ISSN 1361 - 6161

THE UNIVERSITY of MANCHESTER

Computer Science
University of Manchester

Efficient Radiosity Simulation using Perceptual Metrics and Parallel Processing

Simon Gibson

Department of Computer Science

University of Manchester

Technical Report Series

UMCS-99-9-1

Efficient Radiosity Simulation using Perceptual Metrics and Parallel Processing[1]

Simon Gibson

Department of Computer Science
University of Manchester
Oxford Road, Manchester, UK.
sg@cs.man.ac.uk

September 1999

[1]Refereed by: Chris Kirkham

Abstract

The generation of realistically illuminated synthetic images has been at the forefront of computer graphics research since the early 1970s. The core problem that must be addressed in order to achieve this goal is the accurate simulation of light transport throughout an environment. As compute resources have developed, rendering techniques have also progressed from basic local illumination models to physical simulations that consider not only the local interaction of light with a surface, but also important global effects such as shadows and inter-reflections.

The radiosity method is one of the most important global illumination algorithms, and the generation of geometrically complex, view-independent radiosity solutions has received much attention in recent years. This thesis provides two contributions that enable such radiosity solutions to be generated quickly and efficiently.

Firstly, a new framework for radiosity simulation (based on the progressive refinement method) is presented that uses perceptually-based calculations to control the simulation process. It is shown how such an approach enables computational effort to be focussed into those areas that have a visually significant effect on the appearance of the solution, leading to considerable savings in computation time.

Secondly, a novel algorithm for computing radiosity solutions on distributed shared-memory parallel architectures is described. Our approach is based on the notion of a *sub-iteration* being the transfer of energy from a single source to a subset of the scene's receiver patches. By using an efficient queue-based scheduling system to process these sub-iterations, we show how radiosity solutions can be generated without the need for processor synchronisation between iterations of the progressive refinement algorithm. This free-running method of computation exhibits excellent data locality, thereby limiting the amount of inter-processor communication that occurs, and eliminates the need for load-balancing until the end of the solution process. Results are presented indicating an almost linear speedup in the time required to generate a solution.

Contents

1 Image Synthesis **5**
1.1 Realistic Image Synthesis . . . 5
1.2 Local Illumination . . . 6
1.3 Global Illumination . . . 6
1.3.1 Ray-tracing . . . 9
1.3.2 Radiosity . . . 10
1.4 Thesis Contribution . . . 12
1.5 Thesis Outline . . . 12

2 Radiosity Methods **14**
2.1 Radiometry . . . 14
2.1.1 Radiance . . . 14
2.1.2 Irradiance . . . 15
2.1.3 Reflectance . . . 15
2.1.4 Radiosity . . . 15
2.2 Photometry . . . 16
2.2.1 Luminance and Illuminance . . . 17
2.3 The Global Illumination Equation . . . 17
2.3.1 The Radiosity Approximation . . . 18
2.3.2 The Finite-Element Approach . . . 18
2.4 Meshing . . . 21
2.5 Form-Factors . . . 21
2.5.1 Differential Area-To-Area Form-Factors . . . 22
2.5.2 Visibility . . . 23
2.6 Solution Techniques . . . 26
2.6.1 Full-Matrix Methods . . . 27
2.6.2 Progressive-Refinement Radiosity . . . 28
2.6.3 Hierarchical Methods . . . 34
2.7 Rendering . . . 39
2.7.1 Hardware Rendering . . . 40
2.7.2 Ray Tracing . . . 40
2.7.3 Final-Gather . . . 41
2.8 Summary . . . 42

3 Tone-Reproduction **43**
3.1 Introduction . . . 43
3.2 The Human Visual System . . . 47
3.2.1 Adaptation . . . 48
3.2.2 Threshold Sensitivity . . . 49
3.2.3 Brightness . . . 49
3.3 Principles of Tone-Reproduction . . . 52

3.4 Single-Scale Operators . . . 55
3.4.1 Tumblin and Rushmeier . . . 55
3.4.2 Ward . . . 56
3.4.3 Ferwerda *et al.* . . . 56
3.4.4 Adaptation Luminances . . . 58
3.4.5 A Comparison of Models . . . 58
3.5 Multi-Scale Operators . . . 60
3.6 Colour . . . 63
3.7 Tone-Reproduction for View-Independent Solutions . . . 64
3.7.1 Static Adaptation . . . 65
3.8 Summary . . . 67

4 Perceptually-Driven Radiosity **68**
4.1 Introduction . . . 68
4.2 Motivation . . . 69
4.3 Perception of Real-World Luminances . . . 70
4.4 Tone-Reproduction Estimation . . . 73
4.5 Adaptive Refinement . . . 74
4.5.1 Element Refinement . . . 76
4.5.2 Shadow Boundary Refinement . . . 82
4.5.3 Refinement Simplification . . . 82
4.6 Visibility Computation . . . 86
4.6.1 Approximating Visibility . . . 89
4.7 Summary . . . 92

5 Scalable Parallel Radiosity using Distributed Shared Memory **97**
5.1 Introduction . . . 97
5.2 Previous Work . . . 98
5.2.1 Summary of Previous Work . . . 102
5.3 The Silicon Graphics Origin2000 . . . 103
5.4 Data Partitioning . . . 104
5.5 Sub-iteration Scheduling . . . 108
5.5.1 Convergence-Sensitive Scheduling . . . 111
5.5.2 Locality-Sensitive Scheduling . . . 112
5.6 Dynamic Load-Balancing . . . 112
5.7 Visibility Testing . . . 114
5.8 Results . . . 116
5.8.1 Speedup and Efficiency . . . 118
5.8.2 Convergence . . . 121
5.8.3 Dynamic Load-Balancing . . . 124
5.9 Summary . . . 129

6 Conclusion **130**
6.1 Thesis Summary . . . 130
6.1.1 Tone-Reproduction . . . 130
6.1.2 Perceptually-Driven Radiosity . . . 131
6.1.3 Scalable Parallel Radiosity using Distributed Shared Memory . . . 131
6.2 Future Work . . . 132

7 Colour Spaces **135**
7.1 RGB Colour Spaces . . . 135
7.2 CIEXYZ Colour Space . . . 135
7.3 Perceptually-Uniform Colour Spaces . . . 137
7.3.1 Colour Difference Formulae . . . 138
7.3.2 Just Noticeable Differences . . . 139

Chapter 1

Image Synthesis

The generation of realistic imagery has been at the forefront of computer graphics research since the early 1970s. The core problem that must be addressed in order to achieve this goal is the accurate simulation of light transport throughout an environment. As compute resources have developed, rendering techniques have also progressed from basic local illumination models to physical simulations that model not only the local interaction of light with a surface, but also consider important global effects such as shadows and inter-reflections. Including such effects enables the synthesised image to induce a visual experience in the viewer that matches more closely the experience which would occur if a real scene were being observed.

For many applications such as architectural or lighting design, the ability to create such realistic synthetic images eases the processes of visualisation and presentation; a designer can communicate his ideas to colleagues or clients in a more intuitive medium than traditional scale models and illumination plots, and can use rendered images during the design process itself to provide feedback on ideas and design decisions. This has been termed 'The Visual Revolution' in lighting design [Mis94], with both designers and their clients becoming aware of the possibilities that arise from the creation and analysis of photometrically realistic images of architectural spaces.

The aim of the research presented in this thesis is to devise methods for accelerating radiosity simulations, and permit visually accurate solutions to be generated for complex environments. In this chapter, we will review some basic techniques for synthesising realistic images. The next section describes what we are trying to achieve during the process of image synthesis, and discusses some of the problems that arise.

1.1 Realistic Image Synthesis

The ultimate goal of image synthesis techniques is to recreate on a computer monitor (or other display device) the visual experience that is produced when an identical, real environment is observed. In the real world, light propagates through an environment, and enters the eyes of the viewer, forming a pattern on the photo-receptors in the observer's retina. The brain then translates this pattern of chemical signals into a mental picture. When creating a synthetic image, the computer must use a mathematical model of the scene and of the processes of light propagation to simulate the distribution of light entering a virtual eye. The synthetic image is then displayed to the observer on a suitable display device, and a (hopefully identical) mental picture formed when the light emitted by the display enters the observer's eyes. In order to reproduce the same visual experience, we would like to simulate *exactly* the processes of light propagation around a perfectly modelled scene. Unfortunately, there are several fundamental problems to this approach.

First of all, producing an exact model of a real environment is not possible, due to the enormous geometric detail that exists in the real world. Secondly, the propagation of light in the real world is far too complex to simulate completely, even using the world's most powerful supercomputers. A limit must be therefore be placed on the level of detail of both the geometric model and the simulation of light, and image synthesis algorithms must make certain simplifying assumptions regarding the extent to which they model reality. Another related problem is that the exact mechanisms by which our brains process visual data are unknown; we are still unsure as to exactly *what* makes an image appear realistic, and which artifacts provide visual cues indicating to a viewer that the image they are seeing is artificial. This has consequences regarding the limitations and simplifications we make to the illumination process mentioned above – exactly how should we simplify our simulations? And can we be sure that the simplifications we make will not affect our perception that the image we see is real?

A final obstacle arises from the methods used to display images. As will be discussed in detail in later chapters, there are many features of the real world that we are unable to reproduce on a display device due to limitations in spatial and spectral resolution and dynamic range. Images cannot therefore be displayed correctly, and any picture shown to a viewer will always be an approximation of the true view of the scene. Although these display restrictions may make it seem that we are doomed to failure from the start, the success of television indicates that visually realistic images *can* be displayed despite such limitations. Indeed, as we will show in Chapter 4, given these limitations, we can use them to our advantage by restricting the detail of the simulation to that which can be displayed and distinguished by the observer.

1.2 Local Illumination

With the introduction of raster display devices, simple techniques were developed for illuminating computer-generated models. These *local illumination models* describe the light reflected from a surface towards the viewer as a function of the position of a number of idealised point or directional light-sources [Gou71, BT75]. The Phong [BT75] model is a 'standard' local illumination model used in computer graphics, and represents reflected light intensity (I) as a linear combination of a diffuse, specular and ambient term

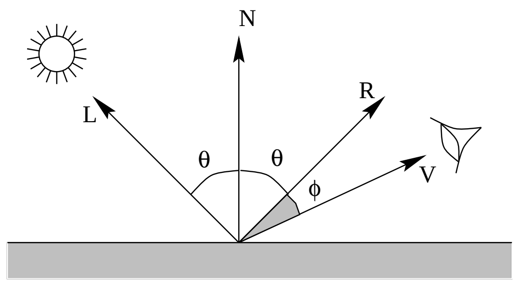
$$I = I_a k_a + \sum_{i=1}^{N} I_i [k_d \cos\theta + k_s \cos^n \phi] \tag{1.1}$$

The intensity at the surface point, I is calculated as a sum of contributions from N light-sources, each having an intensity of I_i. The diffuse term corresponds to the fraction of light that is scattered in all directions and is generally responsible for the 'colour' of the object, and is controlled by a wavelength dependent diffuse reflectivity, k_d. The component corresponding to specular reflection corresponds to light that is reflected around the mirror direction, and is non-zero only for glossy surfaces. Specular reflection is modelled using a specular coefficient, k_s, and an exponent representing the glossiness of the surface finish, n. ϕ is the angle between the direction of ideal specular reflection, R (also called the mirror direction), and the direction from which the surface point is being viewed, V (Figure 1.1). Surfaces that are facing away from all light-sources will not receive any light. In the real world, light will always still reach these surfaces by reflecting off other objects in the environment. These multiple reflections are ignored by local illumination models, and so an ambient term, $I_a k_a$, must be included to ensure that these surfaces do not appear completely black.

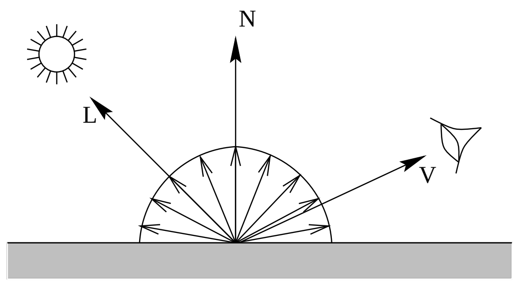
Figure 1.2(a) shows a basic scene rendered using the Phong local illumination model. The lack of important effects such as shadowing and indirect illumination means images rendered with such simple models lack realism. Their simplicity has, however, enabled their implementation in hardware on modern graphical workstations that are capable of rendering tens of millions of shaded polygons per second [MBD97].

1.3 Global Illumination

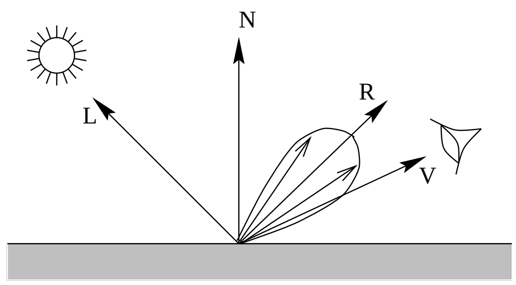
Whenever realistic images are required, *global illumination* effects such as shadowing and indirect illumination must be accounted for in the image synthesis process, as these play an important role in our perception of an environment [MRC$^+$86, Wan92]. Unlike the simple local models described in the



(a) Geometry

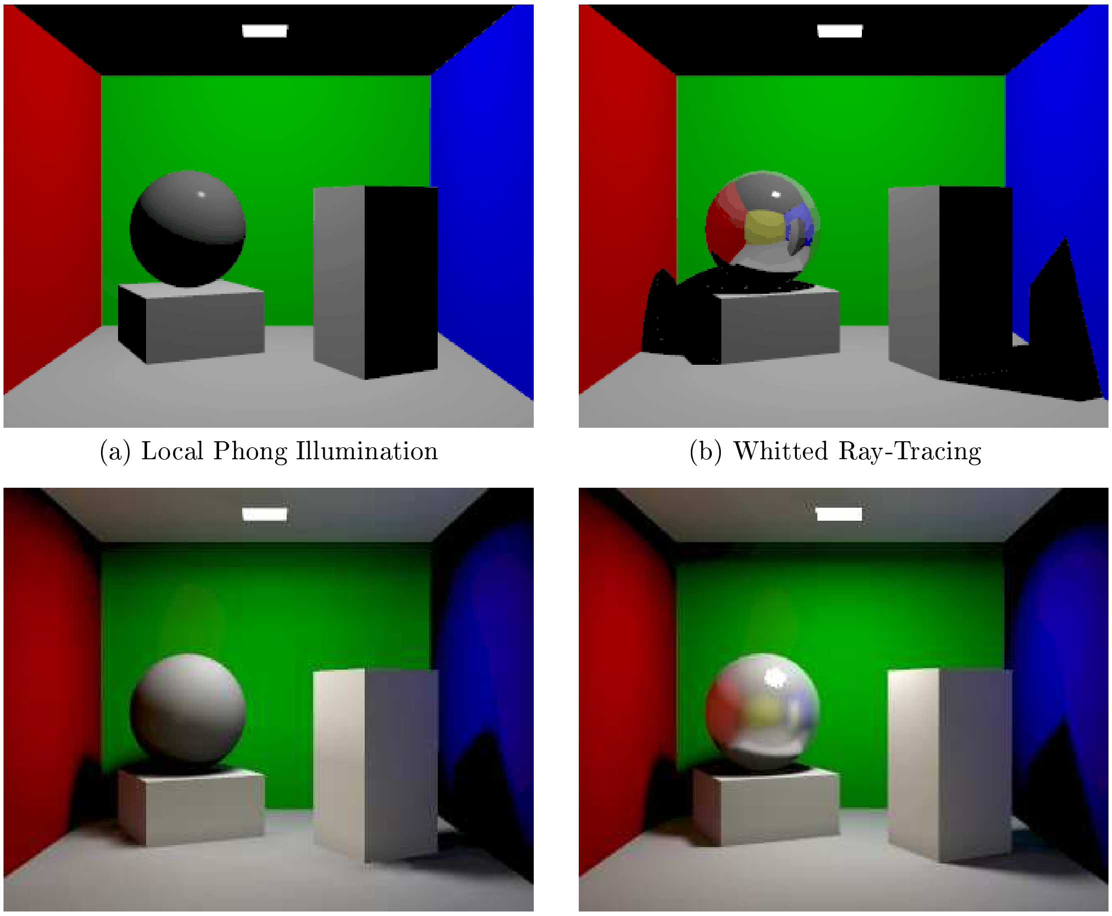


(b) Diffuse reflection



(c) Specular reflection

Figure 1.1: Geometry for the Phong illumination model.



(a) Local Phong Illumination

(b) Whitted Ray-Tracing

(c) Radiosity

(d) Radiosity plus Ray-Tracing

Figure 1.2: A Comparison of different rendering algorithms.

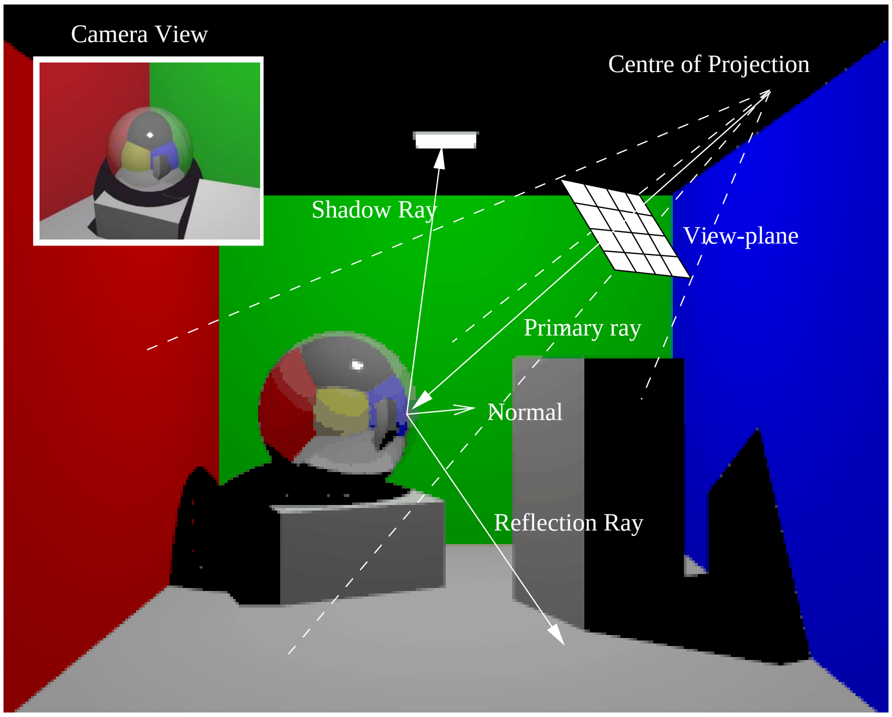


Figure 1.3: Rendering an image using ray-tracing.

previous section, global illumination algorithms account for the relationships between all objects in the scene. Many different algorithms have been proposed for modelling global illumination effects, most of which are based on variations of two different approaches – *ray-tracing* and *radiosity*.

1.3.1 Ray-tracing

Whitted's ray-tracing algorithm [Whi80] has been widely used as an image synthesis technique, as it unifies the processes of hidden surface removal, shading, reflection, refraction, and shadowing. Ray-tracing is a *view-dependent* rendering technique – its goal is to produce a static image of the scene, as viewed from a particular position. Figure 1.3 shows a particular camera view defined by a centre of projection and view-plane. The resulting image (shown in the top-left of Figure 1.3) is generated by subdividing the view-plane into a discrete grid, with each element corresponding to a pixel in the resulting image.

A *primary ray* is traced from the centre of projection through each grid element and into the scene. The ray is tested for intersection with each object, and the nearest intersection point is identified. A *shadow ray* is then traced towards each light-source in turn. Provided this ray does not intersect any other objects, a local illumination model is applied to obtain the contribution of that light-source to the surface point.

If the surface intersected by the primary ray has specular properties, *reflection* and *refraction* rays may be cast (using their respective optical laws), and the local illumination model applied again at the corresponding intersection points. This process is executed recursively, building a tree of contributions to the colour of the image pixel.

Since only single shadow and reflection rays are traced, Whitted's model generates images with unnaturally crisp shadows and sharp mirror reflections (see Figure 1.2(b)). Stochastic sampling techniques were introduced by Cook *et al.* [CPC84, Coo86] in order to model more realistic penumbral shadows from area light-sources, as well as non mirror-like reflections. This is achieved by casting several shadow or reflection rays, each in a slightly different direction and averaging the result of each. The technique

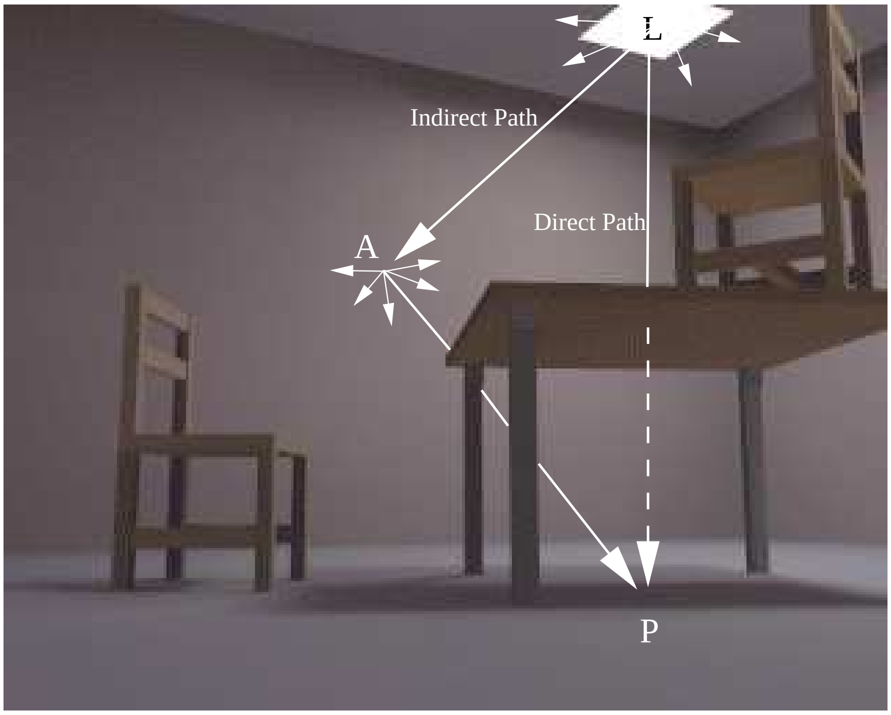


Figure 1.4: Light transport paths for indirect Illumination.

can also model depth of field, due to a simulated camera lens, and motion blur caused by moving objects or camera motion.

1.3.2 Radiosity

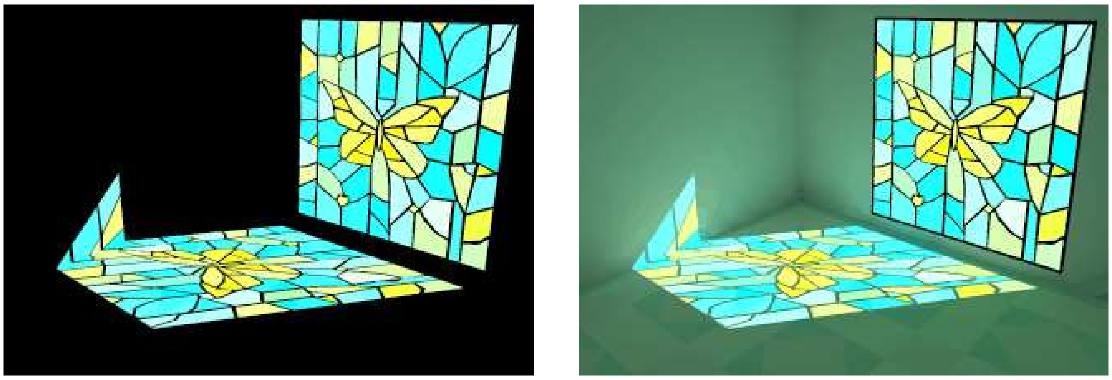
The radiosity method was originally developed by thermal engineers in the 1950s [Hot54, ED59], but not introduced to the computer graphics community until the 1980s, by Goral *et al.* [GTGB84] and Nishita *et al.* [NN85]. Since then, there has been a proliferation of methods and algorithms.

Radiosity methods aim to simulate the *inter-reflection of energy* (in this case, visible radiation) between diffuse surfaces in an environment[1]. Radiosity techniques give rise to visually important features such as soft shadows from area light-sources, and 'colour bleeding' due to secondary illumination effects (note the red tint on the smaller box in Figure 1.2(c), caused by light reflecting off the red wall).

Although Monte-Carlo ray-tracing algorithms have been developed that are capable of simulating the inter-reflection of light [Arv86, Kaj86, WRC88, AK90, SWH$^+$95], the methods by which illumination is computed in radiosity and ray-tracing algorithms are very different; ray-tracing is inherently view-dependent, and illumination is calculated only for those points in the scene that are visible from the current viewpoint. Radiosity, on the other hand, is capable of generating *view independent* solutions that contain illumination information for the entire environment. After computing a radiosity solution for a particular scene, images from any viewpoint may be rendered interactively on a graphic workstation, bringing the possibility of using virtual-reality techniques to immerse a user in a realistically illuminated virtual environment [ARJ90, Moe96].

Figure 1.4 illustrates how inter-reflection affects the illumination at a point in the scene. Although the direct path between the light-source L and point P is blocked by an occluding object, light is still able to reach P by reflecting off another surface (for example, point A). The surface at point A is,

[1]There have been many extensions made to account for specular interactions [ICG86, SAWG91, AH93, CSS96], but in this work we assume completely diffuse surfaces since we are wishing to produce view-independent solutions suitable for display using graphics hardware.



(a) Direct illumination (b) Direct and indirect illumination

Figure 1.5: Direct and indirect illumination.

in this sense, acting as a *secondary light-source*, and may cast shadows in a similar fashion to any primary source. Without accounting for indirect illumination (as in standard ray-tracing techniques, when they are approximated using a constant ambient term), all points with an occluded direct path to the primary light-source will appear to have a constant intensity. Comparing Figures 1.2(b) and (c) shows the improvement in image quality gained by modelling indirect lighting. The ceiling and near side of the tall box are no longer uniformly coloured, but contain subtle variations in their illumination. Note also the soft shadows that occur when the illumination from the area light-source is properly accounted for. Ray-tracing methods can also be used to render an image of a radiosity solution after it has been computed [Rus88, Rei92, LTG93], and allow the combination of indirect illumination with specular effects and accurate soft shadows (Figure 1.2(d)).

There are many situations where indirect reflections are an essential part of the overall level of illumination. Consider the scene shown in Figure 1.5, consisting of an empty room illuminated by a directional light (representing sunlight) through a window[2]. Image (a) shows the result when only direct light is accounted for. Apart from the area of the floor onto which the sunlight falls, the rest of the scene is completely dark. No detail can be seen that helps the viewer recognise the walls and floor. In image (b), the improvement when the indirect component of the illumination is added can be seen.

In order to test the visual realism of a radiosity solution, Meyer *et al.* [MRC$^+$86] performed a set of experiments using a physical model of a real scene, and a radiosity simulation of an identical synthetic scene. Volunteer subjects were presented with two views, one of the real model, and another of the synthetic environment. In order to remove the effect of stereo cues from the real model, each view was projected into a camera, and observed through a frosted glass plate. Subjects were then asked to discriminate between the real and synthetic scene. Approximately half of the subjects chose correctly – no more than would have if they guessed randomly, indicating that the radiosity simulation achieved a high degree of realism.

The cost of achieving this realism is, however, great. The time and memory required to generate radiosity solutions for complex environments containing many tens (or even hundreds) of thousands of surfaces can be enormous. As will be discussed in further detail in the next chapter, radiosity algorithms require the environment to be meshed into many small patches, and the exchange in energy between each *pair* of patches calculated – hence the quadratic time and space complexity of classical radiosity methods. Since each pair-wise exchange requires computing a *form-factor* (the geometric relationship between a source patch and a receiver, which includes the fraction of the source visible from the receiver), calculating the radiosity transfers is expensive, and requires knowledge about the positioning of all other surfaces.

The first radiosity methods required that a complete matrix of form-factors be built before the

[2]The original stained glass window was designed by Elsa Schmid, and was converted into the Materials and Geometry Format by Jim Arvo and Alan Heirich. It is available from *http://csvax.cs.caltech.edu/ arvo/data/StainedGlass.mgf*

radiosity solution could be solved [GTGB84]. For large scenes, the time and memory requirements to do this quickly became prohibitive. Subsequent improvements to the algorithm have include *progressive* refinement and *hierarchical* radiosity methods. Progressive refinement radiosity [CCWG88] re-orders the computation so that the complete form-factor matrix does not need to be stored, and partial solutions can be displayed quickly. Hierarchical radiosity techniques [HS90, HSA91] aim to reduce the number of form-factor computations by approximating blocks of the form-factor matrix using a single value. A fuller discussion of both of these methods is presented in the next chapter.

In summary, radiosity methods are attractive for computing realistic illumination in virtual environments due to the view independence of their solutions. With the increase in power of rendering hardware, we are wanting to visualise ever more complex environments. Even with progressive and hierarchical algorithms, producing radiosity solutions is still a time and memory consuming task. Since the computation of very large radiosity solutions is beyond the compute and memory capabilities of today's workstations, many researchers have proposed utilising the resources available on parallel computer systems [Che89, BW90, CP91, BP94, RAPP97]. Many different approaches have been proposed for performing radiosity computations on a variety of different parallel architectures (see Chapter 5), and solutions have been successfully generated for scenes containing over one million polygons [RAPP97].

1.4 Thesis Contribution

This thesis presents two contributions to generating radiosity solutions for geometrically complex scenes quickly and efficiently. Firstly, we present a new framework for radiosity simulation (based on the progressive refinement method [CCWG88]) that uses *perceptually-based calculations* to control the generation of view-independent radiosity solutions. Perceptual awareness enables computational effort to be moved away from areas that are deemed to have a visually insignificant effect on the solution's appearance, into those that are more noticeable. Perceptually-based metrics are derived using recent *tone-reproduction* techniques [TR93, War94, FPSG96], and provide a method for determining the perceptual impact of radiosity transfers during the solution process. The benefit of such perceptual techniques is that computation times can be reduced without a corresponding decrease in perceived solution quality. Perceptually-driven techniques are applied to several important areas of computation:

- Adaptive refinement – perceptual measures are used to stop the adaptive subdivision of a surface once the difference in radiosity representation between successive levels of refinement becomes perceptually unnoticeable;
- Visibility testing – the perceptual impact of a radiosity transfer is used to control the number of rays used to determine the visibility between a source and receiver;
- Simplification – the resulting view-independent radiosity mesh may need to be simplified in order to meet certain rendering or storage requirements. Perceptual awareness enables those features of the mesh that are visually unimportant to be removed before those features that are more noticeable.

For our second contribution, we present a novel algorithm for computing radiosity solutions on distributed shared-memory parallel architectures. Our approach is based on the notion of a *sub-iteration* being the transfer of energy from a single source to a subset of the scene's receiver patches. By using an efficient queue-based scheduling system to process these sub-iterations, we show how the radiosity solution can be generated without the need for processor synchronisation between iterations of the progressive refinement algorithm. This free-running method of computation eliminates the need for load-balancing until the end of the solution process, and results in an almost linear speedup in the time required to generate a solution.

1.5 Thesis Outline

The remainder of the thesis is as follows: the next two chapters lay the foundations for describing our approach to perceptually-based radiosity computation. In Chapter 2, we discuss the radiosity method in more detail, and Chapter 3 describes recent approaches to tone-reproduction, including a comparison of the most promising algorithms, and a discussion of techniques for applying tone-reproduction to view independent solutions. Following that, in Chapter 4 we introduce the framework for incorporating perceptual measures into the radiosity solution process. Several different applications of the technique are covered, including adaptive subdivision, visibility testing and mesh simplification. Many of the issues that must be raised when attempting to parallelise radiosity computations are then discussed in Chapter 5, and a new algorithm for distributed shared-memory architectures is proposed. Finally, in Chapter 6 we draw conclusions from the work presented and describe some areas raised in this research that are in need of further investigation.

Chapter 2

Radiosity Methods

Since their introduction into the computer graphics field in the mid 1980s [GTGB84, NN85], radiosity methods have received much attention by researchers eager to create ever more photo-realistic images. Radiosity algorithms aim to simulate the inter-reflection of energy between diffuse surfaces in an environment.

In this chapter, we will introduce the basic ideas and techniques used during radiosity computations, and describe many of the algorithms that have been proposed to accelerate the simulation process. Sections 2.1 and 2.2 discuss the two methods used to describe electromagnetic power and energy – *radiometry*, the science of measuring light in the electromagnetic spectrum, and *photometry*, the science of measuring light based on the sensitivity of the human eye. Following that, Section 2.3 presents the *global illumination equation* that describes the distribution of energy in an environment, and then derives the *radiosity equation* that allows a diffuse, view-independent solution to be computed. Section 2.4 briefly discusses methods for constructing the finite-element mesh required to solve the radiosity equation, and then Section 2.5 describes form-factor and visibility computations. Different approaches to generating a radiosity solution are then outlined in Section 2.6. Finally, Section 2.7 describes some of the problems associated with rendering and displaying the results of the simulation.

