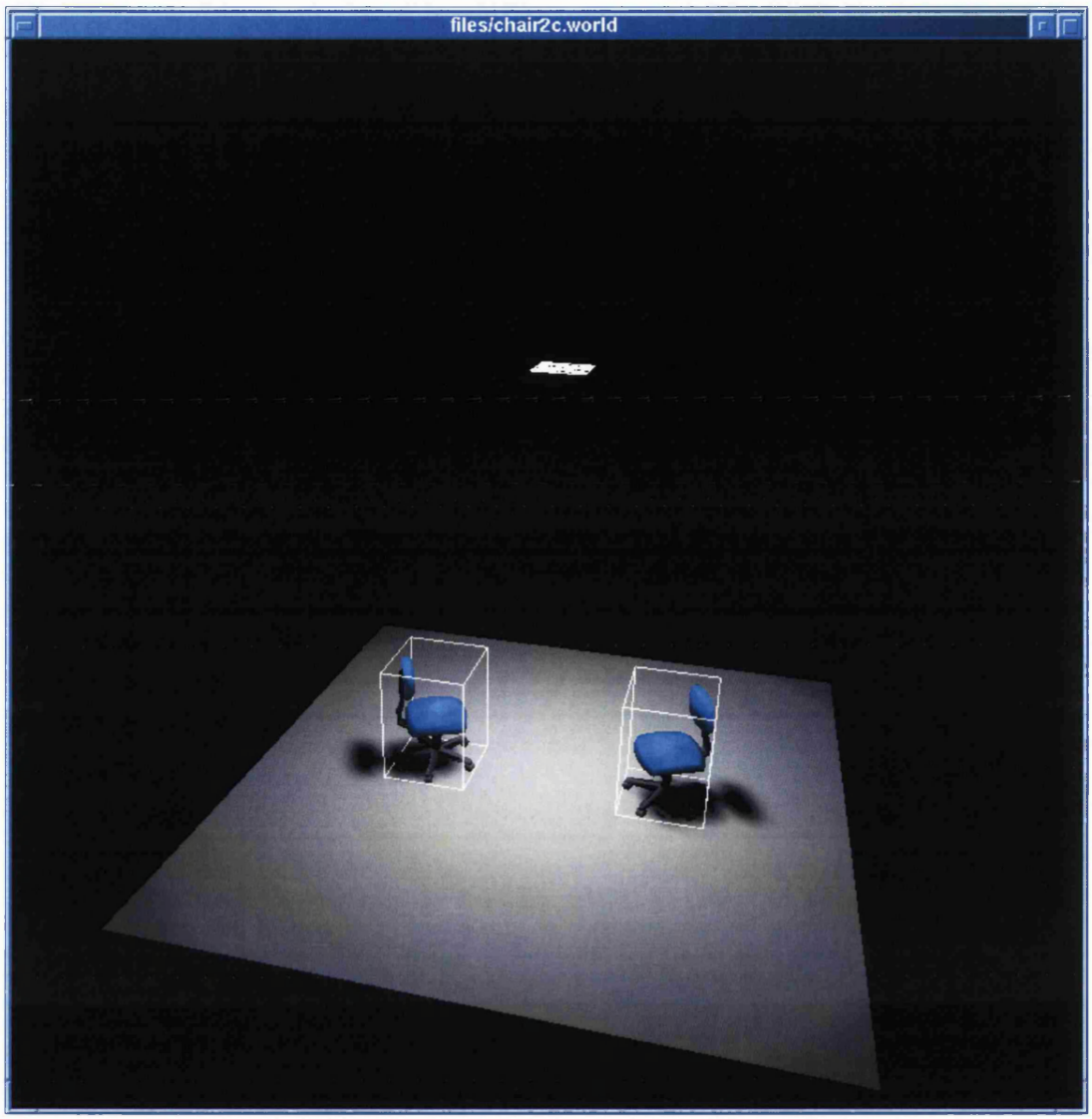


Figure 4.2 The clusters generated from Figure 4.1.

The type of bounding box used for clustering is arbitrary and it is most common to use axially aligned bounding boxes, as these objects are easy to maintain and very quick to compute. Figure 4.2 shows the results of clustering for Figure 4.1 using axially aligned bounding boxes.

Although two tight fitting axially aligned bounding boxes were constructed around the two chairs in Figure 4.2, these clusters are not optimally tightly fitting. Therefore, it is possible for clusters to overlap unnecessarily. More complex bounding shapes could be used to generate much tighter fitting bounding volumes, but axially aligned bounding boxes are very efficient objects to compute and store.

Ideally, the optimal localisation solution would be to reduce the scene into many localised clusters that contain a relatively small number of surfaces. By keeping the number of surfaces within a cluster to a minimum, the $O(n^2)$ running time complexity of the linking process, will also be kept to a minimum. Essentially, it is desirable to create many localised clusters that contain a minimal number of surfaces, than to create few localised clusters that contain many surfaces.

Unfortunately, overlapping bounding boxes will generally have to be merged together, to form a larger cluster. This in turn will reduce the overall efficiency of localisation technique by reducing the total number of clusters that will be created, whilst at the same time, increasing the number of surfaces contained within that cluster. Overlapping bounding boxes mainly occur when two clusters have surfaces that are sufficiently close together, that it is difficult to define a separation boundary.

Assessing the Suitability of Clusters for Localisation

Once suitable clusters have been located, each cluster is assessed for its suitability to be localised. The two main criteria for deciding whether a cluster is detachable from a scene are:

- The light transport between adjacent clusters must be minimal
- The shadow influence of a cluster must not interact with adjacent clusters

Calculating the light transport between clusters can be accomplished with the same methods used in clustered hierarchical radiosity algorithms. In this case, it is the amount of flux or power, transferred between clusters that will be required for assessing the suitability of cluster detachment. Hence if a cluster receives and transmits little light energy from and from surrounding clusters, then that cluster would be deemed suitable for localised rendering. In practice, calculating flux involves calculating the BFA product between pairs of clusters.

Computing the amount of flux transported (i.e. the BFA product) between clusters requires the calculation of radiant exitance (B), the volume-to-volume form factor (F) and the area (A) of a cluster. The two excepted methods for accomplishing this are the

methods by Sillion [57][66] and Smits *et al.* [59]. It must be noted that these methods do not necessarily have to use the BFA product, as there are many other metrics for measuring the light transport between adjacent clusters. For example, Sillion [57][66] used the BF product.

For this localisation algorithm, only an approximate value for light transport between two clusters is required. By using the idea of lazy linking [56], the relevant surfaces within a cluster that directly receive illumination from primary light sources can be found very efficiently. From this, an approximate but useful measure of the 'brightness' of a cluster can be obtained at a minimal computational cost. If lazy linking is used in the initial linking stage in the hierarchical radiosity algorithm, then computing the radiosity of the surfaces within the cluster that receive primary illumination, can be gained at very little extra cost.

Since we are only interested in the light transfer in the direction in which two clusters face each other, it is necessary for the BFA product to include directional information between two clusters. This can be achieved by applying Gibson and Hubbold's method for calculating the *projected-area in the direction of transfer* [73]. The projected area of a surface is calculated by multiplying the cosine of the angle between the normal of the surface, and the direction vector of the two facing clusters. Hence the BFA product will provide a useful estimate of the radiant flux or power, transferred between pairs of clusters.

Before a cluster is deemed suitable for detachment, the effects of the cluster on the surrounding geometry have to be considered. Specifically, it is the shadows cast by the surfaces within a cluster onto the surrounding geometry that will finally determine if that cluster will be detached. Generally, if the shadow cast by a cluster interacts with an adjacent cluster, both clusters will be forced to merge into a larger cluster.

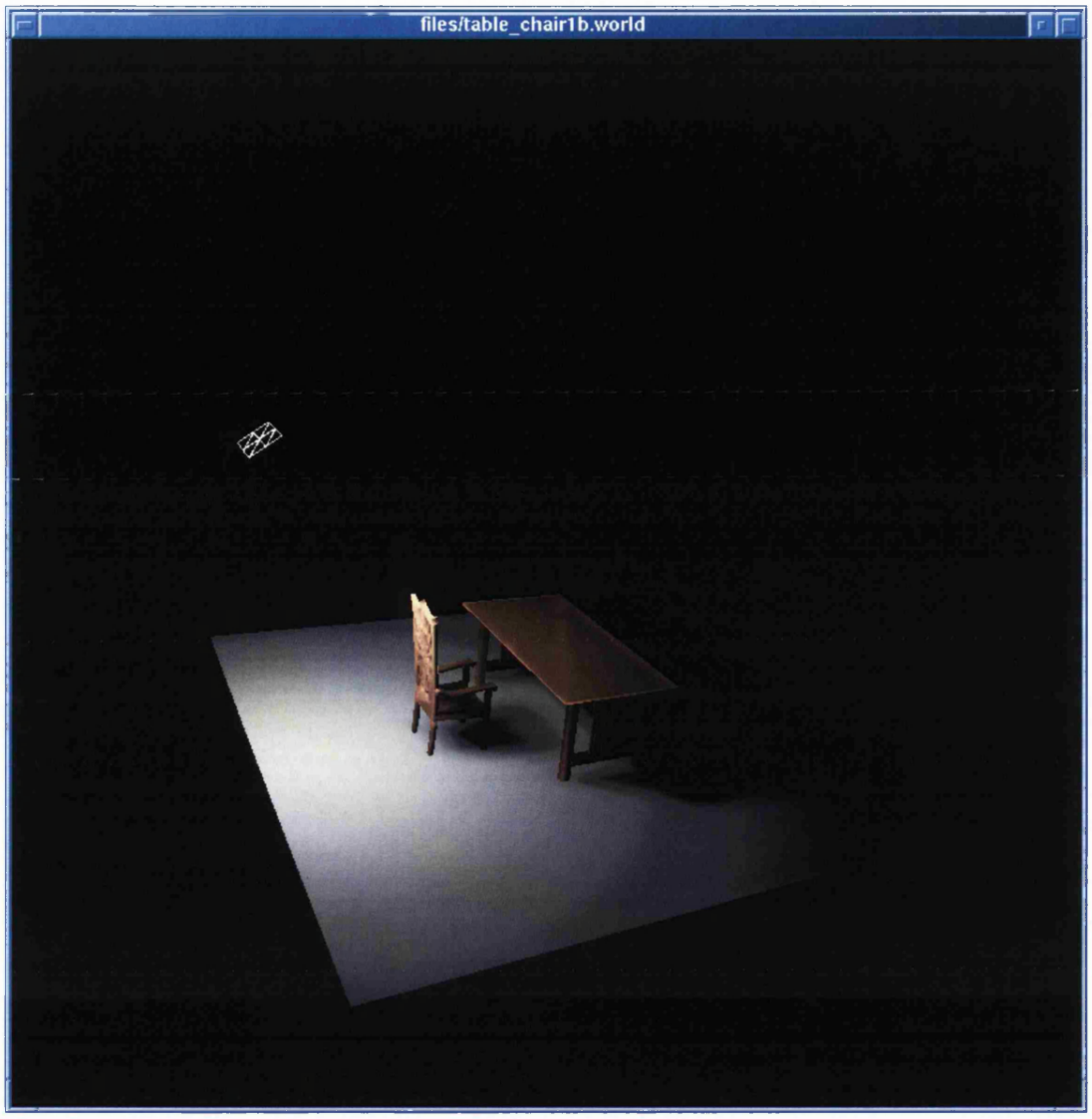


Figure 4.3

Consider the scene in Figure 4.3. The proximity of the chair to the table is sufficient, that both objects would be considered to be part of the same object cluster. However, due to the light source and the surface geometry of the two objects, there is minimal light interaction between the two objects.

By observing the shadow cast by the chair, it can be seen that the shadow does not interact with the table at all. Thus this is a special case situation where the bounding box of two (or more) clusters overlap, but do not need to be merged.

Locating the Extremal Shadow Boundary of a Cluster

The next step is to compute the extents of the shadow cast by the surfaces within a cluster. This can be achieved by casting discontinuity lines for each surface within the cluster, onto the surrounding geometry. However, this is somewhat expensive. A more efficient approach would be to cast an extremal discontinuity mesh (see Section 3.9) instead of a full discontinuity mesh. Once the extremal discontinuity mesh has been cast, an outer convex hull can be generated which can then be used to define a boundary for cluster detachment.

Discontinuity meshing is a very expensive process, but can be optimised by only casting extremal discontinuity lines. However, the scene localisation method does not require a highly accurate shadow boundary for cluster detachment. All that is required is a maximum or upper boundary around the cluster, which will encapsulate the entire shadow cast by the surfaces within a cluster.

Since the bounding box around a cluster represents such a boundary, the discontinuity mesh cast by the axially aligned bounding box of the cluster itself, will always encapsulate the entire shadow cast by the surfaces contained within. By casting an extremal discontinuity mesh, can once again, optimise the efficiency of discontinuity meshing process.

Fortunately, a large proportion of the cost of computing an extremal discontinuity mesh can be further avoided in this algorithm. Since only the maximal boundary of the extremal discontinuity mesh is required to compute the maximum boundary of the bounding box around the cluster, there is no need to compute the extremal mesh of the umbra. Thus half of the discontinuity lines can be immediately dismissed.

A further saving can be made by observing that the discontinuity lines generated by either VE or EV events, will provide sufficient information for the construction of the maximal bounding for a cluster. Hence the total number of discontinuity lines that need to be cast can be reduced by an additional 50%. As a result, the time required for computing an extremal discontinuity mesh for this algorithm, is minimal.

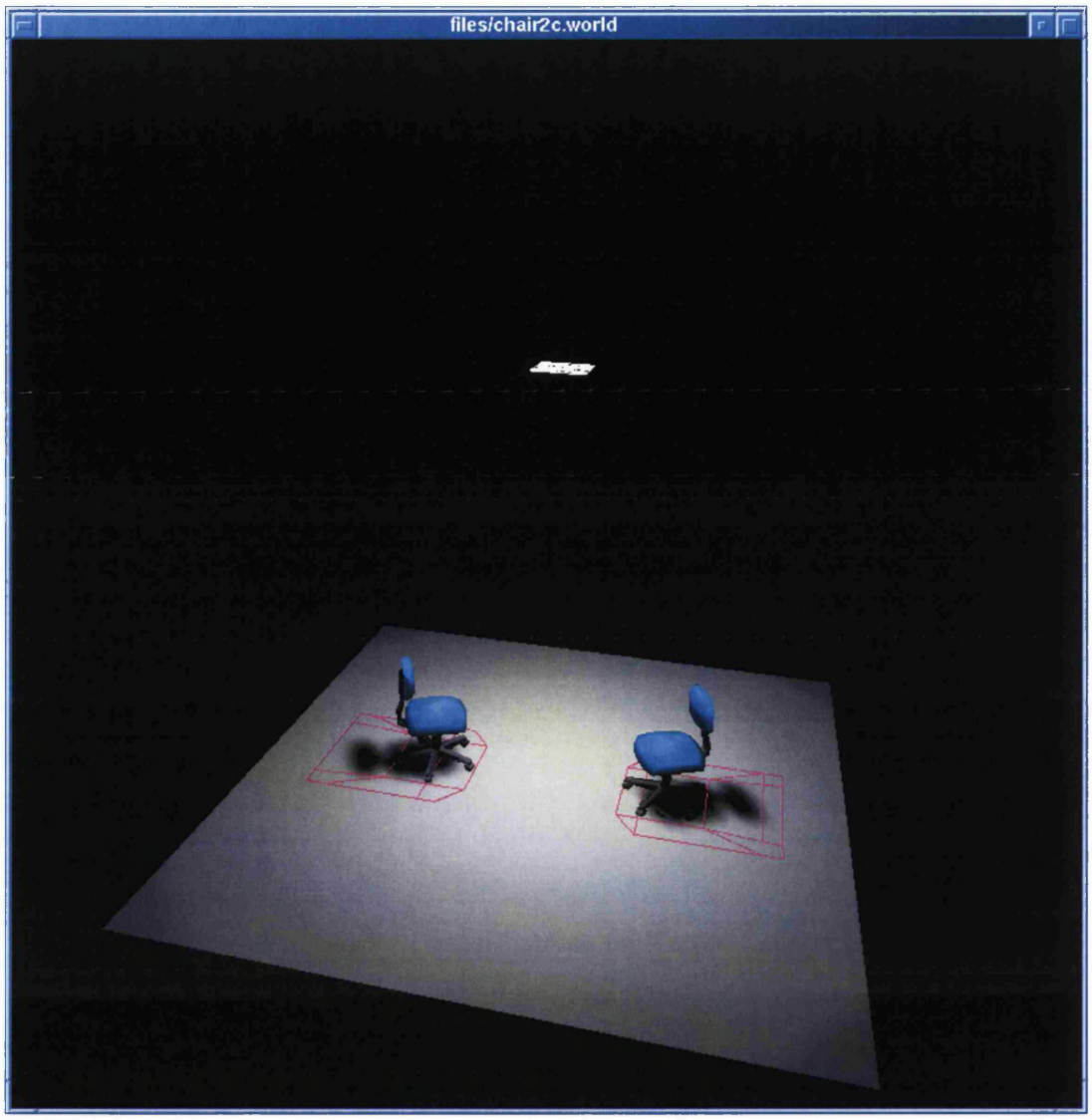


Figure 4.4 The extremal discontinuity meshes that are generated by the clusters shown in Figure 4.2

The results of casting an extremal discontinuity mesh from the axially aligned bounding boxes shown in Figure 4.2, can be seen in Figure 4.4.

Once the extremal shadow boundary has been computed, the bounding box of the cluster is expanded to incorporate the shadow boundary, and this cluster is ready for detachment. This is shown in Figure 4.5.

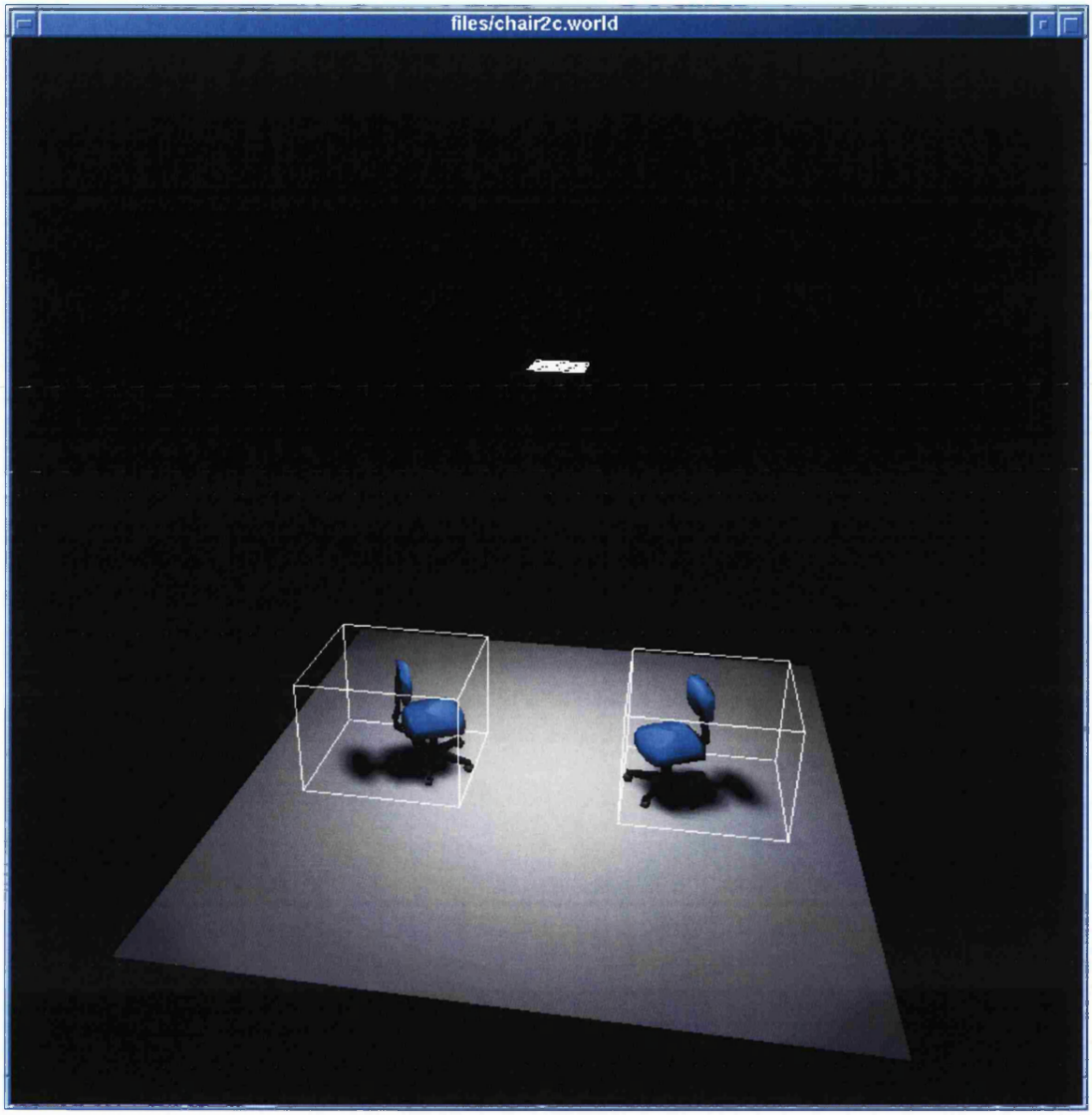


Figure 4.5 The final expanded bounding boxes.

4.3.1 A Simple Two-Pass Method for Clustering Surfaces

There are a variety of clustering methods for clustering surfaces together and they broadly fall into two categories: regular, typically axially aligned space subdivision methods and hierarchical bounding volumes. Since the goal for clustering is to form distinct objects from spatially close surfaces, this can only be achieved reliably by bounding box methods.

The most popular methods for generating hierarchical bounding boxes are based upon the automatic algorithm by Goldsmith and Salmon [26]. The hierarchy produced is versatile and the hierarchy of bounding boxes can be used to accelerate visibility testing. However, if visibility testing is accelerated by other means, then a simpler approach to clustering via bounding boxes, can be achieved.

The algorithm for generating bounding boxes is simple: all spatially nearby surfaces must be grouped together to form a cluster.

The main difficulty in collecting surfaces to form object clusters, is to determine the extents of the boundary of inclusion around a surface. This boundary will determine whether or not a surface is close enough to another surface or cluster, to be merged. If the boundary limit is set too large, then it is possible that two very close but distinct neighbouring clusters may be erroneously merged. Conversely, if the boundary limit is set too small, then many clusters will be formed that only contain a single or at most, a few surfaces.

For scenes that have an even distribution of surfaces, it is possible to set a single boundary limit. Unfortunately, this is usually not that case for general scenes as it is quite common for scenes to have an uneven distribution of densely and sparsely populated areas of surfaces.

To overcome this problem in the simplest possible way is to cluster all surfaces within a scene in two stages. The first stage clusters all surfaces that are very close together. In this stage, it is possible that many clusters may only contain a single or a few surfaces. Hence, the second stage coalesces clusters that are close together including any left over surfaces that were not clustered in the first stage. Each time a surface or cluster is merged, the bounding box around the expanded cluster is updated.

The most efficient method for computing the boundary limit of a surface is to use the minimum and maximum extents of the surface to create an axially aligned bounding box around it. In effect, every surface immediately becomes a cluster.

The advantage of treating individual surfaces as clusters means that a generalised algorithm for grouping surfaces and clusters can be created. This in turn will greatly simplify this two-pass clustering algorithm.

To ascertain whether or not a surface or a cluster is sufficiently close to another surface or cluster, is easily determined by testing whether their bounding boxes overlap. If the bounding boxes overlap, then the two objects (surface or cluster) are deemed to be close enough to be coalesced.

Computing the intersection between bounding boxes is quite complex. However, in this algorithm we are not concerned with the exact intersection details between overlapping bounding boxes. All that is required is a simple indication of whether or not two bounding boxes overlap.

This is easily accomplished by testing to see if any of the corners of one bounding box lie within the extents of the other bounding box. Hence, there are three possible conditions:

- All points belonging to the source bounding box lie totally within the target bounding box.
- Some points belonging to the source bounding box lie within the target bounding box.
- No points belonging to the source bounding box lie within the target bounding box.

The first condition indicates that source object lies totally within the target, whilst the second, indicates that the source object only partially overlaps the target object. However, both conditions require that the source object to be added to the target cluster.

Thus, computing the intersection between a pair of bounding boxes can be accomplished by the following pseudo code:

```
PROCEDURE testBBoxIntersection(BoundingBox A, BoundingBox B)
Set counter c to zero
   FOR all 8 boundingbox corner points DO
      IF a corner point in BoundingBox B lies within BoundingBox A THEN
         Increment counter c by 1
      ENDIF
   ENDFOR
   IF counter c is 8 THEN
      BoundingBox B lies totally inside BoundingBox A
   ELSE IF counter c is > 0 THEN
      Part of BoundingBox B lies inside BoundingBox A
   ELSE
      BoundingBox B does not intersect BoundingBox A
   ENDIF
ENDPROCEDURE
```

The algorithm for the first stage of clustering using bounding boxes is:

```
FOR all surfaces DO
   FOR all currently created clusters DO
      IF current surface bounding box overlaps current cluster THEN
         Add surface to current cluster
      ENDIF
   ENDFOR
   IF current surface did not overlap with any cluster THEN
      Create a new cluster and add current surface
   ENDIF
ENDFOR
```

Figure 4.6 shows the results of the first stage of bounding box creation, for a sample scene.

It can be seen from Figure 4.6 that several features, for example, the wheels and the back of the office chair, have been clearly identified and clustered together, but have not been included in the main bounding box. Thus, the second stage of this bounding box creation algorithm merges these smaller clusters into the main bounding box.

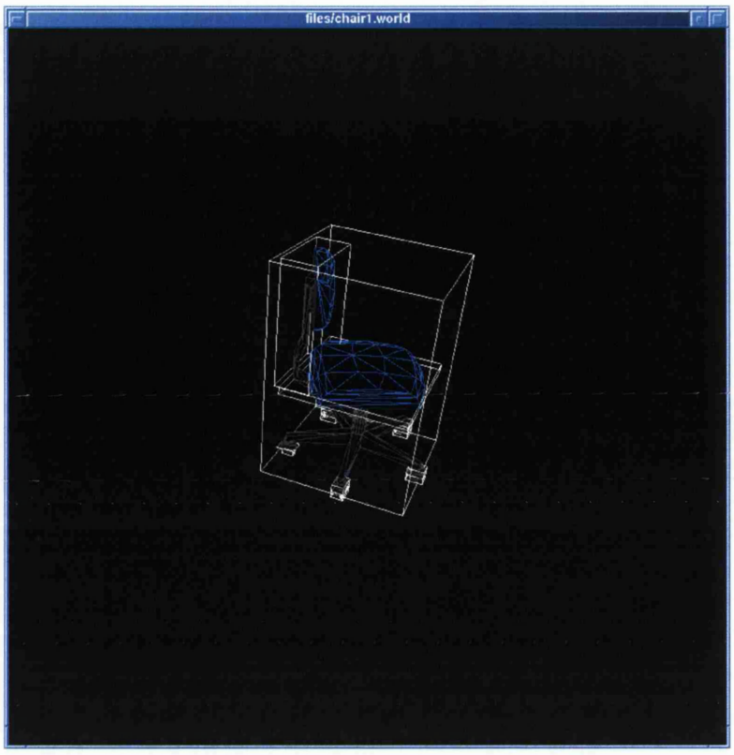


Figure 4.6 The results of clustering after the first stage of bounding box creation.

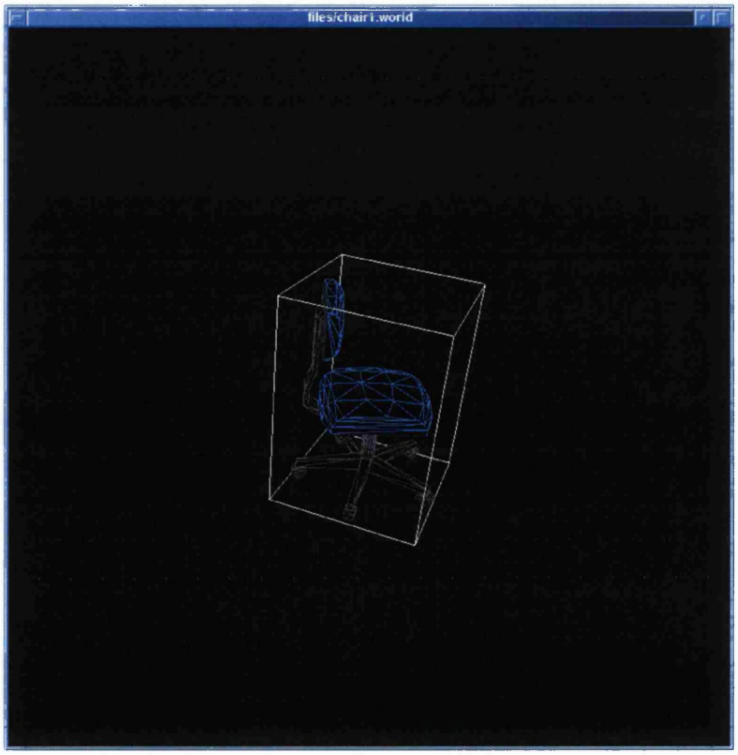


Figure 4.7 The final results of clustering after the second stage of bounding box creation.

The second stage of this two-pass method simply scans through all clusters created in the first stage, to check for any clusters that need to be merged. As with the first stage, exactly the same bounding box testing will be used in this stage.

```
FOR every cluster i DO
   FOR every cluster j DO
      IF i!=j THEN
         Check for bounding box intersection between i and j
         IF we have an intersection THEN
            Merge together clusters i and j
         ENDIF
      ENDIF
   ENDFOR
ENDFOR
```

Figure 4.7 shows the results after the second stage of this bounding box creation algorithm have been applied.

The performance of this algorithm $O(nm^2+m^2)$ where n is the number of input surfaces and m is the number of clusters created. The m^2 quadratic terms come from the simple linear searching algorithm used for merging objects (surfaces or clusters) together.

At first glance, this algorithm performs very poorly, offering $O(n^3)$ time complexity if every surface in the scene forms an individual cluster (i.e. $m=n$). However, general scenes intrinsically contain clusters of surfaces, which means that the worst case running time will only be achieved in unrealistically contrived scenes.

For general scenes, the number of bounding boxes created will be far less that the total number of input surfaces. Therefore, $m << n$ which means that this algorithm will typically have a running time complexity of $O(n)$. Hence, this algorithm should scale well for typical scenes even though the complexity of this algorithm is $O(nm^2)$.

4.3.2 Cluster Detachment

The final step before rendering, for the scene localisation process, is the detachment of the clusters from the main scene. The main algorithm for the cluster detachment process is as follows:

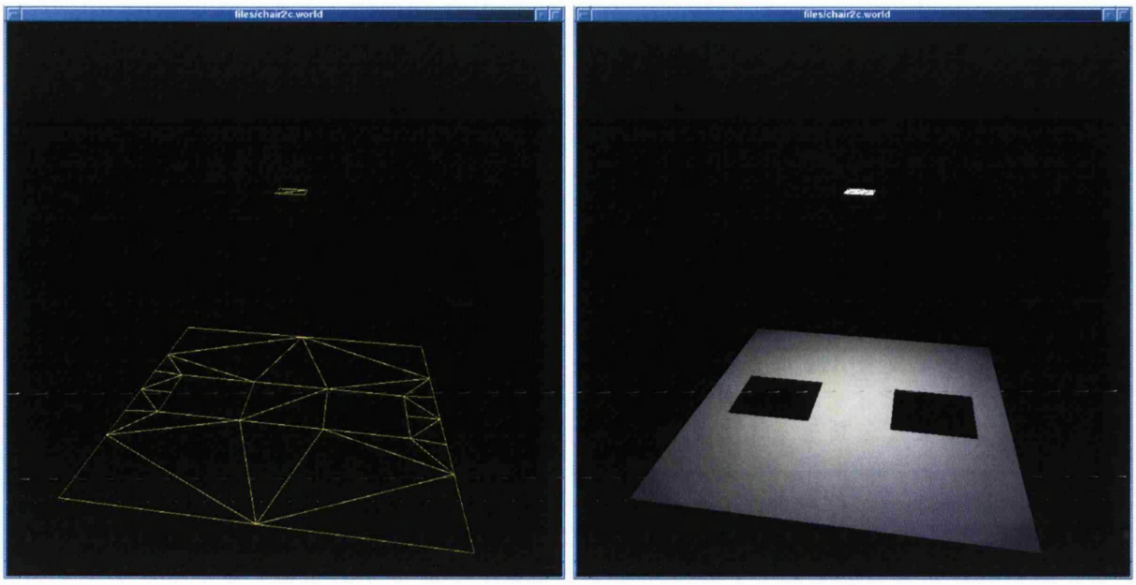
```
FOR every cluster DO
   FOR every surface that intersects the current cluster DO
      Cut out the area of intersection from the current surface
      Add the cut out area as a new surface to the current cluster
   ENDFOR
ENDFOR
```

For the scene in Figure 4.1, Figure 4.5 shows two clusters that are required for detachment. Once the clusters have been removed, two holes will be left in the floor surface. Unfortunately, it is most likely that the hierarchical refinement will ignore the presence of holes in a surface.

To solve this problem, any surface that contains holes will need to be decomposed into triangles, that is, the surface will need to be triangulated. This way, surfaces that contain holes will be represented by a collection of individual triangles and thus will not affect the hierarchical radiosity algorithm.

However, to ensure that all triangles are suitable for hierarchical radiosity refinement, all surfaces should undergo a constrained Delaunay triangulation. This is important since the form factor approximation calculations require all triangles not to be long and thin. A constrained Delaunay triangulation will ensure that the quality of the triangles produced is suitable for hierarchical radiosity.

For the scene in Figure 4.1, Figure 4.8 shows the resulting mesh after a constrained Delaunay triangulation has been applied to the floor surface, and Figure 4.9 shows the triangulated mesh for the two detached clusters.



(a) (b)

Figure 4.8 (a) The CDT mesh of the floor for the scene shown in Figure 4.1 after cluster detachment. (b) The rendered image.

Figure 4.9 The CDT mesh for the two detached clusters in Figure 4.1.

4.3.3 Reintegration

The reintegration process recombines the meshes of the separately rendered clusters, into a single final mesh. Essentially, this involves combining the individual meshes for each patch that was generated by the hierarchical radiosity solver.

The task of merging meshes can be a very complex process, but there is a simple method that will enable individually rendered meshes to be displayed without the need for mesh merging.

When all input surfaces have been triangulated, each triangle essentially becomes an individual patch and is refined separately by the hierarchical radiosity algorithm. Unfortunately, by treating adjacent triangles that are part of the same surface as individual patches, can cause disruptions in the continuity of the radiosity function at the triangle boundaries. By ensuring that adjacent triangles share common edges, this can largely solve the problem. The remaining problem is caused by T-vertices, which are generated during the hierarchical patch refinement process.

During the patch refinement process in the hierarchical radiosity procedure, patches are subdivided by bisecting the edges of the parent patch to produce four new sub-patches (see Section 3.3). Since bisection always takes place at the middle of an edge, every edge belonging to a *root* patch will maintain a binary tree of subdivided edges. Therefore, maintaining shared edges during hierarchical patch refinement is relatively simple.

As all triangular patches are now connected together, each patch can create its own mesh independently, but is still connected to the overall main mesh. Thus the need for merging meshes is no longer needed.

Figure 4.10 shows the results of reintegrating the left cluster in Figure 4.5, into the main scene. The results of the reintegration of the second (right hand) cluster into the main scene can be seen in Figure 4.11. Lastly, the complete scene is shown in Figure 4.12.

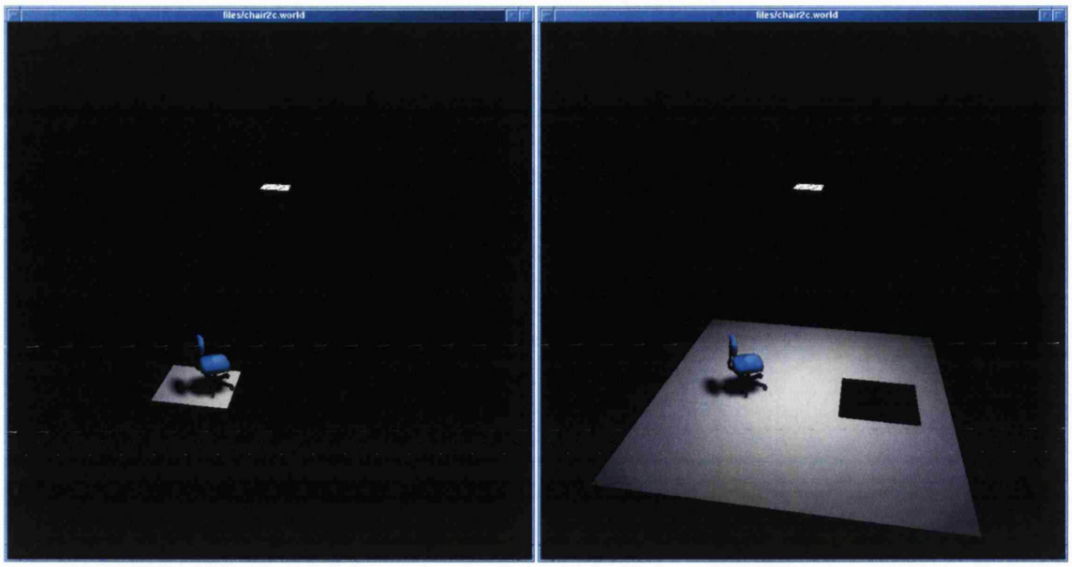


Figure 4.10 The rendered image of the left cluster in Figure 4.5 (left) and the reintegration of the detached cluster into the main scene (right).

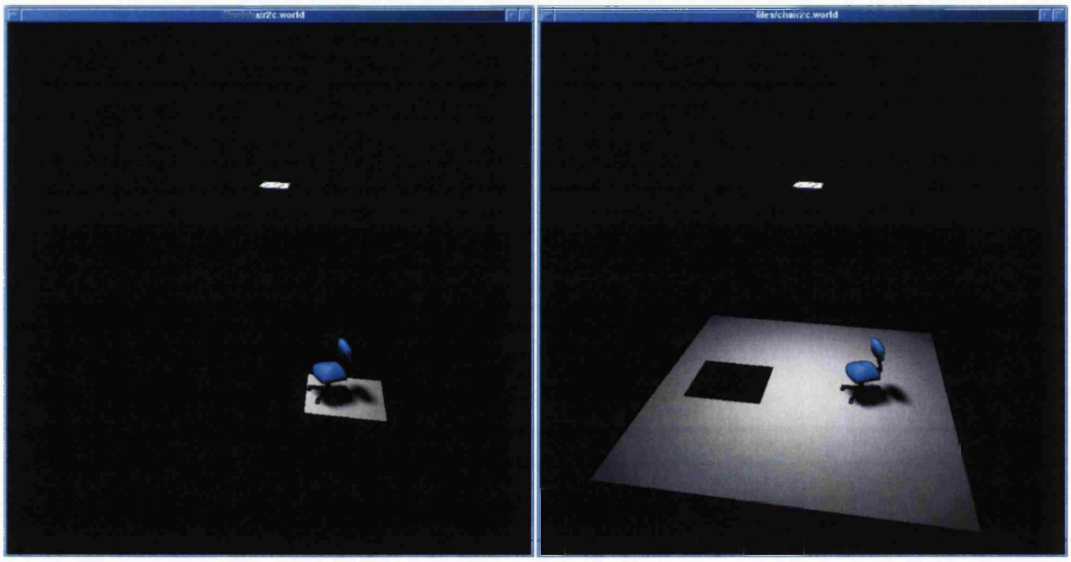


Figure 4.11 The rendered image of the right cluster in Figure 4.5 (left) and the reintegration of the detached cluster into the main scene (right).

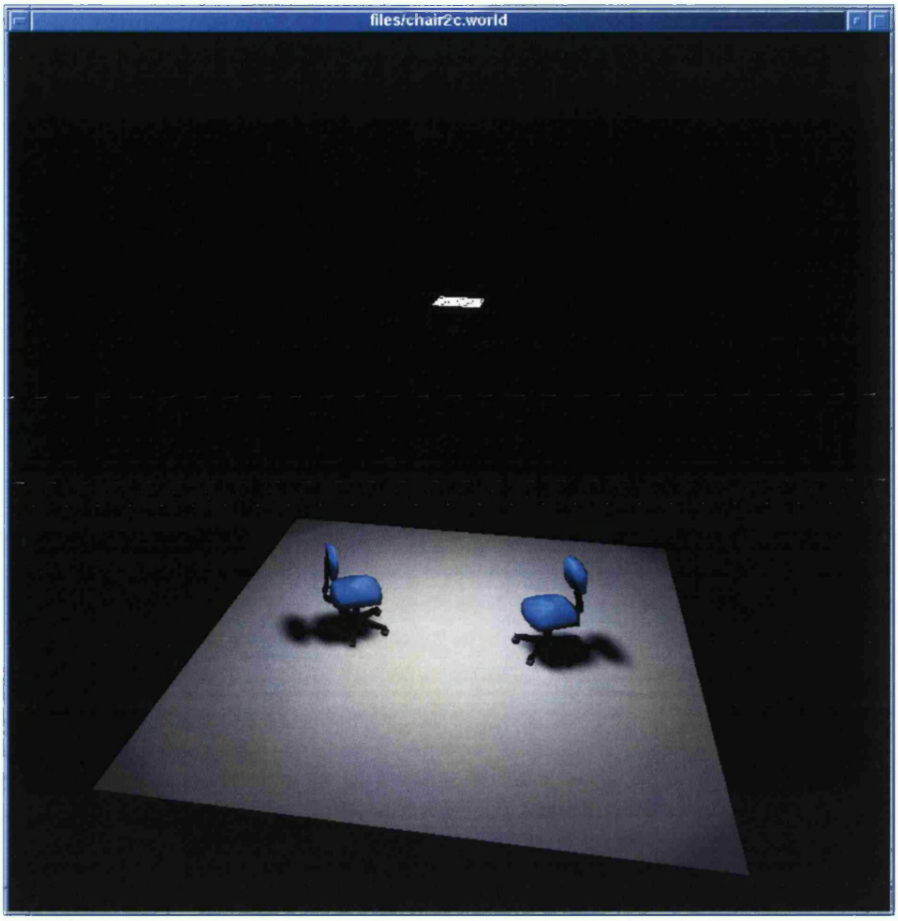


Figure 4.12 The final reintegrated scene.

4.4 Image Artefacts Caused by T-Vertices

It must be noted that one important side effect will occur, which is caused by the triangulation of a surface. The triangles generated by the constrained Delaunay triangulation process were generated externally, without the use of the radiosity function of the source surface to guide the triangulation process. Therefore, it is very likely that some of the boundaries formed by the edges of the triangles will be unsuitable to represent the radiosity function across the source surface. This is because radiosity solutions can be potentially discontinuous across surface/patch boundaries, as well as being continuous in areas without sharp shadow boundaries [37]. Hence, the appropriate level of continuity across all triangles generated by the constrained Delaunay triangulation must be maintained, or artefacts will become visible.

The most visibly obvious artefacts are caused by T-vertices (see Figure 4.13) and are compounded by the patch refinement process in the hierarchical radiosity algorithm. During the patch refinement process, it is possible for two patches that share a common boundary, to have different levels of subdivision.

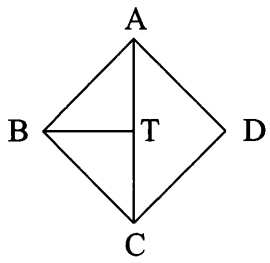


Figure 4.13 An illustration of a T-vertex (from Baum *et al.* [37]). If the vertices ACD define a triangle, then there is a T-vertex at T.

This can be seen on the floor surface in Figure 4.12 and the artefacts caused by T-vertices are particularly evident along the boundary lines formed by the constrained Delaunay triangulation.

Figure 4.14 shows a clearer wireframe view of the final mesh, after the reintegration process. It can be seen that the most noticeable artefacts are in the areas where the two detached clusters have been reintegrated. This is due to the differing levels of refinement of the individual triangular patches.

Unfortunately, the greater the difference between the levels of subdivision of the two adjacent patches that share a common edge, the more pronounced the artefacts would be. This is because a greater number of T-vertices will be present at the boundary between the two patches. Baum *et al.* [37] describe an *anchoring* technique that will eliminate T-vertices.