2.1 Radiometry

Radiometry is the most common representation of energy used in computer graphics. Energy and power are expressed in SI units of *joules* and *watts* respectively. All radiometric quantities are wavelength dependent, but to simplify the notation, this dependence is not written here explicitly. For a more complete discussion of radiometric measurements for computer graphics, see [CW93, SP94, Ash95, Gla95].

2.1.1 Radiance

The basic characterisation of the distribution of energy in a scene is *radiance*, measured in Watts per unit area per unit solid angle (or $W\ m^{-2}\ sr^{-1}$). Radiance (L) is a measure of the energy travelling from a point x, in a particular direction ω, per unit area perpendicular to that direction, per unit solid angle

$$L(x,\omega) = \int p(x,\omega,\lambda)\frac{hc}{\lambda}\,d\lambda \tag{2.1}$$

$p(x,\omega,\lambda)$ represents the density of light photons of wavelength λ at x travelling in a direction parallel to ω, and $\frac{hc}{\lambda}$ represents the energy of a single photon. As light travels along the direction ω, its radiance remains constant (assuming, as we do, that light is travelling in a vacuum). The radiance leaving a surface point x in direction $x \to y$ is therefore equal to that arriving at point y from the same direction.

2.1.2 Irradiance

Another important quantity is *irradiance*, the total radiant energy per unit area *incident* on a point. This is related to incident radiance L_i by integrating over the unit hemisphere Ω, to obtain the total radiance energy, $d\Phi$, incident at a point

$$d\Phi = \int_\Omega L_i \cos\theta \, d\omega \, dA \tag{2.2}$$

where θ is the angle between the surface normal at the point and the hemispherical direction ω. Since irradiance (E) is energy *per unit area*, it is related to $d\Phi$ by

$$E = \frac{d\Phi}{dA} \tag{2.3}$$

or equivalently

$$E = \int_\Omega L_i \cos\theta \, d\omega \tag{2.4}$$

Irradiance is also measured in $W\ m^{-2}\ sr^{-1}$.

2.1.3 Reflectance

Radiance is transported around an environment via reflection off surfaces. The amount of energy incident on a surface point x from a given *incident direction* ω_i that reflects off the surface in another *outgoing direction*, ω_o, is described by the *bidirectional reflectance distribution function* (BRDF). The BRDF (f_r) of a surface point is defined in terms of incoming and outgoing radiances at that point

$$f_r(x, \omega_i, \omega_o) = \frac{L_o(x, \omega_o)}{L_i(x, \omega_i) \cos\theta_i \, d\omega_i} \tag{2.5}$$

where θ_i is the angle between the surface normal at x and ω_i, and $d\omega_i$ is the differential solid angle in direction ω_i. The total amount of light reflected by a surface in direction ω_o is given by the integral over the hemisphere of incident directions

$$L_o(x, \omega_o) = \int_\Omega f_r(x, \omega_i, \omega_o) L_i(x, \omega_i) \cos\theta_i \, d\omega_i \tag{2.6}$$

Standard radiosity methods (see below) assume that surfaces reflect light diffusely – this means that there is an equal probability of light being reflected in any direction, regardless of the incident angle ω_i. Thus, $f_r(x, \omega_i)$ is the same for each incoming direction, and can therefore be moved outside the integral, giving

$$\begin{aligned} L_o(x) &= f_r(x) \int_\Omega L_i(x, \omega_i) \cos\theta_i \, d\omega_i \\ &= f_r(x) E(x) \end{aligned} \tag{2.7}$$

where E represents the irradiance at point x.

2.1.4 Radiosity

Radiosity (also known as radiant exitance) is a similar quantity to irradiance, but rather than representing the total energy per unit area *arriving* at a point, it corresponds to the total energy per unit area *leaving* the point after reflectance. Radiosity is defined as the integral of the outgoing radiance L_o from point x over the hemisphere of directions Ω, and is measured in Watts per square metre ($W\ m^{-2}$)

$$B(x) = \int_\Omega L_o(x, \omega_o) \cos\theta_o \, d\omega_o \tag{2.8}$$

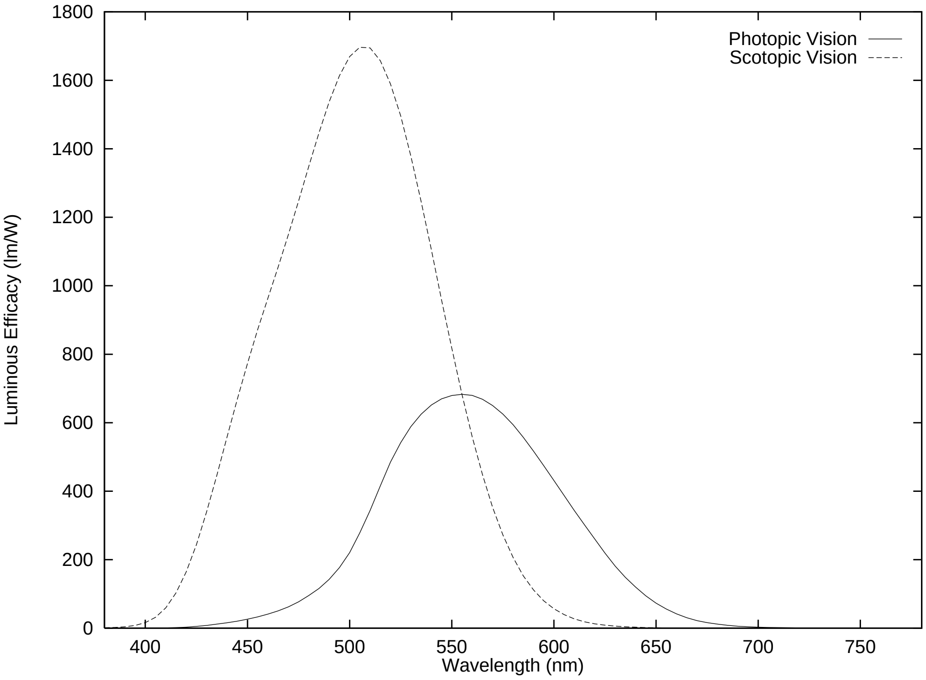


Figure 2.1: Plots of the luminous efficacy for both photopic and scotopic vision.

For diffuse surfaces, L_o becomes independent of direction, and the integral can be solved analytically

$$\begin{aligned} B(x) &= L_o(x) \int_\Omega \cos\theta_o \, d\omega_o \\ &= -L_o(x) \int_0^{2\pi} \int_o^{\pi} \cos\theta_o \, d\cos\theta_o \, d\phi \\ &= L_o(x)\pi \end{aligned} \tag{2.9}$$

From Equations 2.7 and 2.9, the relationship between radiosity and irradiance can be derived

$$B(x) = \pi f_r(x) E(x) \tag{2.10}$$

The *diffuse reflectance coefficient*, $\rho_d(x)$ expresses how much irradiance is reflected as radiosity from a diffuse surface

$$\rho_d(x) = \pi f_r(x) \tag{2.11}$$

2.2 Photometry

Photometry deals with the measurement of light weighted according to the sensitivity of the human eye [Ash95] rather than its objective physical measurement. The eye has differing relative sensitivities for light at different wavelengths, as shown by the *luminous efficacy function* for a 'standard' eye in Figure 2.1 [WS82]. Under daylight (or *photopic*) conditions the eye is most sensitive to light at a wavelength of approximately 555 nm, where the luminous efficacy takes a value of 683 lumens per Watt. For night-time (*scotopic*) conditions, maximum sensitivity is reached at 507 nm, and the luminous efficacy at this wavelength is 1700 lumens per Watt. This function can be used to calculate the perceived response of an observer to a spectral energy function $S(\lambda)$[1]

$$S_v = \int_{380nm}^{780nm} V(\lambda)S(\lambda)\,d\lambda \tag{2.12}$$

where $V(\lambda)$ represents the photopic luminous efficacy function. A similar transformation can be applied for scotopic conditions, using $V'(\lambda)$, the scotopic luminous efficacy function

$$S_{v'} = \int_{380nm}^{780nm} V'(\lambda)S(\lambda)\,d\lambda \tag{2.13}$$

2.2.1 Luminance and Illuminance

Luminance is the photometric equivalent of radiance. It is obtained by weighting spectral radiance by a luminous sensitivity function, and integrating over the visible band of the spectrum

$$L_v = \int_{380nm}^{780nm} V^*(\lambda)L(\lambda)\,d\lambda \tag{2.14}$$

where $V^*(\lambda)$ is either the photopic or scotopic luminous sensitivity function, and $L(\lambda)$ is spectral radiance at wavelength λ. There are many units for expressing luminance, although throughout this work, we will use candelas per square metre ($cd\ m^{-2}$) unless otherwise stated.

Illuminance is the photometric equivalent of irradiance. It is also measured in candelas per square metre.

$$E_v = \int_{380nm}^{780nm} V^*(\lambda)E(\lambda)\,d\lambda \tag{2.15}$$

Given these radiometric and photometric descriptions of light, we are now in a position to discuss how to compute the distribution of energy in an environment. In the following sections, radiometric expressions for this distribution will be presented. Photometric quantities will be used in Chapters 3 and 4, when the response of the human visual system is considered.

2.3 The Global Illumination Equation

To produce an image of an environment using a global illumination model of light transport, the incident radiance at each point on the view-plane of the virtual camera must be computed. Since radiance is known to be constant along a ray when light travels in a vacuum (Section 2.1.1), this is equivalent to calculating the outgoing radiance at each surface point visible to the camera. For a view-independent solution, we require an expression giving a hemisphere of outgoing radiances at all points in the scene. A derivation of what has become known as the *global illumination equation* or the *rendering equation* can be found in [CW93] that describes such an expression. The form of the rendering equation we will use here is as follows

$$L_o(x, \omega_o) = L_e(x, \omega_o) + \int_\Omega f_r(x, \omega_o, \omega_i)L_i(x, \omega_i)\cos\theta_i\,d\omega_i \tag{2.16}$$

where

- $L_o(x, \omega_o)$ is the outgoing radiance at point x in direction ω_o;
- $L_e(x, \omega_o)$ is the *self-emitted* radiance at x in direction ω_o, which will be non-zero only for light-sources;

[1] By convention, a photometric quantity is represented using the same symbol as its radiometric counterpart. In order to avoid confusion where both are discussed, the photometric quantity will be written with a v or v' subscript, depending on whether the photopic or scotopic luminous efficacy curve has been used.

- Ω is the unit hemisphere of incoming directions at x;
- $f_r(x, \omega_o, \omega_i)$ is the BRDF at x for incoming direction ω_i and outgoing direction ω_o;
- $L_i(x, \omega_i)$ is the incoming radiance at x from direction ω_i;

Put more simply, this equation states that the outgoing radiance at a point in a particular direction is equal to the self-emitted radiance in that direction, plus the sum of the incident radiance from each incoming direction over the unit hemisphere, weighted by the BRDF of the surface. Note that the unknown quantities that must be solved for (the radiances) occur inside the integral as well as on the left-hand side of the equation. Such equations are known as *Fredholm integrals of the second kind*, and are impossible to solve analytically in all but the simplest situations.

2.3.1 The Radiosity Approximation

The *radiosity method* introduces several simplifying assumptions that enable Equation 2.16 to be solved. Firstly, all surfaces are assumed to reflect light diffusely. This means that the outgoing and emitted radiances are constant over the hemisphere of outgoing directions, and the BRDF can be substituted for, as in Equation 2.11. These simplifications allow the outgoing radiance at point x to be expressed as follows

$$L_o(x) = L_e(x) + \frac{\rho_d(x)}{\pi} \int_\Omega L_i(x, \omega_i) \cos\theta_i \, d\omega_i \tag{2.17}$$

The hemispherical integral,

$$\int_\Omega L_i(x, \omega_i) \cos\theta_i \, d\omega_i \tag{2.18}$$

must then be transformed into an integral over all *surface points*, rather than directions. This can be achieved by substituting for the *differential solid angle* subtended by a point x' with area dA

$$d\omega_i' = \frac{\cos\theta_o dA}{\|x - x'\|^2} \tag{2.19}$$

where θ_o is the angle between the surface normal at x' and a vector from x' to x. Substituting this into the hemispherical integral gives

$$\int_S L_i(x, \omega_i) \frac{\cos\theta_i \cos\theta_o}{\|x - x'\|^2} V(x, x') \, dx' \tag{2.20}$$

where S is the infinite set of all surface points and $V(x, x')$ is the *visibility function* between points x and x'. The visibility function will be 1 if the points are mutually visible, and 0 if there is an occluding surface between them. Equation 2.9 can then be used to transform radiances into radiosities, resulting in the *diffuse global illumination equation*

$$B(x) = B_e(x) + \rho_d(x) \int_S B(x') \frac{\cos\theta_x \cos\theta_{x'}}{\pi\|x - x'\|^2} V(x, x') \, dx' \tag{2.21}$$

2.3.2 The Finite-Element Approach

Solving complex integral equations such as that in Equation 2.21 requires that the function space defined by the equation be projected down to a finite dimension by evaluating the unknown variables at a number of discrete locations in the environment. The *finite-element* approach [Zie89, HW91b] proceeds by first constructing a *mesh* of N patches[2] over all surfaces in the environment (see Figure 2.2). Classical radiosity methods assume that each patch P_i has radiosity B_i, reflectance ρ_i and emitted radiosity B_{e_i}, all constant across its area (this has several implications regarding the accuracy of the solution that are

[2]We use the term *patch* to denote an initial part of the mesh. The term *element* will be used later when we discuss adaptive subdivision methods to represent part of a subdivided patch.

discussed in more detail in [TL91]). The radiosity of P_i is equal to the area-weighted integral of radiosity values across the patch

$$B_i = \frac{1}{A_i} \int_{x \in P_i} B(x)\, dx \tag{2.22}$$

where A_i is the area of patch P_i. The diffuse global illumination equation may now be reformulated (or more accurately, approximated) as a sum over all *patches* rather than surface points

$$B_i = B_{e_i} + \rho_i \sum_{j=1}^{N} B_j \frac{1}{A_i} \int_{x \in P_i} \int_{x' \in P_j} \frac{\cos\theta_x \cos\theta_{x'}}{\pi \|x - x'\|^2} V(x, x')\, dx\, dx' \tag{2.23}$$

The double integral in Equation 2.23 is dependent only on the geometry of the environment. By denoting this integral as F_{ij}, where

$$F_{ij} = \frac{1}{A_i} \int_{x \in P_i} \int_{x' \in P_j} \frac{\cos\theta_x \cos\theta_{x'}}{\pi \|x - x'\|^2} V(x, x')\, dx\, dx' \tag{2.24}$$

the *radiosity equation* may finally be written as

$$B_i = B_{e_i} + \rho_i \sum_{j=1}^{N} F_{ij} B_j \tag{2.25}$$

This equation expresses the radiosity of patch P_i in terms of its emitted radiosity B_{e_i}, plus some fraction of the radiosity of every other patch P_j, weighted by the diffuse reflectance of P_i, and defines an equilibrium that all patch radiosities must satisfy in order for the solution to be considered as being solved. The quantity F_{ij} is known as the *form-factor*, and expresses the fraction of energy leaving patch P_i that arrives at patch P_j. It may seem strange that in Equation 2.25, the total radiosity of patch P_i is dependent on the fractions of energy *leaving* P_i and *arriving* at other patches. A more intuitive representation can be obtained by weighting the form-factor by patch area, obtaining an expression that deals with the transport of *power* rather than energy

$$A_i F_{ij} = \int_{x \in P_i} \int_{x' \in P_j} \frac{\cos\theta_x \cos\theta_{x'}}{\pi \|x - x'\|^2} V(x, x')\, dx\, dx' \tag{2.26}$$

Note that when expressed in this way, the form-factor is *reciprocal* [HS67]

$$\forall (i, j) \;\; A_i F_{ij} = A_j F_{ji} \tag{2.27}$$

This can be interpreted as the fraction of power that leaves patch P_i and arrives at patch P_j being equal to the fraction that leaves P_j and arrives at P_i. Now, by expressing the radiosity equation in terms of power

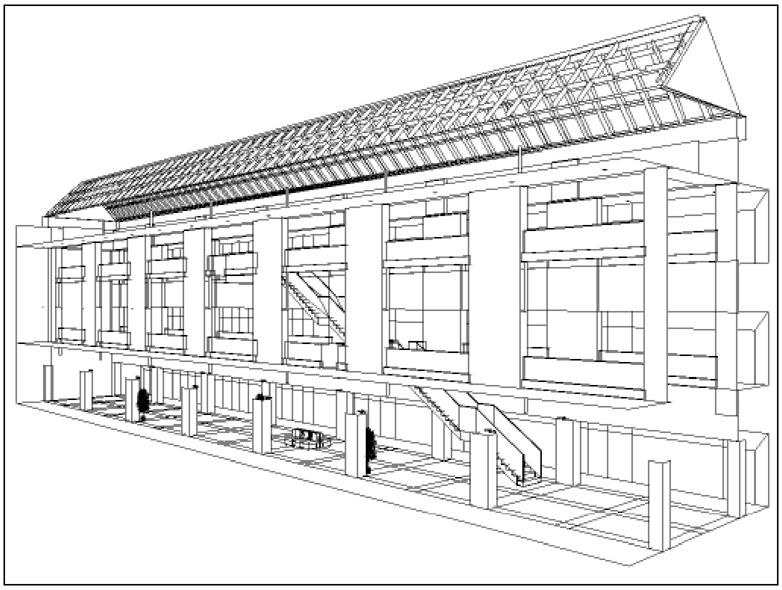
$$A_i B_i = A_i B_{e_i} + \rho_i \sum_{j=1}^{N} A_i F_{ij} B_j \tag{2.28}$$

the reciprocity relationship can be used to obtain

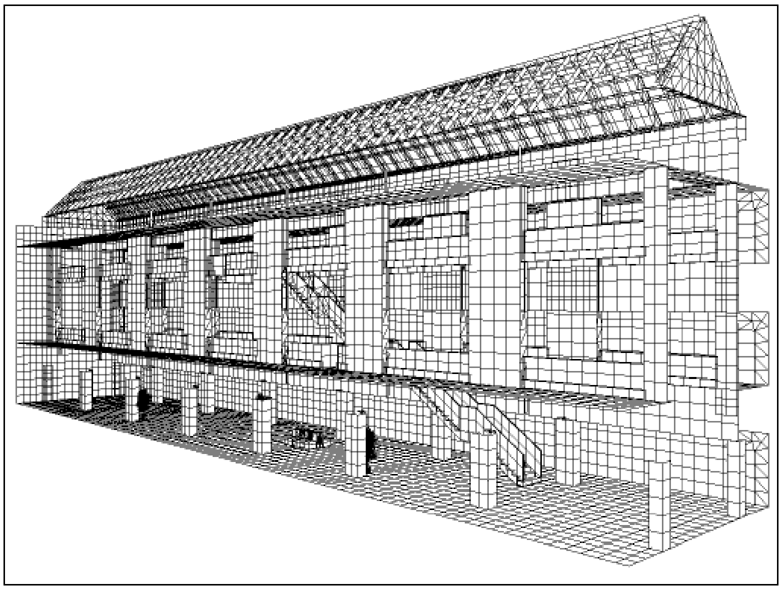
$$A_i B_i = A_i B_{e_i} + \rho_i \sum_{j=1}^{N} F_{ji} (A_j B_j) \tag{2.29}$$

Equation 2.29 has a more intuitive meaning than Equation 2.25; the total power at a patch P_i is equal to the emitted power, plus the sum of the fraction of power of all other patches that reaches P_i, weighted by the diffuse reflectance. Equations 2.29 and 2.25 are equivalent, and either may be used to produce a radiosity solution. Following the conventions set by others [CW93], we will work from the radiosity-based expression given in Equation 2.25.

In order to solve Equation 2.25, all surfaces in the environment must first have a mesh of patches assigned to them. Techniques for producing such a mesh are briefly discussed in the next section. Following that, approaches to calculating form-factors and algorithms for solving the radiosity equation will be presented.



(a) Scene geometry



(b) Patch mesh

Figure 2.2: A patch mesh for a moderately complex scene.

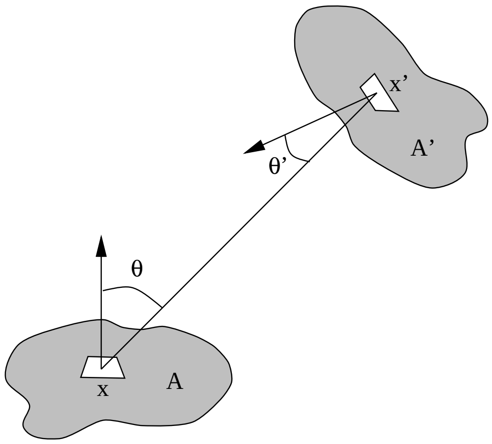


Figure 2.3: Area-to-area form-factor geometry.

2.4 Meshing

Finite-element methods rely on the construction of a *mesh* of patches representing the geometry of the environment. A typical patch mesh for the radiosity method is shown in Figure 2.2(b)[3]. The final accuracy of illumination can depend heavily on the underlying mesh representation, and so it is worth taking care in its generation. [CW93] contains a useful summary of meshing algorithms, and how they relate to the radiosity method.

Typical mesh generation algorithms tessellate surfaces into quadrilateral or triangular patches. Quadrilateral polygons may be meshed into a two-dimensional grid of patches, the size of each patch specified in terms of maximum area or maximum side length. Triangles may be meshed in a similar way, typically into either a lattice of triangular patches, or a mixture of quadrilaterals and triangles. More complex polygons are usually broken down into simpler triangles or quadrilaterals, and each is meshed separately.

Radiosity values may be computed at either the vertices of the mesh [WEH89], or at the patches themselves. Computing patch radiosities requires an interpolation step before the solution can be rendered, since radiosity values are required at the mesh vertices in order to perform the bilinear interpolation necessary to obtain a smooth image [CW93]. This means that certain topological information must be maintained in the mesh. Structures such as a *winged-edge data structure* [Bau75] are usually employed, as they enable information regarding the connectivity of faces, edges and vertices in the mesh to be retrieved quickly and efficiently, and also allow for easy adaptive subdivision of mesh elements (see Section 2.6.2).

2.5 Form-Factors

The form-factor (or configuration factor as it is known in the radiative transfer literature) describes the geometric relationship between two surfaces. It is purely a function of geometry, and does not depend on either the material characteristics of the surfaces or their illumination.

Recall the expression that relates the fraction of the energy leaving a patch P_i that arrives at patch P_j

$$F_{ij} = \frac{1}{A_i} \int_{x \in P_i} \int_{x' \in P_j} \frac{\cos\theta_x \cos\theta_{x'}}{\pi \|x - x'\|^2} V(x, x')\, dx\, dx' \tag{2.30}$$

The geometry for this *area-to-area* form-factor expression is shown in Figure 2.3. Both analytic and numerical approximations have been used to compute F_{ij} (a more complete survey of which is given

[3]This atrium model was created by the University of Aizu, Japan, and is available from `http://wwwsv1.u-aizu.ac.jp/%7Ek-myszk/atrium/`

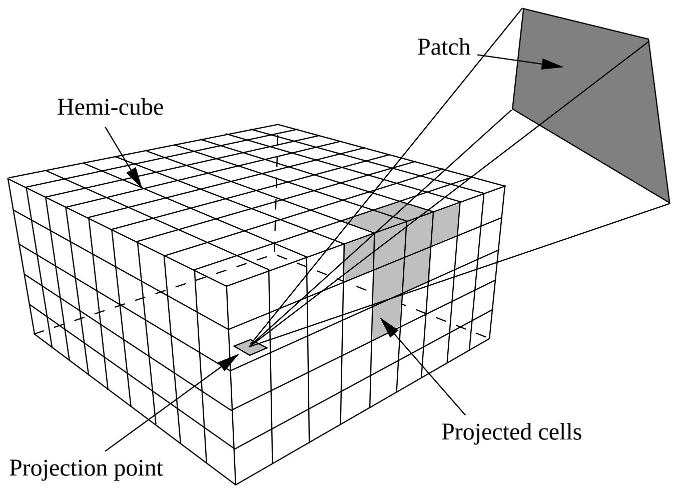


Figure 2.4: Projection of a patch onto the faces of a hemicube.

in [CW93]), but in general, closed form solutions for the full area-to-area expression are complex. A complete analytic solution for mutually visible polygons has been derived by Schröder and Hanrahan [SH93], but it involves evaluating complex and expensive dilogarithmic functions. Because form-factor evaluations often consume a large proportion of the time required to generate a radiosity solution, such costly calculations are of little practical use.

Many early radiosity algorithms used environment projection methods that approximate the form-factor from a point to each patch in the environment by finding the projection of the patch onto a unit hemisphere centered around the surface point, with its pole pointing in the same direction as the surface normal. The *hemicube* [CG85] was proposed as a way of approximating this hemispherical projection as the projection onto a half cube. Using the accelerated graphics hardware that is present in modern workstations, the patches in the environment are scan converted and z-buffered into the hemicube (Figure 2.4), each face of which is discretised into a number of cells that store the identification number of the nearest patch. A delta form-factor is associated with each cell, and the delta form-factors of the cells onto which each patch projects are summed to calculate the total form-factor to that patch.

Hemicube methods require fast and efficient graphics hardware to be used effectively. They are also subject to severe aliasing artifacts due to the limited resolution of each face, causing a noticeable pattern to occur in the resulting radiosity representation. Patches that are smaller than the size of a cell may also be missed altogether [BRW89]. Baum *et al.* discuss the limitations of hemicube methods in more detail in [BRW89]. Because of these problems, as well as the difficulty of applying hierarchical techniques (see Section 2.6.3) to hemicube based algorithms, projection-based form-factor methods will not be discussed further.

2.5.1 Differential Area-To-Area Form-Factors

The inner integral in Equation 2.30 corresponds to the form-factor between a *differential area* and a finite area

$$F_{xj} = \int_{x' \in P_j} \frac{\cos\theta_x \cos\theta_{x'}}{\pi \|x - x'\|^2} V(x, x')\, dx' \tag{2.31}$$

This represents the fraction of energy leaving a differential area around point x that arrives at patch P_j. If random points x_k for $k = 1 \dots n$ are chosen on P_i, and the differential area-to-area form-factors for these points can be found, then the full area-to-area form-factor may be approximated by numerical integration

$$F_{ij} \approx \frac{1}{n} \sum_{k=1}^{n} F_{x_k j} \tag{2.32}$$

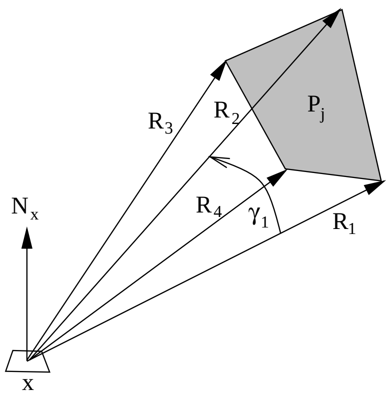


Figure 2.5: Differential area-to-polygon form-factor geometry.

Differential Area-To-Polygon Form-Factor

For planar polygons that are entirely visible to the point x, contour integration may be used to solve Equation 2.31 exactly [BRW89, HS67]. Given the set of edges G of P_j, the vector R_g from x to the start of edge g, and the angle γ_g between vectors R_g and R_{g+1}, define Γ_g as a vector with direction $R_g \times R_{g+1}$ and size γ_g. The differential area-to-area form-factor is then

$$F_{xj} = \frac{1}{2\pi} \sum_{g \in G} n_x \cdot \Gamma_g \tag{2.33}$$

where n_x is the surface normal at point x (Figure 2.5).

Differential Area-To-Disc Form-Factor

A cheaper (but less accurate) differential area-to-area form-factor is obtained when P_j is approximated by a disc of area equal to that of the patch. The analytic form-factor from x to a directly opposing disc of area A_j is [SH92]

$$F_{xj} = \frac{A_j}{\pi r^2 + A_j} \tag{2.34}$$

where r is the distance between x and the centre of the disc. Alternatively, when the differential area is not directly opposite the disc, the form-factor can be approximated using

$$F_{xj} = \frac{A_j \cos\theta \cos\theta'}{\pi r^2 + A_j} \tag{2.35}$$

The geometry for the differential area-to-disc form-factor is given in Figure 2.6.

Form-Factor Subdivision

The disc approximation is only valid when the distance between the differential area and the patch is large relative to the area of the patch. In situations when this is not the case, aliasing artifacts appear [WEH89], and the patch must be subdivided (possibly recursively) until the distance criterion is satisfied (Figure 2.7). Once subdivided, the form-factor to each *delta area* is summed to generate a more accurate result.

2.5.2 Visibility

In all but the simplest of environments, occluding objects affect the form-factor between a pair of patches, and greatly increase the complexity of the form-factor computation if accurate results are required.

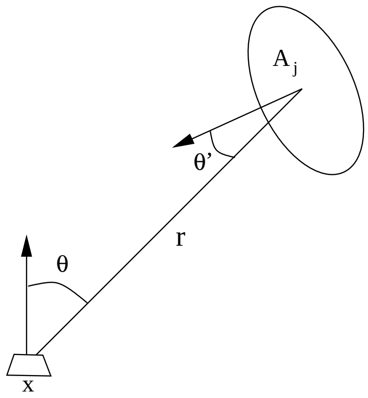


Figure 2.6: Differential area-to-disc form-factor geometry.

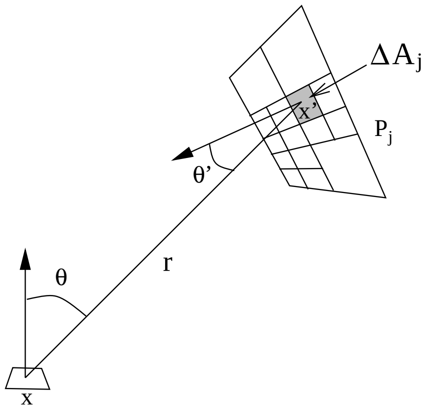
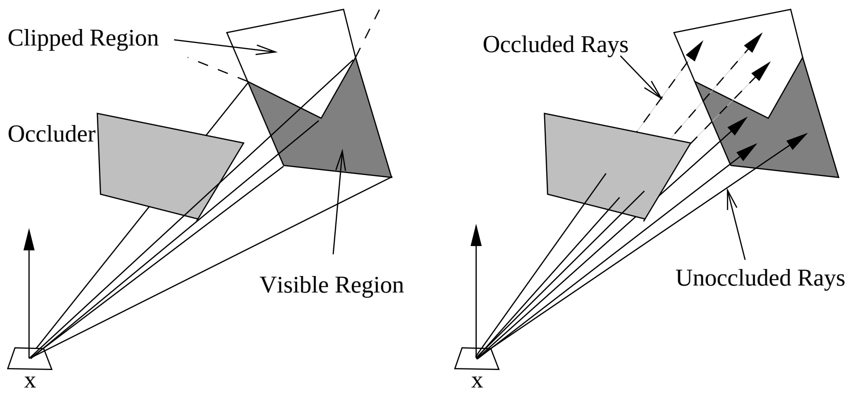


Figure 2.7: Subdivision of a source patch into delta areas.



(a) Analytic (b) Approximate

Figure 2.8: Accounting for occlusion during form-factor computation.

Analytic Methods

If the analytic differential area-to-polygon form-factor is being used, then the part (or parts) of the source that are visible to the point around which the differential area is centred may be identified, and a form-factor evaluation applied to each part (Figure 2.8(a)). This approach was used by Nishita and Nakamae [NN85] in their original radiosity paper. Using their method, the form-factor to the original source polygon is first computed, and then all other polygons are clipped against the source. The form-factor to each clipped part is subtracted from the total to yield the correct result.

This method of accounting for occlusion will always generate correct results for the differential area-to-area form-factor, but is very expensive since each form-factor requires $O(N)$ clipping operations, where N is the number of polygons in the scene. The method is also limited to scenes composed of planar polygons. *Shaft culling* [HW91a] may be employed to reduce the list of candidate polygons that must be clipped against the source, but the complexity of implementation and the fact that no trade-off may be made between computation time and form-factor accuracy limits the appeal of this technique.

Ray Casting

A more cost-effective approach to accounting for occlusion between patches is to approximate visibility using *ray-tracing* [WEH89, HSA91]. A number of rays are cast between two patches, and intersected with the other objects in the scene. The fraction of one patch that is visible from another may then be estimated by counting the fraction of rays that do not intersect any other objects (Figure 2.8(b)).

The ray-tracing method of estimating visibility has a number of attractive features. Firstly, a trade-off between accuracy and computation time can be made – reducing the number of rays will decrease the time required to evaluate the form-factor, but will also decrease its accuracy. Secondly, the process of ray-tracing can be greatly accelerated using *spatial-subdivision structures* such as hierarchical (or recursive) uniform grids [CDP95], octrees [Gla84, FTI86], or bounding-volume hierarchies [GS87]. Finally, when high-level primitives such as spheres and cylinders are present in the scene, ray-tracing algorithms may perform intersection calculations on these primitives directly, rather than deal with the (possibly huge) number of polygons into which they are tessellated.

There are two approaches to using ray-traced visibility estimates in form-factor calculations. Wallace *et al.* use ray-tracing to identify the parts of the source patch that are visible to a differential area, and sum delta form-factors to each visible part [WEH89] (as in Section 2.5.1). Considering Figure 2.7, a ray is cast from the differential area to the centre of each delta area, and the total form factor evaluated as

$$F_{xj} = \sum_{k=1}^{n} \frac{\Delta A_k \cos\theta_k \cos\theta'_k}{\pi r_k^2 + \Delta A_k} V(x, x_k) \tag{2.36}$$

where patch P_j is subdivided into $k = 1 \ldots n$ delta areas, each having area ΔA_k, and $V(x, x_k)$ represents the visibility between x and a point x_k inside area ΔA_k. An alternate method that is potentially faster, but also less accurate, is to evaluate the unoccluded form-factor and visibility estimate separately, and then multiply the two together to obtain an estimate of the true form-factor [HSA91].

Regardless of how form-factors are computed, a solution method is still required to compute the final radiosity values for each patch in the scene. In the following section, several different approaches will be introduced, each of which has its advantages and disadvantages. Following that, approaches to producing a final image of the radiosity solution will be discussed.

2.6 Solution Techniques

The radiosity equation (Equation 2.25) implicitly expresses the radiosity of a patch P_i in terms of the radiosity of all other patches P_j. In order to generate a complete radiosity solution, the radiosity of each patch must be calculated so that this expression is satisfied. A system of N linear equations may be established from Equation 2.25, with N unknown variables representing the patch radiosities [GTGB84]

$$\begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{bmatrix} = \begin{bmatrix} B_{e_1} \\ B_{e_2} \\ \vdots \\ B_{e_N} \end{bmatrix} + \begin{bmatrix} \rho_1 F_{11} & \rho_1 F_{12} & \cdots & \rho_1 F_{1N} \\ \rho_2 F_{21} & \rho_2 F_{22} & \cdots & \rho_2 F_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_N F_{N1} & \rho_N F_{N2} & \cdots & \rho_N F_{NN} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{bmatrix} \tag{2.37}$$

or, equivalently,

$$\begin{bmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \cdots & -\rho_1 F_{1N} \\ -\rho_2 F_{21} & 1-\rho_2 F_{22} & \cdots & -\rho_2 F_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ -\rho_N F_{N1} & -\rho_N F_{N2} & \cdots & 1-\rho_N F_{NN} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{bmatrix} = \begin{bmatrix} B_{e_1} \\ B_{e_2} \\ \vdots \\ B_{e_N} \end{bmatrix} \tag{2.38}$$

Equation 2.38 may be expressed in a more compact form,

$$\mathbf{KB} = \mathbf{E} \tag{2.39}$$

where $\mathbf{K} = \mathbf{I} - \rho\mathbf{F}$ is the matrix containing the form-factors, ρ is a column vector of patch reflectivities, $\mathbf{B}$ is a column vector of all patch radiosities, and $\mathbf{E}$ a column vector of emissivities. Standard matrix inversion techniques could be used to obtain $\mathbf{B}$ [GTGB84] (provided $\mathbf{K}$ is non-singular) by solving

$$\mathbf{B} = \mathbf{K}^{-1}\mathbf{E}$$

although for complex scenes, the storage required to keep the matrix in memory and the complexity of Gaussian elimination (which requires $O(n^3)$ operations for an $n \times n$ matrix) will soon render this method impractical. Given standard inversion methods are of little practical use, *iterative* (or *relaxation*) methods are applied that start with an initial guess (in this case, the vector of emissivities, $\mathbf{E}$), and refine the estimates of the radiosity values until the error in the result is to within a certain tolerance.

There are two main relaxation methods employed in solving radiosity equations – Gauss-Seidel, and Southwell, each of which has different convergence properties and different storage requirements. Each method starts with an *initial guess*, $\mathbf{B}^{(0)}$ for the unknown vector $\mathbf{B}$. At iteration k, the error is defined as $\mathbf{e} = \mathbf{B}^t - \mathbf{B}^{(k)}$ where $\mathbf{B}^t$ is the vector of radiosities we require. Since measuring error in this way is impossible due to the fact that $\mathbf{B}^t$ is not known, the *residual* at iteration k,

$$\mathbf{r}^{(k)} = \mathbf{KB}^{(k)} - \mathbf{E} \tag{2.40}$$

is used to represent the error in practice.

$\forall i \;\; B_i = B_{e_i}$
while (not converged) {
    /* perform a single iteration */
    for (each i)
        $B_i = B_{e_i} + \rho_i \sum_{j=1,j\neq i}^{n} F_{ij} B_j$
    display partial solution
}

Figure 2.9: Pseudo-code for Gauss-Seidel radiosity.

The vector of radiosities $\mathbf{B}^{(k)}$ at the current iteration k is refined by examining the residual vector, and transforming an element of $\mathbf{B}^{(k+1)}$ in such a way that the corresponding element in the residual vector becomes zero. Clearly, if all elements in $\mathbf{r}^{(k)}$ are zero, then we have found a solution since

$$\mathbf{KB}^{(k)} - \mathbf{E} = 0 \tag{2.41}$$

Of course, setting one residual entry to zero may affect others, and many iterations may be required before all entries become small enough that the solution can be considered converged.

2.6.1 Full-Matrix Methods

The earliest algorithms for solving systems of radiosity equations became known as *full-matrix* methods due to the requirement that the entire matrix of form-factors ($\mathbf{K}$) be calculated before the solver could begin. The *Gauss-Seidel* algorithm performs n steps during each iteration, updating entry i of $\mathbf{B}$ at each step by relaxing residual entry r_i

$$B_i^{(k+1)} = \frac{E_i}{K_{ii}} - \sum_{j=1}^{i-1} K_{ij} \frac{B_j^{(k+1)}}{K_{ii}} - \sum_{j=i+1}^{n} K_{ij} \frac{B_j^{(k)}}{K_{ii}} \tag{2.42}$$

The Gauss-Seidel method can be thought of as *gathering* radiosity from the environment onto each patch. At each iteration of the algorithm, all of the scene's patches are visited in turn and considered as receivers of energy. For each receiver, every other patch is considered as a source of energy, and the fraction of each source's radiosity that reaches the current receiver is added to the receiver's total radiosity.

$$\begin{bmatrix} \\ B_i \\ \\ \end{bmatrix} = \begin{bmatrix} \\ B_{e_i} \\ \\ \end{bmatrix} + \begin{bmatrix} \\ \rho_i F_{i1} & \rho_i F_{i2} & \cdots & \rho_i F_{iN} \\ \\ \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{bmatrix} \tag{2.43}$$

Pseudo-code for Gauss-Seidel Relaxation is shown in Figure 2.9. Each iteration corresponds to a 'bounce' of light throughout the environment, and requires access to all $O(n^2)$ form-factors. Once these form-factors have been computed, however, they may be re-used in later iterations since they are independent of the radiosity of a patch. The iterative process is repeated until some *convergence criterion* is reached. For the Gauss-Seidel algorithm, the solution is usually considered converged once the maximum change in any radiosity value between iteration k and $k+1$ ($\|\mathbf{B}^{(k+1)} - \mathbf{B}^{(k)}\|_\infty$) falls below a user-specified threshold value [CW93].

Full-matrix methods are severely limited in the complexity of environment they can handle. A moderately large environment containing 50,000 patches would require over 4.65 GBytes of memory and (we estimate) 23 days to compute and store the entire form-factor matrix[4]. Clearly, if complex radiosity

[4]These estimates are based on experiments performed using a 100MHz MIPS R4400 CPU, using 4 bytes per floating-point form-factor value.

```
∀i  B_i = ΔB_i = B_{e_i}
while (not converged) {
    /* perform a single iteration */
    select j with largest A_j ΔB_j
    for (each i) {
        Δb = ρ_i F_ij ΔB_j
        ΔB_i = ΔB_i + Δb
        B_i = B_i + Δb
    }
    ΔB_j = 0
    display partial solution
}
```

Figure 2.10: Pseudo-code for progressive-refinement radiosity.

solutions are to be computed in a practical amount of time and memory, full-matrix methods are not a viable option.

2.6.2 Progressive-Refinement Radiosity

In 1988, Cohen *et al.* introduced the *progressive refinement* (PR) radiosity algorithm that uses a variation of the Southwell relaxation method to solve the system of radiosity Equations [CCWG88]. Instead of gathering radiosity onto a single receiver, an iteration in the PR algorithm consists of *shooting* radiosity from a single source patch to the rest of the environment.

As well as an 'accumulated' radiosity (B_i), each patch also stores an amount of 'unshot' radiosity, ΔB_i, that corresponds to the residual entry r_i. At each iteration, the patch j with the largest unshot power ($A_j \Delta B_j$) is selected as the current source, and that power is distributed to the other patches in the environment

$$\begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{bmatrix} = \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{bmatrix} + \begin{bmatrix} \Delta B_j \end{bmatrix} \begin{bmatrix} \rho_1 F_{1j} \\ \rho_2 F_{2j} \\ \vdots \\ \rho_N F_{Nj} \end{bmatrix} \tag{2.44}$$

The unshot power of the current source patch is then set to zero, and the process is repeated until convergence. Convergence criteria for the progressive-refinement method are usually based on the sum total of unshot power ($\sum_i A_i \Delta B_i$), with convergence achieved when this sum is less than a user-specified percentage of the total initial power in the environment ($\sum_i A_i B_{e_i}$). For a complete proof of the convergence properties of progressive refinement radiosity see [GCS93]. Pseudo-code for the progressive-refinement algorithm is shown in Figure 2.10.

Progressive-refinement radiosity has several important advantages over gathering algorithms such as Gauss-Seidel. Firstly, one iteration of the progressive-refinement algorithm computes the illumination due to a single patch on the rest of the environment. Several such iterations may be required before the effect of a single gathering iteration has been achieved, but since each iteration only requires the computation of $O(n)$ form-factors, useful (albeit not fully converged) solutions may be displayed much earlier on in the solution process.

The second important aspect of the algorithm is that the form-factor matrix need not be held in memory. Each form-factor may be computed only when it is needed, meaning that the size of environments that may be handled using this method is far greater than for full-matrix techniques. Of course, if a patch is selected to shoot its energy multiple times, the form-factors to that patch will need to be recomputed each time, but this extra cost is far out-weighed by the reduction in storage space.

Ambient Correction

In the initial stages of the progressive refinement algorithm, only a small percentage of the total power in the environment will have been distributed. Any image of early solutions will therefore be much darker than the final, converged solution. Cohen [CCWG88] proposed that an *ambient correction* term, approximating the energy in the environment that has yet to be distributed, can be added to the radiosity of each patch when it is being displayed. The average unshot radiosity in the environment is given by

$$\overline{\Delta B} = \frac{\sum_i \Delta B_i A_i}{\sum_i A_i} \tag{2.45}$$

During each iteration, the average amount of energy that is reflected is related to the average reflectivity of the environment

$$\overline{\rho} = \frac{\sum_i \rho_i A_i}{\sum_i A_i} \tag{2.46}$$

On average, $\overline{\rho}$ will be reflected during one iteration, $\overline{\rho}^2$ during the next iteration, $\overline{\rho}^3$ during the third, and so on. The total *reflection factor* for the infinite number of iterations required to produce a fully converged solution is therefore given by

$$R_t = 1 + \overline{\rho} + \overline{\rho}^2 + \overline{\rho}^3 + \ldots = \frac{1}{1 - \overline{\rho}} \tag{2.47}$$

An estimate for the average amount of radiosity that a patch will receive during the iterations that have not yet been performed is simply the product of the average unshot radiosity and the reflection factor

$$B_a = \overline{\Delta B} R_t \tag{2.48}$$

For display purposes only, the radiosity of each patch is increased to account for this extra energy

$$B_i^{disp} = B_i + \rho_i B_a \tag{2.49}$$

The ambient energy, B_a must be recalculated after each iteration. In this way, the correction added to each patch decreases towards zero as the solution progresses towards convergence.

Figure 2.11 shows a sequence of images after 1, 10 and 100 iterations. The images on the right were rendered without any ambient correction, while those on the left include the ambient radiosity estimate. This example clearly shows the improvement in image quality obtained in the early stages of the solution process.

Adaptive Refinement

The accuracy and computation time of a radiosity solution depends heavily on the number of patches in the environment. Since radiosity will be interpolated between the vertices of each patch, details such as small shadows and high radiosity gradients may not be represented accurately. Figure 2.12 plots a one-dimensional slice of typical radiosity patch representation against an exact reference. The reference solution was obtained by evaluating the analytic point-to-polygon form-factor at a large number of points along the slice. Due to the large size of patches relative to the size of the features in the function, the radiosity is not represented adequately. Although there are points (such as around C) where the approximation is good, many features are missing.

Decreasing the size of the patches will obviously increase the accuracy of the resulting radiosity function, since the density at which the radiosity function is sampled will be increased. Figure 2.13 shows the same reference function as before, along with a patch mesh whose density has been increased four-fold. The radiosity function created by this mesh is better than that in Figure 2.12 – the radiosity gradient at B is improved, but there are still areas where the sampling is inadequate (A). Comparing the radiosity function in the smooth area around point C shows that we have also increased the sampling density in areas that did not require it.

(a) 1 iteration

(b) 10 iterations

(c) 100 iterations

Figure 2.11: Sequence of images computed with (left) and without (right) the ambient radiosity correction.

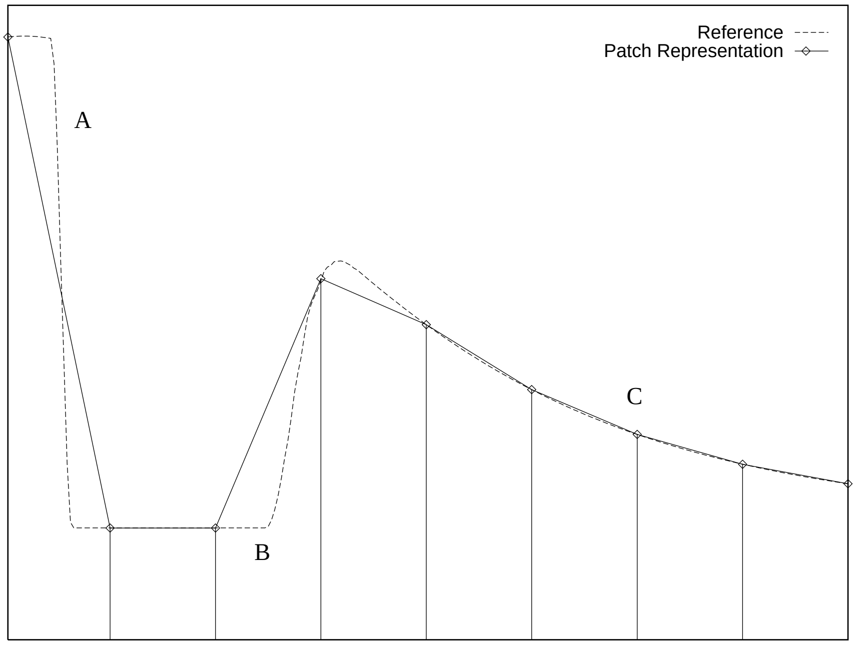


Figure 2.12: Radiosity representation using a patch mesh.

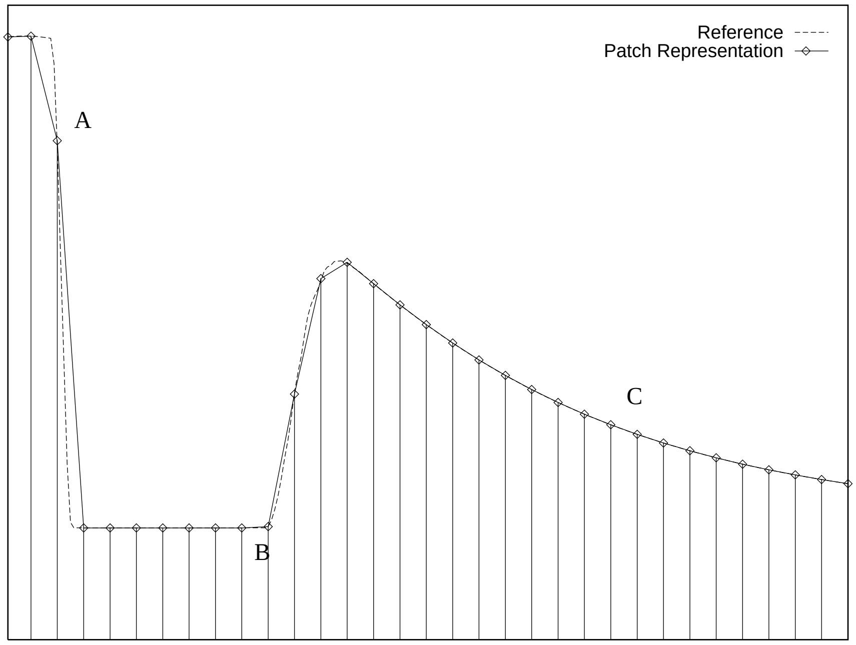


Figure 2.13: Radiosity representation using a finer patch mesh.

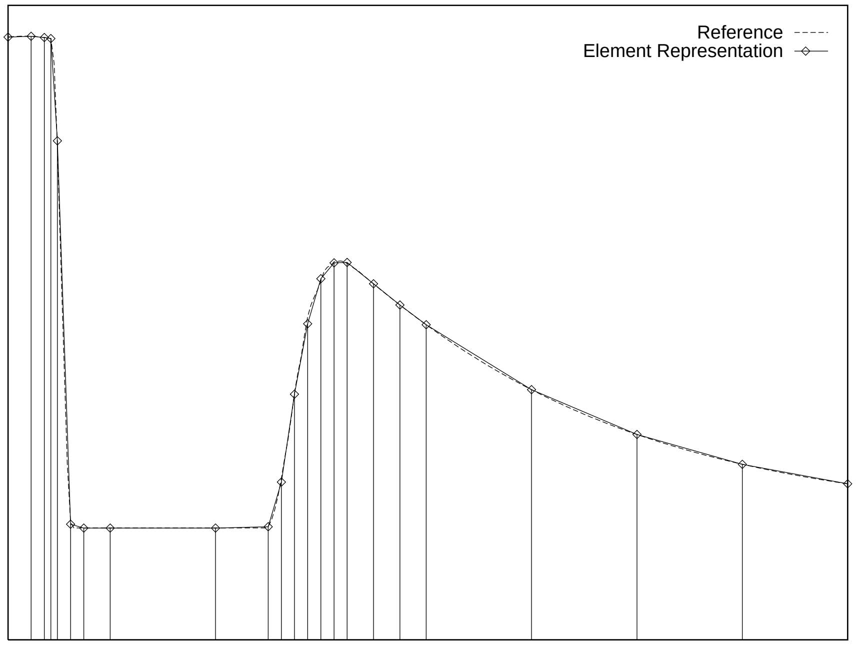


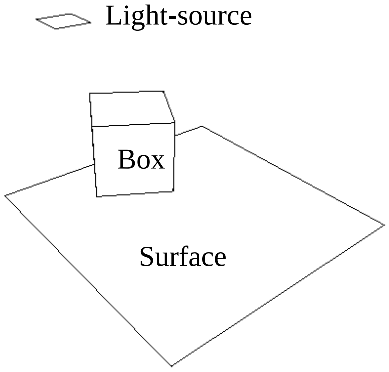
Figure 2.14: Radiosity representation using an adaptive element mesh.

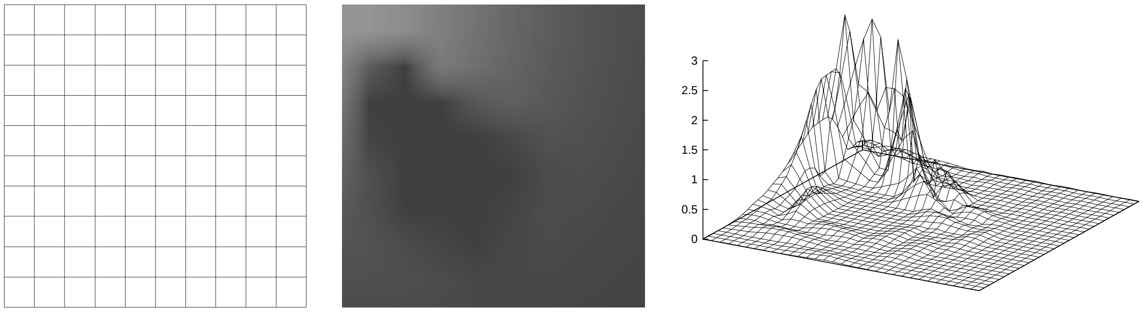
Unfortunately, uniformly decreasing the size of *all* patches also means that a great deal more iterations are required to compute the solution, since more source patches must shoot their energy. Cohen *et al.* [CGIB86] introduced a *sub-structuring* method that maintained a two-level hierarchy of patches and smaller *elements*. After each patch has been subdivided into elements, the initial patch representation is used only when the patches are considered as sources of illumination, and a patch receives radiosity at each element. In this way, a more accurate radiosity function may be calculated over each receiver without the cost of performing more iterations of the solution.

A natural extension to this method of representing receivers at a finer level of detail than sources is to control the density of elements inside each patch *adaptively* [CGIB86]. This allows complex areas such as shadow boundaries to be represented with a suitably large number of elements, whereas other areas that do not exhibit such features may be represented at a lower level of subdivision. Figure 2.14 shows the same radiosity function as before, this time represented using adaptive subdivision. Adaptive subdivision is usually performed by constructing a *quadtree* of elements over each patch, and an *error estimate* which is used to check if the current level of element subdivision is satisfactory.

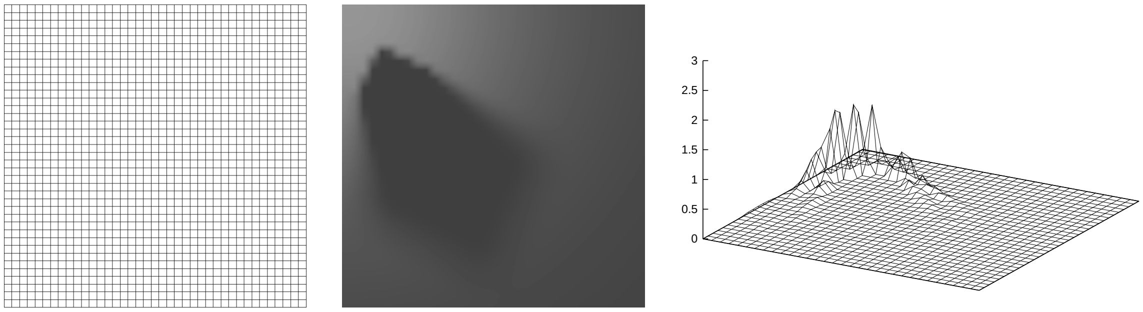
In Figure 2.15, we compare the radiosity representations computed over a two-dimensional surface containing a shadow cast by a box and a single area light-source. For each type of refinement, the element mesh is shown on the left, a shaded image of the resulting radiosity function is shown in the middle, and on the right, a graph is given of the absolute error in radiosity (measured in $W\ m^{-2}$) when compared to an exact reference function. As can be seen by comparing (b) and (c), adaptive refinement clearly outperforms the two-level sub-structuring method, since a more accurate representation of radiosity it produces using fewer elements.

Many refinement oracles have been proposed in the radiosity literature. Based on the requirement that radiosity be constant across an element, Cohen *et al.* [CGIB86] used the difference between the radiosities of an element and its neighbours as the error estimate. In this case, refinement is triggered whenever the radiosity difference is greater than a user-specified threshold. This refinement heuristic favours smaller elements in areas where large radiosity gradients exist, such as along shadow boundaries. Vedel and Puech also use differences in the gradient of the radiosity function as an additional error estimate [VP91]. This avoids the situation when the difference in radiosity is high, but the gradient is constant and the

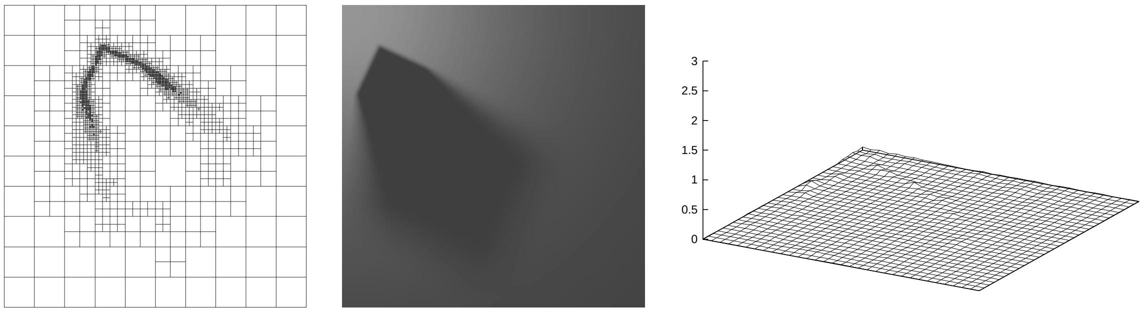




(a) No refinement (332 patches)



(b) Two-level hierarchy (5,312 elements)



(c) Adaptive refinement (3,354 elements)

Figure 2.15: The improvement in accuracy of the radiosity representation when using adaptive refinement.

radiosity function is therefore well represented using bilinear interpolation. Another (more expensive) approach to adaptive subdivision is to evaluate the radiosity function inside the elements, and measure the differences between the interpolated function at these points, and the newly calculated value [Cam91, LTG91, LTG92].

2.6.3 Hierarchical Methods

Both sub-structuring and adaptive subdivision have recently been unified into an elegant approach to radiosity simulation known as *hierarchical radiosity* [HS90, HSA91]. Using similar ideas to those developed for solving the *N-body* problem [App85, BH86, Gre88], hierarchical methods allow many insignificant form-factor calculations to be approximated using a single evaluation.

N-Body Simulation

Traditional N-body algorithms involve calculating the total gravitational force affecting each of N particles, where each particle exerts a force on all other $N-1$ particles. Just as in classical radiosity methods, a naive solution requires evaluating N^2 interactions. As recognised by Appel [App85], several assumptions may be made that enable a faster algorithm to be developed. Firstly, since all numerical simulations are subject to a level of error, the force between two particles need only be calculated to within a certain precision. Secondly, given that a certain error is allowed in the calculations, all forces between two well separated clusters of particles may be replaced using a single interaction, provided that that interaction is a good representation of all the forces it approximates.

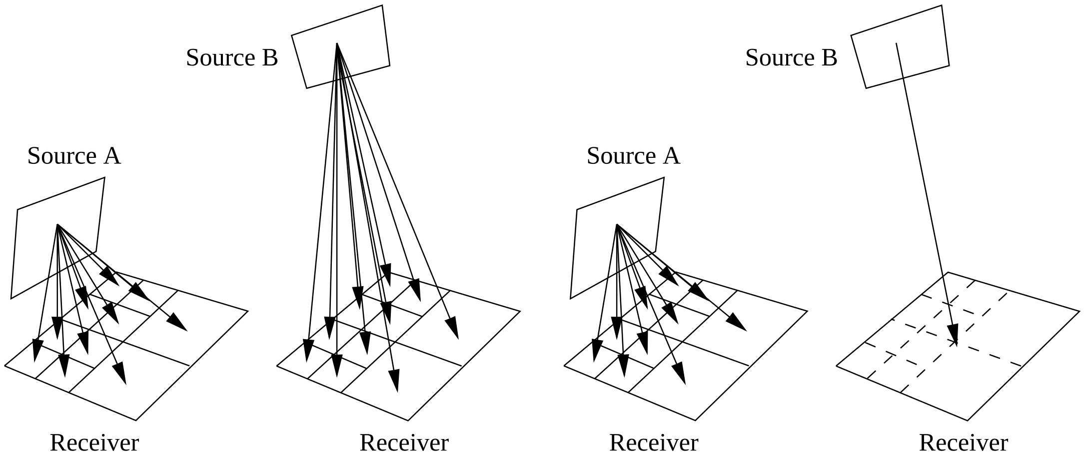
Although similarities exist, there are several differences between the radiosity algorithm and N-body simulations. Both form-factors and gravitational forces fall off with the inverse-square of distance, but the gravitational force between two bodies is unaffected by the positions of other particles. Unlike N-body calculations, form-factors for radiosity simulation are affected by occlusion, and so their calculation requires knowledge about the entire scene. Additionally, instead of recursively clustering particles into groups, hierarchical radiosity methods work in reverse, subdividing large initial surfaces into many smaller elements.

Hierarchical Radiosity

Hierarchical radiosity methods explicitly represent the transport of light throughout an environment using a collection of *links* between elements in multi-level hierarchies created over each surface. When considering the exchange of energy between two surfaces, a level of element representation is chosen for each surface such that the error in radiosity transfer between the surfaces is below a user-specified tolerance. Unlike the simple adaptive subdivision algorithms discussed above, both receivers and sources may be subdivided if required, and it is not necessary to perform any initial meshing.

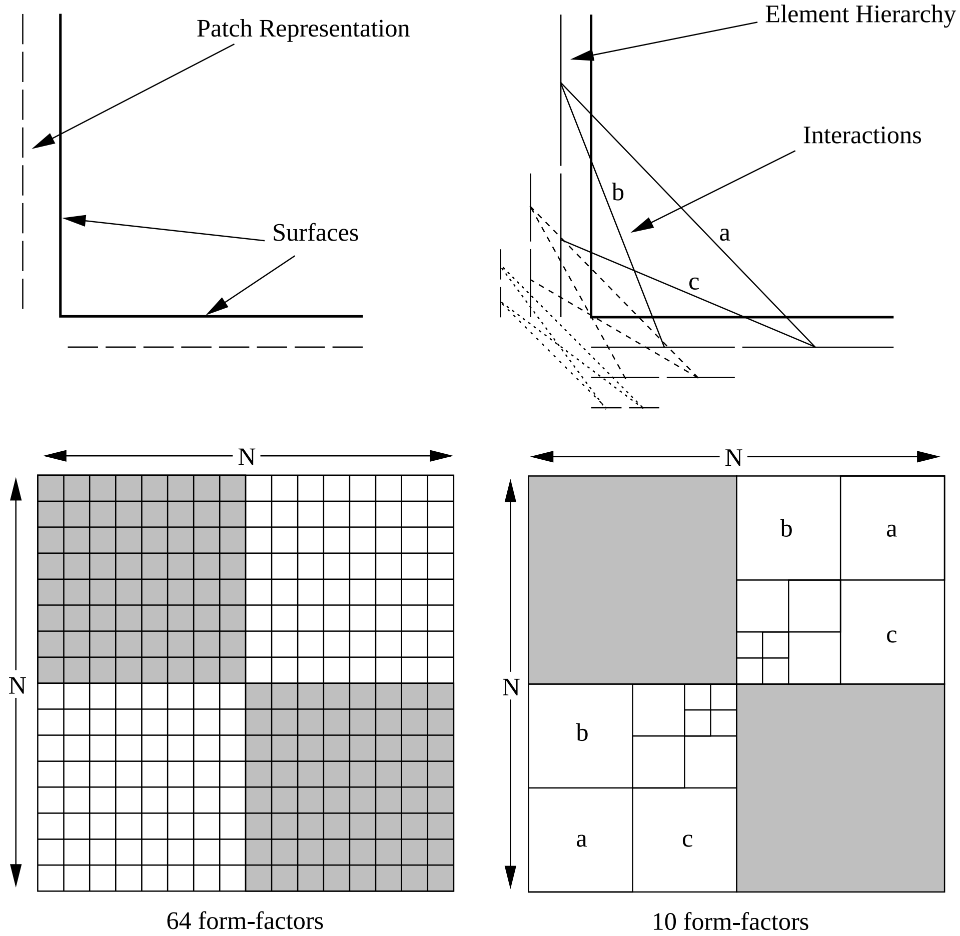
When computing form-factors to the elements of receiving patch using standard adaptive subdivision techniques, the number of form-factors that are calculated depends upon the level of subdivision that is used. Consider the situation shown in Figure 2.16(a). A nearby source patch (A) transfers its radiosity to the elements of the receiver using the level of refinement shown in the diagram. During a later transfer, when patch B distributes its radiosity, it must calculate form-factors to all of the receiver's elements. The radiosity transfer from the distant patch is being calculated to a far greater accuracy than that from the nearby patch. In contrast, the hierarchical algorithm represents all transfers to the same level of error. The transfer from the distant patch may in fact be adequately represented using a single form factor to the entire receiver (Figure 2.16(b)), saving many form-factor calculations.

In terms of the form-factor matrix, hierarchical radiosity methods are capable of approximating large sections of the matrix where the form-factors are similar using a single calculation. Figure 2.17 demonstrates how the form-factor matrix is constructed for a simple one-dimensional pair of perpendicular surfaces. In the top-left is shown the two surfaces divided into a uniform patch mesh, as occurs when full-matrix or progressive radiosity algorithms are used. Underneath is the corresponding 16x16 form-factor matrix. All entries that are shaded require no calculation, since the corresponding source and



(a) Adaptive Subdivision (b) Hierarchical Subdivision

Figure 2.16: Transfer of radiosity from a nearby and distant source to a receiver using both adaptive and hierarchical techniques.