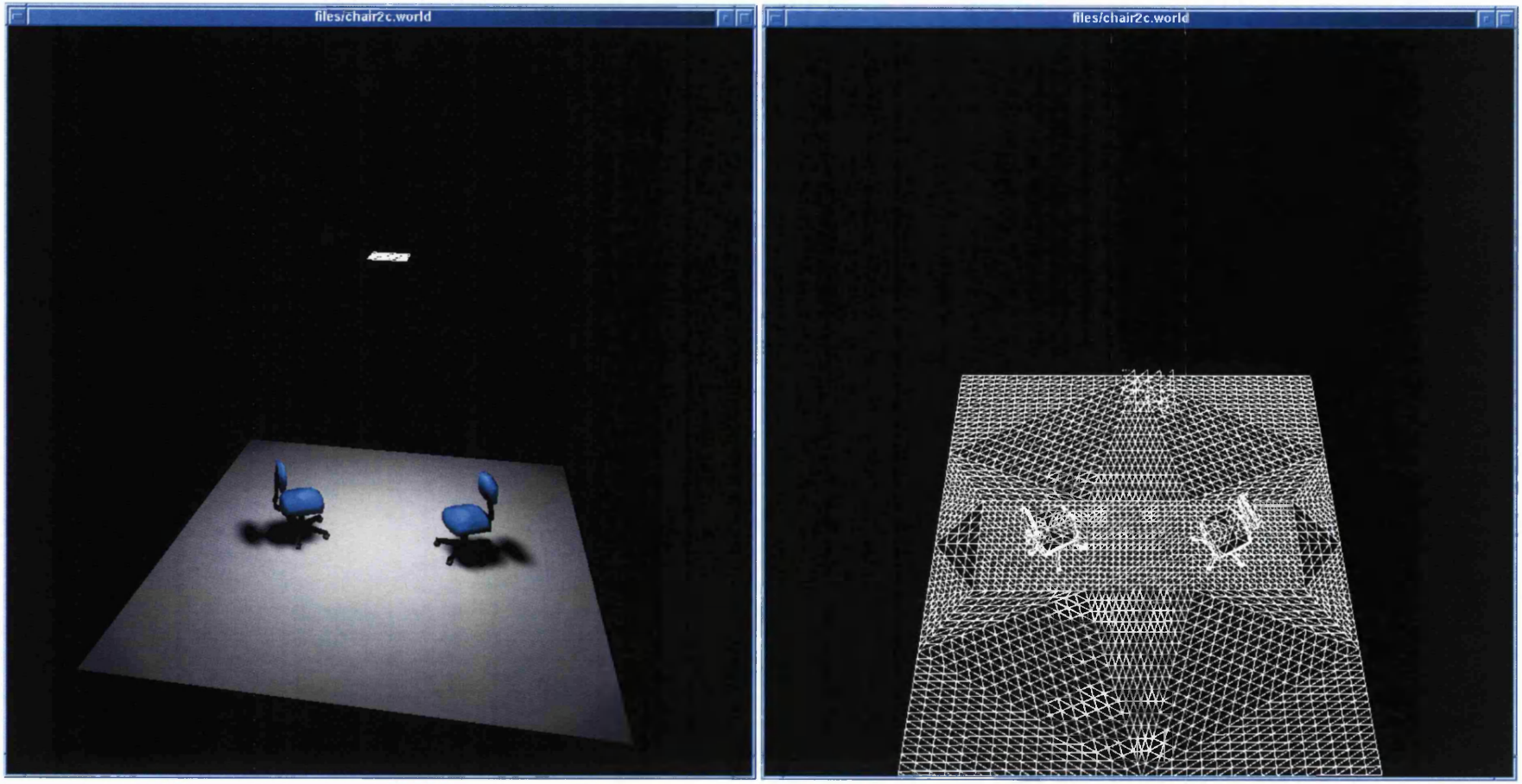


Figure 4.14 Final rendered image (left) and its wireframe mesh view (right).

4.5 Recursive Scene Localisation

Consider the scene in Figure 4.15. If the current scene localisation technique is applied to this scene, then the following clusters shown in Figure 4.16 will be generated.

From Figure 4.16 it can be seen that two clusters were successfully localised from the scene in Figure 4.15. Although only two clusters were generated, it is still more efficient to solve the radiosity system with localisation, rather than without.

A far more optimal clustering solution for the scene in Figure 4.15 can be obtained by reanalysing the clusters that have just been built. From Figure 4.16, it can be seen that it is possible to further localise the surfaces within the larger cluster, to obtain a set of sub-clusters. Clusters can be recursively re-localised because once a cluster has been deemed suitable for localisation, the cluster and its contents effectively becomes a self-contained 'world'. Thus the main light transport will be between the surfaces contained within the cluster.

An important caveat must be noted. Every time a group of surfaces are clustered and localised from the main scene or from its parent cluster, a small amount of energy is lost from the whole scene. This is because when a cluster is localised, any secondary energy that would otherwise be distributed back into the environment, is lost. Thus a limit on the number of times a cluster can be recursively localised must be imposed. Scenes that have been over zealously localised tend to be darker than normal, with clusters that had received little primary light energy becoming very dark.

Typically, a cluster will only need to be re-localised once. It is usually clusters that contain objects that have been stacked on top of other objects, which will be candidates for re-localisation.

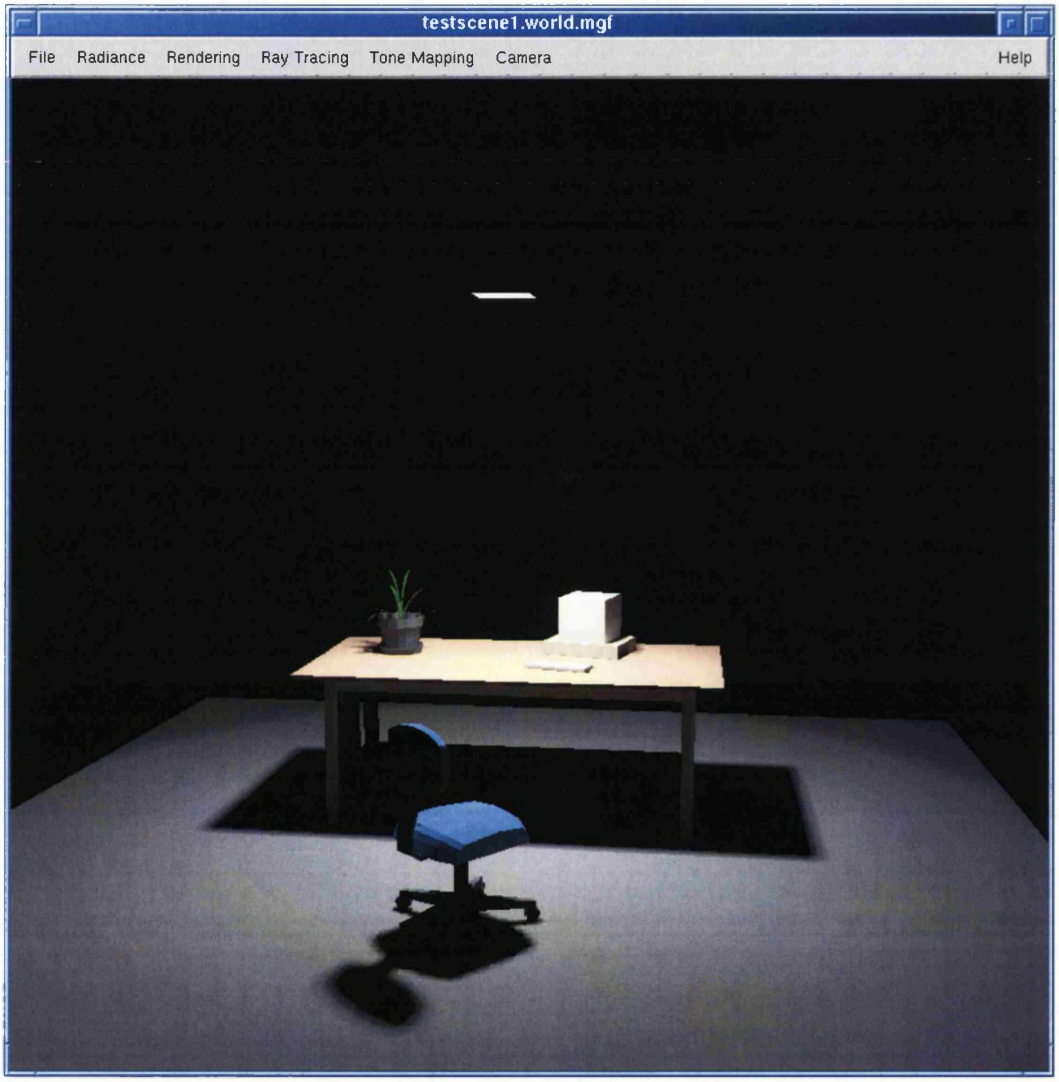


Figure 4.15 A medium complexity scene containing 1678 initial surfaces.

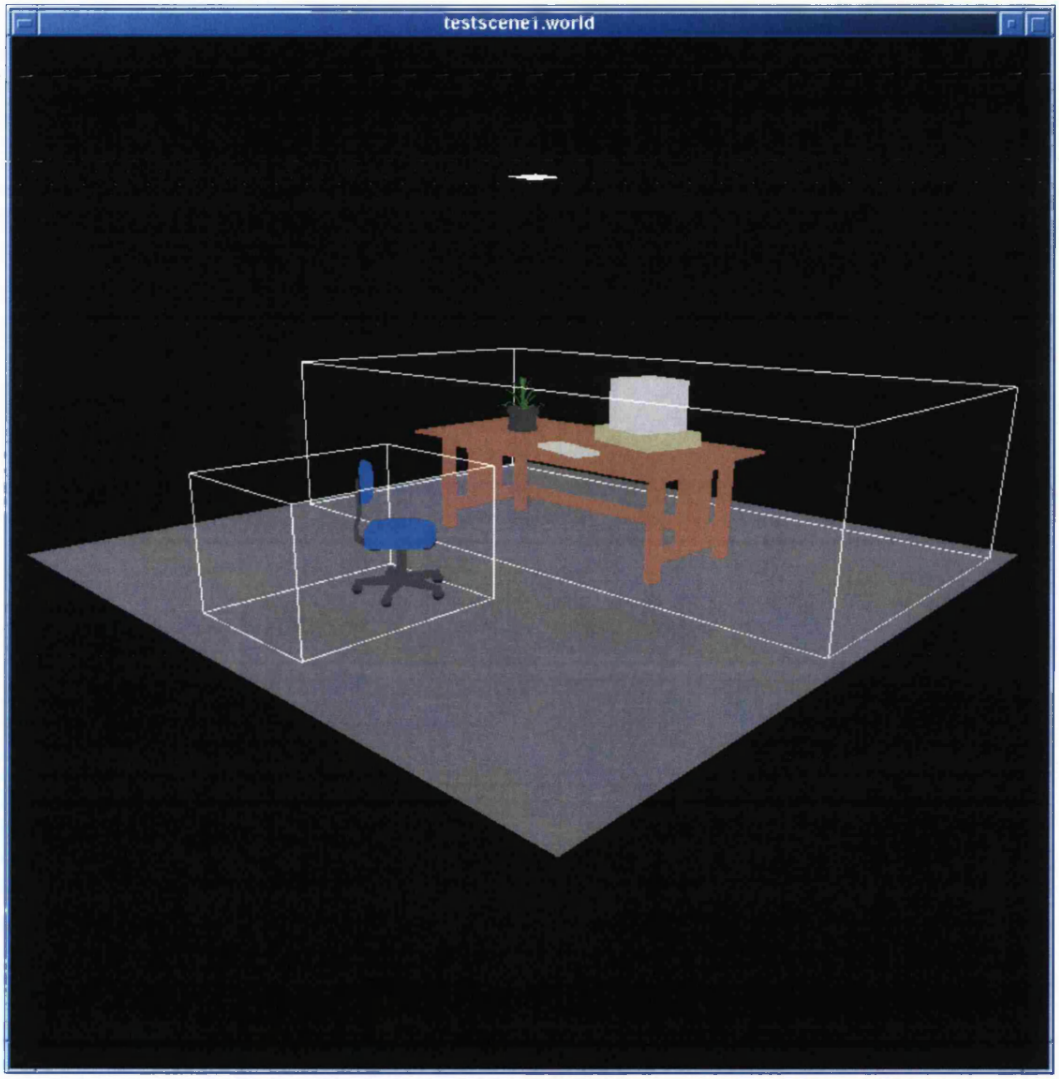


Figure 4.16 The results of cluster location for the scene in Figure 4.15.

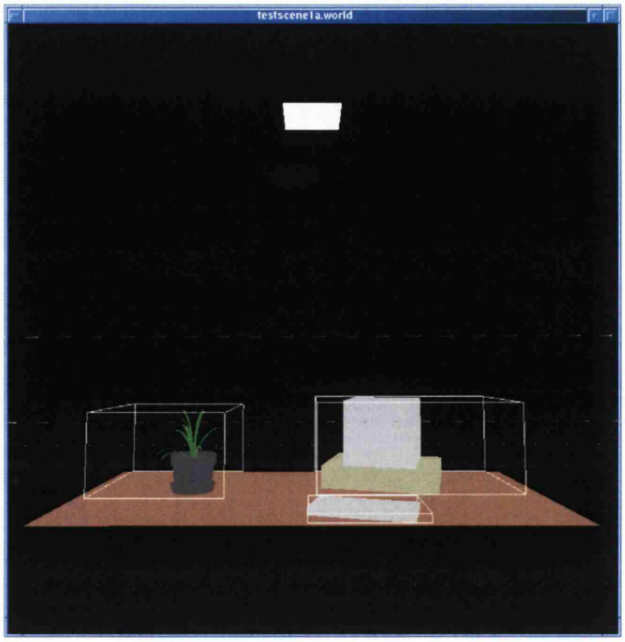


Figure 4.17 The generated clusters for the objects on the table top in Figure 4.15.

Figure 4.17 shows the resulting sub-clusters that were obtained after the localisation technique was reapplied to the surfaces contained within the larger cluster (shown in Figure 4.16). Figure 4.18 shows a diagram of the hierarchy of clusters generated by recursive localisation.

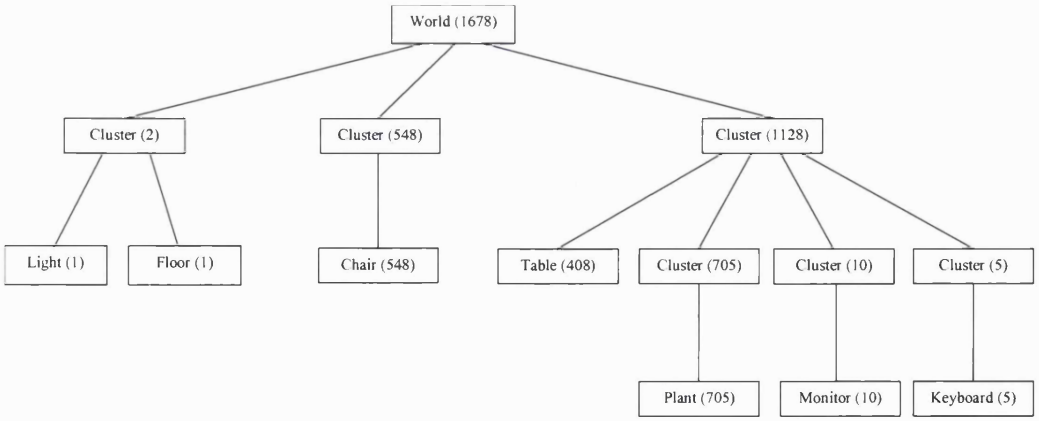


Figure 4.18 A diagram showing the hierarchy of clusters generated for the scene in Figure 4.15 (number of surfaces in parentheses).

The advantages in applying recursive localisation can be seen in the following table.

	Time taken to solve clusters (s)						
	Light + Floor	Chair	Table	Plant	Monitor	Keyboard	Total Time
Localisation	32.88	15.59	164.29				212.76
Recursive Localisation	32.14	15.47	8.49	82.19	1.8	0.23	140.32

Table 4.1 A time comparison between localisation and recursive localisation.

It can clearly be seen that applying recursive localisation improves the total time taken to compute a complete radiosity solution. Thus, for scenes of increasing complexity, the savings will be cumulative.

4.6 Summary

In this chapter, we have introduced a new method for reducing the complexity of a scene for hierarchical radiosity. This method aims to reduce the $O(k^2 + n)$ running time complexity of hierarchical radiosity to $O(n)$ by reducing the initial data input size. It is also a new approach to clustering, where clusters of surfaces are detached from a scene and rendered independently.

The most popular clustering strategies are discussed here, and a simple bounding box clustering strategy is presented. Although there are more efficient algorithms available, this simple clustering strategy is easy to construct and implement, and provides reliable operation during scene localisation.

An important part of the localisation procedure presented in this chapter is the detachment of clusters, and involves casting discontinuity lines to compute the shadow boundary of a cluster. As described in the previous chapter, we restrict our attention to extremal discontinuity meshing.

It was also shown that the localisation method could accumulate significant savings in time spent in the linking phase of hierarchical radiosity. Since the number of initial surfaces within each localised clusters is a smaller portion of the total surfaces within a

scene, the $O(k^2)$ linking time is kept to a minimum. As a result, the number of links created is correspondingly reduced. Thus, the localisation achieves savings in both time and space complexity.

However as the localised clusters are rendered independently, when these clusters are recombined, it is possible for artefacts to appear along the boundaries of the clusters. This is due to differing levels of refinement between adjacent clusters causing T-vertices to appear at the cluster boundaries. T-vertices have been well documented by Baum *et al.* [37] and they describe a method for removing them.

In general scenes, it is normal for objects to rest upon other objects to create more complex objects. Unfortunately these objects are usually spatially close enough to each other, that clustering algorithms only form a single cluster. Since the localisation algorithm works most efficiently when a scene is localised into many clusters that contain relatively few surfaces, we have enhanced the localisation algorithm to recursively localise clusters, to search for sub-clusters.

Having developed this new approach for clustering scenes in hierarchical radiosity, next we undertake a comprehensive series of tests to show the effectiveness and scalability of this method. We report this work in the next chapter.

Chapter 5 Results

5.1 Introduction

To determine the performance of the localisation algorithm described in the previous chapter, this chapter tests the performance and scalability of the algorithm for scenes of increasing complexity. The timing results[1] from our localisation algorithm were compared with timings gained from Bekaert *et al.* [91] 'RenderPark' test-bed system for global illumination.

To enable impartial comparisons between the localisation method and RenderPark, both systems utilise object space acceleration techniques such as uniform grid [27] or octree [16] space subdivision, hierarchical patch refinement and lazy linking [56].

RenderPark implements a Galerkin radiosity solver. Galerkin radiosity [14][51] utilises higher order basis functions to approximate the radiosity across the finite mesh elements generated by the (hierarchical) refinement process. Hence better approximations of the radiosity function across a surface can be potentially obtained with fewer mesh elements.

However, our localisation algorithm implements a traditional hierarchical radiosity solver. Traditional hierarchical radiosity [39] uses constant or Haar basis functions to represent the function across the finite mesh elements. As a result, this potentially means that many more smaller mesh elements would be required to accurately represent the radiosity function across a surface.

Fortunately the Galerkin radiosity module in RenderPark supports a selection of basis functions, including a constant basis function. Thus, to ensure fair testing between RenderPark and our localisation method, both systems will use constant basis functions.

---

[1] All timings were performed on a Pentium III 700Mhz PC with 768MB of RAM, running Slackware Linux 8.0.

Finally, the main aim of the tests between RenderPark and our localisation method is to determine the performance and scalability between our algorithm and existing clustering strategies.

RenderPark implements both isotropic and anisotropic clustering strategies. However, isotropic clustering strategies are well known and have been widely used by a variety of authors, for example, Sillion [57][63], Gibson [73][84], Hazenfratz *et al.* [87], Muller [88]. Thus, the isotropic clustering strategy will be used in all RenderPark tests.

5.2 The Scalability of Simple Scenes

To test the scalability of the new clustering strategy described in Chapter 4 for simple scenes, eight scenes with increasing complexity (shown in Figure 5.1) were tested with this algorithm and RenderPark. The complexity of a scene was altered by increasing the number of chairs in the scene. Each chair was placed at a random location and at a random orientation.

The number of initial surfaces in the scenes shown in Figure 5.1, range from 550 to 4386 surfaces and thus represent scenes of low to medium complexity. The motive behind the set-up and creation of these test scenes shown in Figure 5.1, is to assess how well our localisation technique fairs against a well known clustering strategy.

RenderPark offers a very robust implementation of isotropic clustering, hierarchical refinement and Galerkin radiosity. Thus consistent results will be available for comparison.

The progression of the scenes in Figure 5.1 was designed to test how well the localisation method and RenderPark cope with scenes of increasing complexity. To make this a realistic test, a 'real' object was added to the scene, in preference to adding random surfaces. Thus an office chair was chosen as the object to add to the scenes.

The office chair object is not a trivial object. It contains 548 surfaces of varying sizes and orientations. This object is sufficiently complex that neither the localisation method nor RenderPark will have an unfair advantage.

Table 5.1 shows the timing results gained from the localisation method for clustering, linking and total times for computing a hierarchical radiosity solution. The final rendered scenes are shown in Figure 5.1.

Scene in Figure 5.1	Number of initial surfaces	Clustering time	Linking time	Total Computation Time
(a)	550	0.01	0.46	189.71
(b)	1098	0.03	0.85	205.23
(c)	1646	0.054	1.23	212.724
(d)	2194	0.094	1.67	246.164
(e)	2742	0.105	2.22	268.025
(f)	3290	0.177	2.34	270.567
(g)	3838	0.2	2.89	277.05
(h)	4386	0.337	3.61	317.087

Table 5.1 Timings for the localisation algorithm.

From Table 5.1, it can be seen that the time complexity for clustering is unfortunately quadratic, as can be seen from the graph in Figure 5.2. From the timing results in Table 5.1, clustering takes up a very small proportion of the total time taken to compute a radiosity solution. Thus clustering will only incur a small time penalty on the localisation algorithm, even as the size of the input scene increases. As a result, it is normally worthwhile to cluster a scene. Even if clustering is not used in radiosity computation, it is still very useful for object space acceleration.