(a) Full-matrix radiosity (b) Hierarchical radiosity

Figure 2.17: Comparison of form-factor matrix for full-matrix and hierarchical radiosity methods.

```
Refine_Interaction (p, q) {
    error = Oracle (p → q)
    if (error is acceptable)
        Create_Link (p → q)
    else {
        if (exists a link p → q)
            Delete_Link (p → q)
        if (Preferred_Subdivision (p, q) == p) {
            Subdivide_If_Necessary (p)
            for (each child c of p)
                Refine_Interaction (c, q)
        }
        else {
            Subdivide_If_Necessary (q)
            for (each child c of q)
                Refine_Interaction (p, c)
        }
    }
}
```

Figure 2.18: Pseudo-code for hierarchical refinement.

receiver patches belong to the same planar surface and therefore cannot possibly exchange energy. In total, there are 128 non-zero form-factors, but due to reciprocity, only 64 must be computed.

On the right of Figure 2.17 is shown the same pair of surfaces, as would be represented using the hierarchical algorithm. Starting with a single interaction between each pair of top-level surfaces, a recursive refinement procedure (pseudo-code for which is given in Figure 2.18) is applied to each interaction, taking as arguments the two surfaces, p and q, it connects. At the current level of subdivision, a *refinement oracle* is queried to see if the radiosity transfer between p and q is represented accurately. If so, a *link* is created between the elements p and q. Otherwise, one of the elements is chosen for subdivision (usually the one that is considered to have the largest impact on the error), and the refinement procedure is called recursively for each newly created child.

Referring back to Figure 2.17, we can now see that the number of form-factors that are computed using hierarchical refinement has fallen from 64 to 10. In this case, the top-level interaction was refined, creating interactions a, b and c. The interaction between the two elements that share a vertex was considered inaccurate, however, and the refinement continued with these elements. In total, 10 links were created, with links a, b and c each representing groups of 16 form-factors.

Figure 2.19(a) shows pseudo-code for the hierarchical radiosity solution process. Initially, a single link is created between each pair of surfaces. These links are then repeatedly refined, and radiosity gathered across them, until a convergence criterion is reached. The *Gather_Radiosity* procedure is shown in Figure 2.19(b)[5]. Since links may be created at any level in the element hierarchy, care must be taken that each level maintains a correct amount of radiosity. This requires that the radiosity gathered across a link to an element must be *pushed* down to the leaves of the hierarchy, summed, and then *pulled* back upwards [HS90]. During the upwards traversal, the radiosity of an element is set equal to the area-averaged radiosity of its children.

Schröder [Sch94b] gave a detailed proof that for levels below the top-level in the element hierarchy, each element interacted with at most a constant number of other elements. At the top-level, an interaction must be tested for with each of the other surfaces in the environment. This gives hierarchical radiosity

[5]This code differs from Hanrahan's original implementation [HS90, HSA91] in that Gauss-Seidel iteration is used, rather than the less efficient Jacobi method

```
for (each surface p) {
    B_p = B_{e_p}
    for (each surface q ≠ p)
        Create_Link (p, q)
}

while (not converged) {
    for (each surface p) {
        for (each link l of p)
            Refine_Interaction (p, q)
        Gather_Radiosity (p, 0)
    }
}
```

(a) Solution process

```
Gather_Radiosity (p, B_down) {
    for (each link l of p)
        B_down = B_down + ρ_p F_pq B_q
    if (p has children) {
        B_p = 0
        for (each child c of p) {
            Gather_Radiosity (c, B_down)
            B_p = B_p + B_c A_c / A_p
        }
    }
    else
        B_p = B_{e_p} + B_down
}
```

(b) Gather/Push-Pull procedure

Figure 2.19: Pseudo-code for hierarchical Gauss-Seidel algorithm.

an overall complexity of $O(k^2 + n)$ for k surfaces and n elements [Sch94b].

Many different refinement oracles have been proposed in the literature [HS90, HSA91, LSG94, Gib95, GH96, BW96, SSS97]. Hanrahan's original approach [HS90] used *F-only refinement* – the error estimate for an interaction is assumed to be bounded above by the magnitude of the unoccluded differential area-to-area form-factor (Section 2.5.1). This presumes that the actual error is proportional to this value. Unnecessary subdivision is often produced by this method due to the singularity in the form-factor expression where two surfaces meet at a vertex, since the size of the form-factor is dependent only the *relative* positions and orientations of the elements, so subdividing does not reduce the error. In [HSA91], a technique known as *BF refinement* was proposed, where the error estimate is based on the product of the form-factor and the amount of radiosity being transferred from the source element. This method focuses refinement into those areas that contain a high level of illumination. Another variant, *BFA refinement*, bases error estimates on the product of the form-factor, the source radiosity and the receiver area, thereby tolerating larger errors for small receiver elements.

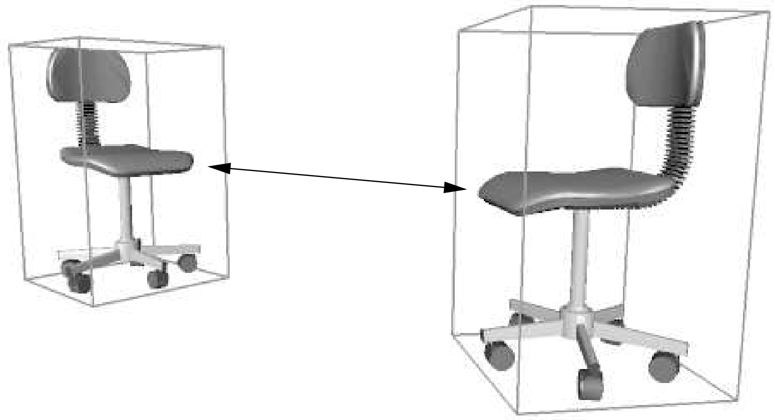
Situations can occur, however, when a relatively large radiosity transfer can be performed without a correspondingly large error. BF refinement will not be able to detect this, however, and will always refine the interactions. Recently, *error bounding* algorithms have been proposed that attempt to calculate upper and lower bounds on the error in the radiosity transfer. The most complete approaches calculate bounds for each component of the radiosity transfer equation, and model the transport of these bounds around the environment, as well as the transport of radiosity [LSG94, BW96]. Less conservative methods that ignore the accumulation of error and generate bounds only for the current transfer [Gib95, GH96, SSS97].

Hierarchical algorithms are also capable of resolving element-to-element visibility queries more efficiently, using a technique known as *visibility coherence* [HSA91]. A visibility estimate must be obtained at each level during the traversal of the element hierarchy in order to calculate a form-factor. If at any time it is found that a receiving element is completely visible to a source patch, then no more visibility calculations need be performed for any children of the receiver. Similarly, if a receiving element is found to be completely invisible, no more refinement is necessary.

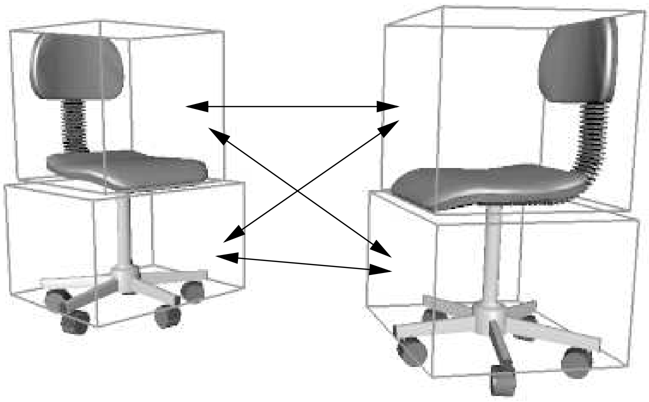
Clustering

The original hierarchical algorithm approximates the form-factors between groups of elements, but recent research has focussed on allowing interactions between collections of surfaces (known as *clusters*) to be grouped together [Kok93, RPV93, SAG94, Sil95, Gib95, GH96, CLSS95, MK95, SSS97].

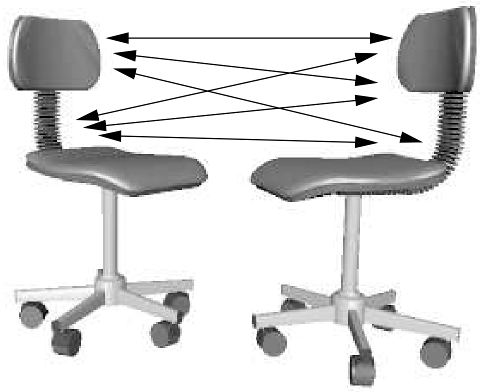
Figure 2.20 shows two complex objects, each composed of many thousands of small polygons. Stan-



(a) Well separated interactions approximated using a single link



(b) Interactions occur at lower levels for closer objects



(c) Close objects require interactions at the polygon level

Figure 2.20: Clustering of interactions between well separated groups of surfaces.

dard hierarchical methods [HSA91] will initially generate $O(k^2)$ links between all k polygons, each link representing the transfer of a very small amount of radiosity between two polygons (Figure 2.20(c)). When the objects are well separated (as in (a) and (b)), clustering algorithms allow form-factor and visibility calculations to be performed between groups of surfaces, and the radiosity transfers between these groups represented using a smaller number of *cluster links*.

The earliest clustering algorithms were based on progressive refinement radiosity. In [RPV93], Rushmeier *et al.* approximated the reflectance properties of clusters of polygons by simple functions obtained using Monte-Carlo sampling techniques. A coarse radiosity simulation was then performed on the simplified environment, and a final image produced using ray-tracing methods, with the indirect component of illumination approximated using the radiosity solution. Kok [Kok93] also proposed an algorithm for performing progressive-refinement radiosity using groups of surfaces, although both of these methods required that the hierarchy of clusters be constructed by hand, and neither attempted to control the error introduced by the simplifications.

Recently, Myszkowski and Kunii [MK95] described a clustering algorithm based on progressive-refinement radiosity that included an error-estimate for refining cluster interactions. Although only a single level of clusters was allowed, the luminance incident on a receiver was used as an error estimate, and a cluster-to-cluster transfer permitted if the luminance fell below a certain threshold. If the incident luminance was above the threshold, interactions were computed between the patches inside the clusters using standard methods.

Extensions to hierarchical radiosity algorithms have also been made to allow automatic surface clustering [SAG94, Sil95, Gib95, GH96, CLSS95, SSS97]. A hierarchy of clusters (similar in appearance and construction to a bounding-volume hierarchy [KK86, GS87]) was first generated, with the root node of the hierarchy containing the entire environment. The costly *initial linking* phase of the algorithm, where the $O(k^2)$ links between pairs of initial surfaces are generated was avoided completely by creating a single link from the root of the cluster hierarchy to itself. Error estimation techniques, based on either the variation in kernel function over the clusters [SAG94, CLSS95], BF-refinement algorithms [Sil95] or error-bounding techniques [Gib95, GH96, SSS97], were applied to refine the cluster links, with standard hierarchical radiosity taking over once the leaves of the cluster hierarchy were reached. Gauss-Seidel based gathering techniques were employed to solve for the surface radiosities once the hierarchy of links was created.

2.7 Rendering

The fundamental difference between the use of radiosity methods in rendering applications and the more traditional heat-transfer analysis problems is that the simulation process does not finish when the surface radiosities are computed. The final stage of the rendering process occurs when the solution or image is displayed and viewed by a human observer.

When an observer looks at a real environment, the variation in light arriving at his eyes is transformed into a mental image of what is being viewed. Ultimately, the goal of all global illumination methods is to be able to synthesise this distribution of light so that when presented to the viewer, the mental image formed is the same as that created by the real environment. As noted by Chiu and Shirley [CS94], if a physically based renderer is being used and a perfect display device were available then this would be a trivial process. Unfortunately, display devices currently in use such as monitors, large projector screens and head-mounted displays are far from perfect, since the range of radiance values that are calculated cannot be properly displayed, and so we can only hope to create images that are approximately correct.

The problem of displaying the distribution of energy as calculated by a global illumination algorithm on a display device such as a CRT screen, is both complex and fundamental to the entire image synthesis process, and is discussed in detail in the next chapter. Assuming that some transformation exists to map radiosity values to display intensities that are suitable for display, there are several approaches that can be taken to generate an image of the environment.

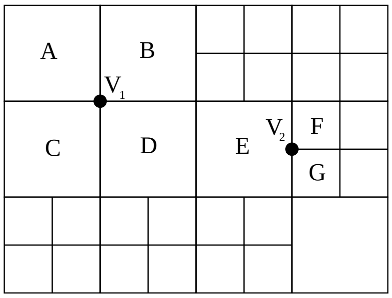


Figure 2.21: The calculation of vertex radiosities by averaging adjacent element radiosities.

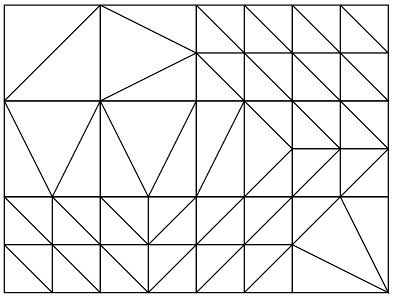


Figure 2.22: A Triangulation of the element mesh to allow for hardware rendering.

2.7.1 Hardware Rendering

Recently developed high-performance graphics workstations are capable of displaying over ten million shaded polygons per second [MBD97]. Given the view-independent nature of radiosity solutions, it is therefore natural to utilise such rendering power to generate a *walk-through* of the illuminated scene.

Unless radiosity values have been calculated directly at the vertices [WEH89], element radiosities must be averaged in order to generate suitable vertex intensities for rendering. For uniformly subdivided meshes (such as around vertex V_1 in Figure 2.21), the vertex radiosity is simply the average of the radiosities of the surrounding elements

$$B_{V_1} = \frac{1}{4}(B_A + B_B + B_C + B_D) \tag{2.50}$$

Whenever a radiosity value is required at a T-vertex (V_2), care must be taken that the total amount of energy received by the vertex is maintained, and so an area-weighted average over the adjoining elements is required

$$B_{V_2} = \frac{B_E A_E + B_F A_F + B_G A_G}{A_E + A_F + A_G} \tag{2.51}$$

Once radiosity values are known at all vertices in the environment, the element mesh associated with each surface must be triangulated and then rendered. Triangulation is required due to rotational variance when Gouraud shading a quadrilateral polygon in image space [CW93] (Figure 2.22). For elements without T-vertices (such as A), where one of two triangulations is possible, connecting the opposite vertices which have the most similar display intensities gives the best results as the element diagonal will run more closely to the colour gradient [ARJ90].

2.7.2 Ray Tracing

If hardware rendering is not available, ray-tracing techniques (see Chapter 1) may also be used to generate an image. Rather than applying a local illumination model at each point of intersection, a radiosity value

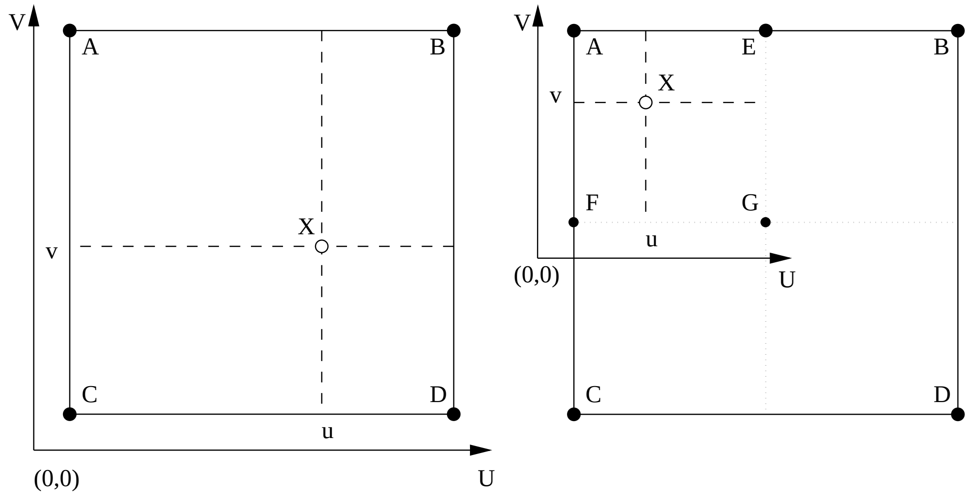


Figure 2.23: Bilinear interpolation.

is obtained from bilinear interpolation of the intersected element's vertex radiosities. Considering the element shown on the left of Figure 2.23, the radiosity at intersection point X is given by

$$B_X = (1-u)vB_A + uvB_B + (1-u)(1-v)B_C + u(1-v)B_D \qquad (2.52)$$

where u and v are the parametric coordinates of the intersection point, in the range $u, v \in [0,1]$ over the element, and B_A, B_B, B_C and B_D are the radiosities of the corner vertices. Since shadows have already been accounted for during the radiosity solution, no shadow rays must be traced, and rendering is much faster than with standard ray-tracing.

This method of image generation requires no triangulation, since bilinear interpolation is performed in object space, and so is rotationally invariant. T-vertices must still be accounted for, however, to prevent 'cracks' appearing between adjacent elements with different levels of subdivision. This is most easily done by temporarily increasing the subdivision of an element containing a T-vertex by one level, and performing bilinear interpolation in whichever child element contains the intersection point. The right hand diagram in Figure 2.23 shows an element (A, B, C, D) with a T-vertex at position E. Increasing the level of subdivision over this element introduces temporary vertices at F and G with radiosities $B_F = \frac{1}{2}(B_A + B_C)$ and $B_G = \frac{1}{4}(B_A + B_B + B_C + B_D)$ (note that A, B, C and D are vertices in this case, not elements). Bilinear interpolation is then performed using the temporary element (A, E, F, G) rather than its parent (A, B, C, D).

2.7.3 Final-Gather

For high quality static images, the accuracy of shadows and shading gradients may be increased by evaluating the final bounce of light at pixel-level accuracy [Rus88, Rei92, LTG93]. These techniques use information computed during the radiosity solution to compute an accurate measure of the illumination at each pixel in the image – in effect performing a final iteration (or *final-gather*) of the radiosity simulation onto pixel-sized elements. As both form-factor and visibility calculations must be performed for each pixel to each major contributor of radiosity, final-gather algorithms are very expensive, often taking longer to generate that the radiosity solutions themselves [LTG93].

Rushmeier proposed an algorithm that traced multiple rays through each pixel and into the scene, casting one ray from each intersection point to a random position on a light-source, and another in a random direction. The contribution from the light-source was used to calculate the direct component of illumination for the pixel, and the radiosity solution was queried at the intersection point of the second ray to provide an estimate of the indirect component [Rus88]. Since Monte-Carlo techniques were used to select the directions of each ray, noisy images resulted unless a large number of rays were cast for each pixel. Kok and Jansen [KJ92] also used stochastic techniques to evaluate the illumination from the most important sources of light at the intersection point seen through each pixel. A similar, although much slower method was described by Reichert [Rei92] where a hemicube was employed to gather radiosity from each patch in the environment. Lischinski *et al.* used the links created during the hierarchical radiosity simulation to select the elements from which to gather energy [LTG93].

2.8 Summary

In this chapter, we have introduced the radiosity method, and discussed several algorithmic improvements that lead to faster solution times and better quality images. Aside from lengthy solution times, one problem common to all the techniques presented so far is that no attempt is made to quantify the impact of any errors on the quality of the *image* that is presented to the user. This requires incorporating measures and calculations into the solution process that attempt to model how the radiosity function is perceived by an observer. The first stage in such an algorithm involves transforming the radiosity function in such a way that it can be displayed on a monitor screen or other device, and this process, known as *tone-reproduction*, is dealt with in the next chapter. Following that, in Chapter 4 we show how such a transformation can be used to measure perceptual differences in the radiosity solution. This enables *visually accurate* radiosity solutions to be generated quickly and efficiently.

As the complexity of environments with which we wish to perform radiosity simulations increases, even the fastest algorithms can still take many hours to generate a complete solution. Parallel processing offers a method of accelerating radiosity computations even further, provided that suitably scalable algorithms can be developed. Utilising the large amount of memory available on many parallel architectures will also allow the size of scenes to be increased beyond the memory restrictions of workstations. In Chapter 5, we will introduce a novel parallel algorithm that has been designed to provide these capabilities.

Chapter 3

Tone-Reproduction

3.1 Introduction

The generation of an image using a global illumination algorithm can be considered as a three stage process: the calculation of the distribution of energy throughout a scene, followed by the calculation of the energy incident on each point of the virtual camera's image plane, and finally, the translation of that energy into an image suitable for display on a device such as a monitor screen.

The first stage has received a great deal of attention by the rendering community, and the radiosity methods described in the previous chapter are good examples of recent developments. A camera model combined with traditional rendering algorithms such as ray-tracing or Z-buffer hidden-surface removal are often applied during the second stage. Until recently, the final stage, known as *tone-reproduction*, had been glossed over or ignored completely, even though it is essential to the production of realistic images. More often than not, the goal of image synthesis algorithms is to generate an image of a scene that is *as close as possible in appearance* to a view of the real scene. This final stage can therefore be seen as one of transforming the image luminances calculated by a global illumination algorithm (and measured in physical units), into a limited set of discrete frame-buffer values, in such a way that the *subjective appearance* of the scene is maintained.

The luminance range that may be experienced in everyday life (and hence may also be calculated by a global illumination algorithm) can vary over a range of 10^{14} $cd\ m^{-2}$ [FPSG96]. Table 3.1 shows some typical background luminances for outdoor scenes [IES84]. Even indoor scenes illuminated by artificial light may contain a wide range of luminance values, varying sometimes over as much as four orders of

Background	Luminance ($cd\ m^{-2}$)
Moonless overcast night	10^{-5}
Moonless clear night	10^{-4}
Moonlit overcast night	10^{-3}
Moonlit clear night	10^{-2}
Deep twilight	10^{-1}
Twilight	1
Very dark day	10
Overcast day	10^{2}
Clear day	10^{3}
Day with sunlit clouds	10^{4}

Table 3.1: Typical background luminances for the sky viewed at the horizon under various illumination conditions.

magnitude [DM97]. Typical display devices are only capable of producing luminances in a much smaller range – CRT monitors, for example, are limited to luminances between 2 to 100 $cd\ m^{-2}$ [TR93]. This raises several problems when attempting to display images of scenes with a high dynamic range on such limited devices. Firstly, it becomes obvious that the luminances calculated by a global illumination algorithm cannot be displayed directly, and a transformation is required to generate a set of display luminances that convey the same subjective impression as the real scene. Secondly, due to the fact that the maximum displayable luminance on a CRT screen is not enough to induce effects such as flare and bloom, the apparent brightness of individual pixels will not match the brightness of the real luminances. Flare corresponds to the multi-coloured 'streaks' that apparently emanate from bright light-sources, and bloom (also referred to as 'veiling luminance') is the 'glow' around bright objects [SSZG95]. Important visual cues such as these that indicate the intensity of an object will therefore be missing from the final image[1].

Most tone-reproduction operations take the form of a transformation from the luminance calculated by a global illumination algorithm, L_w, to a value n that is to be stored in the frame-buffer

$$n = \mathcal{T}(L_w) \tag{3.1}$$

where $\mathcal{T}()$ is the tone-reproduction operator.

When an image is displayed on a CRT, the value present in the frame-buffer at a pixel is directly proportional to the voltage of the electron gun used to illuminate the screen[2]. The relationship between the resulting *luminance* of the pixel, L_d, and the electron gun voltage, V, however, is non-linear

$$L_d = kV^{\gamma} \tag{3.2}$$

where k is a constant of proportionality and γ takes a value of between 2.2 and 2.5, depending on the monitor being used [FvDFH90]. The *actual* luminance that is produced for a frame-buffer value of n (in the range $[0 \ldots 1]$) is

$$L_d = L_{amb} + L_{dmax} n^{\gamma} \tag{3.3}$$

where L_{amb} is the ambient luminance on the CRT screen and L_{dmax} is the maximum displayable luminance (typically around 100 $cd\,m^{-2}$). The exact value of L_{amb} is affected by the level of illumination in the surrounding environment, the reflectance of the screen, and internal reflections inside the CRT [TR93]. The presence of the ambient display luminance means that the dimmest pixel that can be displayed will have a non-zero intensity, even when the frame-buffer value is zero. The ratio of ambient to maximum displayable luminance is known as the *display contrast*, C_{max}. A typical value for C_{max} under normal office viewing conditions is around 35 [TR93].

The earliest tone-reproduction mappings were simple, ad-hoc methods, transforming luminance values in such a way that the smallest and largest luminances in the image were mapped to the smallest and largest displayable luminances on the CRT. Image luminances in the range $[L_{wmin} \ldots L_{wmax}]$, where L_{wmin} is the smallest luminance in the image, and L_{wmax} is the largest, were transformed into display luminances in the range $[L_{amb} \ldots L_{dmax}]$ using a tone-reproduction operator we will denote by $\mathcal{T}_{MAXL}()$

$$\mathcal{T}_{MAXL}(L_w) = \left[\frac{(L_w - L_{wmin})}{(L_{wmax} - L_{wmin})} \right]^{\frac{1}{\gamma}} \tag{3.4}$$

Figure 3.1(a) shows the result of this transformation applied to an image of a computer laboratory. Almost all of the image is very dark because the largest luminance (a light-source that is directly in view, approximately 4200 $cd\ m^{-2}$) is much greater than the luminance of any other surface (an average of around 50 $cd\,m^{-2}$). To improve the quality of the final image, an alternative is to map the largest *reflected*

[1] Spencer *et al.* have recently described a method whereby flare and bloom effects may be added to an image as a post-process [SSZG95]. Since we wish to generate view-independent radiosity solutions and render them using graphics hardware, such methods are not directly applicable.

[2] We will assume here that a monochrome image is being displayed on a grey-scale CRT. Tone-reproduction for colour images will be discussed in Section 3.6.

(a) Application of $\mathcal{T}_{MAXL}()$ (b) Application of $\mathcal{T}_{REFL}()$

Figure 3.1: Comparison of ad-hoc mappings for luminance display.

luminance (the largest luminance that is not a light-source) to the maximum displayable luminance, rather than the largest overall luminance, giving rise to a tone-reproduction operator, $\mathcal{T}_{REFL}()$

$$\mathcal{T}_{REFL}(L_w) = \min\left(1, \left[\frac{(L_w - L_{wmin})}{(L'_{wmax} - L_{wmin})}\right]^{\frac{1}{\gamma}}\right) \tag{3.5}$$

where L'_{wmax} represents the largest non-light-source image luminance. Figure 3.1(b) shows the result of this transformation. The largest reflected luminance in this case is 290 $cd\ m^{-2}$, meaning that the average luminance is mapped to a higher frame-buffer value, and the resulting image is brighter.

Even though acceptable results are often produced, the $\mathcal{T}_{REFL}()$ operator is not suitable for displaying images whenever a *perceptually-accurate* reproduction of the scene is required. If the intensities of light-sources in the scene are reduced, we expect that the resulting image will get darker. Using $\mathcal{T}_{REFL}()$ or $\mathcal{T}_{MAXL}()$ as described above will always result in the *same* image being displayed. This is because the luminances of surfaces in the scene are a linear function of the initial luminances of the light-sources. No matter what scaling is applied to the light-source emissivities, the ratios in Equations 3.4 and 3.5 will remain constant, and the image will not change.

To illustrate this, Figure 3.2 shows three images of the same scene, with light-source luminances adjusted to give average image luminances of 1000, 10, and 0.1 $cd\ m^{-2}$. The simple ad-hoc mapping (results of which are shown on the left) always produces identical images that do not faithfully represent what a real-world observer sees. What we require is a *perceptually-based* tone-reproduction mapping (as has been used to generate the images on the right) that reproduces the expected changes in overall brightness, contrast and visibility.

In this chapter, we will discuss several methods for tone-reproduction that have been published in the literature [CHS$^+$93, TR93, Sch94a, War94, FPSG96, LRP97], most of which employ models of various characteristics of the human visual system. These techniques will form the basis of our approach to incorporating perceptual knowledge into the radiosity simulation, discussed in Chapter 4. In the next section, we present some of the features of our visual system that are important if we wish to obtain a perceptually accurate image. In Section 3.3, we then examine the process of tone-reproduction in more detail. Following that, Sections 3.4 and 3.5 describe some different approaches to tone-reproduction. Section 3.6 discusses how tone-reproduction techniques can be applied to colour images, and finally, in Section 3.7 we investigate how tone-reproduction can be performed on *view-independent* radiosity solutions, rather than static images, as is required when hardware assisted rendering is employed to generate images of the scene.

(a) Average Luminance= 1000 cd/m^2

(b) Average Luminance= 10 cd/m^2

(c) Average Luminance= 0.1 cd/m^2

Figure 3.2: Comparison of (left) ad-hoc and (right) perceptually-based tone-reproduction methods for different average image luminances.

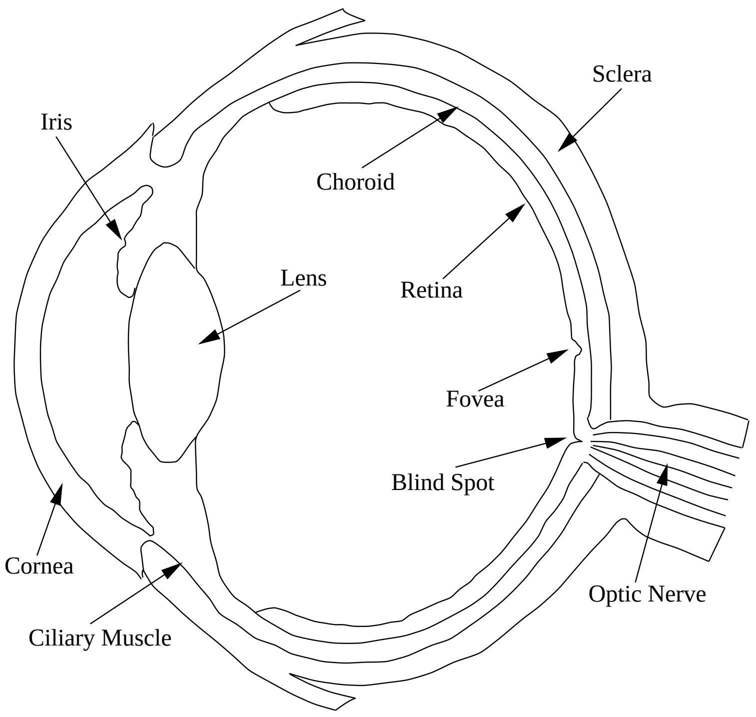


Figure 3.3: A horizontal cross-section of the human eye.

3.2 The Human Visual System

The human eye (a simplified cross-section of which is shown in Figure 3.3) is composed of several parts. The *retina*, which lines the rear part of the vitreous chamber, consists of several layers of nerve tissue, and has a light sensitive layer of cells towards its rear. This layer is composed of two different types of photo-receptors – *rods* and *cones*, both of which are sensitive to light in a range from approximately 400 to 700 nm. Rods provide achromatic vision at low levels of light. There are three types of cones present in the human eye, providing colour vision at higher levels of illumination. Each type of cone is sensitive to either short (S), medium (M) or long (L) wavelengths of light in the visible spectrum.

At scotopic illumination levels (below 0.01 $cd\ m^{-2}$), the rods are more sensitive than the cones, and provide all of our visual input. At photopic levels (above 10 $cd\ m^{-2}$), the cones are active, and are responsible for transmitting sharp, colour images to the brain. In the mesopic region (between 0.01 and 10 $cd\ m^{-2}$), both rods and cones are sensitive. The small area around the *fovea* contains only M and L cones. The lack of S sensitive cones in this region allows the M and L cones to be packed together more tightly, resulting in a greater spatial resolution at the centre of our visual field, but with less sensitivity to light of short (blue) wavelengths [Wan94]. Outside the foveal region, the density of cones falls off rapidly, and the density of rods increases up to a maximum at approximately 20° from the central line of sight. The peripheral region of the retina is composed almost entirely of rods [BG85]. The small area where the optic nerve meets the retina contains no photo-receptors, causing a small 'blind spot' in our visual field.

The rod and cone system can be used to explain many phenomena when comparing day and night-time vision. During daytime, when the average levels of illumination are high, our vision is almost completely cone-dominated. We have good colour vision, and are able to resolve precise spatial detail, especially in the foveal region which corresponds to approximately 1 or 2° of visual angle around the line of sight. As the illumination level falls below 10 $cd\ m^{-2}$, both the rod and cone photo-receptors are active. Once levels drop below 0.01 $cd\ m^{-2}$, however, only the rods are active. Our vision is almost entirely achromatic, and it is difficult to resolve small features, especially when they are viewed directly,

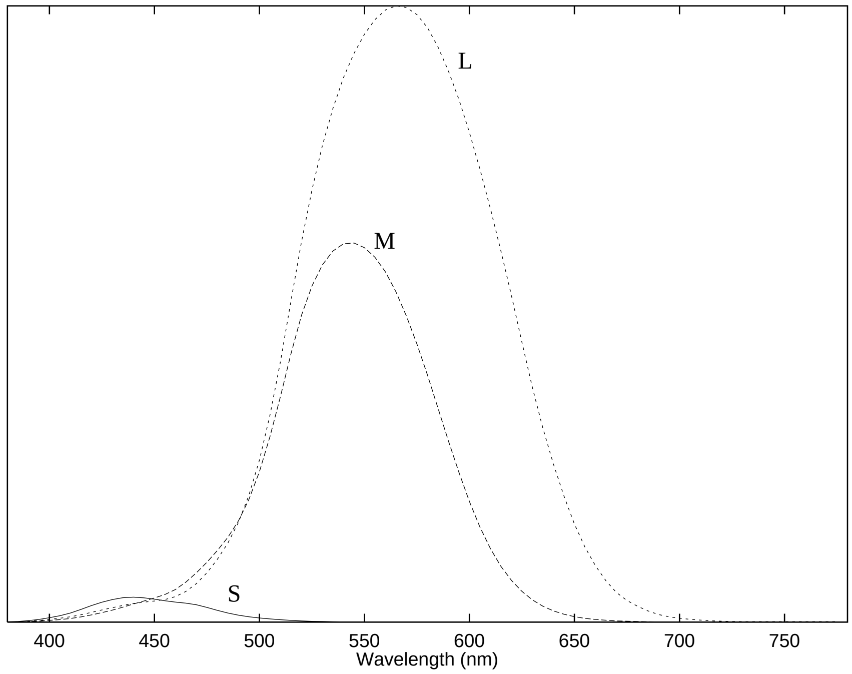


Figure 3.4: Relative sensitivity of the S, M, and L cones over the visible spectrum.

as the photo-receptors around the fovea are completely inactive.

Each photon striking a rod or cone induces an electrical signal which is eventually sent as a nerve impulse to the brain. The signal produced by a photo-receptor only indicates that light has arrived at the eye, and any information regarding the wavelength of light is ignored [DD88]. Colour is therefore only indicated by the type of cone that absorbs the light. The sensitivity of each type of cone over the visible spectrum is shown in Figure 3.4. The nerve fibres connecting the three types of cones to the brain have many intricate inter-connections with each other, indicating that it is unlikely that colour information transmitted to the brain consists of three separate signals, each being proportional to the absorption of light by the three types of cones [Hun67]. Our response to colour has, however, been successfully characterised using *trichromatic colour systems*, where a single colour is specified by the weighted sum of three *colour matching functions* [Hun67]. These matching functions form the basis of a *colour space* in which colours are expressed using three independent numbers. Colour spaces for computer graphics are discussed in more detail in Appendix A.

3.2.1 Adaptation

Our visual system is only effective over a limited range of approximately 100 $cd\ m^{-2}$ [BG85], but we are able to respond to the enormous variety in every-day luminance values through a process known as *adaptation*. When presented with a range of luminances in an environment, an observer's visual system is able to change its sensitivity in such a way as to match the current average light intensity incident on the eye. Two different luminances above the upper limit of the effective range appear to be indistinguishable, as are luminances below the lower limit, but the process of adaptation means that the range of effective luminances can be shifted up or down, depending on the prevailing illumination conditions [BG85]. Research has shown that the eye adapts to the average luminance present in the foveal region of vision [MS43], with this luminance contributing over 91% to the current level of adaptation. The remaining 9% comes from the periphery of the visual field, and is due mainly to scattering of light in the media of the eye [MS43].

Adaptation is caused by mechanical changes (alterations in pupil size), photo-chemical changes (the bleaching of receptor photo-pigments) and alterations in neural processing [FPSG96], and does not happen instantaneously. Whenever a rapid increase in the average level of illumination occurs, it can often take many seconds for the visual system to respond. This process, known as *light adaptation*, can be observed when moving from a dimly lit interior outside into bright sunshine. Initially, little can be seen since everything is painfully bright. Light adaptation in the rod system occurs quickly, with almost 75% of sensitivity recovered in the first 200 ms, and over 80% within 2 seconds [FPSG96, Ade82]. The cones respond much more slowly – after approximately 3 minutes, most of the sensitivity is recovered, but full adaptation does not occur until more than 10 minutes after exposure [FPSG96, Bak49]. Moving from very high to very low levels of illumination is accounted for by the process of *dark adaptation*. Sensitivity increases at a much slower rate, taking up to 35 minutes for the visual system to reach its maximum response [FPSG96, Hec34].

Adaptation affects the way we perceive light, adjusting the sensitivity of the visual system and changing parameters such as brightness, contrast, and visibility. In the next sections we will discuss how each of these attributes is affected.

3.2.2 Threshold Sensitivity

An important quantity that changes with adaptation is *threshold sensitivity* – the smallest difference in luminance between a target and a background that is detectable when an observer is fully adapted to the luminance of the background. Blackwell described a comprehensive analytic model of how the visual performance potential of observers changes with adaptation level [Bla81]. It was determined that the luminance difference, ΔL, between a circular target and uniform background which is just noticeable when the observer is adapted to the luminance of the background, L_a, is given by

$$\Delta L = 0.054(1.219 + L_a^{0.4})^{2.5} \tag{3.6}$$

where all luminances are expressed in $cd\ m^{-2}$. More recently, Ferwerda *et al.* presented threshold sensitivity functions that describe the sensitivity of the rods and cones separately [FPSG96]. Their function representing cone sensitivity is

$$\log_{10}(\Delta L) = \begin{cases} -0.72 & \text{if } \log_{10}(L_a) \leq -2.6 \\ \log_{10}(L_a) - 1.255 & \text{if } \log_{10}(L_a) \geq 1.9 \\ (0.249 \log_{10}(L_a) + 0.65)^{2.7} - 0.72 & \text{otherwise} \end{cases} \tag{3.7}$$

where luminances are again expressed in $cd\ m^{-2}$, and the rod sensitivity is given by

$$\log_{10}(\Delta L) = \begin{cases} -2.86 & \text{if } \log_{10}(L_a) \leq -3.94 \\ \log_{10}(L_a) - 0.395 & \text{if } \log_{10}(L_a) \geq -1.44 \\ (0.405 \log_{10}(L_a) + 1.6)^{2.18} - 2.86 & \text{otherwise} \end{cases} \tag{3.8}$$

Figure 3.5 shows plots of these three functions. It can be seen that Blackwell's model closely matches the cone sensitivity function given in Equation 3.7.

Between 0.1 and 100 $cd\ m^{-2}$ for the rod function, and above 100 $cd\ m^{-2}$ for the cones, the graphs are very nearly linear. These regions obey *Weber's law* [CF75, Hec24]

$$\frac{\Delta L}{L} = \Delta c \tag{3.9}$$

where Δc is known as the *contrast* between a luminance L and luminance $L + \Delta L$. Weber's law states that the contrast in these regions is constant, approximately 0.02. This suggests that equal increments in the logarithm of luminance will be perceived as being equally different ($\Delta \log_{10} L \propto \Delta c$).

3.2.3 Brightness

Brightness is a quantity that measures the subjective sensation produced by a particular luminance. Since brightness depends on the response of the human eye, it will change depending on the current level

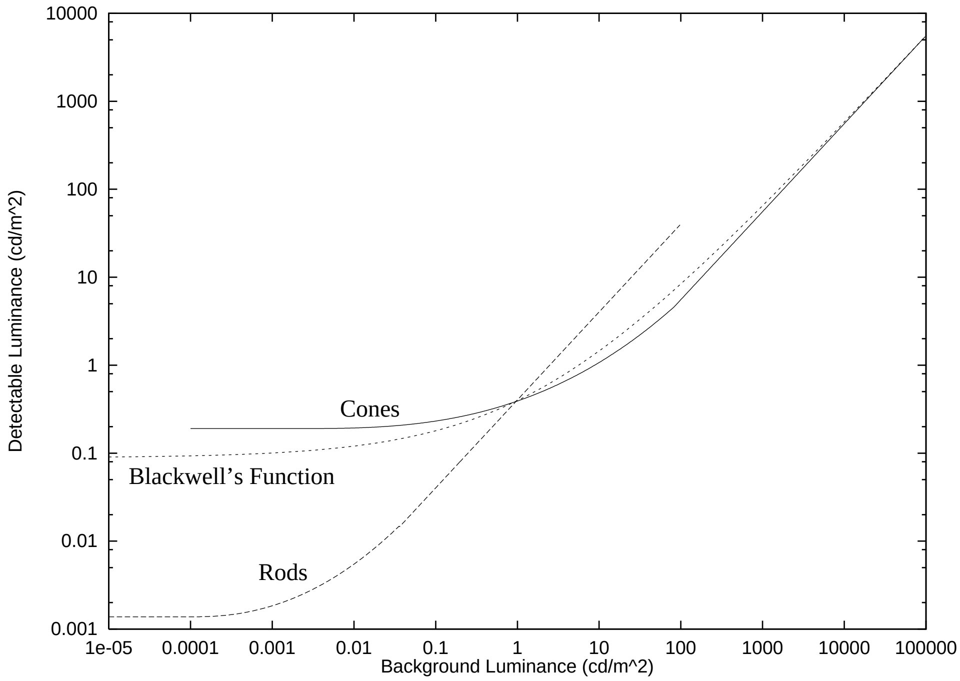


Figure 3.5: Luminance difference (in $cd\ m^{-2}$) between a target and background required for detection when the observer is fully adapted to the background luminance. Ferwerda's rod and cone sensitivity model is shown, along with Blackwell's function.

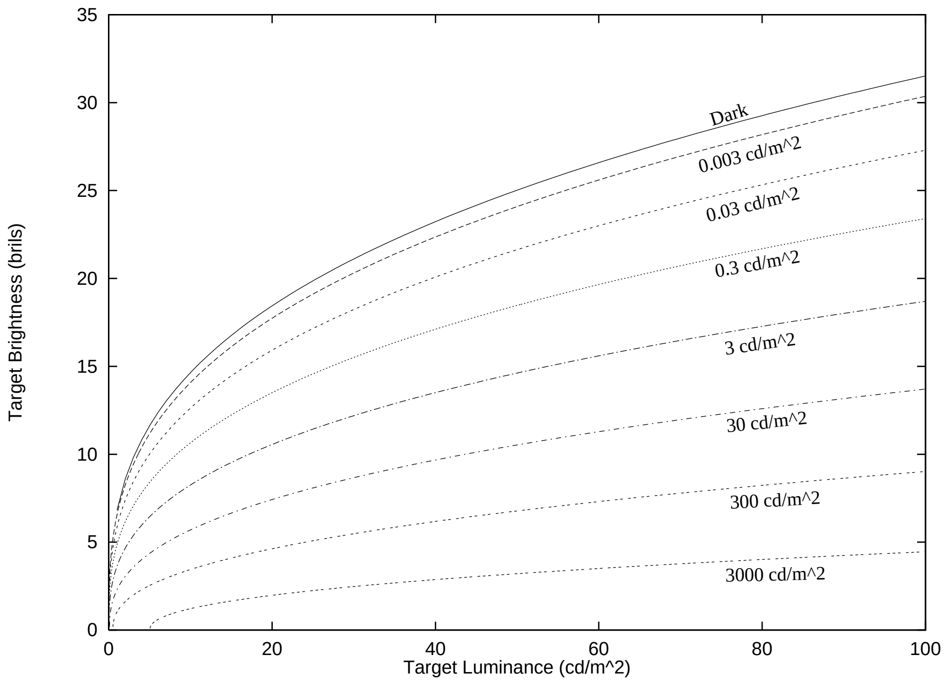


Figure 3.6: Brightness functions relating target luminance in $cd\ m^{-2}$ to brightness in brils for various levels of adaptation. Each plot is labelled with the adaptation level.

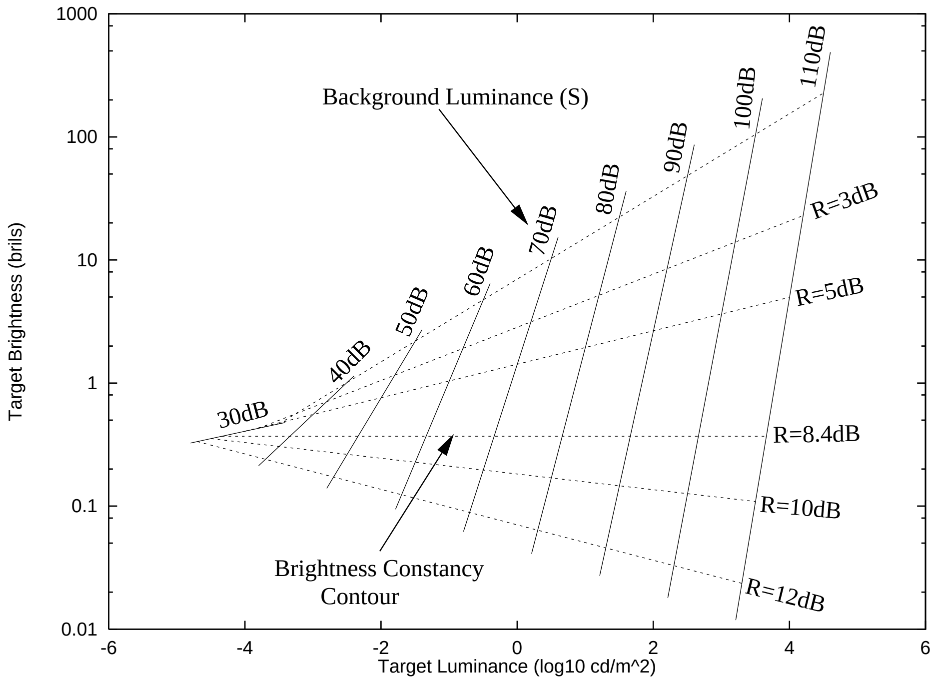


Figure 3.7: Reported brightness (in brils) of target luminances (measured in $cd\ m^{-2}$), for various background luminances (measured in decibels), shown on a log-log scale.

of adaptation. Stevens and Stevens proposed a linear scale for brightness known as 'brils', where one bril is equivalent to the brightness induced by a 1 second exposure to a 5° white target of $1/(\pi \times 10^2)$ $cd\ m^{-2}$, or 1 microlambert [SS63]. They demonstrated that subjective brightness, B, grows as a power function of luminance

$$B = k(L - L_0)^\alpha \tag{3.10}$$

where k is a constant, L_0 is the threshold luminance (the minimum luminance that can be seen), and α is an exponent between 0.333 and 0.49, depending on the level of adaptation [SS63]. Plots of this function (using data from [SS63]) are shown in Figure 3.6.

Stevens and Stevens determined that the relationship between the luminance of the target and its apparent brightness (as reported by the observer) was linear on a log-log scale. Figure 3.7 shows plots of this relationship for various background luminances [SS60], measured in *decibels*[3]. By denoting the ratio of background luminance to target luminance as the *target reflectance*, R (in decibels)

$$R = 10 \log_{10}(L_b/L_t) \tag{3.11}$$

Stevens' results for the brightness of a grey target on a white background illuminated at photopic levels may be expressed as follows [SS60]

$$\log_{10}(B) = 0.004((S - 27)(8.4 - R) - 108) \tag{3.12}$$

where B is the brightness in brils, and S is the background luminance, measured in decibels. Equation 3.12 may be written in a more convenient form, as a power function with luminances measured in lamberts [TR93]

$$B = 10^{\beta(L_b)} L_t^{\alpha(L_b)} \tag{3.13}$$

[3]The decibel is used as a measure of luminance because it provides a convenient logarithmic unit [SS63]. 0 dB is equivalent to 3.183×10^{-7} $cd\ m^{-2}$, and an increment of 10 decibels is equivalent to an increase of one $\log_{10}$ unit in $cd\ m^{-2}$. Luminance in $cd\ m^{-2}$ can be converted to decibels using $dB = 100 + 10 \log_{10}(\pi L \times 10^{-4})$.

where

$$\alpha(L_b) = 0.4\log_{10}(L_b) + 2.92 \tag{3.14}$$

and

$$\beta(L_b) = -0.4(\log_{10}(L_b))^2 - 2.584\log_{10}(L_b) + 2.0208 \tag{3.15}$$

Although these expressions provide a convenient relationship between luminance and brightness at photopic levels, Tumblin and Rushmeier noted that brightness response is controlled by many interconnecting processes, and as such is very difficult, if not impossible, to quantify accurately [TR93]. Bartleson and Breneman stated that although Stevens' power law may be applied to brightness perceptions of simple targets, more complex fields are not as easily represented. They performed experiments to measure the apparent brightness of elements in highly complex images such as photographs and projected transparencies. Rather than obeying a power law, the resulting brightness functions were found to be non-linear in log-log coordinates, and exhibited an exponential decay from the simple power form, as a function of target and background luminance [BB67]. The performance of Stevens' functions over a wide range of luminance values will be examined in Section 3.4.5.

3.3 Principles of Tone-Reproduction

Several researchers have developed *tone-reproduction operators* [CHS+93, TR93, War94, Sch94a, FPSG96, LRP97] that are able to transform luminances calculated by a global illumination algorithm into frame-buffer values, such that when they are displayed on a monitor screen, the visual experience of the *display observer* (the person looking at the monitor screen) would be similar to that experienced by a *real-world observer* viewing the real environment. This is achieved by first simulating the processes of adaptation in order to determine how luminances in the real-world and on the display device are perceived, and then attempting to match the subjective appearance of the image as seen by both display and real-world observers. Although the range of luminance values that can be displayed is often far less than would be needed to provide a correct image, the human visual system is not sensitive to absolute luminances, but rather to relative differences [Gla95]. This characteristic can be used to ensure that the same subjective qualities that occur in a real environment can be reproduced on a display device, even if the luminances themselves cannot. Indeed, it has been known by painters and artists for many centuries that the perceived intensity of light (the brightness) at a point is influenced by the contrast between that point's luminance and the luminance of the surrounding area – Leonardo da Vinci noted that "If you wish to produce an excellent darkness, give it an excellent lightness by way of contrast" [Alb96, dV56].

The principle of tone-reproduction is illustrated in Figure 3.8 (after [TR93]). Starting with a set of real-world luminances[4], the real-world observer views the scene and forms a mental picture of its contents (anti-clockwise from top). We wish to generate a set of frame-buffer values, and create an image on the display device that will be viewed by the display observer (clockwise from bottom). In order to ensure that the quantities Q_w and Q_d (such as brightness or visibility) that are perceived by each observer are equal, the frame-buffer values that are calculated must account for:

- the perception of real-world luminance by the real-world observer;
- the relationship between frame-buffer value and displayed luminance;
- The perception of displayed luminance by the display observer.

Provided we are able to construct a mathematical model of each of these functions, they may be concatenated as shown on the right of Figure 3.8 to ensure that the match is as close as possible.

Figure 3.9 shows a classification of recent tone-reproduction algorithms. For the purposes of discussion, these have been split into two classes – *single-scale* and *multi-scale* operators. Single-scale operators use the same scaling transformation for each pixel in the image, and that scaling is dependent only upon the current levels of adaptation, not on the real-world luminances themselves. The simple

[4] These are luminance values in the real scene, which we assume are the same as those calculated using the global illumination algorithm.

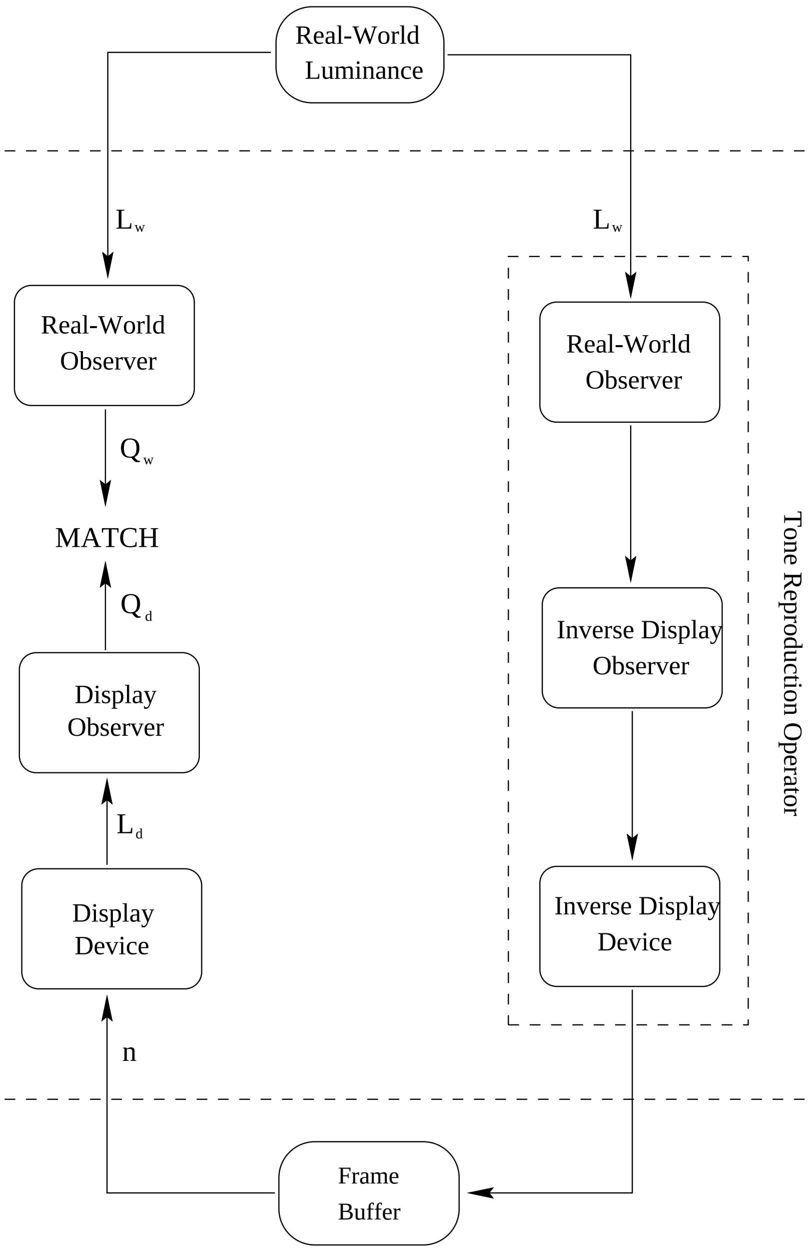


Figure 3.8: The process of tone-reproduction.

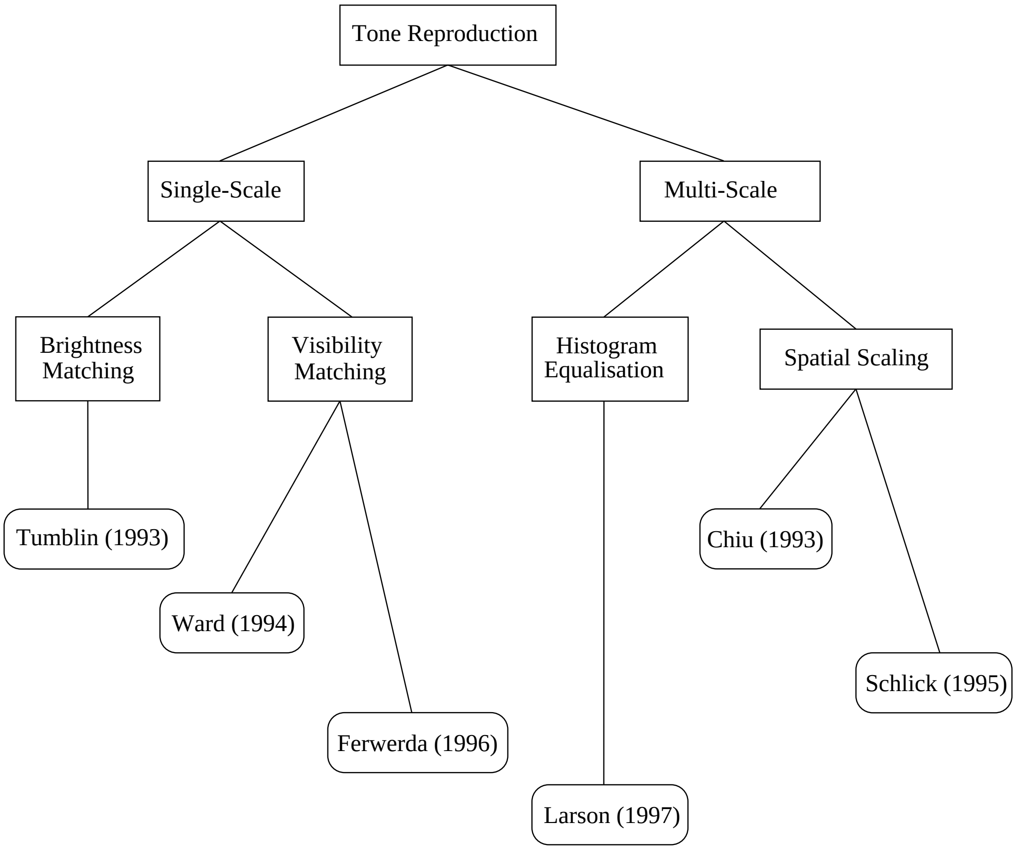


Figure 3.9: A classification of tone-reproduction algorithms.

tone-reproduction operators introduced in Section 3.1 are examples of single-scale models. Multi-scale operators may apply a different scale to each pixel in the image that is dependent upon factors other than just the adaptation levels. This distinction will become important in Chapter 4, when we discuss how tone-reproduction operators may be estimated before the global illumination solution has been computed.

In the next section, several single-scale operators will be introduced, each of which attempts to preserve a different aspect of the visual experience (such as brightness or contrast) when displaying an image on screen. The performance of these operators is then compared for a typical environment at varying levels of illumination.

3.4 Single-Scale Operators

3.4.1 Tumblin and Rushmeier

The first perceptually-based tone-reproduction operator was developed by Tumblin and Rushmeier [TR91, TR93]. Using the equations relating brightness to luminance that were derived by Stevens and Stevens, and discussed in Section 3.2.3, they produced a tone-reproduction operator that attempted to match the brightness of the real-world scene to the brightness of the image displayed on a CRT. This was achieved by constructing an *observer model* that described how real-world and display luminances are perceived, and a *display model* that described how a frame-buffer value is converted into displayed luminance on the display device.

Given an image calculated by a global illumination algorithm, a hypothetical real-world observer would view the scene, and through the process of adaptation discussed above, his eyes would adapt to a luminance $L_{a(w)}$. The perceived brightness of a real-world luminance value, L_w, at this level of adaptation can be determined by applying Stevens' equation relating brightness to target luminance (Equation 3.13)[5]

$$B_w = 10^{\beta(L_{a(w)})}(\pi \times 10^{-4} L_w)^{\alpha(L_{a(w)})} \tag{3.16}$$

where $\alpha(L_{a(w)})$ and $\beta(L_{a(w)})$ are functions of the real-world adaptation level

$$\alpha(L_{a(w)}) = 0.4 \log_{10}(L_{a(w)}) + 1.519 \tag{3.17}$$

$$\beta(L_{a(w)}) = -0.4(\log_{10}(L_{a(w)}))^2 - 0.218 \log_{10}(L_{a(w)}) + 6.1642 \tag{3.18}$$

and all luminances are expressed in $cd\ m^{-2}$. Assuming a display observer viewing a CRT screen is adapted to a luminance of $L_{a(d)}$, the brightness of a displayed luminance value L_d can be expressed similarly

$$B_d = 10^{\beta(L_{a(d)})}(\pi \times 10^{-4} L_d)^{\alpha(L_{a(d)})} \tag{3.19}$$

where $\alpha(L_{a(d)})$ and $\beta(L_{a(d)})$ are defined as in Equations 3.17 and 3.18. In order to match the brightness of a real-world luminance to the brightness of a displayed luminance, we must have $B_w = B_d$. Using Equations 3.16 and 3.19, the display luminance required to achieve the match can be determined

$$L_d = \frac{1}{\pi \times 10^{-4}} 10^{\frac{\beta_{a(w)} - \beta_{a(d)}}{\alpha_{a(d)}}} (\pi \times 10^{-4} L_w)^{\frac{\alpha_{a(w)}}{\alpha_{a(d)}}} \tag{3.20}$$

This equation represents the first two stages of the tone-reproduction operator shown in Figure 3.8 – the concatenation of the real-world observer model and the inverse display observer model. All that is left is to invert Equation 3.3 in order to find a frame-buffer value n that will produce the required display luminance

$$n = \left[\frac{L_d - L_{amb}}{L_{dmax}}\right]^{1/\gamma} \tag{3.21}$$

[5] The following equations differ from those presented in [TR91, TR93], since we assume luminance values are given in SI units of $cd\ m^{-2}$ instead of lamberts.

giving,

$$\mathcal{T}_{TUMB}(L_w) = \left[\frac{10^{\frac{\beta_{a(w)}-\beta_{a(d)}}{\alpha_{a(d)}}} (\pi \times 10^{-4} L_w)^{\frac{\alpha_{a(w)}}{\alpha_{a(d)}}}}{\pi \times 10^{-4} L_{dmax}} - \frac{L_{amb}}{L_{dmax}} \right]^{1/\gamma} \tag{3.22}$$

Because of the limited dynamic range of the CRT, a display luminance will often be required that is above or below the displayable limits. In this case, the frame-buffer value n must be clipped to the range $[0 \ldots 1]$.

3.4.2 Ward

A simpler model was proposed by Ward, who derived a single linear scaling value that maintained the impression of contrast (equivalent to visible changes in luminance) between the real scene and the displayed image [War94]. Using Blackwell's model of the luminance difference that is just noticeable at a given adaptation level (Equation 3.6), the operator preserved the minimum discernible difference that the real-world observer would be able to detect. Recalling Equation 3.6, the smallest luminance detectable by a real-world observer who is adapted to a luminance of $L_{a(w)}$ is

$$\Delta L_w = 0.054(1.219 + L_{a(w)}^{0.4})^{2.5} \tag{3.23}$$

A similar expression can be used for the display observer, adapted to $L_{a(d)}$

$$\Delta L_d = 0.054(1.219 + L_{a(d)}^{0.4})^{2.5} \tag{3.24}$$

In order to maintain contrast between the real and displayed image, Δc (Equation 3.9) must be equal for both the real-world and display observers

$$\frac{\Delta L_w}{L_w} = \frac{\Delta L_d}{L_d} \tag{3.25}$$

and the required display luminance may therefore be expressed in terms of the real-world luminance and the adaptation levels

$$L_d = L_w \left[\frac{1.219 + L_{a(d)}^{0.4}}{1.219 + L_{a(w)}^{0.4}} \right]^{2.5} \tag{3.26}$$

Finally, Equation 3.21 must be applied to calculate a frame-buffer value which is then clipped to the range $[0 \ldots 1]$ if necessary. The resulting operator is given by

$$\mathcal{T}_{WARD}(L_w) = \left[\frac{L_w}{L_{dmax}} \left[\frac{1.219 + L_{a(d)}^{0.4}}{1.219 + L_{a(w)}^{0.4}} \right]^{2.5} - \frac{L_{amb}}{L_{dmax}} \right]^{1/\gamma} \tag{3.27}$$

3.4.3 Ferwerda *et al.*

The most recent single-scale operator was developed by Ferwerda *et al.* [FPSG96]. This model was conceptually similar to that proposed by Ward [War94], but, as well as preserving threshold visibility, changes in colour appearance, visual acuity (the ability of the eye to resolve small spatial details), and temporal sensitivity were also included.

In order to match visibility in the real-world to visibility in the display, Δc must be equal for both luminances, as in Ward's model.

$$\frac{\Delta L_w}{L_w} = \frac{\Delta L_d}{L_d} \tag{3.28}$$

If the level of adaptation of the real-world observer is in the photopic range (above 10 $cd\ m^{-2}$), a photopic tone-reproduction operator is applied. Considering the cone-based threshold sensitivity given in Equation 3.7, a *threshold versus intensity* (t.v.i.) function is produced for both the display and real-world observers at their appropriate levels of adaptation[6]

$$\Delta L_w = \begin{cases} 0.1905 & \text{if } L_{a(w)} \leq 2.512 \times 10^{-3} \\ 0.0556 L_{a(w)} & \text{if } L_{a(w)} \geq 79.433 \\ 0.190510^{(0.249 \log_{10}(L_{a(w)})+0.65)^{2.7}} & \text{otherwise} \end{cases} \tag{3.29}$$

and similarly for the display, using ΔL_d and display adaptation level $L_{a(d)}$. A linear scale factor is then calculated, relating real-world luminance to photopic display luminance

$$L_{d(p)} = L_w \frac{\Delta L_d}{\Delta L_w} \tag{3.30}$$

When the real-world observer is adapted to a luminance in the scotopic range (below 0.01 $cd\ m^{-2}$), the rod data from Equation 3.8 is used to determine ΔL_w

$$\Delta L_w = \begin{cases} 1.3804 \times 10^{-3} & \text{if } L_{a(w)} \leq 1.148 \times 10^{-4} \\ 0.4027 L_{a(w)} & \text{if } L_{a(w)} \geq 0.0363 \\ 1.3804 \times 10^{-3} 10^{(0.405 \log_{10}(L_{a(w)})+1.6)^{2.18}} & \text{otherwise} \end{cases} \tag{3.31}$$

Note that the real-world luminance values that are used for this equation must be generated using the scotopic luminous sensitivity function (Figure 2.1). A very good approximation of the scotopic luminance can be calculated from the photopic luminance using [LRP97]

$$L = Y \left[1.33 \left(1 + \frac{Y+Z}{X} \right) - 1.68 \right] \tag{3.32}$$

where X, Y and Z are the CIEXYZ primaries of the photopic colour (see Section 3.6). Regardless of the state of adaptation of the real-world observer, the display observer will still be adapted to a photopic luminance, and so the cone t.v.i. function is used when calculating ΔL_d.

The display luminance required on the display device is then calculated by combining the photopic and scotopic display luminances, using a parametric constant k

$$L_d = L_{d(p)} + k(L_{a(w)}) L_{d(s)} \tag{3.33}$$

where k varies between 1 and 0 as the real-world adaptation level goes from the bottom and top of the mesopic range [FPSG96]. Because rod-dominated vision is achromatic, the rod t.v.i function is used to produce a grey-scale display luminance [FPSG96]. When combined with the colour luminance values generated by the cone t.v.i. function (see Section 3.6), this preserves the loss in colour that is observed when the average luminance drops into the rod-dominated levels of vision.