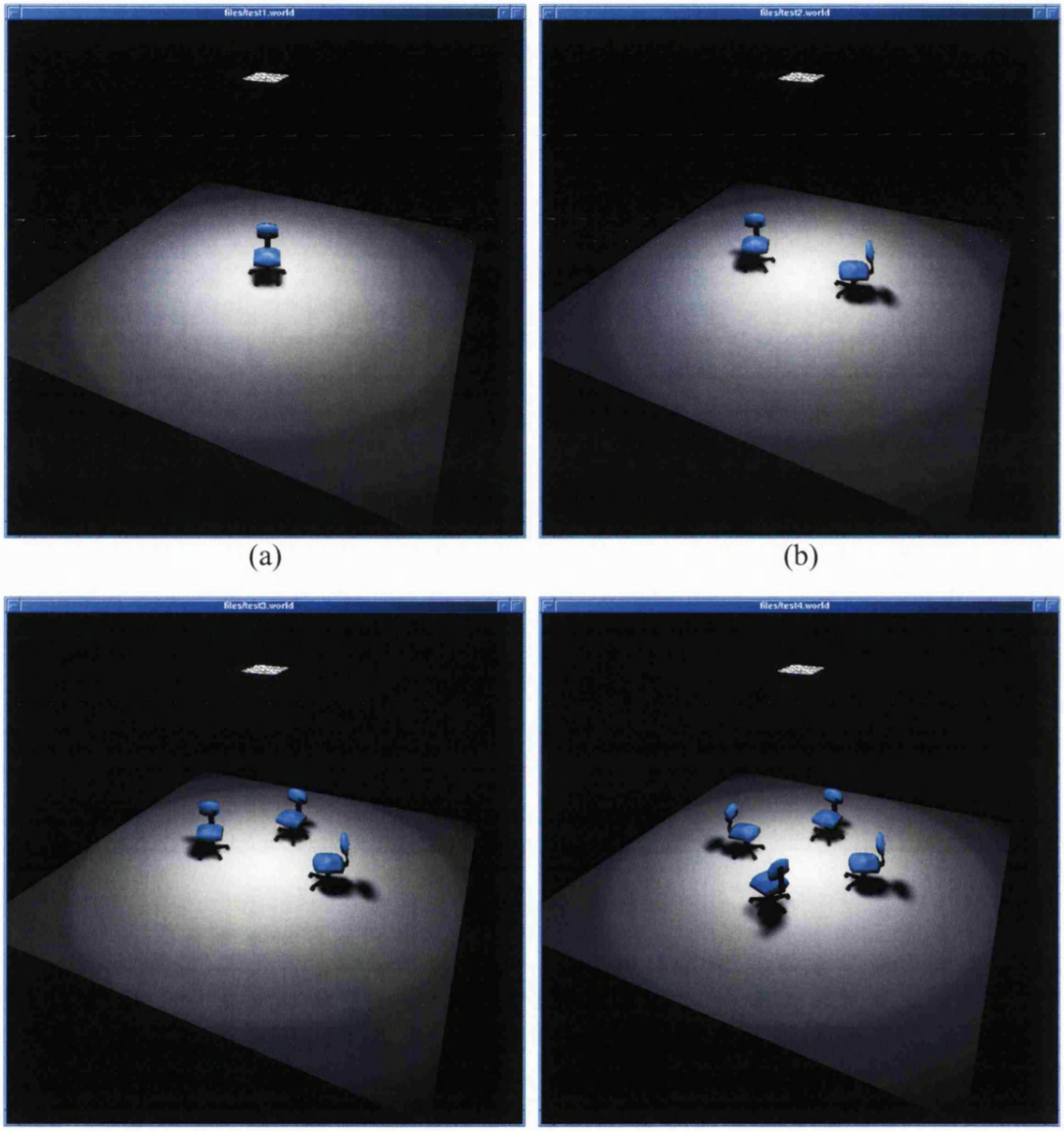
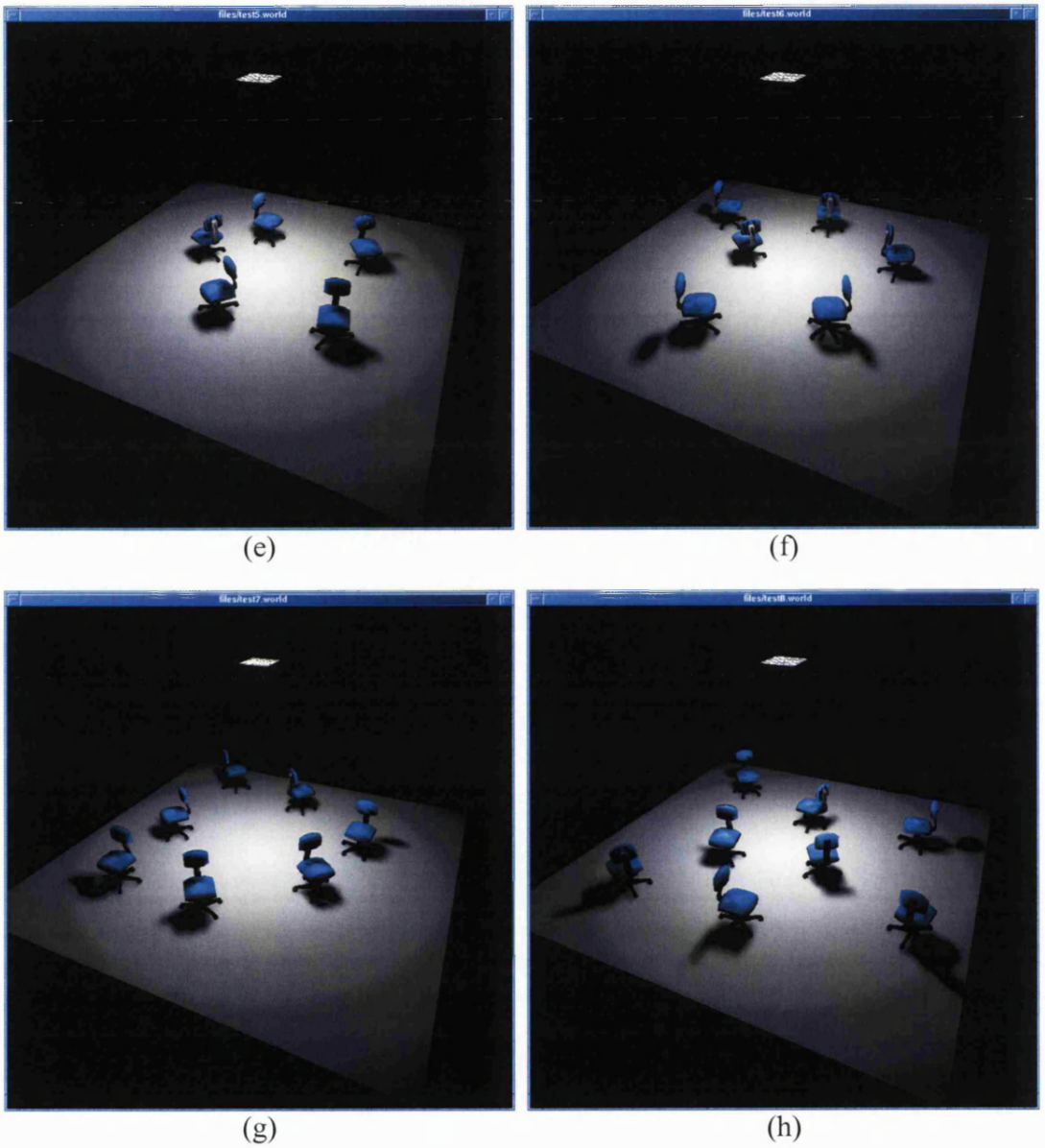


Figure 5.1 Solutions from the localisation method.



(e) (f)

(g) (h)

Figure 5.1 ***cont.***

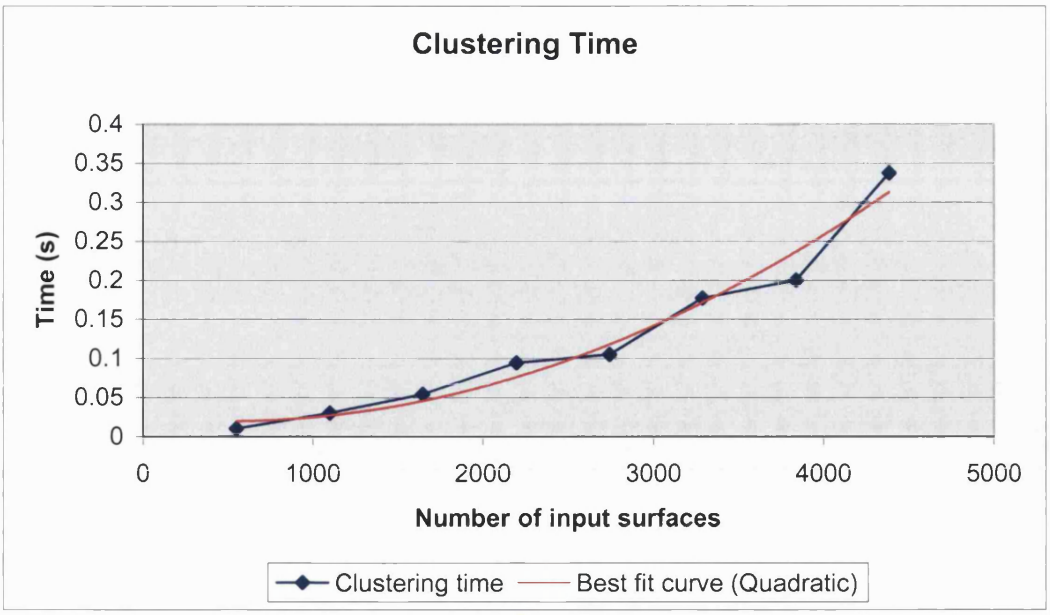


Figure 5.2 A graph showing the time taken to cluster the scenes in Figure 5.1.

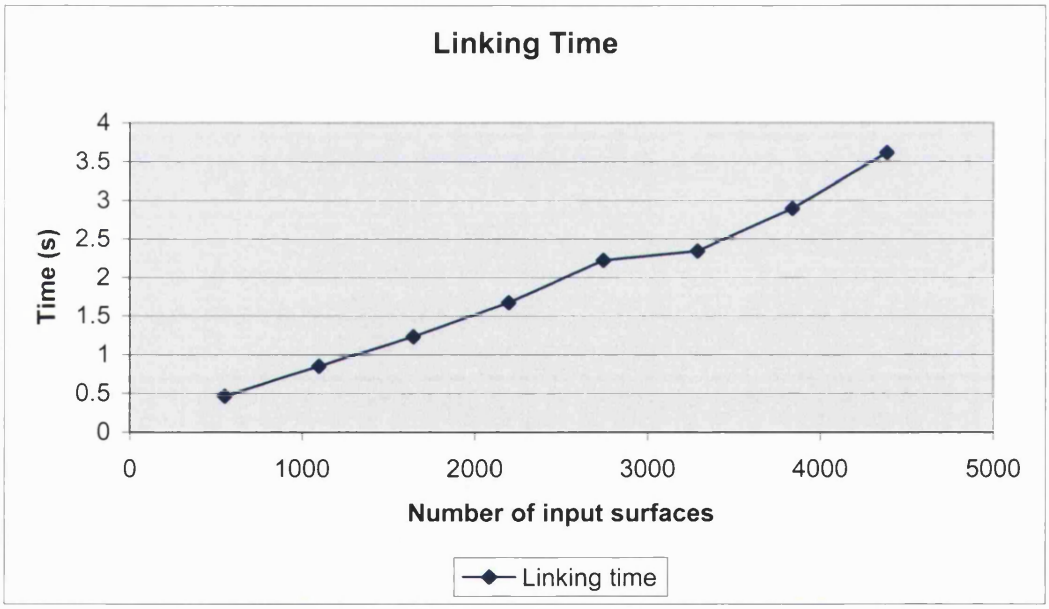


Figure 5.3 A graph showing linking times for the scenes in Figure 5.1.

Figure 5.3 shows the timing results for the linking phase for each scene in Figure 5.1. This graph shows that as the complexity of the scene increases, the linking time, which is the most expensive part of hierarchical radiosity computation, now increases linearly instead of the normal quadratic time complexity.

Figure 5.3 shows an important result. The linking phase is an integral part of hierarchical radiosity. This phase essentially records (for every patch in the scene) the light transported throughout the entire scene. Modelling every light interaction takes $O(n^2)$ time to compute and accounts for the k^2 term in the $O(k^2 + n)$ total running for hierarchical radiosity.

The results shown in Figure 5.3 show that as the complexity of a scene increases, the overall linking time for the localisation method increases linearly. It must be noted that the linking time *within* the localised clusters still takes quadratic time to compute. It is the overall linking time for the *entire* scene, with respect to increasing scene complexity, which now increases linearly. Thus, the reduction due to scene localisation, in the number of initial surfaces the hierarchical radiosity system needs to consider, is sufficient that the original quadratic running time complexity has been effectively reduced to a linear running time complexity.

Finally, Figure 5.4 shows a graph of the total rendering times for each scene in Figure 5.1. The results show that as the complexity of the scene increases, the total rendering time is linear. A comparison of the running time complexities between this algorithm and a variety of radiosity algorithms is shown in Table 5.2.

Method	Running time complexity
Localisation	$O(nm^2 + n)$ but in practice, achieves $O(n)$
Progressive Refinement Radiosity	$O(n^2)$
Hierarchical Radiosity	$O(k^2 + n)$
Clustered Hierarchical Radiosity	$O(k\log k + n)$
Face Clustering (Willmott [90])	$O(n\log n)$ but in practice, achieves $O(n)$

Table 5.2 Time complexity comparison between a variety of radiosity algorithms.

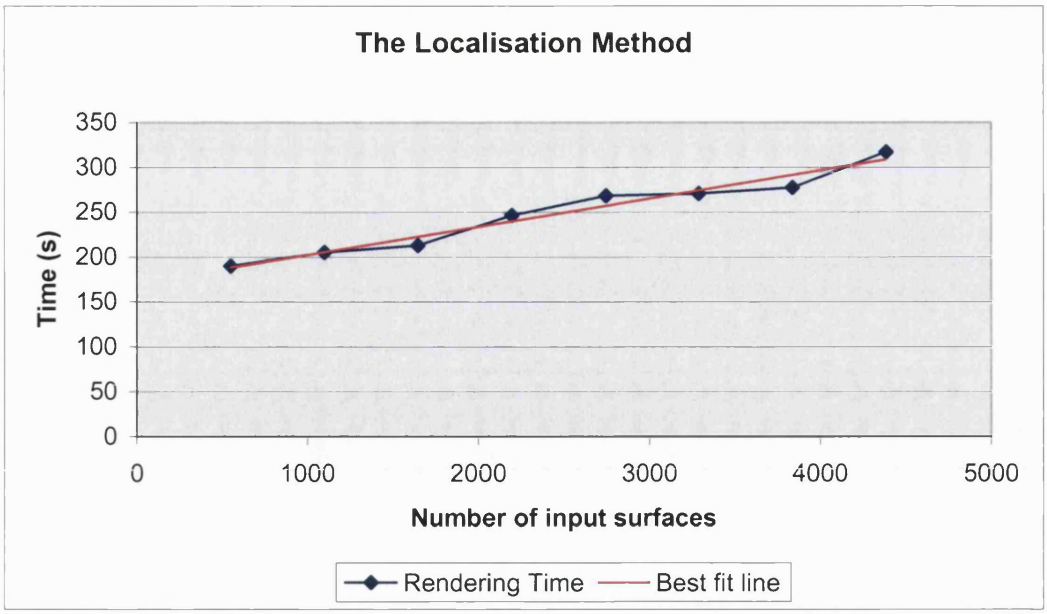


Figure 5.4 A graph showing the total rendering times for the scenes in Figure 5.1

The graph shown in Figure 5.4 demonstrates the performance and scalability benefits gained by the localisation method. The best fit line shown in Figure 5.4 shows that the running time complexity of this algorithm scales linearly with the number of input surfaces. Therefore, localisation should be applied to complex scenes.

The variations in the timings about the best-fit line are mainly due to random positions and orientations of the extra chairs within the scene. The positioning and orientations of the objects within the scene affect the number of visibility calculations that have to be made. More importantly, the number of triangles generated after constrained Delaunay triangulation is quite sensitive to the layout of the objects within a scene and can be seen in Figure 5.5. This illustration shows that as the cluster approaches the boundary of a surface, the constrained Delaunay triangulation algorithm will generate a greater number of small triangles.

It is quite possible for the triangulation stage of any radiosity method to contain many tiny triangles, generated by the constrained Delaunay triangulation algorithm. In some cases this is beneficial since it can help in reduce the time spent on hierarchical refinement. However, it can be possible for the constrained Delaunay triangulation algorithm to generate vast numbers of tiny triangles.

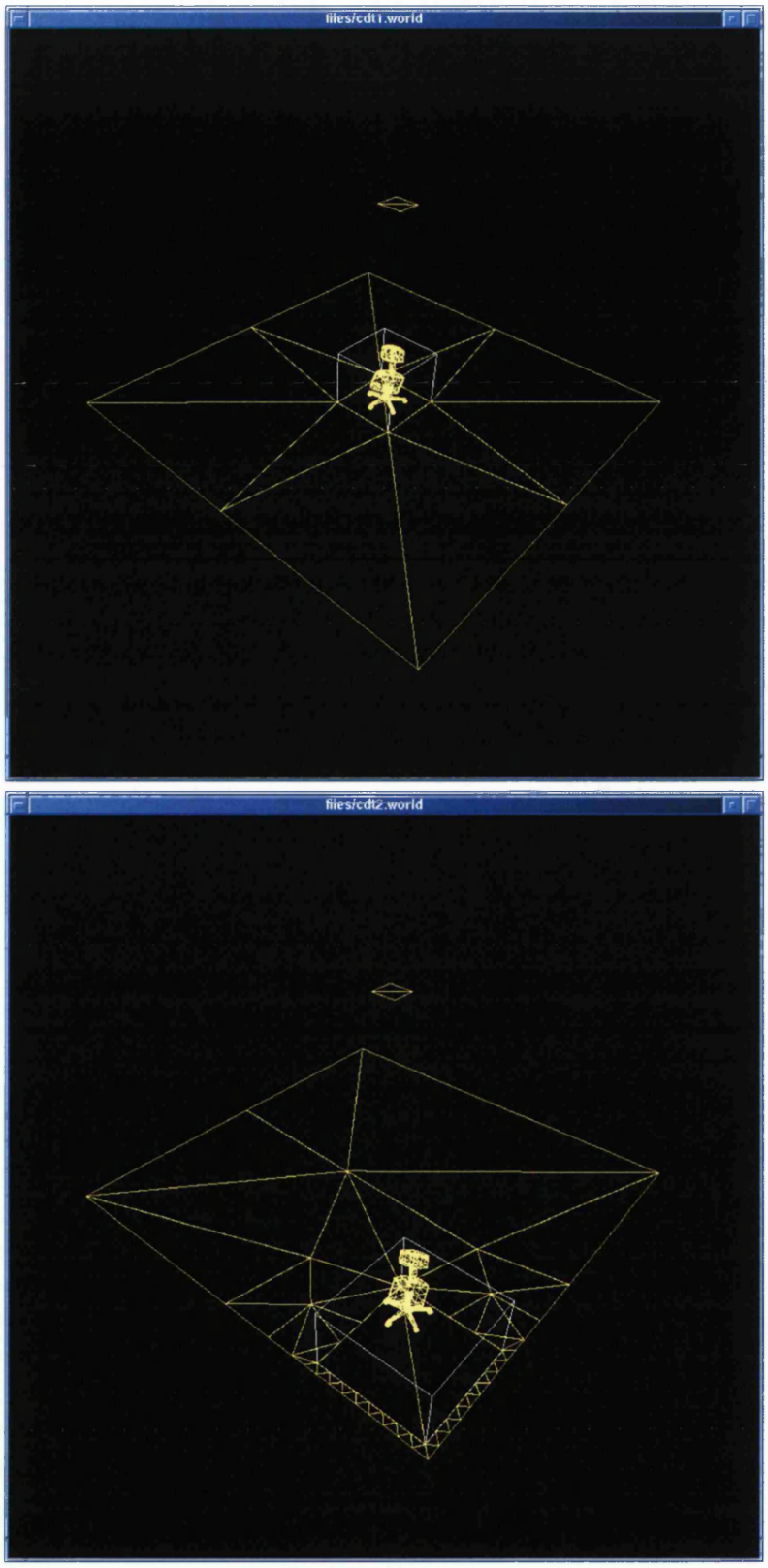


Figure 5.5 A diagram showing how the placement of an object can affect the number of triangles generated by constrained Delaunay triangulation.

Although these tiny triangles will never be subdivided by hierarchical refinement, each triangle will be considered as an initial surface. As the time and space complexity of the linking stage in hierarchical radiosity is $O(k^2)$, each of the unnecessarily small triangles will add considerably to the amount of work that needs to be done by the hierarchical radiosity solver.

To stop the size of the triangles from becoming too small, the parameters controlling the limits of area size and angle of the triangles generated, should be altered in the constrained Delaunay triangulation.

In fact, over refined meshes actually degrades the performance of hierarchical radiosity. Since hierarchical radiosity has its own refinement algorithm, it is far better to let the hierarchical radiosity perform its own refinement. Thus, in general, the size of initial surfaces supplied to hierarchical radiosity should be as large as possible.

Interestingly, if the best-fit line from the graph in Figure 5.4 is extrapolated back such that it intersects the time axis, a noticeable offset can be observed. This is actually caused by the light source and the floor surfaces.

As it turns out, the linking and hierarchical refinement between the light source and floor surface dominates the total computation time. Figure 5.6 shows a graph of the ratios between the time taken to compute a radiosity solution between the light source and the floor surface, and the total computation time for solving the entire scene.

For the scenes in Figure 5.1 that contain very few clusters, the total computation time is entirely dominated by the light source and floor surfaces. This is due the large amount of time that has to be spent hierarchically refining a relatively large floor surface.

However as the number of clusters increases, the total area of the floor surface will reduce significantly, as sections of the floor surface are cut away due to localisation. Hence, this will result in a reduction in the amount of time spent on refining the floor surface.