Ferwerda's tone-reproduction operator also included models of the loss in visual acuity when the real-world adaptation level is reduced. They used data obtained by Shaler [Sha37] that related the detectability of square wave gratings of different spatial frequencies to changes in background luminance [FPSG96]. A Gaussian convolution filter was then applied to remove frequencies from the image that could not be resolved by the real-world observer when adapted to the real-world adaptation level.

Light and dark adaptation were also included in the operator. This was achieved by adding a parametric constant, b, to the display luminance, the value of which changes over time

$$L_d = L_w \frac{\Delta L_d}{\Delta L_w} + b \tag{3.34}$$

Because no quantitative data was available for the change in b, Ferwerda *et al.* set b so that

$$L_d = L_{a(w)} \frac{\Delta L_d}{\Delta L_w} + b \tag{3.35}$$

remains constant over time. This caused the overall luminance of the displayed image to remain the same during the time-dependent adaptation process [FPSG96].

[6]These equations differ to those presented in [FPSG96], since we use a non-logarithmic representation, for consistency with Ward's model.

3.4.4 Adaptation Luminances

All of the above tone-reproduction operators require adaptation levels for both the real-world and display observers. Several methods for calculating these have been proposed. Tumblin and Rushmeier [TR93] assumed that the eye adapts so as to keep most of the real-world brightnesses around the *brightness constancy* contour of Figure 3.7 (8.4 dB). This contour corresponds to the ratio of target to background luminances for which the target luminance appears at a constant brightness over a wide range of background levels. Their real-world adaptation level was therefore

$$\log_{10}(L_{a(w)}) = \mathcal{E}[\log_{10}(L_w)] + 0.84 \tag{3.36}$$

where $\mathcal{E}[]$ is the expected value of the logarithm of image luminances, expressed in lamberts [TR93].

Ward took a simpler approach, assuming that the eye adapts to the logarithmic average of image luminances, but without the correction to maintain brightness constancy. Ferwerda *et al.* used a real-world adaptation value that is simply half the maximum visible luminance in the image [FPSG96].

As noted in [TR93], the actual level of display adaptation cannot be computed exactly, since it requires knowing the final display luminances of the image. The display adaptation level is therefore usually set to some fraction of the maximum displayable luminance of the CRT. Tumblin and Rushmeier assumed that the display luminances are uniformly distributed on a logarithmic scale, with $\log_{10}(L_{a(d)})$ as their midpoint. Their display adaptation level was set to

$$L_{a(d)} = \frac{L_{dmax}}{\sqrt{C_{max}}} \tag{3.37}$$

where C_{max} is the maximum display contrast [TR93]. Both Ward and Ferwerda assumed that the display luminances are uniformly distributed over the displayable range, and set the display adaptation level to half the maximum displayable luminance

$$L_{a(d)} = \frac{L_{dmax}}{2} \tag{3.38}$$

3.4.5 A Comparison of Models

A quantitative comparison of a displayed image against an observer's perception of a real scene is difficult to obtain. This is mainly due to the difficulties in recording high dynamic range images of real-world luminance values, and comparing a displayed image to a real scene without altering the observer's perception of the appearance of the real scene. Because of this, the best we can do is to judge the models against each other, and compare the qualitative appearance of each image with what would be expected if a real scene were viewed.

Each operator was constructed with a different idea of what the most important factor is that should be preserved, and as such, defining a single quality that should be present in the resulting images is not possible. We will assess the relative differences in contrast and brightness between models, although from experience, we have found that contrast is perhaps a more important feature that should be preserved. This is supported by evidence suggesting that the human visual system is sensitive to ratios of intensity, rather than the absolute levels themselves [Gla95].

Although the printing process should be considered as an extra step in the tone-reproduction algorithm, since the relationship between image values and luminance perceived by a 'print observer' is affected by the quality of the printer as well as the conditions under which the print is viewed, we have not included this information in these comparisons. The images in the following figures do, however, bear a close relationship to those displayed on the monitor screen, and the conclusions drawn are the same regardless of the medium on which they are compared.

Figure 3.10 compares the three single-scale tone-reproduction operators over a wide range of luminance values that are typically found in global illumination images. Each row shows results for Ward's model on the left, Ferwerda's model in the middle, and Tumblin's model on the right. Each operator was applied to a set of images with average luminances ranging from 1000 $cd\ m^{-2}$, which represents typical

Ward | Ferwerda *et al.* | Tumblin and Rushmeier

(a) Average Luminance= 1000 cd/m^2 (daylight)

(b) Average Luminance= 10 cd/m^2 (interior)

(c) Average Luminance= 0.1 cd/m^2 (moonlight)

(d) Average Luminance= 0.001 cd/m^2 (starlight)

Figure 3.10: A comparison of single-scale tone-reproduction operators for a wide range of average real-world luminances.

daylight illumination levels, to 0.001 $cd\ m^{-2}$, approximating starlight illumination[7]. The acuity and time-dependent adaptation changes modelled by Ferwerda's operator were not used in the comparison, because, as will be discussed later, we wish to apply these operators to view-independent solutions. The real-world observer was assumed to be fully adapted to the average image luminance, and the display adaptation level was set to half the maximum displayable luminance of the CRT.

We can see that the performance of all models at photopic levels of illumination (rows (a) and (b)) is approximately the same – both brightness and contrast are reproduced as would be expected for a scene illuminated at this level. The contrast matching operators of Ward and Ferwerda give almost identical images due to the similarity of the threshold sensitivity curves plotted in Figure 3.5. Tumblin's operator also performs well, since contrast and brightness are also as expected, with the overall level of brightness being slightly greater.

As the average level of illumination decreases and enters the mesopic and scotopic ranges (rows (c) and d), we start to see the differences between the operators more clearly. Because the model of threshold sensitivity that Ward's operator is based on does not account for the change between cone and rod sensitivity, the images quickly become unrealistically dark, even at levels of illumination corresponding to moonlight (row (c)). Assuming that the real-world observer had adapted to these low levels of illumination, we would expect that some of the larger luminance contrasts in the scene would still be visible. Ferwerda's operator models this change from cone to rod-dominated vision more accurately – contrast is still discernible even at 0.001 $cd\ m^{-2}$. Tumblin's operator is less successful at preserving the appearance of scenes at low levels of illumination, and luminances in the image appear to approach a medium-grey level rather than black.

Figure 3.11 shows the compression in dynamic range that is applied to luminance values over a wide range of adaptation levels. The solid line in each graph represents an average real-world luminance, to which we assume the real-world observer is fully adapted. The upper and lower dashed plots in each graph represent luminances that are 10 and 0.1 times the average real-world luminance respectively. These curves can be considered as representing the brightest and dimmest pixels in a typical image. The vertical axis in each graph represents the value stored in the frame-buffer when the image is displayed.

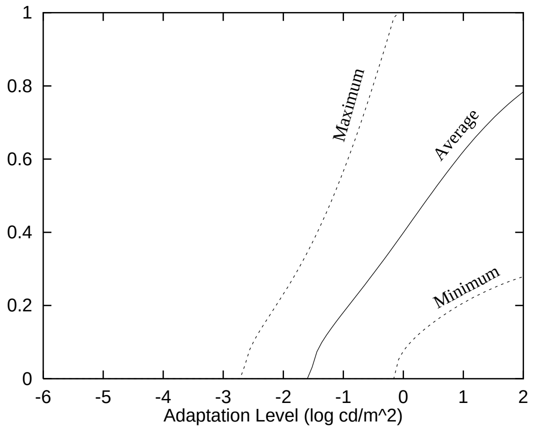
When adapted to a high level of illumination, all three operators perform similarly, as demonstrated in the images shown above. Comparing the plots of Ward's model against Ferwerda's, the change between rod and cone-sensitive vision as the level of adaptation reduces can be clearly seen at around 0.1 $cd\ m^{-2}$ where the display value levels off slightly before falling again.

The interesting behaviour of Tumblin and Rushmeier's operator can be seen in the lower graph, where the average display value remains relatively high, even at very low adaptation levels below 10^{-3} $cd\ m^{-2}$. The transformation of the image to an average medium-grey intensity can be seen as all three real-world luminance plots converge to a single display value when the real-world observer is adapted to approximately $10^{-3.75}$ $cd\ m^{-2}$. This plot also shows an interesting behaviour below this level of adaptation – the smallest real-world luminance is mapped to a display value *greater* than that of the largest real-world luminance. These artifacts can probably be explained by the fact that Steven's data (Equation 3.12), on which the transformation is based, was obtained for the particular case of a gray target on a white background, where the illumination conditions are in the photopic range [SS60], and so do not extrapolate well to scotopic vision.

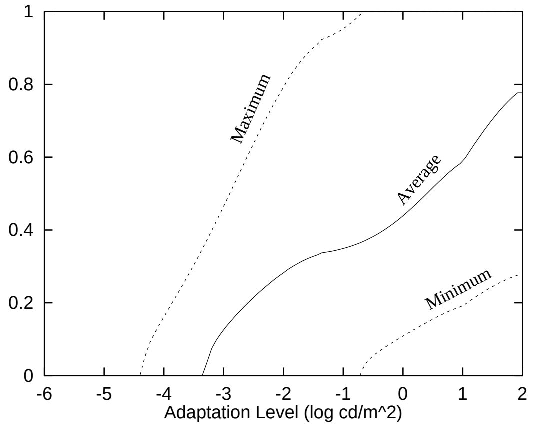
3.5 Multi-Scale Operators

It has been argued that using a single adaptation level to derive a tone-reproduction transformation is fundamentally incorrect, since the eye adapts to the luminance present in the small foveal region around the centre of the visual field [LRP97, MS43]. The single-scale operators discussed in the previous section ignore this, meaning that the display observer's perception of the scene does not change as his eyes move over the image, as would that of the real-world observer.

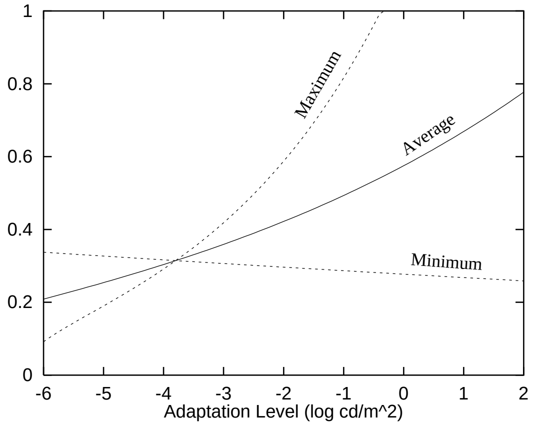
[7]These images were not produced using daylight or starlight lighting models – the levels of illumination were simply scaled to give an equivalent average luminance.



(a) Ward



(b) Ferwerda *et al.*



(c) Tumblin and Rushmeier

Figure 3.11: Plots of the dynamic range compression that occurs with the application of each operator, (a) Ward, (b) Ferwerda *et al.* and (c) Tumblin and Rushmeier.

Figure 3.12: Comparison of images generated using a (right) single adaptation level against (left) images with a changing adaptation level based on the focus of the observer.

Figure 3.12 shows a sequence of images comparing the difference between an image-based adaptation level and a foveal based level. The left-hand images, reading from top to bottom, show the results obtained when a viewer moves his direction of gaze across the scene. A box is positioned in each image showing the area corresponding approximately to the foveal region, and the appropriate adaptation level for each region is given. As the level of adaptation decreases, the image becomes correspondingly brighter as expected. Comparing these images to those shown on the right, which were generated using a single image-based adaptation level, we can see the difference that the foveal-based adaptation makes. When the observer is looking at an area of the image with a high luminance, the adaptation level is increased, meaning that sensitivity around that region is also increased and more detail can be seen. Similarly, when a dark region is being viewed, sensitivity is increased around low luminance levels.

Tone-reproduction operators that mimic these effects do so by generating different transformations for different regions of the image, in effect assuming that when a particular area in the image is viewed, the real-world observer adapts immediately to the average luminance present in that region. Although strictly speaking, the process of adaptation does not happen instantaneously, recent algorithms that use this technique have been shown to be superior to single-scale models, which often generate large areas of display intensities that are beyond the displayable limits of the CRT [LRP97].

The first approach to multi-scale tone-reproduction was introduced by Chiu *et al.*, who applied a spatially non-uniform scaling function over the entire image [CHS$^+$93]. This scaling function was constructed by first blurring the original image to remove all high frequencies, and then inverting the result. Although this approach was able to reproduce all detail that is visible in the original image, reverse intensity gradients appeared in the transformed image when very bright and very dark areas were in close proximity. More recently, Schlick [Sch94a] derived a similar transformation, based on a rational tone-reproduction mapping instead of a logarithmic one. Neither of these methods took account of differing adaptation levels, nor did they aim to use psycho-visual models of the human visual system.

Larson *et al.* [LRP97] developed a histogram-equalisation technique that used a spatially-varying map of foveal adaptation levels to transform a histogram of image luminances in such a way that the resulting image lay within the dynamic range of the display device, and characteristics such as contrast and visibility were preserved. A histogram of brightness (approximated as the logarithm of real-world luminance) was first created for a filtered image in which each pixel corresponds to approximately 1° of the visual field. A histogram and cumulative distribution function were then obtained for this reduced image. After an automatic adjustment algorithm was applied using threshold visibility data from [FPSG96], an image was created in which the dynamic range of the original scene had been compressed into the range available on the display device, subject to certain restrictions regarding the limits of contrast sensitivity of the human eye.

When applied as a post-process to an image, Larson's multi-scale operator produces results that are subjectively better representations of the real scene than can be obtained using single-scale methods [LRP97]. The clipping of out-of-range display values that occurs with single-scale methods is avoided in most cases, since varying adaptation levels (and hence sensitivities) across the image are properly accounted for. As will be described later, for virtual-reality applications, tone-reproduction algorithms are required that are applicable in view-independent situations, and the multi-scale methods described above, which are inherently image-based, are of no direct use. The application of such operators is an interesting area for future research.

3.6 Colour

So far we have only dealt with transforming a set of real-world luminances into grey-scale frame-buffer values. Typical global illumination algorithms calculate luminance values at a number of discrete wavelengths [Mey86] or use one of the many standard colour spaces, such as a monitor's RGB space, or the device-independent CIEXYZ space. Regardless of how colour is represented in the illumination calculations, a colour image, usually displayed on a monitor screen, is usually required as a final product.

Combining tone-reproduction with colour images first requires luminance values to be transformed from whatever colour space is used in the global illumination algorithm, into the device-independent

CIEXYZ space (see Appendix A for more details). Radiosity values represented in this system are expressed as a vector (X_w, Y_w, Z_w), with each value corresponding to the weight applied to the X Y or Z colour matching function. The Y channel corresponds directly to the luminance, and the 'pure' colour of the vector (the colour when brightness is ignored) is represented by two *chromaticity coordinates*

$$x = \frac{X_w}{X_w + Y_w + Z_w} \tag{3.39}$$

$$y = \frac{Y_w}{X_w + Y_w + Z_w} \tag{3.40}$$

A vector of radiosity values calculated in a colour space may therefore be specified by its luminance, Y_w, and its chromaticities x and y. Any of the tone-reproduction techniques discussed in previous sections may then be used to derive a display value Y_d from the real-world luminance Y_w. The X and Z channels of the display colour (also represented using the CIEXYZ system) may then be calculated in such a way that the chromaticity of the original colour is preserved

$$X_d = Y_d \frac{x}{y} \tag{3.41}$$

$$Z_d = Y_d \frac{1 - x - y}{y} \tag{3.42}$$

The display colour represented by the vector (X_d, Y_d, Z_d) can then be transformed into the RGB colour space of the target monitor. Colours outside the gamut of the display device need to be clipped or transformed in some way so that they become displayable. The simplest approach consists of simply clipping the display value Y_d to the range $[0 \ldots 1]$ before applying Equations 3.41 and 3.42, and then transforming from XYZ colour space to the monitor's RGB space. Other approaches, such as shifting the out-of-range values towards the white point have been suggested [Rus94], but this will cause a highlight from a white light-source on a diffuse red wall to appear pink or white rather than a more intense red. This is the reason why we have employed the simpler technique described above, as it preserves the chromaticity of the real-world colour on the display device.

This is a rather simplistic approach to colour tone-reproduction, since complex (and still relatively unknown) effects such as colour adaptation are ignored. The data on which colour systems such as CIEXYZ are based were obtained under very specific conditions. Just as apparent brightness is affected by the brightness of the surround, the actual colours that are perceived by a real-world observer also depend on everything else in view. For example, a bright orange light with a dark surround is perceived as brown when a white surround is used [Spe43]. Assuming that the real-world observer adapts to an achromatic luminance ignores these changes. For the types of environments to which global illumination algorithms are usually applied (such as architectural interiors), these subtle effects are rarely noticeable, but a comprehensive adaptation model that is correct for all situations would need to include these chromatic changes.

3.7 Tone-Reproduction for View-Independent Solutions

When radiosity algorithms are being employed to generate realistic lighting for virtual environments, we require a view-independent radiosity solution rather than a single image. This requires performing tone-reproduction operations on the luminance values stored at the vertices of the element mesh, rather than those of the pixels in a rendered image.

All of the single-scale tone-reproduction operators discussed in Sections 3.4 may be applied to view-independent solutions. The result of applying a non-linear tone-reproduction operator to two vertex luminances and then interpolating between them to find the intermediate display values (as will happen with Gouraud shading) will be slightly different, however, from interpolating between the two vertex luminances, and then applying the operator. In Chapter 4, we will discuss how the perceptibility of this difference can be used to control the level of adaptive subdivision that is required for a visually accurate solution.

The first problem encountered when performing view-independent tone-reproduction is one of generating a suitable adaptation level. Techniques based on *view-frustum culling* may be employed. View-frustum culling is normally used in conjunction with a bounding-volume hierarchy to quickly determine the set of surfaces that are potentially visible, given an eye position and view direction [RH94]. This involves a top-down depth first traversal through the bounding-volume hierarchy, testing each node to see if it is contained entirely within the view frustum. If so, the polygons contained within that node are potentially visible to the viewer, and must therefore be drawn into the Z-buffer. If the node is completely outside the frustum, then the polygons can be ignored. Nodes that partially intersect the frustum have the test repeated recursively for each child. As each potentially-visible object or polygon is found, it can be rendered. For large models where only a small fraction of the scene is visible at any one time, this technique can provide a large increase in frame-rate, as hierarchy traversal is typically much faster than rendering [TS91, FS93].

View-frustum culling may also be used in a similar way to determine the super-set of luminance values that can contribute to the current adaptation level. Polygons cannot be rendered as soon as they are found to be potentially visible, however, since the complete set must first be determined, an estimate of the adaptation level calculated, and then the appropriate tone-reproduction operator derived. Once the operator is known, the potentially-visible set of polygons may be traversed. For each leaf element of each polygon, the tone-reproduction operator must be applied to the luminance value stored at each vertex, and then the element rendered.

The limiting factor in the performance of this technique is the extra cost per frame that the application of the tone-reproduction operator incurs. From preliminary experiments, we have found that approximately $250,000$ applications of Ferwerda's tone-reproduction operator may be performed each second on a 150MHz MIPS R10000 CPU. Assuming that the rendering hardware is capable of displaying $250,000$ elements each second, and each element requires an average of 1.5 vertex tone-reproduction transformations, we estimate that just over $100,000$ elements may be transformed and rendered each second. For a frame-rate of 10 frames per second, this limits the number of elements that can be rendered each frame to around $10,000$.

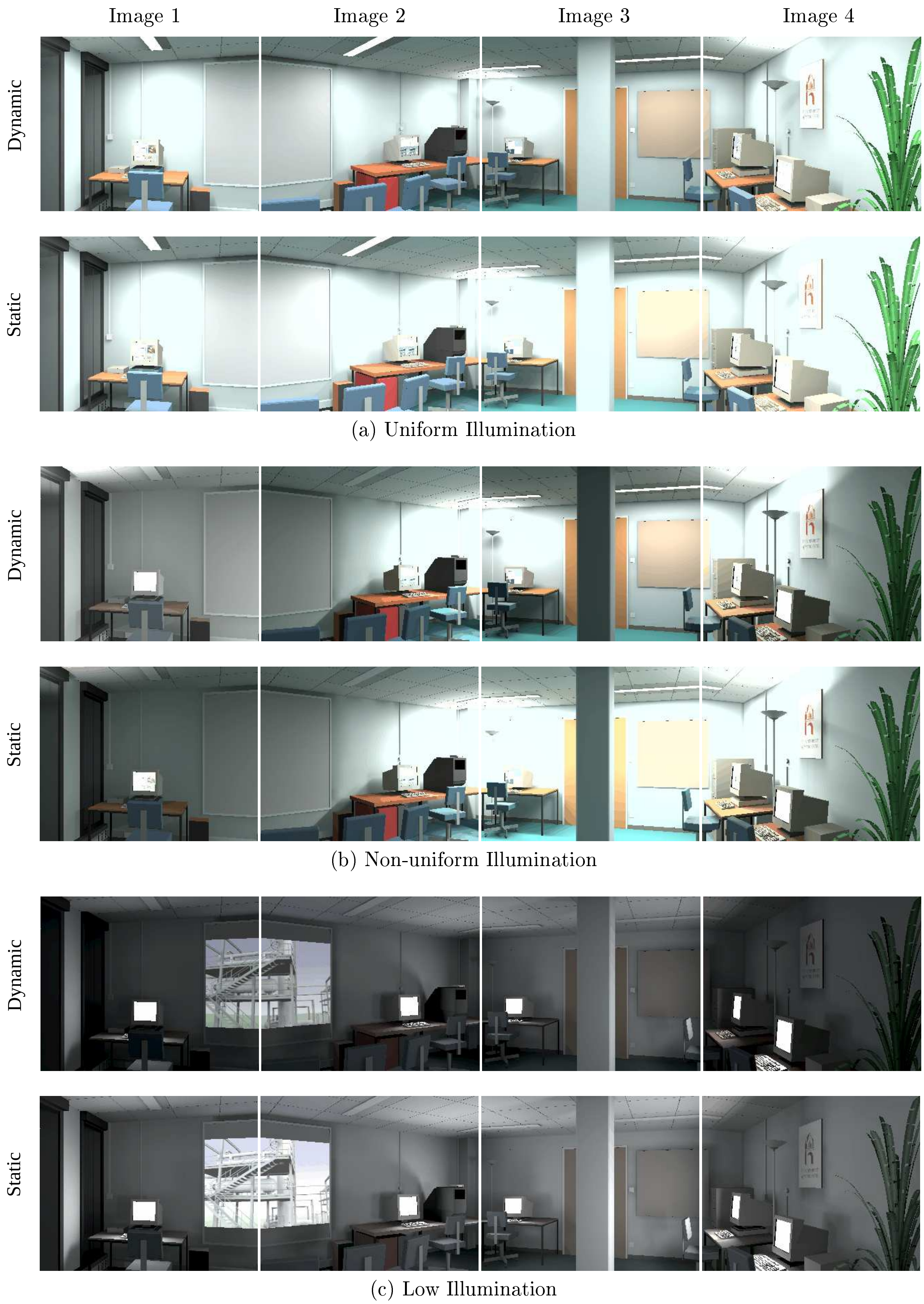
No implementation has been done to confirm these estimates, but performing dynamic tone-reproduction mappings during a real-time walk-through of the scene, especially when combined with the time-dependent adaptation changes described by Ferwerda *et al.* [FPSG96] is an interesting area for future research. We have instead opted for a simpler approach of estimating a single adaptation level suitable for the entire scene, as will be described in the next section. Tone-reproduction is then performed as a pre-process to rendering, as this places fewer computational limits on the size of scene that can be displayed.

3.7.1 Static Adaptation

When a single image of a radiosity solution is being processed, the adaptation level that is used to generate the tone-reproduction operator can be thought of as *dynamic*, in the sense that for an observer looking at a fixed surface point, his adaptation level will change depending on the position from which he is looking. In order to generate a single adaptation level for a view-independent radiosity solution, we need to assume that the adaptation induced at a point remains *static*, regardless of the luminances of other surface that are visible from any position in the scene. This means that a tone-reproduction transformation may be applied to the luminances stored at the vertices of the mesh elements as a pre-process, and the radiosity solution rendered from any viewpoint without recalculating the vertex colours at each frame.

Once a radiosity solution has been calculated, a static real-world adaptation level may be determined using any of the methods outlined in Section 3.4.4. We have found that a simple average of all element luminances generates a static adaptation level that more closely matches dynamic levels over a wide range of conditions than when a logarithmic average is used.

Figure 3.13 compares the results of applying Ferwerda's tone-reproduction operator to a radiosity



(a) Uniform Illumination

(b) Non-uniform Illumination

(c) Low Illumination

Figure 3.13: Comparison of static and dynamic adaptation levels, using Ferwerda's tone-reproduction operator.

Sequence	Image 1	Image 2	Image 3	Image 4	Static
(a)	79.11	99.37	117.05	95.82	68.26
(b)	7.57	28.56	88.25	44.19	29.46
(c)	2.98	2.40	2.26	4.21	0.96

Table 3.2: Adaptation levels for the images in Figure 3.13, measured in $cd\,m^{-2}$.

solution when using a static adaptation level against a dynamic one. Figure 3.13(a) shows a radiosity solution in which the overall level of illumination is approximately uniform throughout most of the scene. The camera for each image was panned across the scene, mimicing the real-world observer changing his viewing direction. The upper sequence of images shows the results when a dynamic adaptation level was used for each frame, whereas the lower sequence uses a static level. Table 3.2 shows the dynamic and static adaptation levels (measured in $cd\,m^{-2}$) for the images in each sequence. Because the illumination in the environment is approximately uniform, the static adaptation level is a acceptable estimate of each dynamic level. The overall brightness, however, appears slightly greater since (for this particular case) the static adaptation level is less than that of any individual image.

Figure 3.13(b) shows similar sequences for a radiosity solution with a less uniform overall level of illumination. In this case, the static adaptation level is nearer the average dynamic level, but the variance in dynamic levels means that the view-independent tone-reproduction appears noticeably different in images 1 and 3. Finally, Figure 3.13(c) shows image sequences for a radiosity solution with a relatively low level of illumination. Even though there is less variation in terms of absolute luminance levels between the images, the increased sensitivity of the visual system means that differences between the sequences are more visible. Most apparent is the overall increase in the brightness of the image – at these low adaptation levels, small differences in adaptation luminance can greatly affect rod sensitivity.

Overall, we have found that the results obtained using view-independent tone-reproduction are satisfactory, especially when considering that the comparisons shown above are not entirely fair since the display observer will never actually perform a side-by-side comparison of static and dynamic adaptation. All images shown in the following chapters have been computed using this view-independent tone-reproduction method.

3.8 Summary

In this chapter, we have discussed various methods of transforming the luminance values calculated by a global illumination algorithm in such a way that they can be displayed on a display device with limited dynamic range. This process, known as tone-reproduction, is essential to the production of realistic-looking images, and the most successful methods attempt to mimic the psycho-visual processes that occur in human vision so that the subjective appearance of the image matches that of the real scene. Several different tone-reproduction operators have been discussed and compared, and methods for applying them to view-independent solutions have also been presented. In the next chapter, we will see how view-independent tone-reproduction operators may be employed to calculate perceptually-based error estimates during the radiosity simulation, allowing us to generate a view-independent radiosity solution more efficiently than when radiometric measures are used.

Chapter 4

Perceptually-Driven Radiosity

4.1 Introduction

When computing radiosity solutions for models containing hundreds of thousands or millions of surfaces, large amounts of CPU time and memory are required. As discussed in Chapter 2, many researchers have proposed algorithms that are capable of accelerating the radiosity process, the most successful of which are progressive refinement [WEH89], and cluster-based hierarchical radiosity [HSA91]

All of the radiosity algorithms discussed previously have aimed to generate solutions that are *radiometrically accurate* – the actual radiosities that are calculated at each point in the scene are as close as possible to the 'exact' values that would be calculated by the diffuse global illumination equation given in Section 2.3.1. Radiometric accuracy is essential if the final solution is being used for traditional radiative transfer applications such as lighting analysis, where an accurate representation of luminance is required over each surface. Producing such accurate results using these algorithms usually involves reducing the thresholds that control the simulation process to a suitably low level so that no artifacts appear in the final solution. Decreasing these thresholds causes the simulation to spend more time and memory on generating a solution whose radiometric values will be nearer the exact values that are required. The selection of threshold levels involves both experience and understanding of the algorithms as well as a certain amount of guesswork. There is also no guarantee that the choice is optimal, that is, that all visible errors have been eliminated without excessively increasing the computation time.

When the final goal of the radiosity simulation is a solution or set of images that will be viewed by a human observer, such a high level of numeric accuracy is not always necessary. What is more important in these situations is how accurate the images *appear* to be when viewed by the user. In this chapter, we propose a new approach to accelerating radiosity computations that aims to produce solutions that are visually accurate, rather than fully accurate in the radiometric sense. The fundamental observation upon which this work is based is that a large proportion of the computations performed during the course of generating a solution have little visible impact on the final image when considered individually. For the progressive radiosity algorithm discussed in Section 2.6.2, only the radiosity from the initial light-sources and main secondary reflectors is likely to cause visible shadows, with the remainder of the radiosity transfers contributing only to the overall level of illumination. By ensuring that only those radiosity transfers that cause *perceptually important* changes in the appearance of the solution are computed to any great accuracy, we will show how a useful solution may be generated more efficiently. Moreover, with this approach, the decreases in computation and memory requirements achieved do not manifest itself as a corresponding decrease in visual accuracy, as often occurs with traditional radiometric-based algorithms.

Our approach is based on the *tone-reproduction* methods described in the previous chapter. Until now, tone-reproduction operators have only been used to post-process images of global illumination solutions in order to recreate as accurately as possible what the user would see when viewing a real scene. This involves calculating how the real-world luminances computed by the global illumination algorithm should be mapped to intensity values on an output device.

In order to determine the perceptual impact of changes in the radiosity solution over the course of the simulation, we need to know how differences in real-world luminance correspond to differences in the image, as perceived by the display observer. This involves estimating how the real-world luminances will be transformed to frame-buffer values *before* the radiosity solution has been generated. If we can produce such a tone-mapping, the perceptual impact of changes to real-world luminances may be determined. Perceptual knowledge can then be incorporated into the simulation, increasing efficiency by distinguishing between those aspects of the solution that are visually important and those that are not.

Rushmeier suggested that such a method could be employed when rendering an image of a radiosity solution using a Monte-Carlo based second pass [Rus94, RW94]. A coarsely rendered image of the radiosity solution was first produced to enable a simple linear-scaling tone-reproduction operator to be derived. This operator mapped the average real-world luminance in the image to the average display value of the frame-buffer. The mapping was then used to predict the number of samples required to ensure that the estimated error for radiance calculations associated with each pixel in a high-resolution rendering was less than 1 part in 255 when displayed on using an 8-bit frame-buffer. The result of this calculation was a reduction in the computation time for the image by a factor of 5.5, without any visible degradation in quality. This approach was limited to generating images of an already computed radiosity solution using a simple, perceptually unrealistic tone-reproduction operator, and no perceptual knowledge was used in the first pass when the view-independent radiosity solution was generated.

In this chapter, we show how tone-reproduction algorithms can be used to accelerate the generation of *view-independent* radiosity solutions, and outline some of the benefits perceptual awareness brings. This work is based on, and extends previously published research by the author [GH97].

The remainder of this chapter is presented as follows: Section 4.2 describes some of the problems associated with current radiosity methods, and shows how a perceptually-based algorithm can be used to overcome them. Section 4.3 then presents the core of our perceptually-driven radiosity algorithm – a technique for measuring the perceived difference between two real-world luminances. Our algorithm requires the *a priori* construction of a tone-reproduction operator, and an approach for estimating such an operator is given in Section 4.4. Following that, Sections 4.5 and 4.6 discuss how perceptual knowledge can be used to generate a radiosity solution more efficiently, and presents experimental results regarding the performance of the new algorithm.

4.2 Motivation

We have chosen a progressive shooting algorithm rather than a hierarchical gathering algorithm such as [HS90, HSA91] primarily because it allows for more immediate feedback on the progress of the solution, as well as requiring an amount of memory that is linear in the number of elements in the scene. Progressive radiosity has also been shown to be more efficient than surface-based hierarchical radiosity for large environments [WH97a, WH97b]. Although cluster-based hierarchical algorithms have proven very useful in quickly generating radiosity solutions [Gib95, GH96], their methods of error control are not entirely reliable. As will become clear below, our adaptive refinement algorithm is based upon the error in the *representation* of radiosity over each element, rather than the error in energy *transport*. Controlling error in this way for current cluster-based hierarchical approaches is not possible without reducing the benefits clustering brings.

Considering each iteration of a progressive radiosity algorithm, radiosity must be transferred from a source patch to each receiver. Adaptive refinement attempts to increase the accuracy of the solution in areas where the current representation of radiosity is deemed to be inadequate. A refinement criterion and user-specified threshold are used in order to decide whether or not each receiving patch needs subdividing. Typical criteria that have been proposed, such as the difference between the radiosities of an element and its neighbours [CGIB86] or the differences in radiosity gradient [VP91], were discussed briefly in Section 2.6.2.

These refinement oracles do not, however, account for the way in which the radiosity solution will be viewed. The mechanism by which real-world luminances (or equivalently, radiosity values) are perceived by the display observer is highly non-linear, and any error metric based solely on radiometric comparisons cannot guarantee that additional errors will not be introduced during the display process. We have already seen in Chapter 3 that a particular difference in absolute luminance may be perceived very differently depending on the level of adaptation of the viewer. Differences in radiosity values that pass the refinement test (and therefore cause subdivision) may in fact be undetectable after the tone-reproduction transformation has been applied. If we are aiming to produce a solution that is only *visually* accurate, this extra refinement was in fact unnecessary because no extra *visible* detail was introduced. This often occurs around bright light sources, where areas with luminancse that are large (relative to the real-world adaptation level) appear 'washed-out' when displayed. Similarly, a radiometric refinement oracle may also fail to recognise where subdivision is needed, causing visible artifacts in the final solution.

Reducing the radiometric refinement threshold by a sufficient amount will generally eliminate any artifacts, but might also cause increased subdivision in areas that do not need it. This raises the question of how to select a suitable threshold – the user requires knowledge and experience of the specific radiosity algorithm and refinement technique. The threshold should also depend upon the tone-reproduction transformation applied when viewing the solution. Consider, for example, a radiometric refinement algorithm that subdivides an element whenever the difference between the luminance of any of its vertices is greater than 1 $cd\ m^{-2}$. If the overall level of illumination is in the photopic region, this difference may not be noticeable after tone-reproduction because the real-world observer will be adapted to a relatively high luminance. The threshold will therefore be small enough to ensure visible errors are eliminated. If the overall level of illumination is in the mesopic or scotopic regions, however, increased rod sensitivity may mean that a difference of 1 $cd\ m^{-2}$ is very noticeable, and a smaller threshold is needed. When using perceptually-based transformations, we will show how a visually reliable refinement criterion can be employed, and a more intuitive threshold level selected.

Another important part of each radiosity transfer is form-factor computation, the most time consuming aspect of which is determining the fraction of one patch that is visible from another (Section 2.5.2). In many implementations this involves casting a number of rays between the two patches, intersecting each ray with any potentially occluding objects, and counting the number of rays that are found to be unoccluded [HSA91]. For complex environments, visibility computations are often the most computationally intensive phase of the solution process, since a large number of rays must be used for each test if accurate form-factors are required. Reducing the number of rays will reduce the computation time, but can also introduce unpleasant artifacts in the radiosity solution such as noise in the radiosity representation where smooth gradients should occur [WEH89], or the absence of small shadow features.

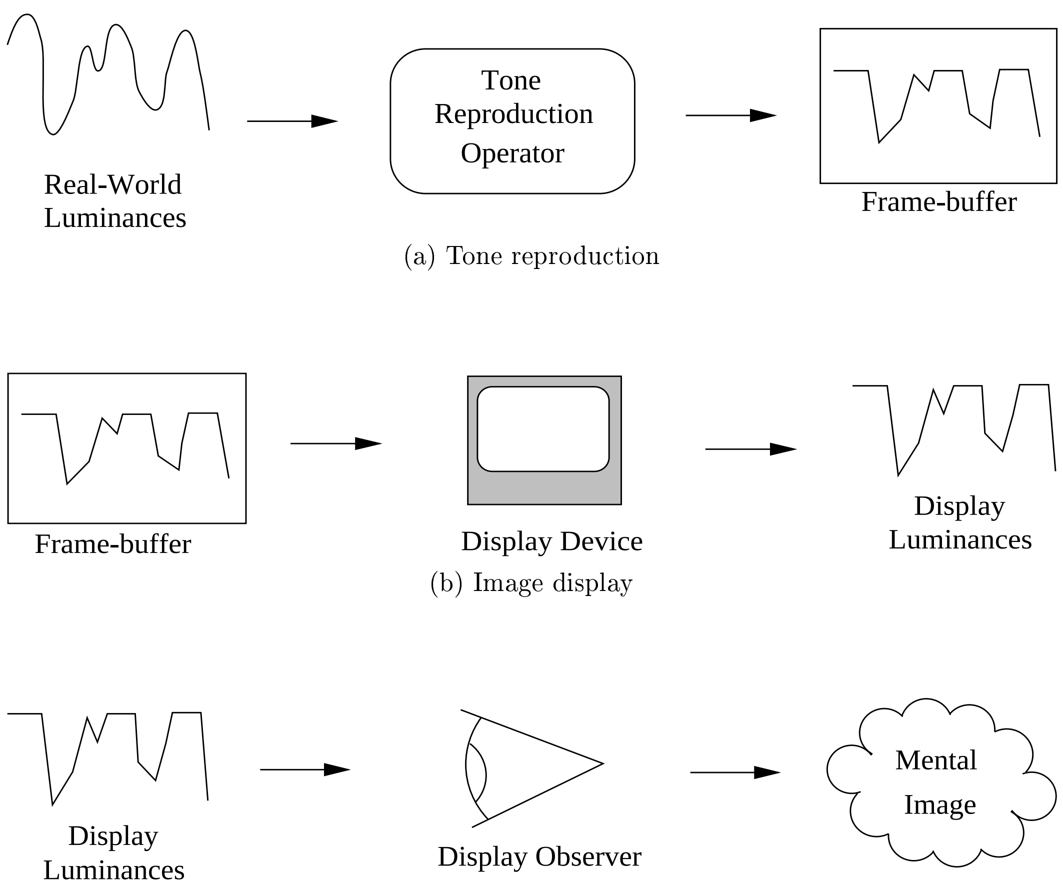
Not all shadows are as important as others, however. Over the course of an entire radiosity simulation, only a very small percentage of patches that distribute their energy to the rest of the environment will actually cause visible shadows. For a particular source and receiver pair, we will show how to determine the *perceptual impact* that the energy being transferred from the source has on the receiver, and how the perceptual importance of any potential shadow can be estimated before the visibility test is performed. Shadows that will not be visible can therefore be tested for very cheaply, since obtaining a more accurate visibility estimate will not increase the visual accuracy of the solution.

All of these techniques rely on being able to measure the difference between two real-world luminance values, as perceived by the display observer. In the next section we will consider this problem and propose a solution based on the principles of tone-reproduction.

4.3 Perception of Real-World Luminances

We wish to ensure that, when viewed by a human observer, the radiosity solution has no visible errors. In order to achieve this, we must guarantee that any radiometric differences between the radiosity representation and an exact solution do not manifest themselves as perceivable errors when the solution is displayed on a monitor screen or other display device.

Figure 4.1 shows the transformations that occur when the real-world luminances calculated by a global illumination algorithm are shown on a monitor screen and seen by a human observer. The first stage is shown in (a) – a tone-reproduction mapping from real-world luminances to frame-buffer values. The frame-buffer values are then used to control the voltage of the electron gun in the CRT, which



(a) Tone reproduction

(b) Image display

(c) Image viewing

Figure 4.1: The process of perceiving real-world luminances values

.

in turn cause pixels on the screen to radiate light with particular display luminances (b). Finally, the display luminances are seen by the display observer, and a mental image is formed (c).

If we wish to measure the perceived differences between real-world luminance values, we must in fact consider the perceived difference between the display luminances on the display device. For all experiments detailed in this chapter, the tone-reproduction operator proposed by Ferwerda *et al.* [FPSG96] (and described in detail in Section 3.4.3) has been used, although the techniques are equally applicable to other single-scale operators. A transformation $\mathcal{F}()$ may be derived using tone-reproduction techniques that calculates the *actual* luminance, L_{da}, that is output from the display device

$$L_{da} = \mathcal{F}(L_w) \tag{4.1}$$

Note that Equation 4.1 is different from those relating real-world to display luminance given in the previous chapter (Equations 3.20, 3.26, and 3.33) since $\mathcal{F}()$ calculates the actual (rather than required) display luminance, and therefore includes the clipping of out-of-range luminances to the displayable range.

The calculation of $\mathcal{F}()$ begins with the relationship between real-world and required display luminance (L_{dr}) given by Ferwerda's model (from Equation 3.33)

$$L_{dr} = L_{d(p)} + k(L_{a(w)})L_{d(s)} \tag{4.2}$$

A vector of required display luminances, $\overline{L}_{dr} = (L_{dr_X}, L_{dr_Y}, L_{dr_Z})$, in CIEXYZ colour space, can then be obtained using the chromaticity preserving transformation described in Section 3.6. The frame-buffer values required to produce this luminance are

$$\overline{n} = \left[\frac{\overline{L}_{dr} - \overline{L}_{amb}}{\overline{L}_{dmax}}\right]^{1/\gamma} \tag{4.3}$$

$\overline{n}$ must then be transformed into the RGB colour space of the target monitor, and each value clipped to the range $[0 \ldots 1]$. A vector of the actual luminances that are displayed may then be determined (as in Equation 3.3) by applying

$$\overline{L}_{da} = \overline{L}_{amb} + \overline{L}_{dmax}\overline{n}^{\gamma} \tag{4.4}$$

and then transforming $\overline{L}_{da}$ back into CIEXYZ colour space.

In order to quantify the difference between two real-world luminances, as perceived by the display observer, we must measure the perceptual difference between the actual display luminances that are calculated for each. Most colour spaces (including CIEXYZ) are perceptually non-uniform – equal numeric differences do not correspond to perceptually equal changes in colour. In 1931, the CIE defined two alternative colour spaces, known as CIELAB and CIELUV which are more perceptually uniform. These colour spaces are discussed in more detail in Appendix A. Unlike CIELAB colour space, the CIELUV system was developed for representing colours when viewing additive sources of light [CC83], and so is the colour space most applicable to measuring differences on self-luminous objects such as CRT screens. The transformation to a triple of values (L^*, u^*, v^*) in CIELUV space is given with reference to a standard white, (X_w, Y_w, Z_w), scaled so as to make $Y_w = 100$:

$$\begin{aligned} L^* &= \begin{cases} 116(Y/Y_w)^{1/3} - 16 & \text{if } Y/Y_w \geq 0.008856 \\ 903.3(Y/Y_w) & \text{otherwise} \end{cases} \\ u^* &= 13L^*(u' - u'_w) \\ v^* &= 13L^*(v' - v'_w) \end{aligned} \tag{4.5}$$

where

$$u' = \frac{4X}{X + 15Y + 3Z} \tag{4.6}$$

$$v' = \frac{9Y}{X + 15Y + 3Z} \tag{4.7}$$

and u'_w and v'_w are defined similarly. The transformation of two colours from CIEXYZ to CIELUV colour space allows the perceptual difference between them to be determined using a *colour difference formula*. In CIELUV colour space, the Euclidean distance between the two colours $L^*_A u^*_A v^*_A$ and $L^*_B u^*_B v^*_B$ gives a measure of their perceptual similarity

$$E^*_{uv} = \sqrt{(L^*_A - L^*_B)^2 + (u^*_A - u^*_B)^2 + (v^*_A - v^*_B)^2} \tag{4.8}$$

To summarise, the perceived difference between two real-world luminance values can be quantified using the following steps:

- the application of a tone-reproduction operator to determine the actual displayed luminances;
- the transformation of the displayed luminances into a perceptually-uniform colour space;
- the application of a colour difference formula to calculate E^*_{uv}.

This process will be denoted by the operator $\mathcal{P}()$ when it is used below.

The minimum value of E^*_{uv} that is visible to an observer is known as a *just-noticeable difference*, ΔE^*_{uv}. Real-world luminances that have a perceived difference of less than ΔE^*_{uv} are classified as perceptually similar – there is no apparent difference between them when they are viewed by a human observer. Various values for ΔE^*_{uv} have been proposed in the literature, ranging between 1 and 10 [SFB90, Bar90, Red97] (see Appendix A). The larger the value of ΔE^*_{uv}, the greater number of real-world luminance values will be classified as perceptually similar. As we will show later on, increasing this value will decrease the apparent accuracy of the solution, but also reduce computation time. A balance must therefore be drawn between the quality of the solution and its computation time. The actual value below which colours are truely perceptually similar is very difficult to obtain, since it will vary between viewers and is also affected by viewing conditions. Imperfections in the perceptual-uniformity of the colour space mean that just-noticeable differences also change slightly depending on the chromaticity of the colours being tested. We have therefore chosen the minimum difference that has been reported in the literature ($\Delta E^*_{uv} = 1$), and this has proven satisfactory in all situations we have encountered.

4.4 Tone-Reproduction Estimation

Applying the previous discussion to the problem of measuring the perceived differences between real-world luminance values during the radiosity solution requires that a suitable tone-reproduction operator be available *before* the simulation is performed. Since a real-world adaptation level is required to calculate the tone-reproduction transformation, obtaining an accurate tone-mapping is not possible until the radiosity solution has been completed. All of the tone-reproduction algorithms that were presented in Chapter 3, however, are insensitive to relatively small changes in the real-world adaptation level. It has also been noted that the human visual system is not sensitive to absolute luminances, but rather relative differences [Gla95]. If we are able to estimate the static real-world adaptation level so that the resulting transformation reproduces the relative levels of illumination acceptably, we will obtain a final solution that matches (perceptually) that obtained using the correct *a posteriori* level.

In order to approximate the static real-world adaptation luminance, $L_{a(w)}$, we need an estimate of the average luminance a patch will have after the radiosity solution has converged. We propose using a technique similar to that for the *ambient display term* [CGIB86] for progressive radiosity that was derived in Section 2.6.2.

The average self-emitted luminance for the environment is calculated

$$\overline{L_e} = \frac{\sum_i A_i L_{e_i}}{\sum_i A_i} \tag{4.9}$$

where L_{e_i} is the self-emitted luminance of a patch i with area A_i. Using the area-averaged reflectivity of the environment,

$$\overline{\rho} = \frac{\sum_i A_i \rho_i}{\sum_i A_i} \tag{4.10}$$

Sequence	Image 1	Image 2	Image 3	Image 4	Static	Estimated
(a)	79.11	99.37	117.05	95.82	68.26	59.81
(b)	7.57	28.56	88.25	44.19	29.46	24.37
(c)	2.98	2.40	2.26	4.21	0.96	0.81

Table 4.1: Adaptation levels for the images in Figure 4.2, measured in $cd\ m^{-2}$.

where ρ_i is the diffuse reflectivity of patch i, an *ambient illuminance* value can be found that represents the average illuminance of a patch after an infinite number of bounces of light [CGIB86]

$$E_{amb} = \frac{\overline{L_e}}{1 - \overline{\rho}} \tag{4.11}$$

The ambient illuminance, E_{amb}, allows the estimation of the luminance of patch i in a fully converged solution

$$\widetilde{L_i} = L_{e_i} + \rho_i E_{amb} \tag{4.12}$$

Finally, to obtain an *a priori* estimate of the final static real-world adaptation level, we calculate the average estimated luminance of all patches in the environment

$$L_{a(w)} \approx \frac{\sum_i A_i \widetilde{L_i}}{\sum_i A_i} \tag{4.13}$$

Figure 4.2 shows how tone-reproduction using this estimated adaptation level compares to the images obtained using *a posteriori* static adaptation. The environment used for this experiment is the same as in Figure 3.13 (see Section 3.7.1). Again, two sequences of images are shown for three different illumination conditions. The overall results of tone-reproduction using the estimated adaptation level are very similar to those using a static level. Table 4.1 shows the adaptation levels in $cd\ m^{-2}$ for each sequence, as well as the static levels from Table 3.2. Overall, the results of this estimation technique have proved acceptable in most situations to which it has been applied, especially when compared to the static adaptation technique that was applied to view-independent solutions in Section 3.7.1. Problems do exist, however, when judging results at low levels of illumination against dynamic adaptation, due to the increased rod sensitivity. This means that small changes in the calculated adaptation level can significantly affect the overall brightness of the image, as can be seen by comparing sequence (c) in Figures 4.2 and 3.13.

4.5 Adaptive Refinement

Our radiosity algorithm uses a progressive refinement solution method [CCWG88], coupled with adaptive refinement to select an appropriately accurate representation of radiosity over each surface. In the progressive algorithm we have employed, radiosity values are calculated directly at the vertices of each mesh element [WEH89] in order to avoid an interpolation step before the elements may be displayed (Section 2.7.1). Element subdivision is performed hierarchically, as in the algorithm developed by Hanrahan and Salzman [HS90, HSA91].

In any radiosity algorithm, adaptive refinement must be carefully controlled in order to obtain an accurate radiosity solution without excessively increasing the complexity of the element mesh. The adaptive refinement method we present in this section is capable of halting subdivision once the differences between successive levels of refinement become perceptually unnoticeable. When this occurs, any extra refinement is unnecessary since it does not introduce extra visible detail into the solution.

Recently, Martin *et al.* have developed an image-space refinement criterion for hierarchical radiosity that is based on the principles of tone-reproduction [MPT97], although the work presented in this thesis was performed independently. Their approach consisted of using a tone-reproduction operator

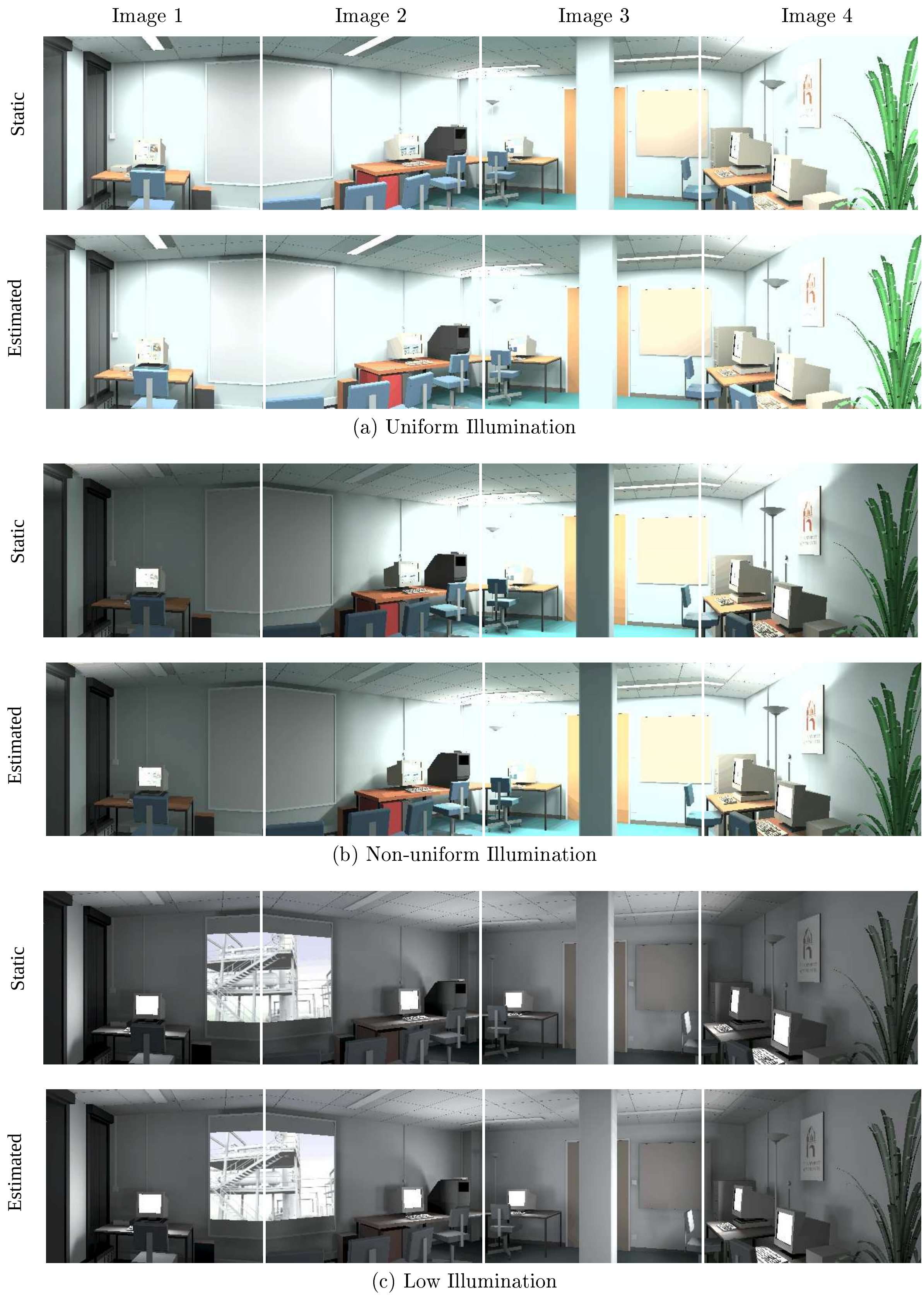


Figure 4.2: Comparison of estimated and static adaptation levels, using Ferwerda's tone-reproduction operator.

to base element refinement decisions on the expected smoothness of the Gouraud shading across the edges between elements. The oracle function measures the change in continuity over each edge in the mesh, refining adjacent elements whenever the error exceeds a threshold. Although tone-reproduction is applied, no description is given of how the adaptation levels are generated before the simulation is performed, and their approach is limited to generating *view-dependent* radiosity solutions.

In the following sections we will discuss an an adaptive refinement algorithm that can be applied to the generation of view-independent radiosity solutions. The technique will be described for elements that are fully visible to a source patch in Section 4.5.1, and then for elements that occur along shadow boundaries in Section 4.5.2.

One problem with shooting algorithms such as progressive refinement is that the subdivision caused during one iteration may become unnecessary in later iterations. Consider for example a light-source casting a sharp shadow across a surface. Additional iterations may transfer radiosity in such a way that the original shadow boundary becomes 'washed-out' and is no longer visible. What is needed therefore, in addition to a refinement algorithm, is a mesh *simplification* technique that is capable of detecting if prior refinement has become perceptually unnoticeable. Section 4.5.3 describes how simplification of the mesh can be integrated into the refinement algorithm.

4.5.1 Element Refinement

For each refinement test, we need to construct a numerical measure that bounds the perceived difference in luminance between successive levels of element subdivision. Consider the case when no occlusion occurs between a particular source and receiver pair. In Figure 4.3(a), an element ABC is shown, with the magnitude of luminance at each vertex on the vertical axis, along with the radiosity representation obtained when the element is rendered using bilinear interpolation. (b) shows the same element after it has been subdivided into 4 child elements. The improved radiosity representation is also shown, along with new luminance values at vertices D, E and F.

The refinement test must be able to determine if subdividing element ABC will introduce any extra visible detail. Put another way, we wish to detect if the representation of radiosity between successive levels is perceptually different, using the method proposed in Section 4.3. Figure 4.3(c) shows the radiosity representation of element ABC overlayed on top of that of the child elements. Because radiosity is calculated directly at the vertices of each element, the luminance at vertices A, B and C will be the same in each case. The real-world luminances at these vertices (which are the luminances that will be used by the tone-reproduction operator when displaying the solution) are

$$\begin{aligned} L_w(A) &= L(A) + \rho(A)(F_{Aj}\Delta L_j + E_{amb}) \\ L_w(B) &= L(B) + \rho(B)(F_{Bj}\Delta L_j + E_{amb}) \\ L_w(C) &= L(C) + \rho(C)(F_{Cj}\Delta L_j + E_{amb}) \end{aligned} \tag{4.14}$$

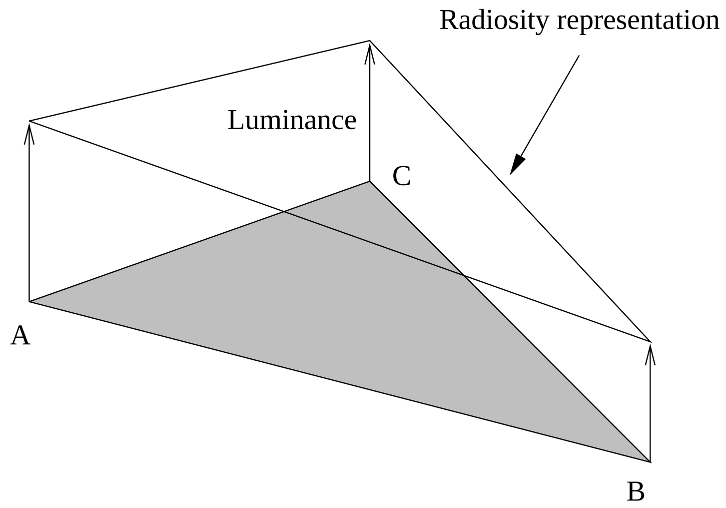
where L_w is the real-world luminance, L is the vertex luminance before this iteration, ρ is the reflectivity at the vertex, F_{Xj} is the analytic differential area to polygon form-factor between the vertex X and patch j, ΔL_j is the unshot luminance of patch j, and E_{amb} is the ambient illuminance. The ambient illuminance must be included to approximate the extra luminance received at the vertex during later iterations of the radiosity simulation. Without it, the initial iterations will produce subdivision that becomes perceptually unimportant during the later stages of the simulation.

If element ABC is not subdivided, the real-world luminances at the positions of vertices D, E and F will be

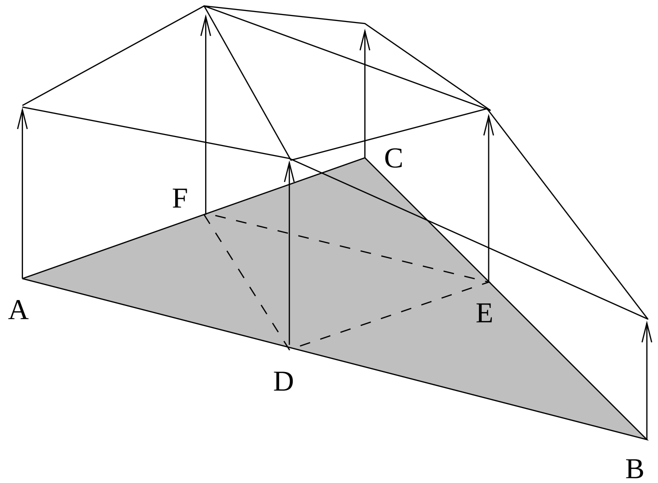
$$\begin{aligned} L'_w(D) &= 0.5(L_w(A) + L_w(B)) \\ L'_w(E) &= 0.5(L_w(B) + L_w(C)) \\ L'_w(F) &= 0.5(L_w(C) + L_w(A)) \end{aligned} \tag{4.15}$$

If the element is subdivided, however, these values will differ by ΔD, ΔE and ΔF (as shown on the right of Figure 4.3), and the real-world luminances will be

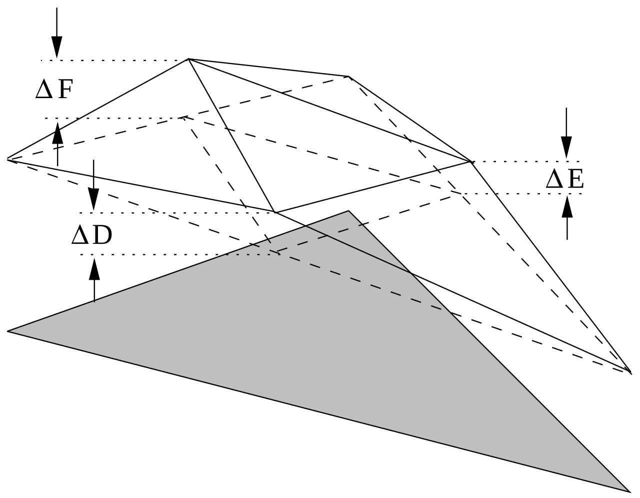
$$\begin{aligned} L_w(D) &= L(D) + \rho(D)(F_{Dj}\Delta L_j + E_{amb}) \\ L_w(E) &= L(E) + \rho(E)(F_{Ej}\Delta L_j + E_{amb}) \\ L_w(F) &= L(F) + \rho(F)(F_{Fj}\Delta L_j + E_{amb}) \end{aligned} \tag{4.16}$$



(a) Initial element ABC



(b) Element ABC after one level of refinement



(c) Difference between radiosity representations

Figure 4.3: Two successive levels of element refinement over a triangular element, and the difference between each level's representation of radiosity.

The perceived difference between the real-world luminances associated with, for example, vertex D with and without the extra subdivision may be written as

$$E^*_{uv} = \mathcal{P}(L'_w(D), L_w(D)) \tag{4.17}$$

where $\mathcal{P}()$ is the three-stage operator described in Section 4.3. Because bilinear interpolation is used to obtain luminance values at each point over an element, the largest perceived difference associated with vertices D, E and F will be an upper bound on the perceived difference between the luminances at all points over element ABC. The quantity

$$E^*_{uv} = \max(\mathcal{P}(L_w(D), L'_w(D)), \mathcal{P}(L_w(E), L'_w(E)), \mathcal{P}(L_w(F), L'_w(F))) \tag{4.18}$$

can therefore be used to determine if increasing the level of subdivision will cause a perceptually noticeable improvement. If E^*_{uv} is less than the minimum just noticeable difference, ΔE^*_{uv}, then subdivision will not improve the perceived accuracy of the solution.

For quadrilateral elements, the same process may be applied by assuming that the quadrilateral is split into two triangular elements by connecting opposite corner vertices. Since the choice of which corners to connect is arbitrary (and in fact, should depend upon which corners have the most similar final radiosity values [ARJ90]), Equation 4.18 must be applied twice at the centre point, using edges connecting both opposite corner pairs, as well as for the midpoint of each perimeter edge, and the maximum of these values taken when calculating E^*_{uv}.

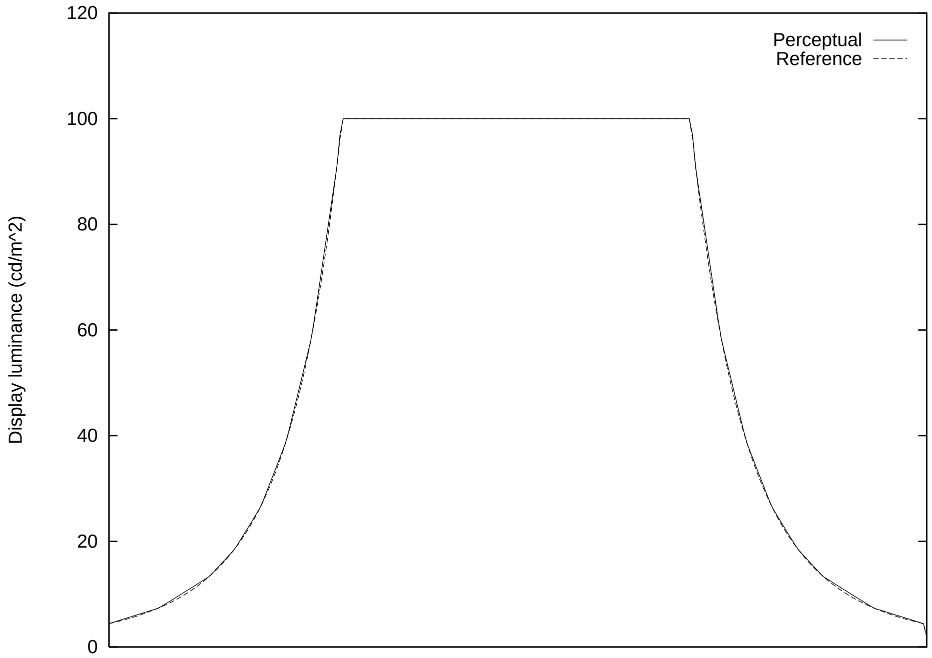
Figure 4.4 compares the real-world and display luminances generated using this refinement algorithm when applied to a one-dimensional 'surface' with a single light-emitting patch positioned above. In this case, the tone-reproduction operator produced a range of display luminances beyond the capabilities of the display device, causing many luminances to be clipped to the maximum displayable value (100 $cd\ m^{-2}$). Although the calculated display luminance in graph (a) closely matches that of the reference solution (which could be calculated analytically for this simple situation), differences can be seen in graph (b) between the real-world luminance and the reference, in those areas in which the display luminance has been clipped. This caused a slight energy loss in this case, which amounted to less than 0.4%.

In Figure 4.5, we compare the perceived quality of the radiosity representation over a two-dimensional surface, against that obtained using a radiometric refinement criterion. As a metric for the perceived similarity between two radiosity representations, we have chosen the average number of just noticeable differences between display luminances in CIELUV colour space over each image. Other measures, such as RMS error, do not account for the perceptual response of the human visual system, and so are only of use when radiometric accuracy is being measured. Although several monochromatic perceptually-based comparison metrics have been proposed by Rushmeier *et al.* [RWP$^+$95], our method was chosen due to its computational simplicity, and because the error values may be considered relative to the perceptually unnoticeable threshold of $\Delta E^*_{uv} = 1$.