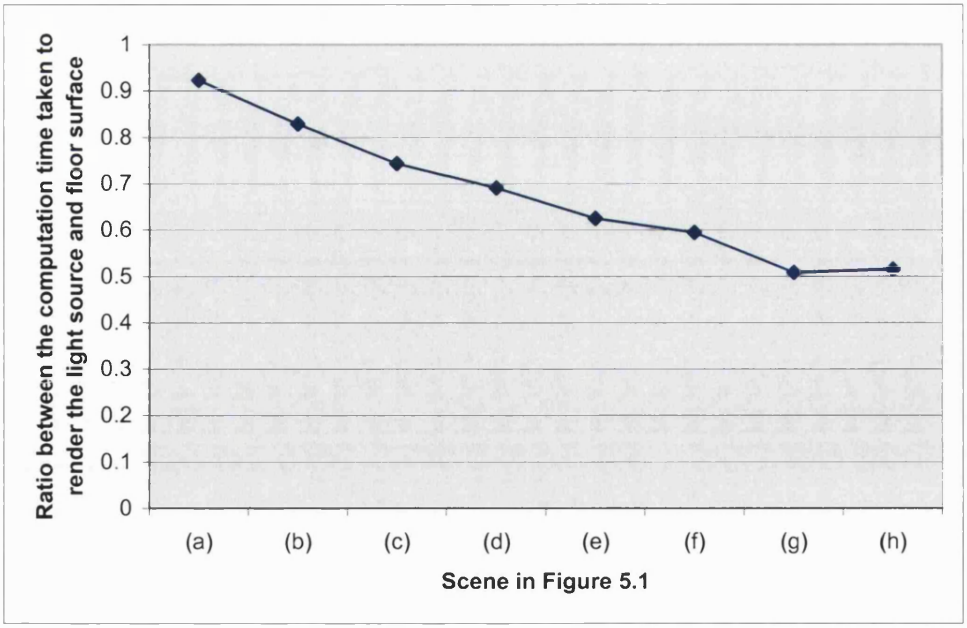


Figure 5.6 The ratio between the computation time taken to render the light source and floor surface, and the total computation time.

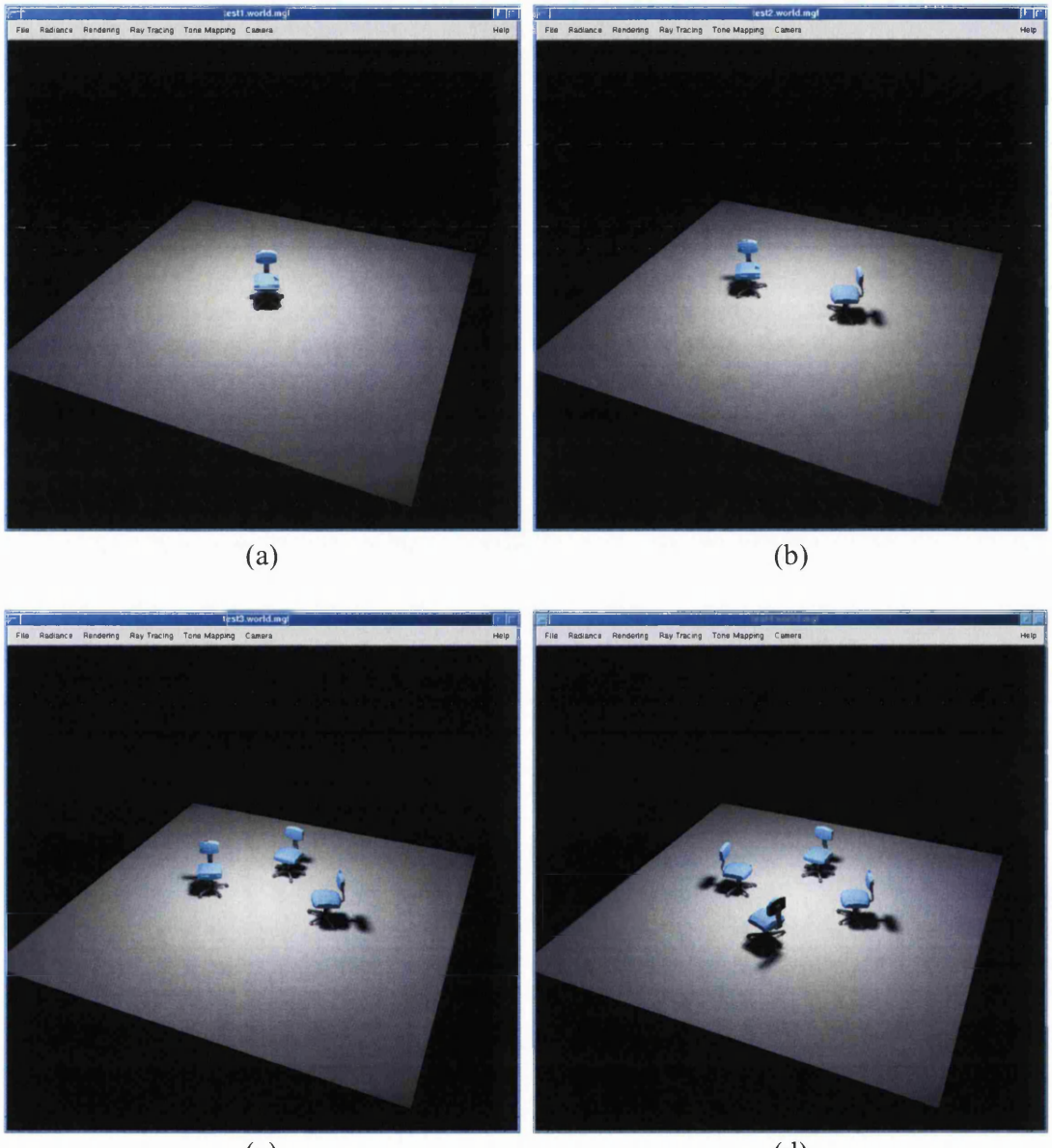
As a result, it can be seen from the graph in Figure 5.6 that the ratio of the computation time spent on the light source and floor surface, and the whole scene, reduces significantly from over 90% for the scene in Figure 5.1(a) to approximately 50% for the scene in Figure 5.1(h).

Thus for scenes with low complexity, the cost of applying the whole localisation procedure, which includes, surface clustering, computing and casting extremal discontinuity lines, cluster detachment, is greater than what it would cost to solve these types of scenes the traditional hierarchical radiosity technique.

For example, the following table (Table 5.3) shows the times taken to render the simple scene in Figure 5.1(a) with RenderPark, hierarchical radiosity and the localisation method.

	RenderPark	Hierarchical Radiosity	Localisation Method
Time (s)	88.01	169.68	189.71

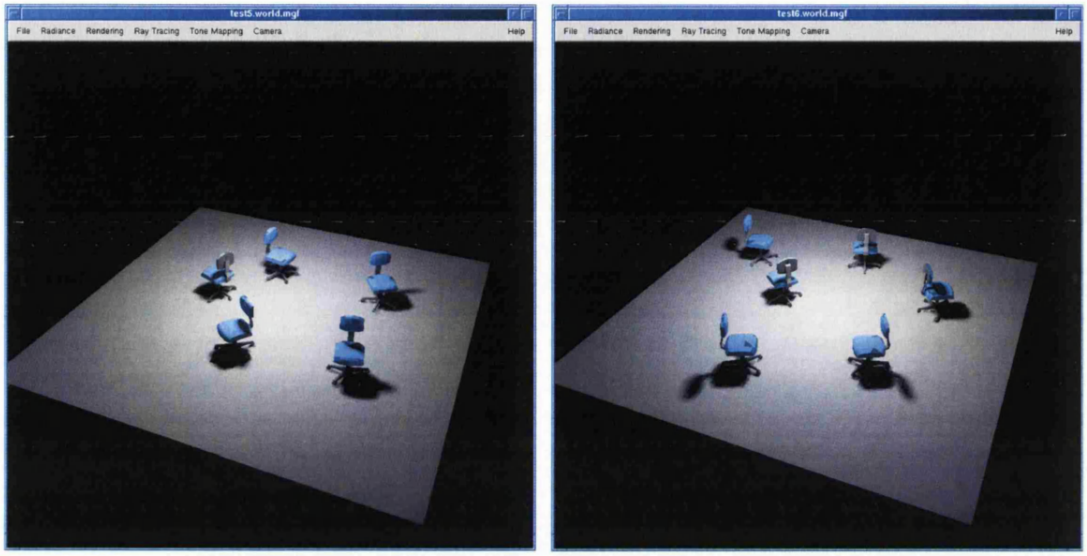
Table 5.3 Rendering time comparisons for Figure 5.1(a).



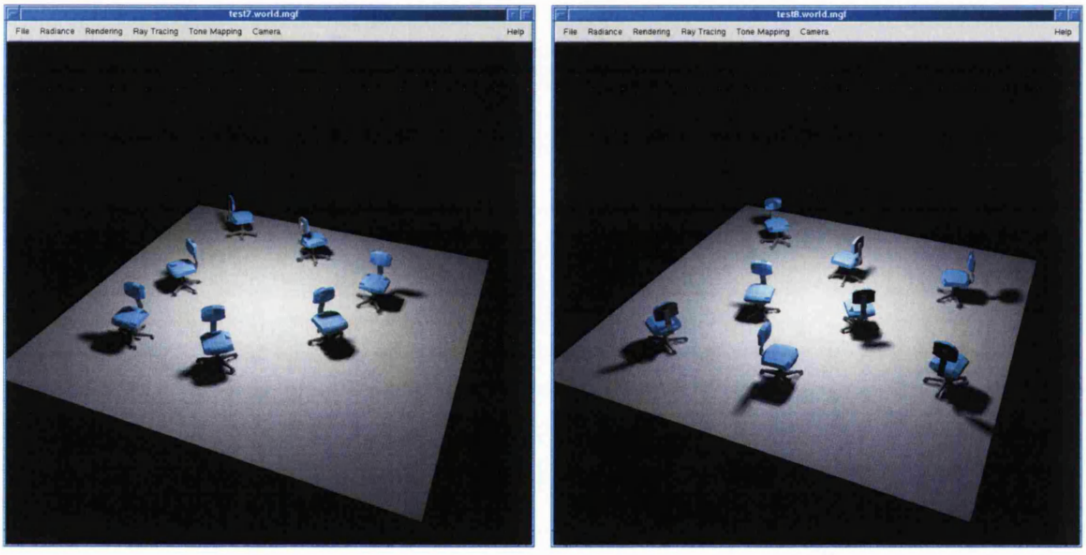
(a) (b)

(c) (d)

Figure 5.7 Solutions from RenderPark for the scenes in Figure 5.1.



(e) (f)



(g) (h)

Figure 5.7 ***cont.***

To compare the results gathered by the localisation algorithm with RenderPark, the data files for the test scenes shown in Figure 5.1 were converted to the *materials and geometry format* [76]. To ensure consistency between the results generated by RenderPark and our localisation method, the dimensions of all geometry were maintained.

Figure 5.7 shows the equivalent scenes in Figure 5.1 that have been rendered by RenderPark. Two main differences can be noted between the two sets of rendered images. Firstly, the colour of the chairs is slightly different. This was due to the conversion between the RGB colours used in the localisation algorithm to the CIE-XYZ colour space used in RenderPark. Secondly, the lighting intensities of the rendered images in Figure 5.7 are slightly different from the images in Figure 5.1. This was due to the tone mapping function present in RenderPark that was absent from our localisation method. These are purely visual differences. The energy transported within the scenes is identical in both systems.

Figure 5.8 shows a graph of the times taken to render the scenes in Figure 5.7. For these scenes, RenderPark also shows linear time complexity in rendering time.

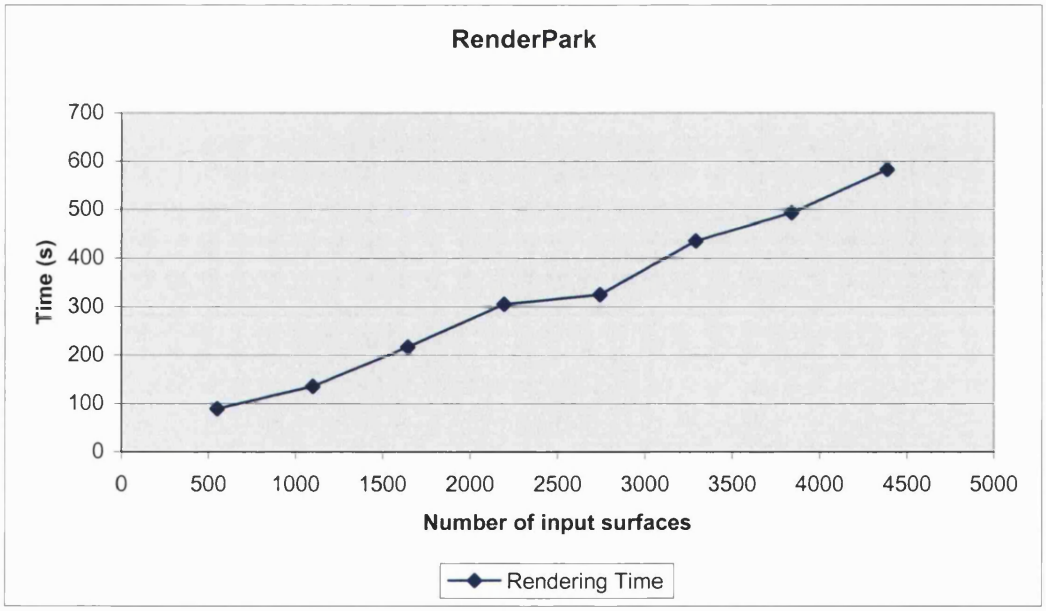


Figure 5.8 A graph showing the rendering times for the scenes in Figure 5.7.

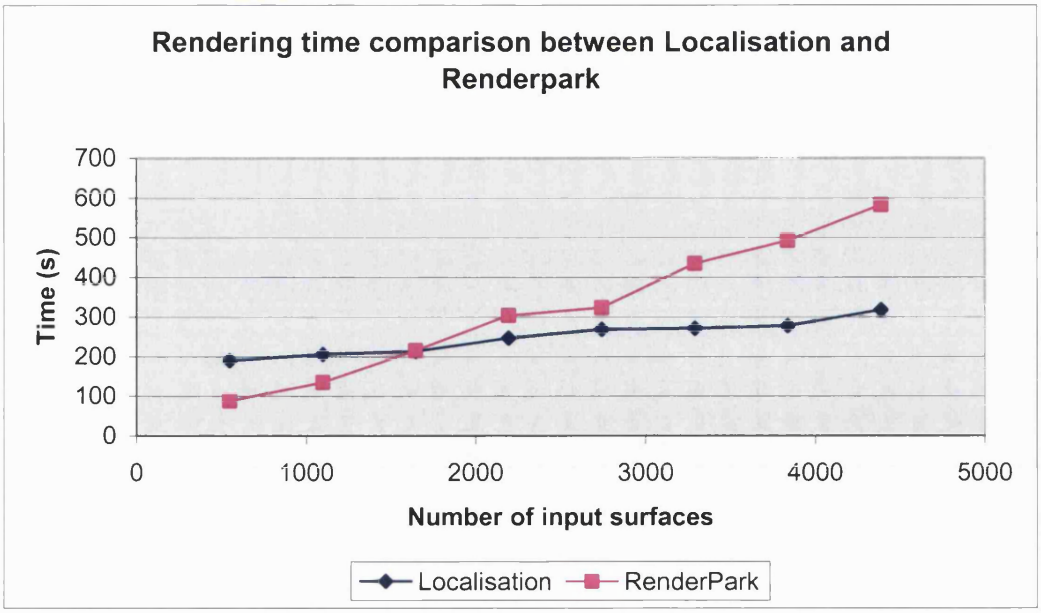


Figure 5.9 A comparison between the localisation method and RenderPark.

Figure 5.9 shows a comparison of the rendering times between our localisation method and RenderPark. The most notable difference between the two systems is the offset of the localisation method.

As described above, scenes of minimal complexity may cause the localisation method to perform relatively badly, in comparison to RenderPark or any system that uses comparable algorithms to RenderPark. For the first two test scenes in Figure 5.1 (and Figure 5.7), the localisation algorithm takes almost twice as long to render these images. Hence, localisation would not be used for such scenes.

However, for the latter test scenes shown in Figure 5.1 (and Figure 5.7), the localisation method out performs RenderPark. It can be seen from the graphs in Figure 5.4 (and also Figure 5.8) that adding more chairs to the scene adds only a small increase to the total computation time.

Both RenderPark and the localisation method exhibit linear running time complexities, hence it can be superficially concluded that both algorithms have equal scalability. However, upon closer inspection of the results shown in Figure 5.9, comparing the graph of the localisation method with the graph of RenderPark, shows that the

localisation method has an almost constant running time complexity in comparison. This is in fact only an illusion created by the vertical scale of the graph, but it does emphasise the differences in performance of the two systems, even though they share the same running time complexity.

5.3 The Scalability of Complex Scenes

The series of test scenes presented in this section, aims to test the performance and scalability of the localisation and recursive localisation algorithms for scenes that contain a greater range of complexity, than the scenes presented in the previous section (Section 5.2). The test scenes presented in Figure 5.10 and Figure 5.11 also aim to represent more realistic scenes that would be found in the 'real' world.

The number of initial surfaces for these test scenes (shown in Figure 5.10 and Figure 5.11) has been increased from 1678 to 26833 surfaces and thus represents scenes of low to high complexity. Also the complexity of the input geometry has been increased to allow for more complex objects to be included in the scene. Thus objects may now have other objects resting upon them. Hence the performance of the recursive localisation algorithm can be tested.

Figure 5.10(a) corresponds to a *base* scene, from which more complex scenes will be created. The test scenes shown in Figure 5.10 were extended from the base scene by adding and repositioning Figure 5.10(a), hence scenes of low to medium complexity were generated.

For the test scenes in Figure 5.11, the test scene shown in Figure 5.10(d) was used as the *base* scene, and so scenes of medium to high complexity were generated. The test scenes in shown Figure 5.11 were produced in the same way as Figure 5.10.

All test results gained from the localisation and recursive localisation algorithms are compared against timings obtained from RenderPark. To ensure consistency, exactly the same parameters, as used in the previous section (Section 5.2), will be applied in this section.

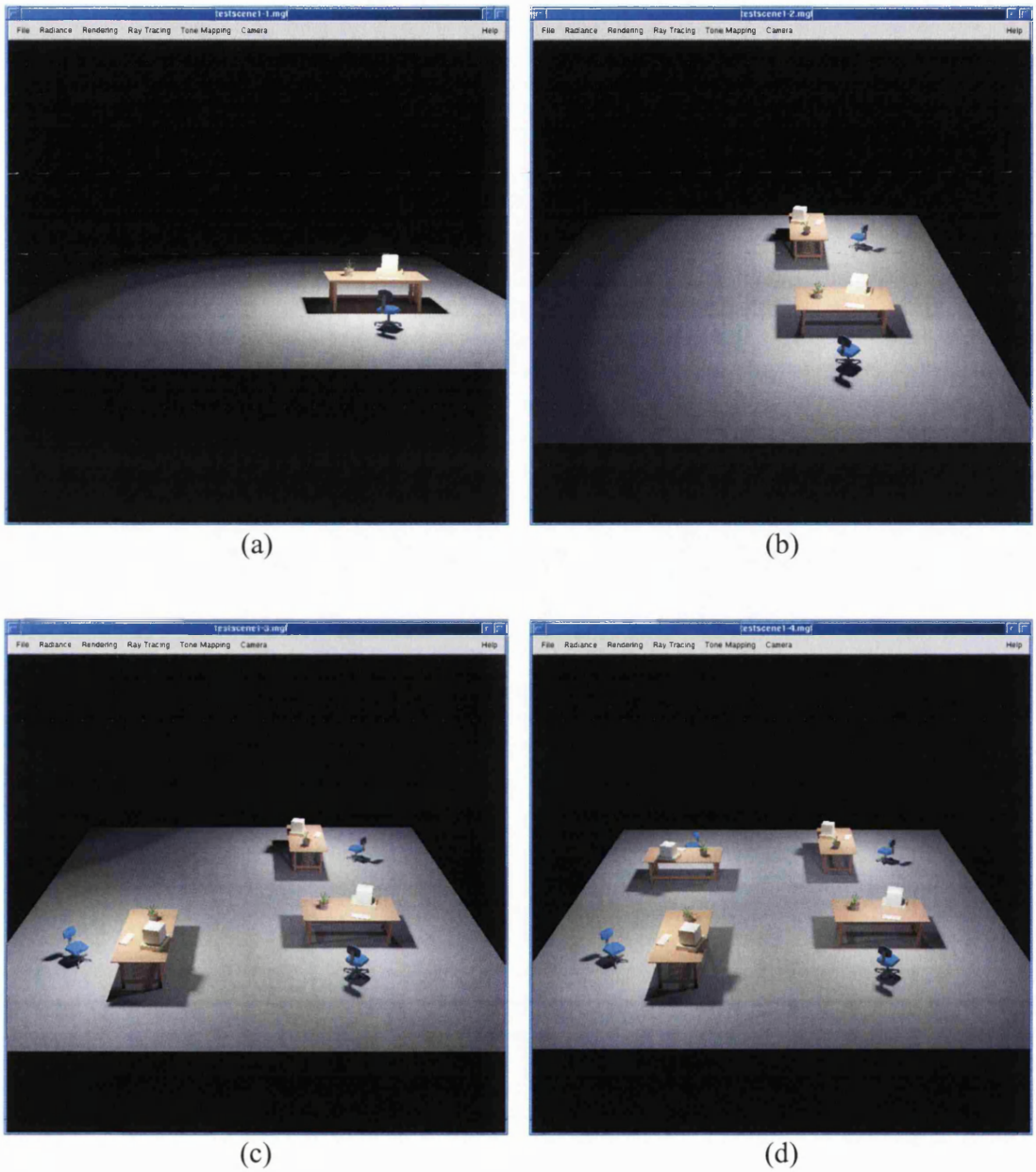
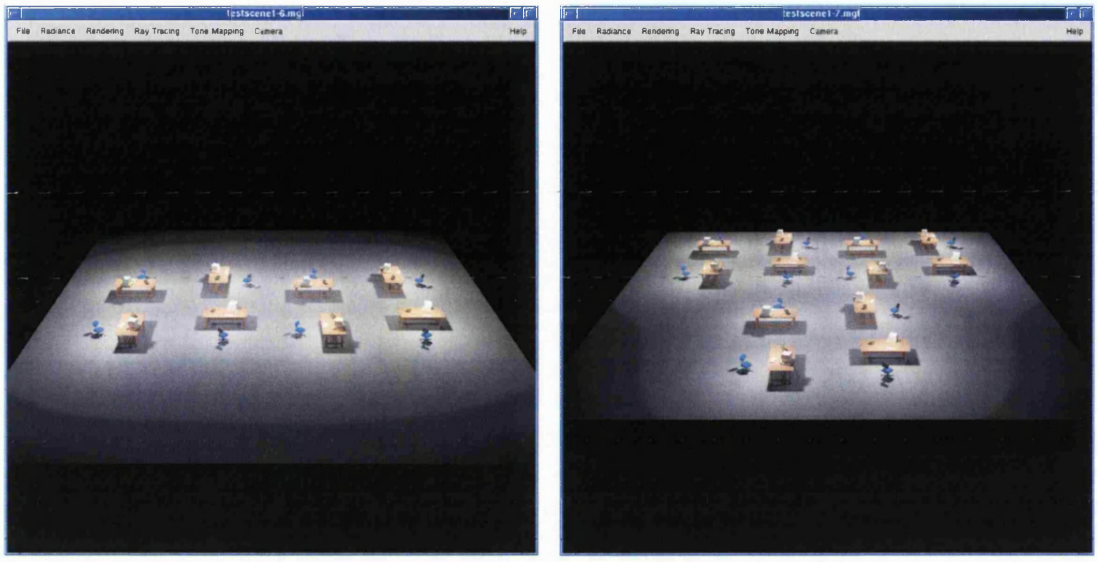
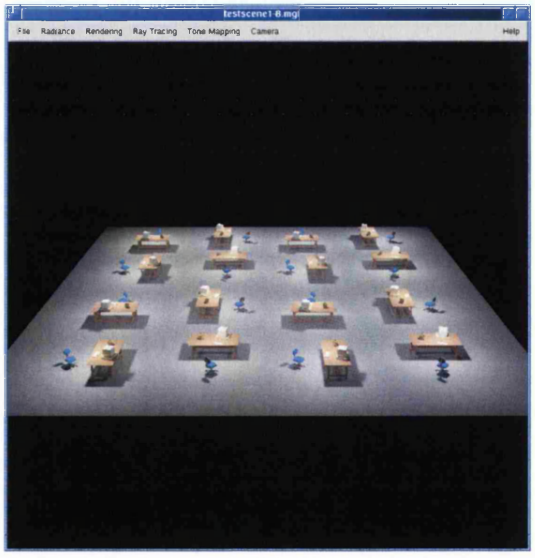


Figure 5.10 Low to medium complexity test scenes that contain (a) 1678, (b) 3355, (c) 6789 and (d) 5032 surfaces, respectively.



(a) (b)



(c)

Figure 5.11 High complexity test scenes that contain (a) 13417, (b) 20125 and (c) 26833 surfaces, respectively.

The results of the localisation and recursive localisation algorithms for the scenes shown in Figure 5.10 and Figure 5.11, are shown in the following table.

Number of surfaces	Localisation algorithm: rendering time (s)	Recursive localisation algorithm: rendering time (s)
1678	557.55	764.01
3355	1403.95	1416.68
5032	1897.95	1911.58
6709	2443.97	2167.94
13417	3678.76	2237.45
20125	5674.45	3852.95
26833	7022.81	4549.94

Table 5.4 Timings for the localisation and recursive localisation algorithms, for the scenes in Figure 5.10 and Figure 5.11.

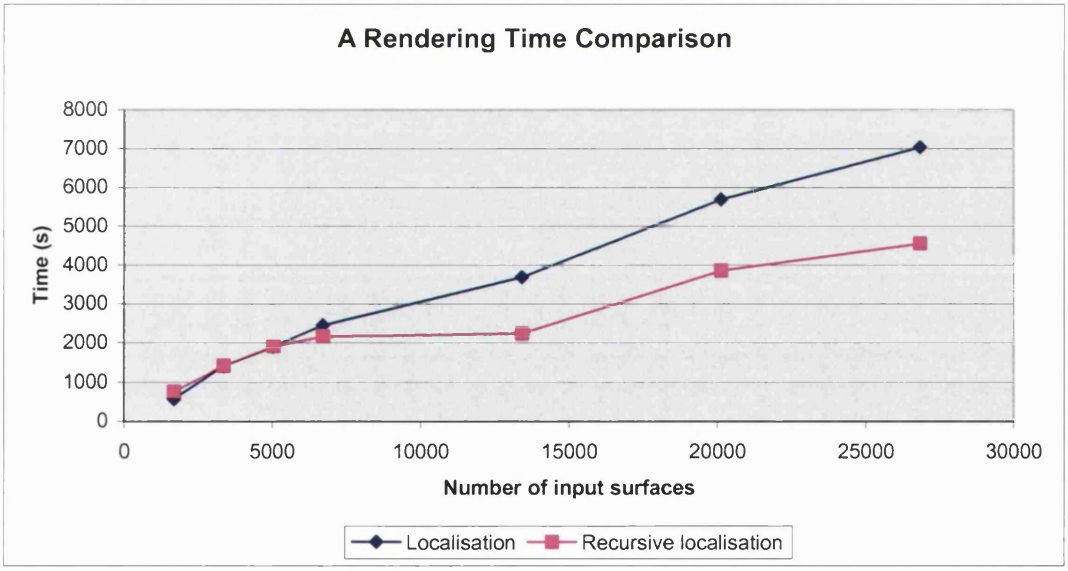


Figure 5.12 A comparison between the rendering times generated from the localisation and recursive localisation algorithms.

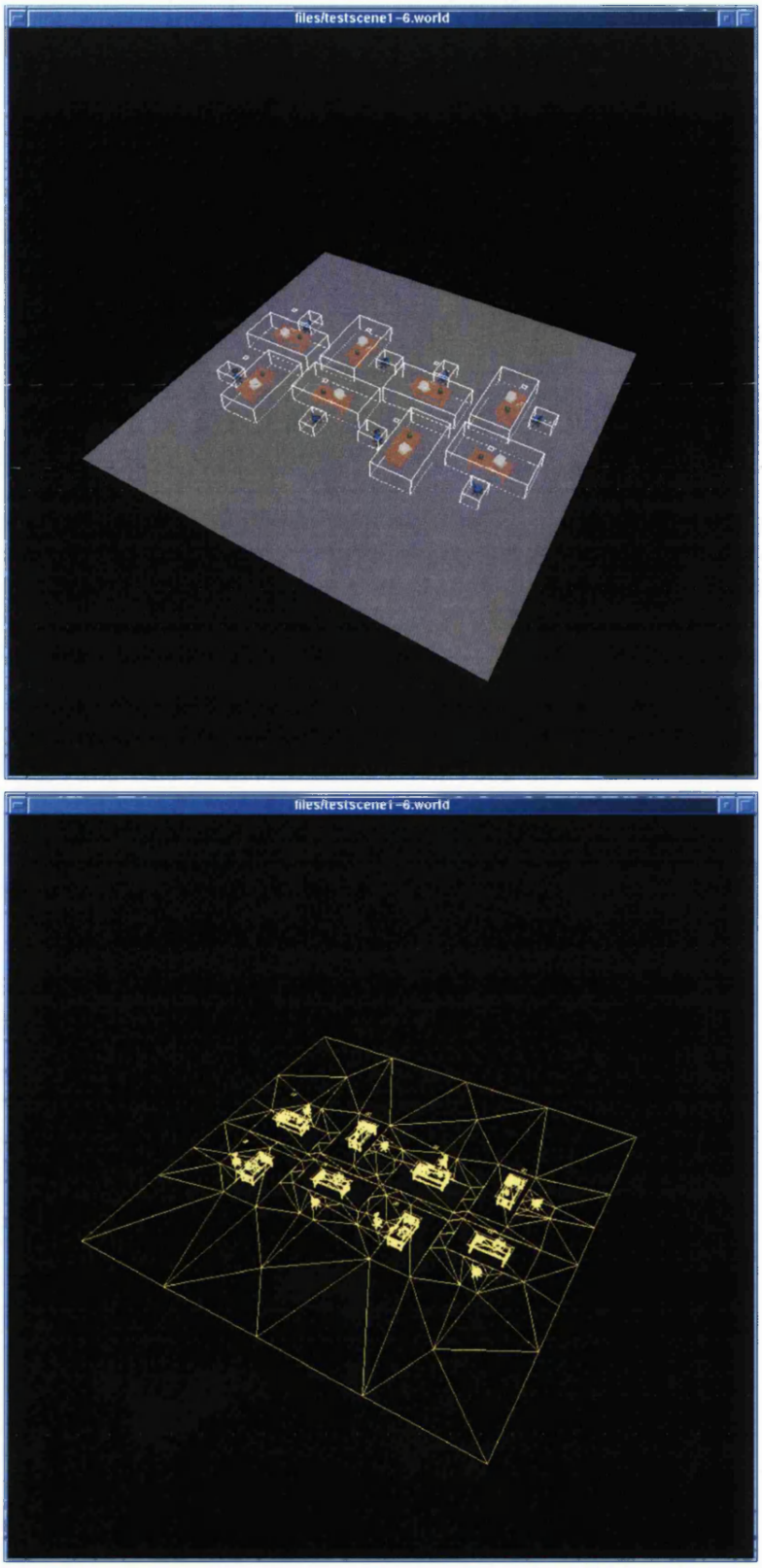
Figure 5.12 shows a graph displaying the results shown in Table 5.4. It can be seen from this graph, that the benefits of recursive localisation become apparent for the more complex scenes shown in Figure 5.11. Both the localisation and recursive localisation algorithms work most efficiently for scenes that can be clustered into many individual objects.

For the scenes in Figure 5.10, there is little difference in rendering times between the two algorithms. As described in the previous section (Section 5.2), this is mainly due to the total rendering time being dominated by the computation of a radiosity solution between the light sources and the floor surface.

As a result, only a minimal increase in performance will be gained by performing recursive localisation on scenes of low to medium complexity. In fact, the graph shown in Figure 5.12 shows that the cost of performing the recursive localisation procedure actually increases the total computation time over the normal localisation method, for the scenes in Figure 5.10.

However, as the number of clusters increases, an increasing amount of the floor surface will be cut away by the localisation algorithm. This will result in a corresponding reduction of the total area of the floor surface and thus a reduction in the amount of time taken to hierarchically refine this surface.

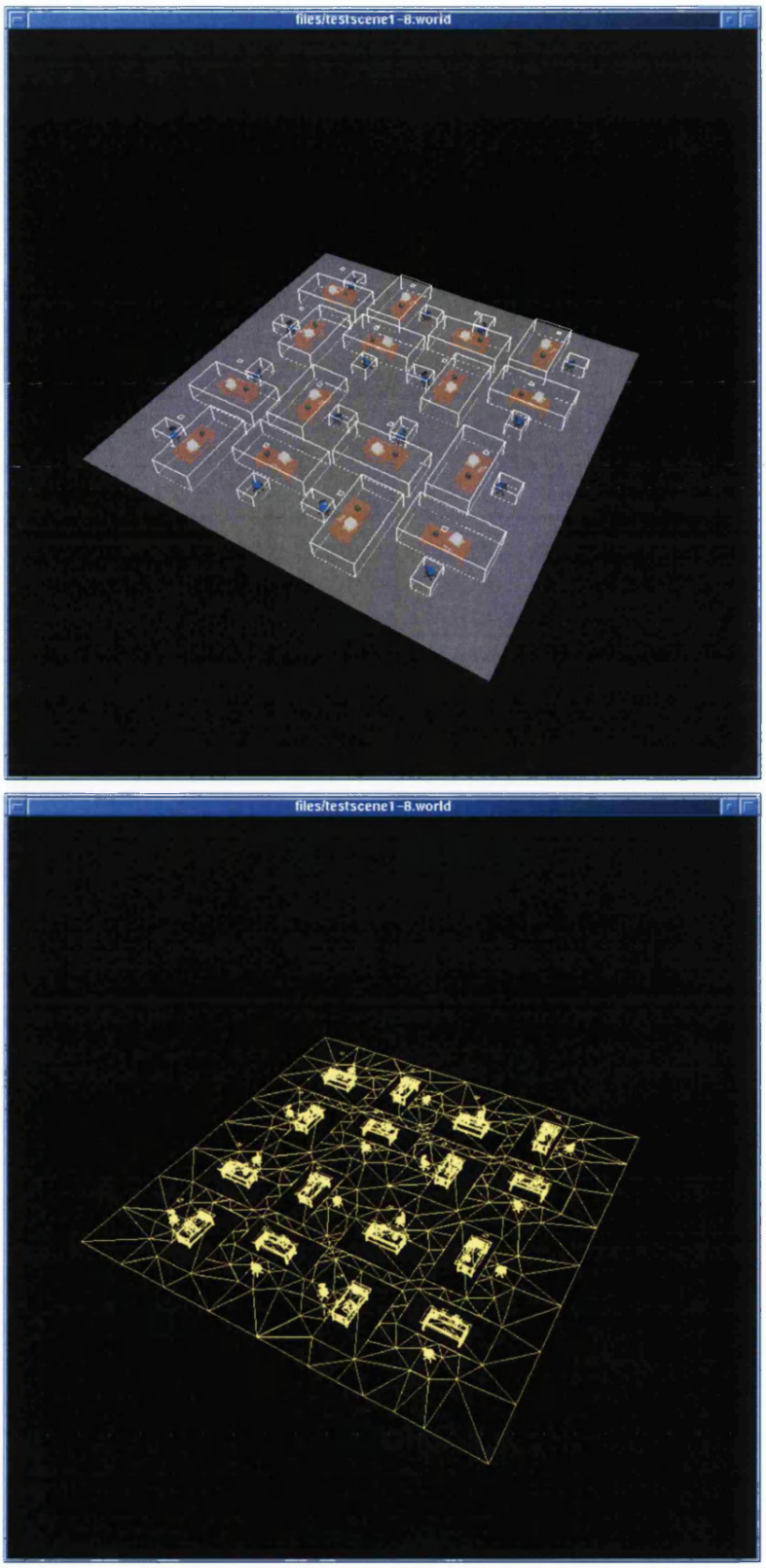
In general, the greater the number of clusters generated by a scene (see Figure 5.13(a) and Figure 5.14(a)), the greater the number of holes on the floor surface after localisation. This in turn will reduce the total area of the floor surface, but will increase the number of triangles generated by a constrained Delaunay triangulation (see Figure 5.13(b) and Figure 5.14(b)). However, the triangles generated by a constrained Delaunay triangulation will be small in comparison to the size of the original surface, so the time taken to hierarchically refine these triangles will be correspondingly much less.



(a)

(b)

Figure 5.13 The results showing the (a) clusters and the (b) CDT mesh generated for the test scene shown in Figure 5.11(a).



(a)

(b)

Figure 5.14 The results showing the (a) clusters and the (b) CDT mesh generated for the test scene shown in Figure 5.11(c).

For the more complex scenes in Figure 5.11, the benefits of increased clustering and hence increased localisation, becomes more apparent. As a result, the performance of recursive localisation is much more noticeable than with normal localisation. This is because these scenes contain sufficient complexity that there are many more sub-clusters generated. Again, see Figure 5.13 and Figure 5.14 for an illustration.

Since the time savings gained by recursive localisation are cumulative, the total time savings for large scenes become significant. This is particularly evident in the graph shown in Figure 5.12.

Comparing the timing results (see Table 5.4) for the most complex scene, Figure 5.11(c), the recursive localisation method is over 50% faster than the standard localisation method.

The following table (Table 5.5) show the timing results from RenderPark, for the scenes in Figure 5.10 and Figure 5.11.

Number of surfaces	Rendering time (s)
1678	253.66
3355	628.16
5032	988.73
6709	1412.75
13417	4554.23
20125	9104.47
26833	13465.1

Table 5.5 Timings from RenderPark, for the scenes in Figure 5.10 and Figure 5.11.

Finally, a comparison between the results from RenderPark, localisation and recursive localisation methods for the scenes in Figure 5.10 and Figure 5.11 can be found in Figure 5.15. The following table (Table 5.6), shows that actual timing results from the localisation, recursive localisation and RenderPark systems.

Number of surfaces	Localisation: Time (s)	Recursive localisation: Time (s)	RenderPark: Time(s)
1678	557.55	764.01	253.66
3355	1403.95	1416.68	628.16
5032	1897.95	1911.58	988.73
6709	2443.97	2167.94	1412.75
13417	3678.76	2237.45	4554.23
20125	5674.45	3852.95	9104.47
26833	7022.81	4549.94	13465.1

Table 5.6 A comparison of rendering times for the scenes in Figure 5.10 and Figure 5.11.

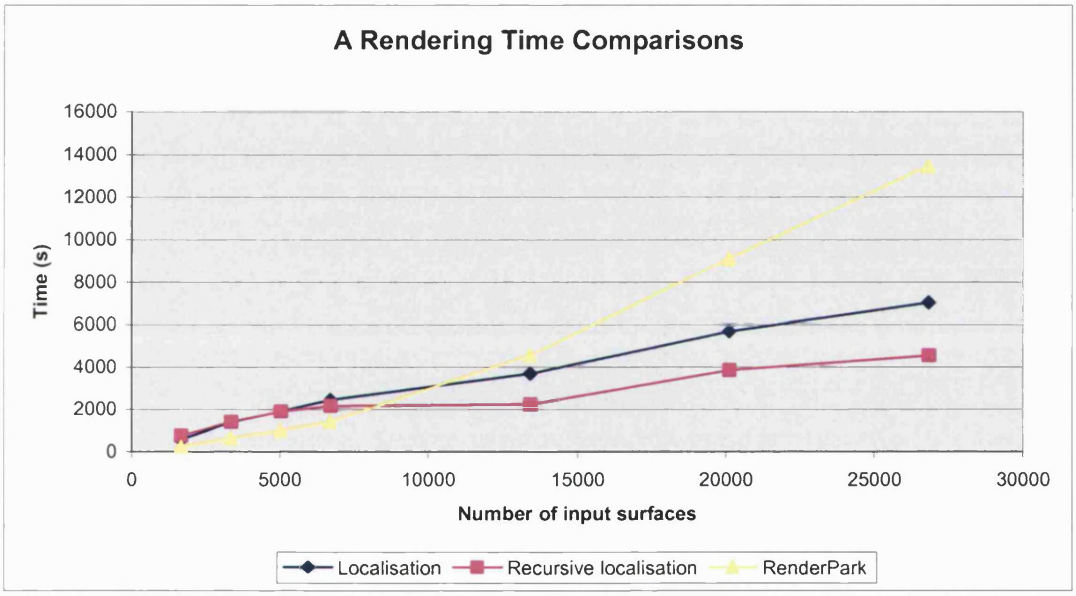


Figure 5.15 A comparison of rendering times for the localisation and recursive localisation methods, and RenderPark.

It can be seen that RenderPark performs well for the scenes in Figure 5.10, consistently outperforming both the localisation and recursive localisation methods. However, RenderPark does not perform as well as the localisation methods for the large scenes in Figure 5.11.

5.4 Summary

The results in this chapter show that the localisation and recursive localisation methods perform well for large complex scenes, where the surfaces in the scene cluster into many objects. Since the cost of computing a hierarchical radiosity solution is expensive, the only way to improve the efficiency of hierarchical radiosity is to reduce the input size, and hence the amount of work done.

The localisation methods presented in Chapter 4 show that this can be achieved, enabling large scenes containing more than 20000 surfaces, to be solved by traditional hierarchical radiosity in linear time. This is an important result for those who need to handle photo-realistic scenes in applications such as VR.

We have achieved the following:

- A new localisation algorithm for reducing the complexity of an input scene, for hierarchical radiosity.
- A recursive localisation algorithm for optimal cluster generation.
- Linear running time complexity for solving large scenes with hierarchical radiosity.
- Super-linear performance when compared to existing clustering algorithms.

The results shown in this chapter demonstrate that as the complexity of a scene grows, the efficiency of our localisation method improves and hence the more appropriate our methods become.

Chapter 6 Multi-resolution Modelling

6.1 Introduction

The vast majority of acceleration techniques applied to radiosity have focused upon efficiently solving the radiosity equation and form factor computation. For complex scenes, the meshes that are generated can be overwhelming. Research carried out by Hedley [83] and Gibson [84], have attempted to reduce the complexity of the generated meshes using perception-based metrics.

Hedley [83] addressed the problem of scalability. Discontinuity meshing does not scale well with increasing input size. Hedley [83] proposed four techniques for reducing the number of discontinuity surfaces that need to be processed:

- Only building discontinuity surfaces with those parts of occluders which are in silhouette with respect to the light source.
- Using a perception based metric to determine when it is not necessary to trace discontinuity surfaces from given occluders.
- A discontinuity line is only placed on receiving surfaces if the light source, which caused it, makes a perceptible difference to the radiosity leaving that surface.
- Considering only those discontinuity surfaces that form the extremal boundary of the shadow.

When combined, these four techniques significantly reduce the amount of processing that needs to be carried out and thus improves the scalability characteristics of discontinuity meshing.

Another way to address the problem of scalability is to analyse and reduce the complexity of the input scene before discontinuity meshing or radiosity computation is performed. Currently, this problem has only been addressed by Willmott [90] and by the work presented in this thesis.

Unfortunately the methods by Hedley [83] and Gibson [84] do not perform any optimisations on the initial scene geometry. Therefore, the size and complexity of the initial scene geometry will ultimately determine the overall performance of these methods. Hence, the true scalability of any radiosity algorithm is dependent upon how well it can organise and optimise the input scene geometry, before it is committed to the radiosity solver.

6.2 LOD: An Alternative Method for Reducing Scene Complexity

To date, the vast majority of radiosity systems model each object in a scene with a single representation. Such a model is called a *fixed resolution model.* The disadvantage of using a fixed resolution model is that, regardless of what distance an object maybe from the viewer, the representation of that object is always the same. However, the further away an object is from the viewer, the less detail the viewer is able to resolve. A much more efficient method of modelling objects is *multi-resolution modelling.* With multi-resolution modelling each object has a hierarchy of representations that vary from coarse representations to a high-detail models. Depending upon the distance away from the viewer, the correct *level of detail* (LOD) of the object is chosen.

Figure 6.1 shows an example of *levels of detail* for a model of a cat. The top row of cats shows the various levels of detail the cat model may represent in its hierarchy and the bottom row shows the distance at which an application might decide to draw each level of detail. Unfortunately there were no results included by Erikson [69] for the cat model in Figure 6.1, but it does demonstrate the effectiveness of LOD in multi-resolution modelling.

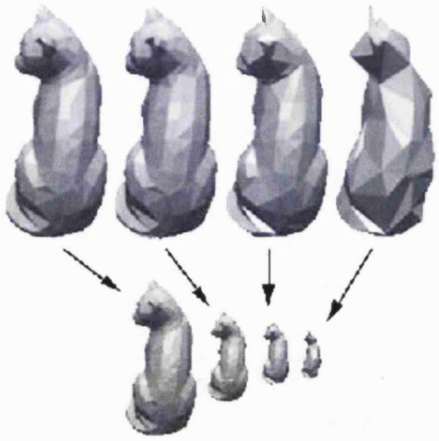


Figure 6.1 An example of Levels of Detail (from Erikson [69]).

Heckbert and Garland [55] state that multi-resolution modelling is useful in radiosity. Traditional radiosity algorithms require each polygon in a scene to be pre-meshed into many small elements, in order to capture the radiosity gradients across each polygon. As a result a large proportion of time is spent computing insignificant light transfers between distant elements.

Although two-level patch-element hierarchies [22] can reduce the complexity to $O(nm)$ and hierarchical radiosity [39] to $O(n)$, Heckbert and Garland [55] propose that if multi-resolution modelling is used with hierarchical radiosity, an algorithm with linear or *better* complexity can be achieved.

Unfortunately creating a multi-resolution model is difficult and such models are most often created by hand. Heckbert and Garland [55] discuss and evaluate six possible methods for generating multi-resolution models:

- Image pyramids
- Volume pyramids
- Texture and reflectance models
- Pictures from multiple angles
- Ray space
- Polygonal models

Out of the six possible methods, polygonal models have received the most work. The principle challenge of using the polygonal model for multi-resolution modelling is *polygonal simplification.* Polygonal simplification is the automatic conversion of a detailed model to a simpler one.

There has been a recent explosion in research into polygon simplification [69][77]. The main area of research for multi-resolution modelling has been in the areas of CAD and VR [64][74], where typical scenes may contain hundreds of thousands of polygons.

To date, there has only been one significant piece of research into multi-resolution modelling and radiosity, which is by Willmott [90]. Willmott combined Garland's [86] polygon simplification algorithm with hierarchical radiosity, to enable huge scenes to be rendered in a tractable amount of time.

Willmott's method is particularly effective for scenes that contain models that are constructed from very finely tessellated surfaces. Here, the polygonal simplification algorithm can contract vast numbers of neighbouring surfaces into larger, but approximate surfaces.

Level of detail (LOD) representation of an object is a complex topic. With non-automatic multi-resolution modelling, LODs introduce a 'popping effect' when a transition occurs between two LOD representations. This is because there is a fixed number of representations stored within the multi-resolution hierarchy.

Research performed by Lounsbery *et al.* [54] introduced a multi-resolution representation of a mesh that consists of a simple base mesh and a sequence of local correction terms called *wavelet coefficients*. Each wavelet coefficient represents the detail of the input mesh at various resolutions. Thus, depending upon what level of detail is required, coefficients can be added or removed to achieve the desired result.

The major advantage of the method by Lounsbery *et al.* [54], is that the resulting approximation is guaranteed to be within a specified error tolerance of the original mesh. However, Eck *et al.* [65] found that the method proposed by

Lounsbery *et al.* [54] had a serious shortcoming. It is restricted to meshes with *subdivision connectivity*, that is, it is restricted to the simple base mesh. Eck *et al.* [65] propose a method that reduces this shortcoming.

Hoppe [72][81] introduced the *progressive mesh*, which was a new scheme for storing and transmitting arbitrary triangular meshes. The progressive mesh incorporates a new mesh simplification procedure that aims to preserve both the geometry of a mesh and its overall appearance. The most significant contribution this algorithm presents that has not been introduced before, is that it offers efficient, loss-less and continuous representation of an arbitrary mesh.

6.3 Combining Radiosity and Multi-resolution Modelling

Multi-resolution modelling offers the most logical approach for improving the performance and scalability of radiosity algorithms. Hierarchical and clustered hierarchical radiosity algorithms provide the most optimal methods for solving the radiosity equation. However, modern day rendering requirements demand that hundreds of thousands, if not millions of polygons must be rendered at interactive speeds. It is currently not possible for radiosity methods to render such scenes using current techniques.

The fundamental problem with the radiosity method lies with the way radiosity models achieve global illumination. Thus, the penalty for obtaining extremely high quality rendered images is paid in the amount of time needed to compute a solution.

Since it has been established that clustered hierarchical radiosity algorithms provide the most optimal methods for solving the radiosity equation, then the solution to the problem of scalability must be dealt with elsewhere. The only other source of control left, is over the input scene geometry. Hence, if the method for solving the radiosity equation cannot be made more efficient, then the quantity of input will have to be reduced.

Thus, the aim of any algorithm that exploits the initial scene geometry is to optimise the input polygons such that the radiosity algorithm operates at peak efficiency, whilst generating high quality images with the reduced input. Multi-resolution modelling offers the best means for achieving this goal. Currently, level of detail in multi-resolution modelling is mainly used for reducing the complexity of distant objects. Thus, objects that are further away from the users view point, will be represented by lower resolution definitions.

However, the relative positioning of objects within a scene is not the only factor that influences the level of detail an object should be represented at. The lighting within a scene is the most influential factor for determining the level of detail of objects. For example, objects that are totally occluded or lie within areas of a scene that receive relatively little light, should be represented by a corresponding lower resolution model, even though those objects may be close to the users viewpoint. Unfortunately, in the main application areas of multi-resolution modelling (e.g. VR and CAD), the influence/effects of lighting on the resolution of objects within are scene are typically overlooked.

6.3.1 The Proposed Algorithm

To achieve an algorithm that incorporates multi-resolution modelling and hierarchical radiosity is relatively simple, but such an algorithm would require the following non-trivial algorithms:

- A flexible clustering algorithm that form tight bounding boxes around distinct objects.
- A polygon/mesh simplification algorithm.
- A hierarchical radiosity algorithm.

The results presented in Chapter 5 have already shown that the localisation method provides a highly scalable algorithm for solving large, complex scenes. Hence, the proposed algorithm would essentially utilise multi-resolution modelling to further optimise the surfaces contained within the clusters, which were generated by the localisation method described earlier in this thesis. Therefore, we present a prototype

to demonstrate the feasibility of constructing such a system and present preliminary results showing the performance gained by this system.

We begin by clustering a scene in the same fashion as the localisation method. Then, using the same light transport information gathered for the current cluster, we determine an appropriate level of detail representation for that cluster.

There are two basic control factors for determining the level of detail a cluster should be represented at:

- The level of illumination falling on a cluster.
- The distance between the cluster and the user's viewpoint.

Ideally, all multi-resolution control factors should be view independent, due to the view independence of radiosity. Unfortunately the second control factor is an entirely view dependent metric. However, it is an essential parameter in multi-resolution modelling and hence is included here.

The primary control of level of detail representation will be from the illumination of the scene. Thus, for clusters that lie in dark areas of the scene, either because there is little surrounding illumination or the cluster is occluded, a relatively coarse level of detail representation is required. Conversely, clusters that are well illuminated would be represented at a correspondingly finer level of detail. However, it is generally likely that there will be clusters that are very well lit, but are relatively distant. These clusters would be better represented at a coarser level of detail. Hence, we apply the second control factor to revise the current estimate for the level of detail representation of a cluster.

Once all clusters have been represented at an appropriate level detail, the clusters are then detached from the scene and rendered independently by hierarchical radiosity. Figure 6.2 shows the basic algorithm.

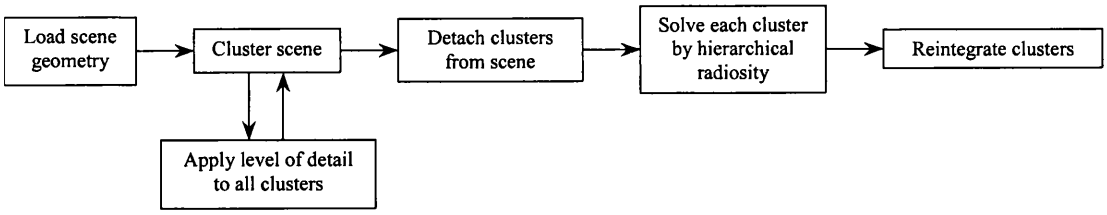


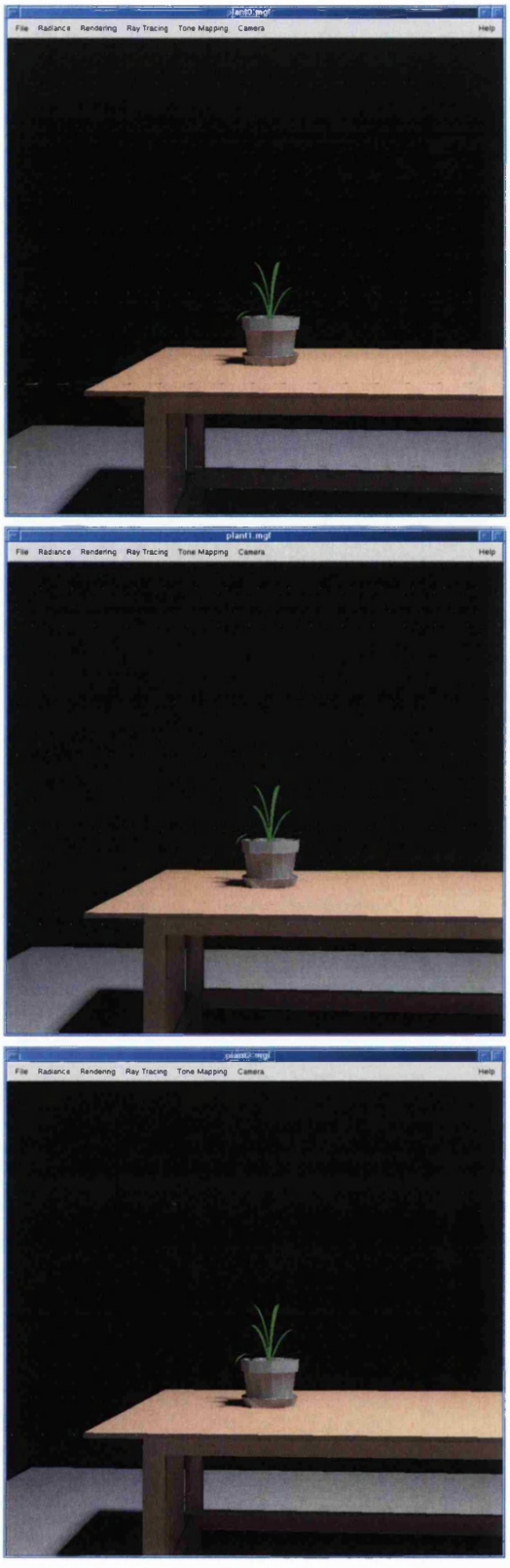
Figure 6.2 A basic algorithm for combining level of detail and the localisation method.

To generate multi-resolution models, we use Garland's polygon simplification algorithm [86] to generate polygon meshes at various levels of detail. Incidentally, this polygon simplification algorithm was also used as a base algorithm by Willmott [90].

Garland's polygon simplification algorithm takes a single argument that specifies the desired number faces the input model will be reduced to. Unfortunately, there were no other mechanisms for controlling the level of detail produced by this algorithm. Hence, with no metrics for guiding the simplification, the levels of detail representations were performed manually.

Figure 6.3 shows an example of the effectiveness of multi-resolution modelling. In this example we show the model of the plant at two levels of detail. The first image, Figure 6.3(a), is the control image that shows the plant model at it original resolution (705 surfaces). Figure 6.3(b) shows the same scene but the level of detail of the plant model has been reduced to 600 surfaces. Finally, Figure 6.3(c) shows the plant model that is constructed from only 229 surfaces.

It can clearly be seen from Figure 6.3 that there is very little perceptible difference between the lower resolution models and the original image. As a result, for this particular scene the plant model can be represented with only a third of its original number of surfaces. Hence, a significant saving is made.



(a)

(b)

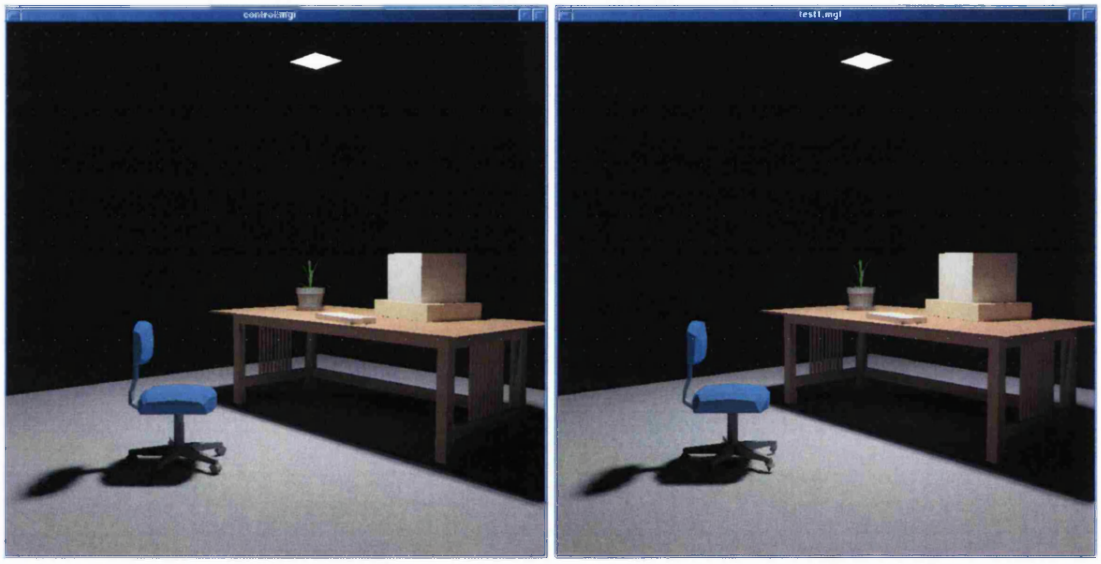
(c)

Figure 6.3 Multi-resolution models of a plant with (a) 705 (original), (b) 600 and (c) 229 surfaces.

6.3.2 Results

To test the performance of the combined localisation method and multi-resolution modelling, we reuse the test scenes shown in Figure 5.10 and Figure 5.11. This will enable a direct comparison between the localisation method and this combined method.

Figure 6.4 shows the results for a portion of the scene in Figure 5.10(a) obtained by the localisation (Figure 6.4(a)) and localisation with multi-resolution modelling (Figure 6.4(b)) methods.



(a) (b)

Figure 6.4 (a) Original model containing 1678 surfaces. (b) Lower resolution model containing 1134 surfaces.

It can be seen from Figure 6.4 that there is little perceptible difference between the two rendered images. In this scene, multi-resolution modelling was primarily used to represent lower levels of detail for the chair and the pot plant on the desk.

The level to which polygon simplification was applied to Figure 6.4(b) was conservative. Hence, it would be possible to obtain even more coarse levels of detail for the objects in the scene. However at this level of detail, multi-resolution modelling

achieved a reduction of 30% in the total number of surfaces in the scene, whilst enabling a high quality image to be rendered.

The following table (Table 6.1) shows the total computation times for rendering the scenes in Figure 5.10 and Figure 5.11, for the combined LOD and localisation method.

	Rendering Time (s)		
Scene	Localisation	Recursive Localisation	LOD+Recursive Localisation
Figure 5.10(a)	557.55	764.01	701.07
Figure 5.10(b)	1403.95	1416.68	1248.8
Figure 5.10(c)	1897.95	1911.58	1569.99
Figure 5.10(d)	2443.97	2167.94	2162.33
Figure 5.11(a)	3678.76	2237.45	1864.79
Figure 5.11(b)	5674.45	3852.95	2831.76
Figure 5.11(c)	7022.81	4549.94	3651.18

Table 6.1 Time comparisons between the localisation methods and LOD+localisation.

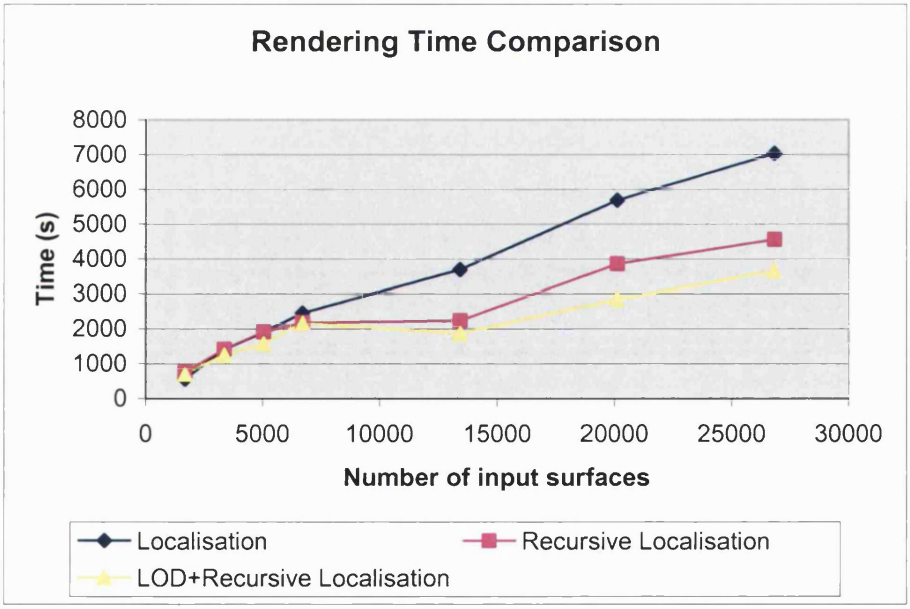


Figure 6.5 Rendering time comparisons between the localisation, recursive localisation and LOD+recursive localisation methods.

Figure 6.5 shows a graph comparing the total rendering times obtained by the localisation algorithms in Section 5.3 and the combined LOD and recursive localisation algorithm described in this chapter.

As mentioned above, conservative estimates for LOD representation were used in these experiments. This was because the LOD models were constructed manually and so the results shown in Table 6.1 and Figure 6.5 could be improved. However, the results gained by this prototype, show that combining LOD with the localisation algorithms enhances the overall performance for solving hierarchical radiosity.

6.4 Summary

In this chapter, we have discussed multi-resolution modelling. The benefits of multi-resolution modelling in VR and CAD simulations, is impressive. This technique has enabled walkthroughs of scenes containing millions of triangles [74] at interactive rates.

Multi-resolution modelling would be an ideal method for reducing the scene complexity for radiosity. Work done by Willmott [90] has confirmed this. However, Willmott's method does not make use of the lighting within the environment to guide the level of detail of objects within a scene. Instead, Willmott replaces the patch hierarchies that would be generated by hierarchical radiosity, with multi-resolution models.

Although significant speedups have been obtained by Willmott [90], the construction of the test scenes are such that the majority of all surfaces are contained within a few highly complex objects. In general scenes, this is not usually the case. A scene will typically contain many, less complex objects. Hence the results of Willmott's algorithm [90] will be less dramatic.

However, if multi-resolution modelling was applied to the localisation algorithm presented earlier in this thesis, the complexity of the localised clusters could be further

reduced. Thus in this chapter, we have combined level of detail with the localisation algorithm.

We have shown that multi-resolution models can represent objects contained within clusters generated by the localisation algorithm described in previous chapters. The results gained in this chapter show that the time taken to render the large, complex scenes in Figure 5.11 for the combined LOD and recursive localisation algorithm produces a consistent 100% performance increase. Even though the LOD models were constructed by hand, this is still a significant result and thus multi-resolution modelling should always be considered in radiosity systems that solve large, complex scenes.

Unfortunately, the current implementation for this combined LOD and recursive localisation algorithm method has two major shortcomings:

- LODs are performed manually.
- No facility is provided for LOD storage, thus LODs must always be recomputed on the fly.

These two shortcomings must be resolved if large, complex scenes are to be solved by radiosity methods. However, the purpose for the implementation and experimentation of a combined hierarchical radiosity and multi-resolution modelling system presented in this chapter, is to demonstrate the feasibility and practicability of such a system.

Chapter 7 Conclusion

As we have stated in many sections of this thesis, the demand for high quality computer graphics images is ever-increasing. Simple images that were acceptable two decades ago are no longer satisfactory. Hence, the demand for realism now extends throughout the various user communities and embraces such diverse applications as architectural representations, computer games, CAD, medical imaging and the film and entertainment industry. Unfortunately, the cost of realism is high in terms of computational complexity and real time construction of such images, is still beyond the power of available systems.

The highest quality realism is obtained from radiosity images, but, computing the radiosity solution for a scene is a time consuming process, requiring $O(n^2)$ space and time complexity to solve the radiosity matrix. Thus, early work in radiosity concentrated upon the efficient computation of this matrix. As a result, efficient methods for computing form-factors and iterative methods for solving the radiosity matrix were developed.

This led to the implementation of the 'progressive refinement radiosity' algorithm, which was the first radiosity system to implement the new paradigm of 'shooting' light into an environment, and was also the first to attempt a multi-resolution mesh representation of the surfaces within a scene, albeit a two-level sub-structure of patches and elements [22][31].

Although progressive refinement radiosity still required $O(n^2)$ time to arrive at a complete radiosity solution, it was a unique algorithm that enabled partial radiosity solutions to be rendered after every iteration, in $O(n)$ time. This was a very important feature, as it enabled scenes to be previewed at increasingly more accurate stages of progressive refinement. Hence a decision could be made on whether a solution should run to completion or be terminated.

The hierarchical radiosity algorithm [39] eventually superseded progressive refinement radiosity. Hierarchical radiosity has been without a doubt, the most valuable method for the accurate and rapid radiosity solution for scenes of modest sizes. The $O(n)$ running time complexity of this algorithm represents the most optimal efficiency for arriving at a complete solution. However, the $O(k^2)$ linking time limits the overall size of scenes that hierarchical radiosity can manage.

Clustering techniques [57][59] have since been incorporated into hierarchical radiosity algorithms and have significantly improved the running time complexity from $O(k^2 + n)$ to $O(k\log k + n)$. Thus, clustered hierarchical radiosity algorithms represent the most efficient methods for solving the radiosity of a scene.

In this thesis, we have presented a new approach to clustering for hierarchical radiosity. The objective for this new approach was to improve the running time complexity of hierarchical radiosity and thus its scalability, so that large, complex scenes can be rendered in a tractable amount of time.

We have shown that the use of clustering to localise clusters of surfaces, reduces the size of input that needs to be solved by hierarchical radiosity. As a result, significant increases in performance have been achieved. The results of our clustering strategy show that our localisation technique scales linearly with increasing complexity of the input scene.

The most notable achievements obtained from our localisation algorithms are:

- A simple clustering algorithm that can be recursively reapplied,
- The quality of rendered shadows are unaffected by clustering,
- The running time complexity for linking is linear instead of quadratic,
- The total running time complexity for rendering complex scenes is linear,
- For very complex scenes, the recursive localisation method is more than 50% faster than the localisation method,
- When compared to existing clustered hierarchical algorithms, both the localisation and recursive localisation methods achieve super-linear performance.

The simple bounding box technique proposed earlier, has proved to be a quick and simple method for generating clusters. It also has the advantage of being able to recursively locate sub-clusters.

One problem with clustering algorithms is their insensitivity to the illumination of a scene. Since clustering algorithms operate solely on the scene geometry, it is usually not possible to control the behaviour of clustering in a fashion more suitable to the illumination of a scene. Thus it is possible for clusters to contain deep hierarchies of sub-clusters. By designing a clustering method that initially gathers surfaces into general clusters, only clusters that are in areas with significant illumination will need to be refined. As a result, we were able to keep the cost of clustering to a minimum.

An important feature of our localisation method is that there is no degradation in the quality of the shadows cast by the surfaces within a cluster. One of the main problems with isotropic clustering algorithms is that the surfaces contained within, very rarely behave isotropically. As a result, shadows tend to become washed out. This is not the case for our algorithm. In our algorithm, we add all portions of the surrounding geometry that lie within the extremal shadow boundary cast by a cluster, into the cluster. Then each cluster is solved independently by hierarchical radiosity. With this approach, there is no loss of shadow quality.

However, the current implementation of the localisation technique is not without drawbacks. The first major limitation of this method is that a single, user defined, BFA threshold is used to determine whether clusters are deemed suitable for detachment. This scheme works well for scenes that have an even illumination level throughout the environment. However, for scenes that have varying levels of illumination, this simple metric is insufficient and hence clusters will not be correctly selected for detachment. Thus, further research should be carried out into finding an automatic and dynamic metric for determining cluster detachment.

The second drawback is that when a cluster is detached from the main scene, any energy contributions it would have made back into the scene are lost. The main effect of this is usually a reduction in the overall brightness of a scene. However, careful selection of the BFA threshold can keep the amount of energy lost to a minimum. A

viable solution to this problem would be to leave the shell of the cluster behind. The shell would represent an approximation of the energy transported by the surfaces within the cluster, in much the same way as traditional clustering methods for hierarchical radiosity.

Today, there is an ever increasing demand for complex environments to be rendered. Not only are these environments required to be rendered at interactive rates, they must also be of photo-realistic quality. Due to the view-independence of radiosity, it is a logical choice for rendering virtual environments. However, the running time complexity of current radiosity algorithms means that they cannot compute radiosity solutions for complex scenes at interactive rates. Hence radiosity is rarely used in modern day VR simulations.

Since there is a large demand for realistic rendering of complex VR environments, especially in the computer games industry, there has been a sizeable amount of research into level of detail representation of objects in multi-resolution modelling. Multi-resolution modelling has been used successfully to enable huge virtual environments to be rendered at interactive rates [74]. However most of these simulations utilise very simple lighting and shading techniques for rendering. Therefore, to achieve photo-realistic rendering would require radiosity or ray-tracing algorithms.

Multi-resolution modelling is the next step forward for photo-realistic rendering algorithms to achieve the performance and scalability for rendering huge scenes. It has already been shown by Willmott [90], that multi-resolution modelling can be applied to hierarchical radiosity. However, Willmott's method [90] does not make use of the illumination within the scene to guide the level of detail which objects should be represented at. Instead, Willmott only demonstrates the effectiveness of the polygon simplification algorithm by Garland [86]. However, the results by Willmott [90] confirm that reducing the complexity of the input geometry is the only method that will significantly improve the performance of hierarchical radiosity. This is also confirmed by the results gained from our localisation algorithms.

The localisation method presented in this thesis aims at reducing the scene complexity by using the light transported between clusters to determine whether a cluster of surfaces can be rendered independently. Thus, each cluster contains a reduced number of surfaces that need to be solved by hierarchical radiosity. In contrast, Willmott uses a direct approach of simplifying the geometry within the scene. Either way, both methods aim to reduce the computation time of hierarchical radiosity by reducing the number of surfaces.

At the present time, our localisation method does not optimise the surfaces contained within the localised clusters. Therefore, it is very possible that objects within the scene could be optimised considerably by multi-resolution modelling. Preliminary results have been obtained from our prototype system that combines the recursive localisation method with multi-resolution modelling. The results shown in Section 6.3 suggest that a 100% increase in performance is attainable, even with conservative LOD representations for large complex scenes. Hence, with improved metrics it is possible to further reduce the level of detail of the objects within the clusters in the scene.

Unfortunately, the prototype system is limited in two important ways. Firstly, the construction of the LOD representations was performed manually. Secondly, there was no facility to store the LOD representations within the system. However, research into *progressive meshes* by Hoppe [72][81] appears to provide the necessary framework for the storage of multi-resolution models and also the smooth progression between adjacent LOD representations.

Thus, as a proposal for further work, an automatic system for generating and storing multi-resolution models combined with hierarchical radiosity should be investigated. Such a system would require perception based metrics for the automatic determination of the level of detail for all clustered objects, according to the illumination within a scene.

Bibliography

[1] Nusselt W. 1928. *Grapische Bestimmung des Winkelverhältnisses bei der Wärmestrahlung.* Zeitschrift des Vereines Deutscher Ingenieure. 72(20):673.

[2] Eckert E R G. 1935. *Bestimmung des Winkelverhältnisses beim Strahlungsaustausch Durch das Lichtbilb.* Zeitschrift des Vereines Deutscher Ingenieure. 79:1495-1496.

[3] Moon P. 1936. *The Scientific Basis of Illumination Engineering.* New York: McGraw-Hill.

[4] Eckbert E R G, Drake R M. 1959. *Heat and Mass Transfer.* New York: McGraw-Hill.

[5] Sutherland I E, Hodgman G W. January 1974. *Reentrant Polygon Clipping.* Communications of the ACM. 17(1):32-42.

[6] Baumgart B G. October 1974. *Geometric Modeling for Computer Vision.* Artificial Intelligence Project Memo AIM-249 (CS-TR-74-463), Computer Science Department, Stanford University.

[7] Baumgart B G. 1975. *A Polyhedral Representation for Computer Vision.* Proc. National Computer Conference. Pages 589-596.

[8] Atkinson K E. 1976. *A Survey of Numerical Methods for the Solution of Fredholm Integral Equations of the Second Kind.* Society for Industrial and Applied Mathematics.

[9] Crow F. July 1977. *Shadow Algorithms for Computer Graphics.* Computer Graphics (ACM SIGGRAPH '77 Proceedings). 11(3):242-247.

[10] Siegel R, Howell J R. 1978. *Thermal Radiation Heat Transfer.* Hemisphere Publishing Corp.

[11] Murdoch J B. January 1981. *Inverse Square Law Approximation of Illuminance.* Journal of the Illuminating Engineering Society. 11(2):96-106.

[12] Howell J R. 1982. *A Catalog of Radiation Configuration Factors.* McGraw-Hill.

[13] Nishita T, Nakamae E. November 1983. *Half-tone Representation of 3-D Objects Illuminated by Area Light Sources or Polyhedron Sources.* The IEEE Computer Society's Seventh International Computer Software and Applications Conference. Pages 237-242.

[14] Fletcher C A J. 1984 *Computational Galerkin Methods.* Springer-Verlag, New York.

[15] Goral C M, Torrence K E, Greenberg D P, Battaile B. July 1984. *Modelling the Interaction of Light between Diffuse Surfaces.* Computer Graphics (ACM SIGGRAPH '84 Proceedings). 18(3):212-222.

[16] Glassner A S. October 1984. *Space Division for Fast Ray Tracing.* IEEE Computer Graphics and Applications. 4(10):15-22.

[17] Appel A. January 1985. *An efficient program for many-body simulation.* SIAM Journal of Sci. Stat. Computing. 6(1):85-103.

[18] Nishita T, Nakamae E. July 1985. *Continuous Tone Reproduction of Three-Dimensional Objects Taking Account of Shadows and Interreflections.* Computer Graphics (ACM SIGGRAPH '85 Proceedings). 19(3):23-30.

[19] Cohen M F, Greenberg D P. August 1985. *The Hemi-Cube: A Radiosity Solution for Complex Environments.* Computer Graphics (ACM SIGGRAPH '85 Proceedings). 19(3):31-40.

[20] ANSI/IES. 1986. *Nomenclature and Definitions for Illumination Engineering.* ANSI/IES RP-16-1986. New York: Illumination Engineering Society of North America.

[21] Cook R L. January 1986. *Stochastic Sampling in Computer Graphics.* ACM Transactions on Graphics. 5(1):51-72.

[22] Cohen M F, Greenberg D P, Immel D S, Brock P J. March 1986. *An Efficient Radiosity Approach for Realistic Image Synthesis.* IEEE Computer Graphics and Applications. 6(3):26-35.

[23] Kajiya J T. August 1986. *The Rending Equation.* Computer Graphics (ACM SIGGRAPH '86 Proceedings). 20(4):269-278.

[24] Bergeron P. September 1986. *A General Version of Crow's Shadow Volumes,* IEEE Computer Graphics and Applications. 6(9):17-28.

[25] Maxwell G M, Bailey M J, Goldschmidt V W. September 1986. *Calculations of the Radiation Configuration Factor using Ray Casting.* Computer-Aided Design. 18(7):371-379.

[26] Goldsmith J, Salmon J. May 1987. *Automatic Creation of Object Hierarchies for Ray Tracing.* IEEE Computer Graphics and Applications. 7(5):14-20.

[27] Amanatides J, Woo A. August 1987. *A Fast Voxel Traversal Algorithm for Ray Tracing.* In Proceedings of Eurographics '87. Pages 3-10.

[28] Blinn J F. January 1988. *Me and My (Fake) Shadow.* IEEE Computer Graphics and Applications. 8(1):82-86.

[29] Haines E. April 1998. *Automatic Creation of Object Hierarchies for Ray Tracing.* Ray Tracing News, volume 6 number 1. http://www1.acm.org/pubs/tog/resources/RTNews/html/rtnews3a.html

[30] Malley T J V. June 1988. *A Shading Method for Computer Generated Images.* Master's Thesis. University of Utah.

[31] Cohen M F, Chen S E, Wallace J R, Greenberg D P. August 1988. *A Progressive Refinement Approach to Fast Radiosity Image Generation.* Computer Graphics (ACM SIGGRAPH '88 Proceedings). 22(4):75-84.

[32] Sillion F, Puech C. July 1989. *A General Two-Pass Method Integrating Specular and Diffuse Reflection.* Computer Graphics (ACM SIGGRAPH '89 Proceedings). 23(3):335-344.

[33] Wallace J R, Elmquist K A, Haines E A. July 1989. *A Ray Tracing Algorithm for Progressive Radiosity.* Computer Graphics (ACM SIGGRAPH '89 Proceedings). 23(3):315-324.

[34] Chen S E. 1991. *Implementing Progressive Radiosity with User-Provided Polygon Display Routines.* Graphics Gems 2. Pages 295-298, 583-597.

[35] Beran-Koehn J C, Pavicic M J. 1991. *A Cubic Tetrahedral Adaptation of the Hemi-cube Algorithm.* Graphics Gems 2. Pages 299-302.

[36] Heckbert P S. June 1991. *Simulating Global Illumination Using Adaptive Meshing.* PhD Thesis, Department of Electrical Engineering and Computer Science, University of California at Berkeley.

[37] Baum D R, Mann S, Smith K P, Winget J M. July 1991. *Making Radiosity Usable: Automatic Preprocessing and Meshing Techniques for the Generation of Accurate Radiosity Solutions.* Computer Graphics (ACM SIGGRAPH '91 Proceedings). 25(4):51-60.

[38] Chen S E, Rushmeier H E, Miller G, Turner D. July 1991. *A Progressive Multi-Pass Method for Global Illumination.* Computer Graphics (ACM SIGGRAPH '91 Proceedings). 25(4):165-174.

[39] Hanrahan P, Salzman D, Aupperle L. July 1991. *A Rapid Hierarchical Radiosity Algorithm.* Computer Graphics (ACM SIGGRAPH '91 Proceedings). 25(4):197-206.

[40] Campbell A T, Fussell D S. August 1991. *An Analytical Approach to Illumination with Area Light Sources.* Technical Report R-91-25, Department of Computer Sciences, Univerisity of Texas at Austin.

[41] Lischinski D, Tampieri F, Greenberg D P. August 1991. *Improving Sampling and Reconstruction Techniques for Radiosity.* Technical Report TR-91-1202, Department of Computer Science, Cornel University.

[42] Campbell A T. December 1991. *Modeling Global Diffuse Illumination for Image Synthesis.* PhD Thesis, Department of Computer Sciences, University of Texas at Austin.

[43] Beran-Koehn J C, Pavicic M J. 1992. *Delta Form Factor Calculation for the Cubic Tetrahedral Algorithm.* Graphics Gems 3. Pages 324-328.

[44] Heckbert P S. May 1992. *Discontinuity Meshing for Radiosity.* Third Eurographics Workshop on Rendering. Pages 203-216.

[45] Lischinski D, Tampieri F, Greenberg D P. November 1992. *A Discontinuity Meshing Algorithm for Accurate Radiosity.* IEEE Computer Graphics and Applications. 12(6):25-39.

[46] Stuerzlinger W. 1993. Constrained Delaunay Triangulation Software. Available from http://www.cs.yorku.ca/~wolfgang/software.html

[47] Gortler S, Cohen M F, Slusallek P. February 1993. *Radiosity and Relaxation Methods: Progressive Refinement is Southwell Relaxation.* Technical Report CS-TR-408-93, Department of Computer Science, Princeton University.

[48] Tampieri F. May 1993. *Discontinuity Meshing for Radiosity Image Synthesis.* PhD Thesis, Department of Computer Science, Cornell University.

[49] Rushmeier H E, Patterson C, Veerasamy A. May 1993. *Geometric Simplification for Indirect Illumination Calculations.* Graphics Interface '93. Pages 227-236.

[50] Kok A J. June 1993. *Grouping Patches in Progressive Radiosity.* In Proceedings of the Fourth Eurographics Workshop on Rendering. Pages 221-231.

[51] Zatz H R. August 1993. *Galerkin Radiosity: A Higher Order Solution Method for Global Illumination.* Computer Graphics (ACM SIGGRAPH '93 Proceedings). Pages 213-220.

[52] Ashdown I. 1994. *Radiosity A Programmer's Perspective.* John Wiley & Sons, Inc.

[53] Drettakis G. January 1994. *Structured Sampling and Reconstruction of Illumination for Image Synthesis.* CSRI Technical Report 293, Department of Computer Science, University of Toronto.

[54] Lounsbery M, DeRose T, Warren J. January1994. *Multiresolution Analysis for Surfaces of Arbitrary Topological Type.* Technical Report 93-10-05b. Department of Computer Science, University of Washington.

[55] Heckbert P S, Garland M. May 1994. *Multiresolution Modelling for Fast Rendering.* Computer Graphics (ACM SIGGRAPH '94 Proceedings). Pages 43-50.

[56] Holzschuch N, Sillion F, Drettakis G. June 1994. *An Efficient Progressive Refinement Strategy for Hierarchical Radiosity.* In Proceedings of the Fifth Eurographics Workshop on Rendering. Pages 343-357.

[57] Sillion F. June 1994. *Clustering and Volume Scattering for Hierarchical Radiosity Calculations.* In Proceedings of the Fifth Eurographics Workshop on Rendering, Darmstadt, Germany. Pages 105-117.

[58] Drettakis G, Fiume E. July 1994. *A Fast Shadow Algorithm for Area Light Sources Using Backprojection.* Computer Graphics (ACM SIGGRAPH '94 Proceedings). Pages 223-230.

[59] Smits B, Arvo J, Greenberg D. July 1994. *A Clustering Algorithm for Radiosity in Complex Environments.* Computer Graphics (ACM SIGGRAPH '94 Proceedings). Pages 435-442.

[60] Lischinski D. 1995. Constrained Delaunay triangulation software. Available from http://www.cs.huji.ac.il/~danix

[61] Christensen P, Lischinski D, Stollnitz E, Salesin D. January 1995. *Clustering for Glossy Global Illumination.* Technical Report 95-01-07, Department of Computer Science and Engineering, University of Washington.

[62] Lischinski D. January 1995. *Accurate and Reliable Algorithms for Global Illumination.* PhD Thesis, Cornell University.

[63] Sillion F, Drettakis G. March 1995. *Feature-based Control of Visibility Error: A Multi-resolution Clustering Algorithm for Global Illumination.* INRIA (iMAGIS) Research Report #2503.

[64] Maciel P W C, Shirley P. April 1995. *Visual Navigation of Large Environments Using Textured Clusters.* Symposium on Interactive 3D Graphics. Pages 95-102

[65] Eck M, DeRose T, Duchamp T, Hoppe H, Lounsbery M, Stuetzle W. August 1995. *Multiresolution Analysis of Arbitrary Meshes.* Computer Graphics (ACM SIGGRAPH '95 Proceedings). Pages 173-182.

[66] Sillion F. September 1995. *A Unified Hierarchical Algorithm for Global Illumination with Scattering Volumes and Object Clusters.* IEEE Transactions on Visualization and Computer Graphics. 1(3).

[67] Gibson S. October 1995. *Efficient Radiosity for Complex Environments.* MSc Thesis, Department of Computer Science, University of Manchester.

[68] Worrall W, Willis C P, Paddon D J. December 1995. *Dynamic Discontinuities for Radiosity.* Edugraphics and Compugraphics Proceedings. Pages 367-375.

[69] Erikson C. 1996. *Polygon Simplification: An Overview.* Technical Report TR96-016. Department of Computer Science, UNC-Chapel Hill.

[70] Shewchuk J R. 1996. *Triangle.* Constrained Delaunay Triangulation software. Available from http://www.cs.cmu.edu/~quake/triangle.htm

[71] Chrysamthou Y. January 1996. *Shadow Computation for 3D Interaction and Animation.* PhD Thesis, Department of Computer Science, Queen Mary and Westfield College, University of London.

[72] Hoppe H. August 1996. *Progressive Meshes.* Computer Graphics (ACM SIGGRAPH '95 Proceedings). Pages 99-108.

[73] Gibson S, Hubbold R J. December 1996. *Efficient Hierarchical Refinement and Clustering for Radiosity in Complex Environments.* Computer Graphics Forum. 15(5):297-310.

[74] Aliaga D, Cohen J, Zhang H, Bastos R, Hudson T, Erikson C. 1997. *Power Plant Walkthrough: An Integrated System for Massive Model Rendering.* Technical Report TR97-018. Department of Computer Science, UNC-Chapel Hill.

[75] Atkinson K. 1997. *The Numerical Solution of Integral Equations of the Second Kind.* Cambridge University Press.

[76] Ward G, Shakespeare R, Ashdown I, Rushmeier H. March 1997. *The Materials and Geometry format Version 2.0.* Available from http://radsite.lbl.gov/mgf

[77] Heckbert P S, Garland M. May 1997. *Survey of Polygonal Surface Simplification Algorithms.* Draft of Carnegie .Mellon University Computer Science Technical Report.

[78] Loscos C, Drettakis G. September 1997. *Interactive High-Quality Soft Shadows in Scenes with Moving Objects.* In Proceedings of Eurographics '97. 16(3):220-230.

[79] Stamminger M, Slusallek P, Seidel H. September 1997. *Bounded Radiosity – Illumination on General Surfaces and Clusters.* In Proceedings of Eurographics '97. 16(3).

[80] Stamminger M, Slusallek P, Seidel H. 1997. *A Comparison of Different Refiners for Hierarchical Radiosity.* 3D Image Analysis and Synthesis '97. Pages 27-34.

[81] Hoppe H. January 1998. *Efficient Implementation of Progressive Meshes.* Technical Report MSR-TR-98-02, Microsoft Research, Microsoft Corporation. Also in Computers & Graphics '98. 22(1):27-36.

[82] Worrall A. April 1998. *Dynamic Discontinuity Meshing.* PhD Thesis, Department of Computer Science, University of Bristol.

[83] Hedley D. August 1998. *Discontinuity Meshing for Complex Environments.* PhD Thesis, Department of Computer Science, University of Bristol.

[84] Gibson S. September 1998. *Efficient Radiosity Simulation Using Perceptual Metrics and Parallel Processing.* PhD Thesis, Department of Computer Science, University of Manchester.

[85] Stamminger M, Schirmacher H, Slusallek P, Seidel H P. September 1998. *Getting Rid of Links in Hierarchical Radiosity.* Computer Graphics Forum. 17(3):165-174.

[86] Garland M. May 1999. *Quadric-Based Polygonal Surface Simplification.* PhD Thesis, School of Computer Science, Carnegie Mellon University.

[87] Hasenfratz J, Damez C, Sillion F, Drettakis G. September 1999. *A Practical Analysis of Clustering Strategies for Hierarchical Radiosity.* Eurographics '99. 18(3):221-232.

[88] Muller G, Schafer S, Fellner D W. October 1999. *Automatic Creation of Object Hierarchies for Radiosity Clustering.* 7th Pacific Conference on Computer Graphics '99. Pages 21-29.

[89] Soler C, Sillion F X. June 2000. *Hierarchical Instantiation for Radiosity.* Eurographics Workshop on Rendering Techniques 2000. Pages 173-184.

[90] Willmott A J. November 2000. *Hierarchical Radiosity with Multiresolution Meshes.* PhD Thesis, School of Computer Science, Carnegie Mellon University.

[91] Bekaert P, de Laet F S, Peers P, Masselus V. August 2001. *RenderPark.* Test-bed software system for Global Illumination. Available from http://www.renderpark.be

[92] Slater M, Steed A, Chrysanthou Y. 2002. *Computer Graphics and Virtual Environments.* Addison-Wesley.