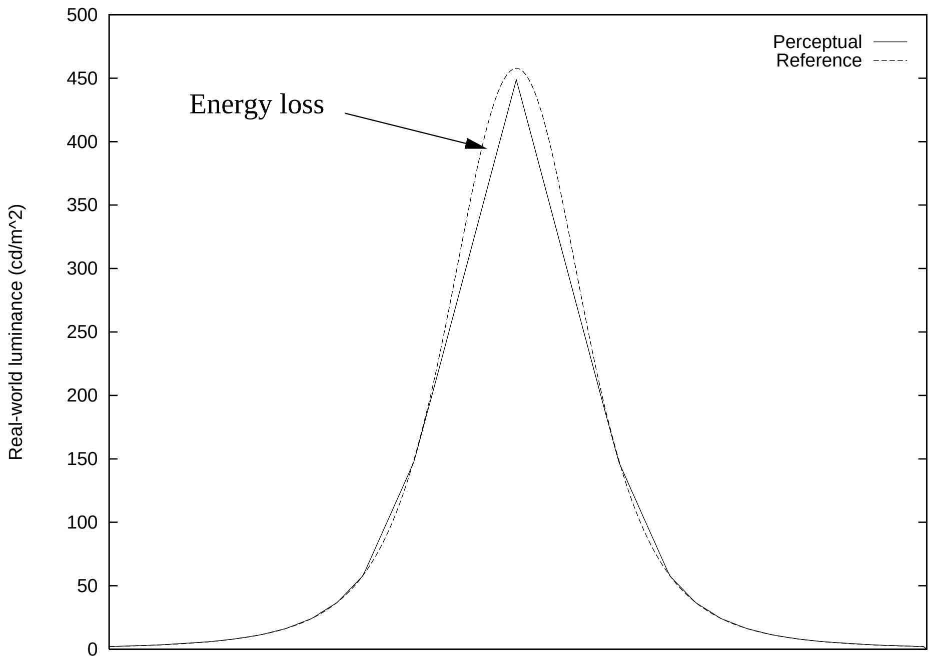
Figure 4.5(a) shows the display intensities and element mesh obtained using the perceptual refinement algorithm, along with a graph showing the number of CIELUV just-noticeable differences by which the intensities deviates from the reference solution. With an error threshold of $\Delta E^*_{uv} = 1$, the maximum difference was 0.98, and the average 0.16. Figure 4.5(b) shows the element mesh generated by a radiometric refinement algorithm that is identical to the perceptual algorithm, except that subdivision decisions are performed directly on real-world luminances, rather than using the tone-reproduction operator and transformation into CIELUV colour space. This mesh contains almost the same number of elements, but the maximum deviation of the radiosity representation from the reference solution is 2.15 and the average 0.38 [1]. In Figure 4.5(c), an element mesh with approximately the same deviation from the reference as was found using the perceptual algorithm is shown (maximum error of 1.02 and average of 0.17). This mesh contains 2.6 times more elements, most of which are in the central region where they become unnecessary once the tone-reproduction operator is applied.

Finally, we wish to compare the efficiencies of our refinement algorithm against other radiometric methods that are commonly employed in radiosity simulations, in order to see how the perceived error

[1] Although it is difficult to identify differences between these images, due to the poor quality of printing reproduction, Mach bands and linear intensity gradients can be clearly seen when the images are viewed on a monitor screen

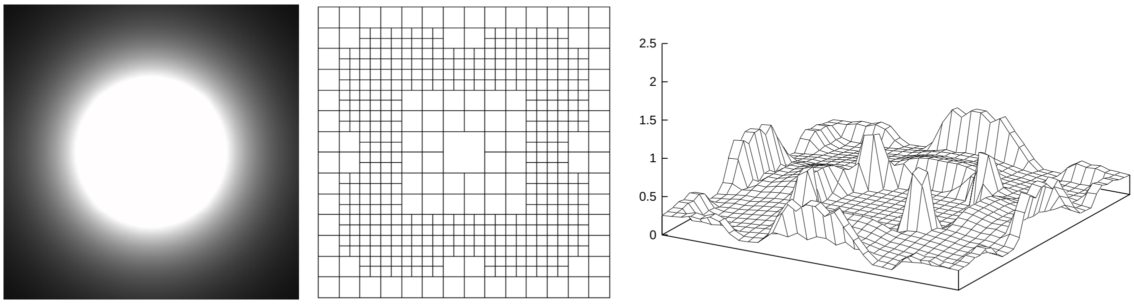


(a) Display luminance

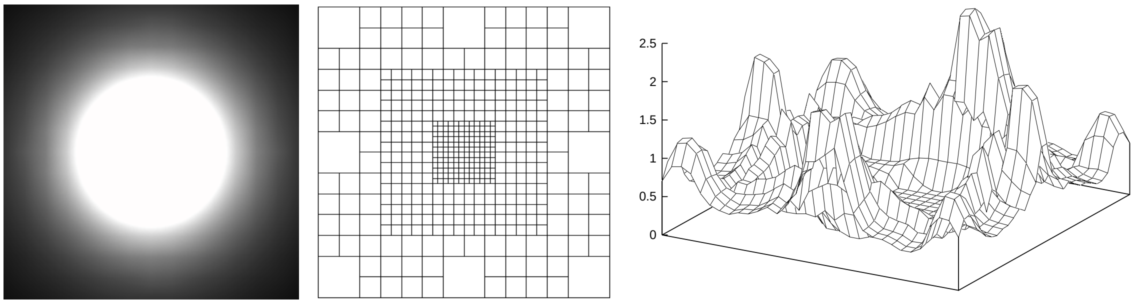


(b) Real-world luminance

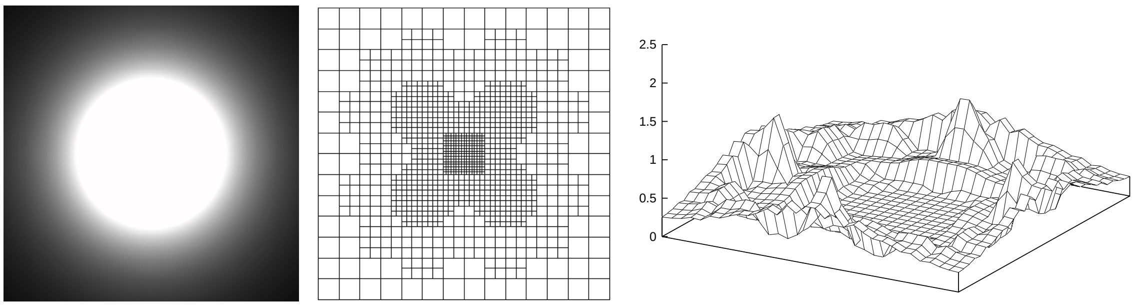
Figure 4.4: Display and real-world luminances calculated by the perceptual refinement algorithm.



(a) Perceptual refinement (626 elements)

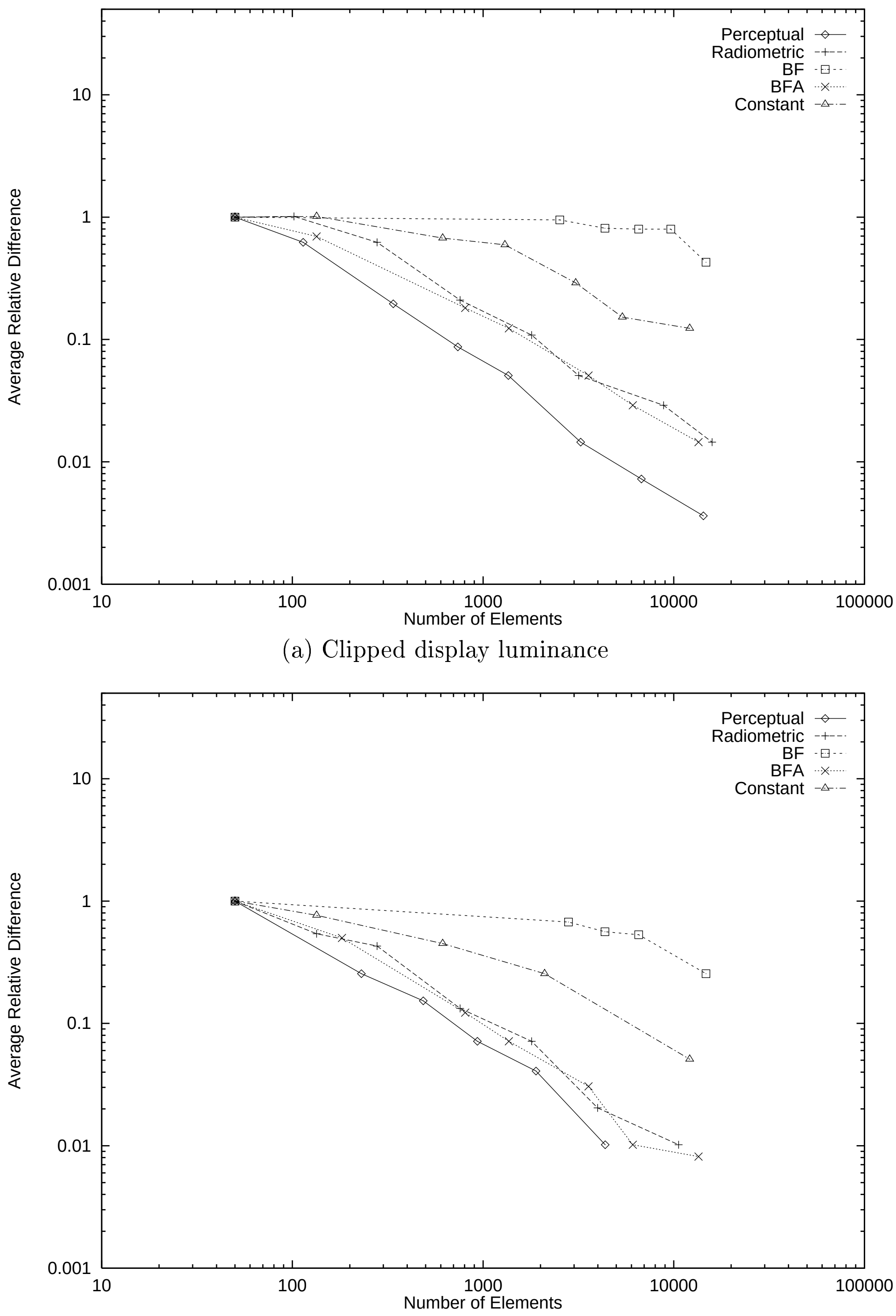


(b) Radiometric refinement (614 elements)



(c) Radiometric refinement (1638 elements)

Figure 4.5: Comparison of radiometric and perceptual refinement for a simple unoccluded scene.



(a) Clipped display luminance

(b) Unclipped display luminance

Figure 4.6: Comparison of the average relative perceptual difference for increasing numbers of elements when using different adaptive refinement oracles.

decreases as increasing numbers of elements are created. In Figure 4.6 we show the average number of just-noticeable differences between the approximate and reference radiosity solutions. Graph (a) was generated using the same geometry as in Figure 4.5, with the resulting radiosity function containing a region where the display luminances needed to be clipped to the displayable range. For graph (b), the radiosity of the light-source patch was reduced so that no clipping occured. Several refinement algorithms were compared:

- 'Perceptual' – the perceptually-based algorithm described above;
- 'Radiometric' – same as 'Perceptual', but refinement decisions are performed without the tone-reproduction transformation;
- 'BF' - Hanrahan's BF-refinement algorithm, where the error is assumed to be proportional to the product of the form-factor and source radiosity [HSA91];
- 'BFA' - a commonly used variant of BF-refinement, where the error is weighted by the area of the receiving element, tolerating larger errors for smaller elements;
- 'Constant' - the error is taken as the maximum difference between the radiosities of an element's vertices, in an attempt to keep the radiosity function constant over each element.

As the graphs show, when ranges of the display luminance are clipped, the perceptual algorithm clearly out-performs all others that were tested, due to its ability to recognise when refinement would not introduce extra visible detail (note the logarithmic scale on both the horizontal and vertical axes). When the display luminances did not require clipping, the perceptual algorithm was also able to produce a smaller error, although to a lesser extent when compared to its radiometric counterpart and BFA-refinement.

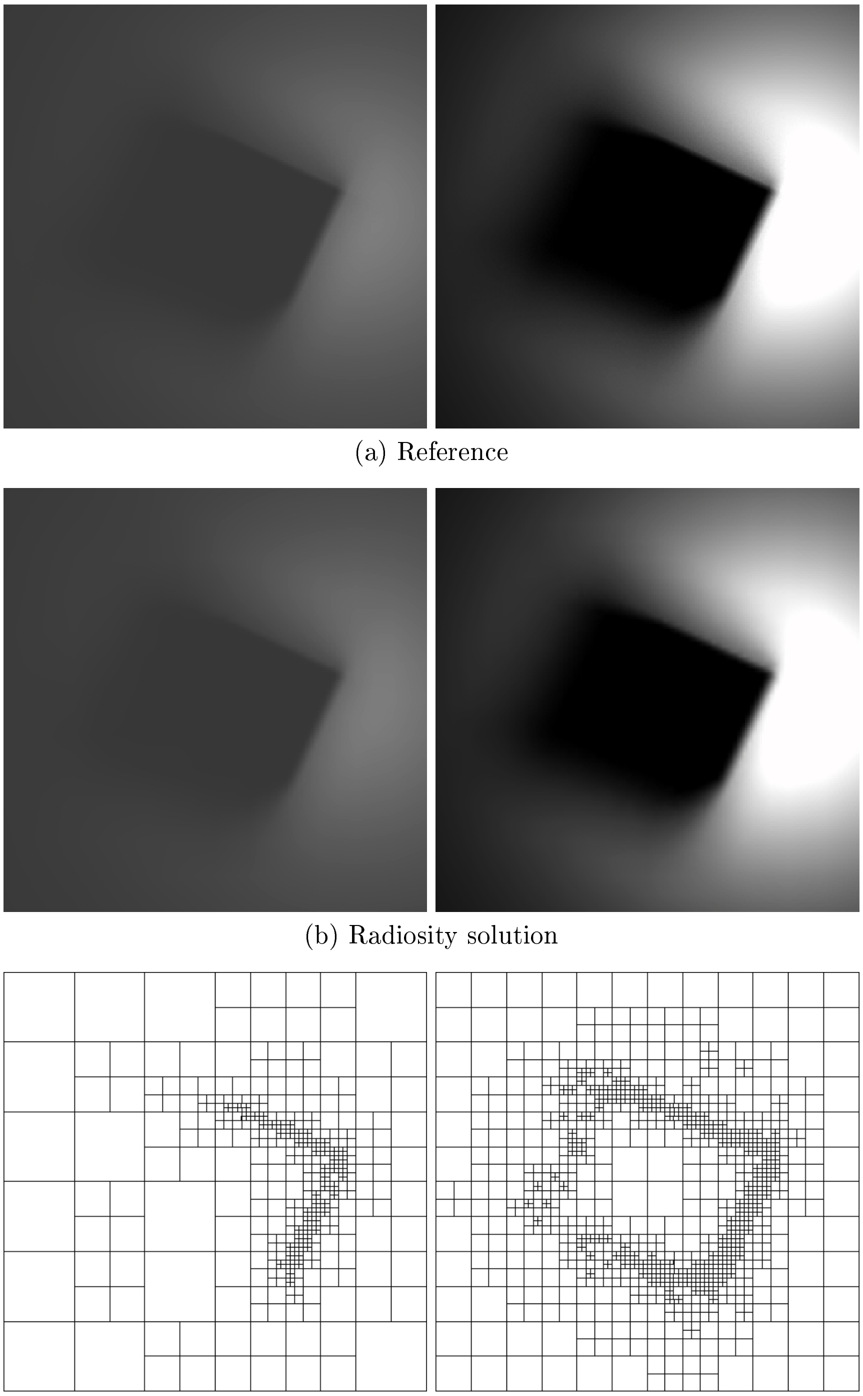
4.5.2 Shadow Boundary Refinement

Whenever the source patch is not completely visible, the radiosity representation over the receiver can exhibit many changes in continuity due to the variations in visibility across its area. In a similar fashion to the refinement technique introduced above, we wish to subdivide the element mesh whenever doing so will introduce extra visible detail. This involves re-evaluating a visibility term, as well as a form-factor, at each new vertex. The same refinement criterion (Equation 4.18) may then be employed, and the element refined if necessary.

Figure 4.7 shows the shadow cast by a cube onto a polygon when illuminated by a single area light-source. Two cases are shown – (left) when the contrast between the light and umbra regions is low, and (right) when the contrast is higher. For each shadow, a reference image is shown in (a), the radiosity approximation is shown in (b), and the element mesh used for that approximation is given in (c). The reference images were computed using a large number of rays to determine the visible fractions of the light source, and analytic form-factor computations. As can be seen by examining the element meshes, the refinement oracle only creates as much subdivision as is necessary to generate a visually accurate representation of radiosity. On the left side of the low contrast shadow, where the visual differences between the areas in shadow and those that are not is small, little refinement has been performed. The average number of just-noticeable differences between the approximate and reference solutions are 0.84 for the high contrast shadow, and 0.64 for the low contrast shadow. An error threshold of $\Delta E^*_{uv} = 1$ was used for both solutions.

4.5.3 Refinement Simplification

Adaptive refinement can produce meshes with a very large number of elements, especially when complex shadow boundaries are present. Even though we have used perceptually-based metrics to ensure that the mesh is computed efficiently for the current iteration, and included an ambient term in our comparisons (Equation 4.14), whenever many elements are generated to account for a sharp shadow boundary, later iterations may transfer radiosity in such a way that the high level of refinement becomes unnecessary.

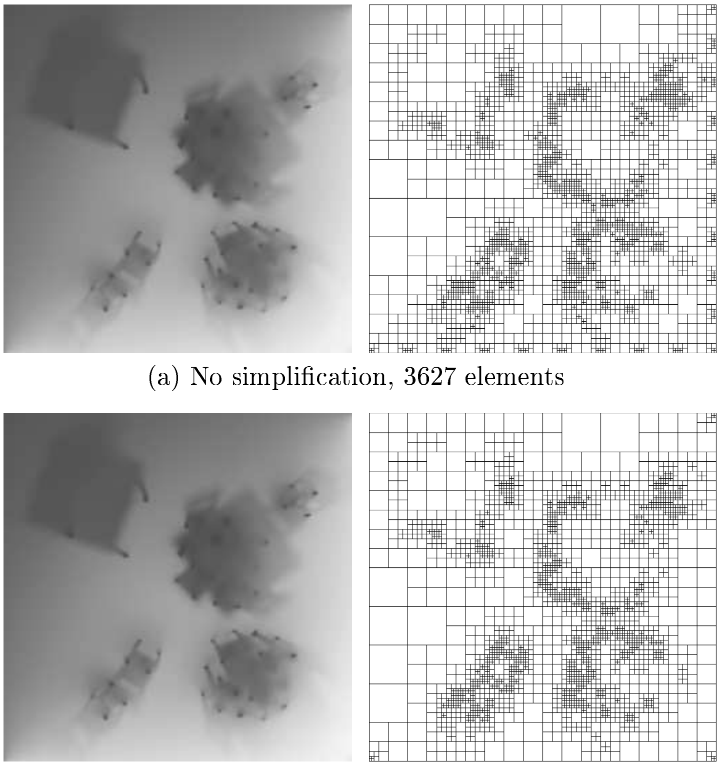


(a) Reference

(b) Radiosity solution

(c) Element mesh

Figure 4.7: Varying element refinement caused by (left) low and (right) high shadow contrasts. A reference image is shown in (a), the radiosity image in (b), and the element mesh in (c).



(a) No simplification, 3627 elements

(b) Simplification with $\Delta E^*_{uv} = 1$, 3135 elements (13.5% reduction)

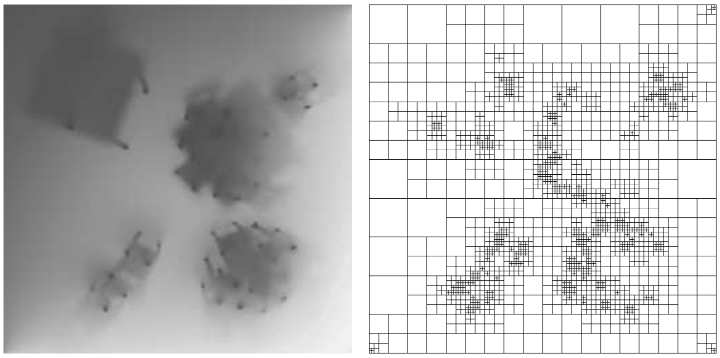
Figure 4.8: Comparison of radiosity representations generated with and without mesh simplification.

The same technique as was used for comparing the quality of the radiosity representation for successive levels of refinement may also be used to see if the elements immediately below the current element can be removed without any perceivable change in the solution. Given an element with four children, we can measure E^*_{uv} to compare the actual luminance values at each edge midpoint against the values interpolated between the vertices. If the maximum difference over all edges is below the just-noticeable threshold, then removing the child elements will not detract from the visual quality of the solution.

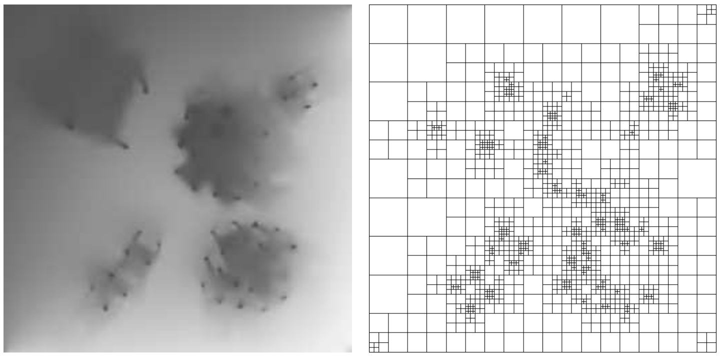
Figure 4.8(a) shows a rendered image and element mesh for the floor polygon in a test scene containing several light-sources and complex shadows cast by tables and chairs onto the floor. This scene is illuminated by a strong light-source in the centre of the room, as well as windows along the right-hand and lower walls. In total, 3627 elements were generated during the computation of the direct component of illumination. Figure 4.8(b) was generated using the simplification technique described above combined into the adaptive refinement algorithm. For this scene, a saving of 13.5% in the number of elements was achieved without any loss in image quality. The error, measured using the same technique as in Section 4.5.1, has a maximum value of 0.963 (below the just-noticeable threshold), and an average of only 0.013.

The proposed method of element mesh simplification can also be applied as a post-process to reduce the number of elements in the solution prior to rendering. Figures 4.9(a) and (b) show the results of simplifying the element mesh using thresholds of $\Delta E^*_{uv} = 2$ and $\Delta E^*_{uv} = 4$. The corresponding statistics are given in Table 4.2.

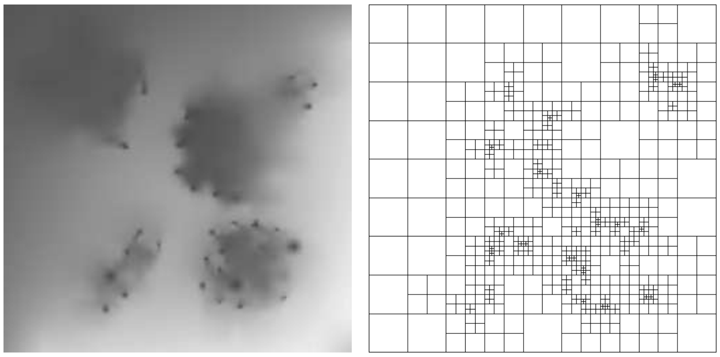
Whenever rendering speed is essential, such as in real-time walk-through applications, the simplification technique may be used to generate a mesh with a given number of elements. Figure 4.9(c) shows an element mesh containing 798 elements. This was generated by iteratively removing the child elements with the minimum perceived error until less than a given number (800 in this case) remained. Most of the important areas of shadow are retained, but the lower contrast shadows have been removed. Again,



(a) $\Delta E^*_{uv} = 2$, 2316 elements (36.1% reduction)



(b) $\Delta E^*_{uv} = 4$, 1617 elements (44.6% reduction)



(c) Target of 800 elements (78.0% reduction)

Figure 4.9: Comparison of radiosity representations generated using increasing simplification thresholds.

ΔE^*_{uv}	Original	Optimised	Reduction	Max. Error	Av. Error
1	3627	3135	13.5%	0.963	0.013
2	3627	2316	36.1%	1.929	0.070
4	3627	1617	44.6%	3.948	0.177
n/a	3627	798	78.0%	14.74	0.493

Table 4.2: Reduction in the number of elements, along with maximum and average perceptual errors for varying simplification thresholds.

the maximum and average errors are given in Table 4.2.

4.6 Visibility Computation

Over the course of an entire radiosity simulation, patches shoot varying amounts of radiosity. For many non-primary sources, or primary sources very far from a receiver, the effect that the source radiosity has on the receiver is often perceptually unnoticeable. The computation of a form-factor between each receiver and the source is required in order to perform the radiosity transfer, and this involves estimating the fraction of the source patch that is visible from the receiver element or vertex. Ray-casting techniques are often employed to determine this fraction [WEH89, HSA91]. This involves tracing a number of rays from the receiver towards the source, and intersecting each with any potentially occluding objects that are encountered (Section 2.5.2).

A trade-off between accuracy and computation time can be made by increasing or reducing the number of rays that are traced when calculating each form-factor. Fewer rays take less time to intersect with any potential occluders, but also decrease the accuracy of the form-factor calculation. We wish to limit the number of rays used whilst still producing a perceptually accurate representation of radiosity. As will be described below, for those radiosity transfers that are perceptually unimportant, we will use an approximate (but much faster) visibility estimate, since computing an accurate value will not improve the perceived accuracy of the solution.

The maximum increase in luminance that can occur at a vertex i when transferring radiosity from a source j is

$$\rho(i)F_{ij}\Delta L_j \tag{4.19}$$

where $\rho(i)$ is the diffuse reflectivity at vertex i, F_{ij} is the *unoccluded* form-factor between the vertex and source patch, and ΔL_j is the unshot luminance of the patch j. Assuming that no shadow is generated across the element during this transfer, the perceived difference between the luminance at the vertex before and after the radiosity transfer will be

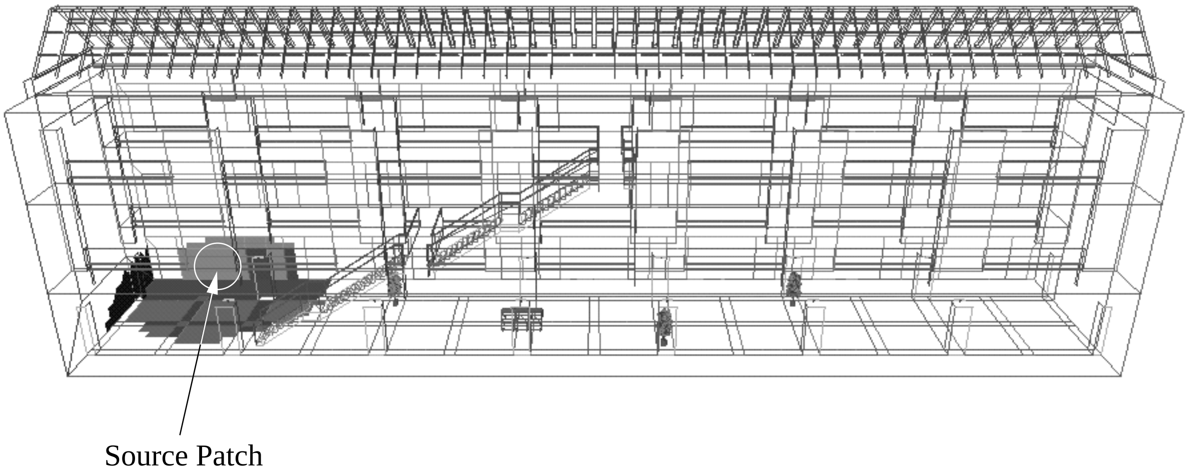
$$E^*_{uv} = \mathcal{P}(L(i) + \rho(i)E_{amb}, L(i) + \rho(i)(F_{ij}\Delta L_j + E_{amb})) \tag{4.20}$$

where $\mathcal{P}()$ is the same operator as was used in Equation 4.17.

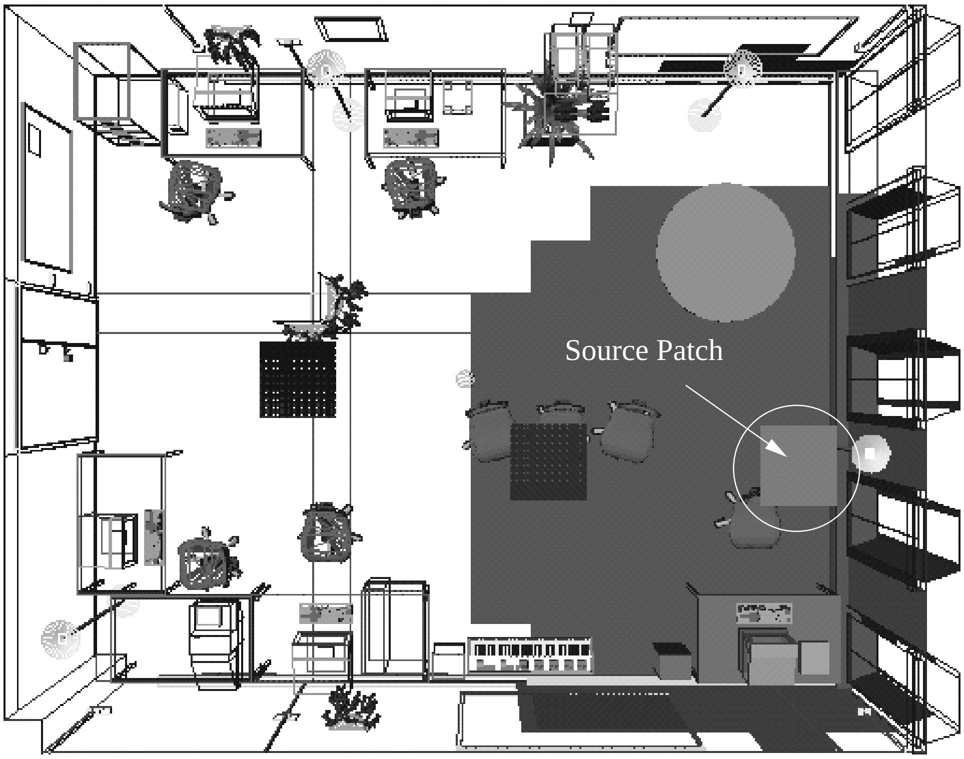
Equation 4.20 can now be used as a measure of the perceptual importance of any potential shadow boundary caused by the light from the source arriving at the vertex. If E^*_{uv} is less than the just-noticeable difference, there will be no perceptible difference between the solutions before and after the radiosity transfer, and so any shadow that falls across the element will not be detectable. In this case, the appearance of the solution will not be improved by the calculation of an accurate visibility term. Visibility must still be tested for, however, since many such transfers to a vertex that would not otherwise receive any illumination will cause an energy imbalance, as well as perceivable difference in the appearance of the solution.

Figure 4.10 shows two test scenes, and the sets of patches for which $E^*_{uv} > 1$ for a single source patch. The source patch in each case is one of the primary light-sources. The first scene, 'atrium', contains $37,454$ patches, 132 of which are light-sources. This is a large environment (in terms of physical size rather than number of patches), where most of the light-sources affect only a small local area. The second scene, 'lab', is a more compact model containing $78,790$ patches and 23 light-sources. For the 'atrium' model, only 520 out of $37,454$ transfers (less than 1.4%) were considered perceptually important, and these patches have been shaded for clarity. For the 'lab' model, the effect of each light-source is greater due to the relative positions of the other patches. In this case, $16,485$ out of $78,790$ transfers (20.9%) observed a change of $E^*_{uv} > 1$. Figure 4.11 shows the frequency of occurrence of perceived changes in the solution (measured by Equation 4.20, and shown in the same units as E^*_{uv}), that occur for these two moderately complex test environments.

Figure 4.11(a) shows a histogram and cumulative frequency distribution for the perceived changes over 1000 iterations with the 'atrium' model. Over 99.17% of the radiosity transfers caused a change of $E^*_{uv} \leq 1$. Figure 4.11(b) shows similar graphs for the 'lab' model. Even though this environment is

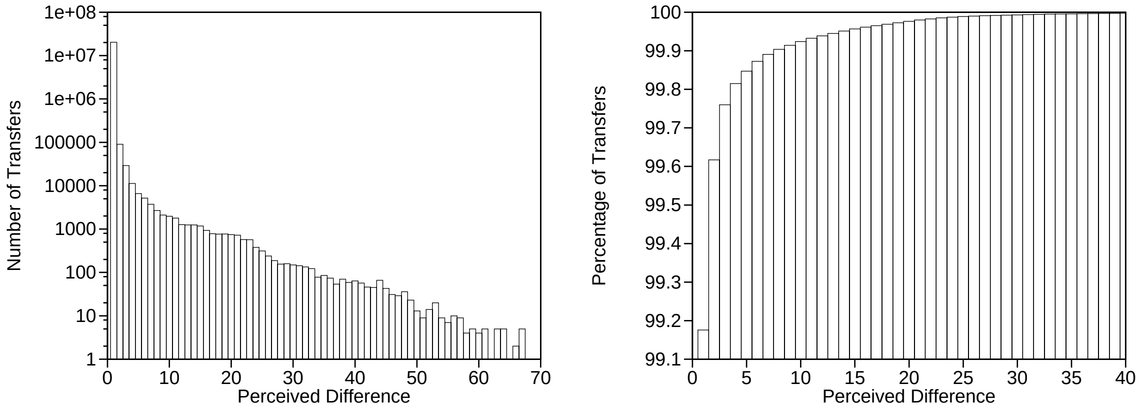


(a) ‘atrium’ model

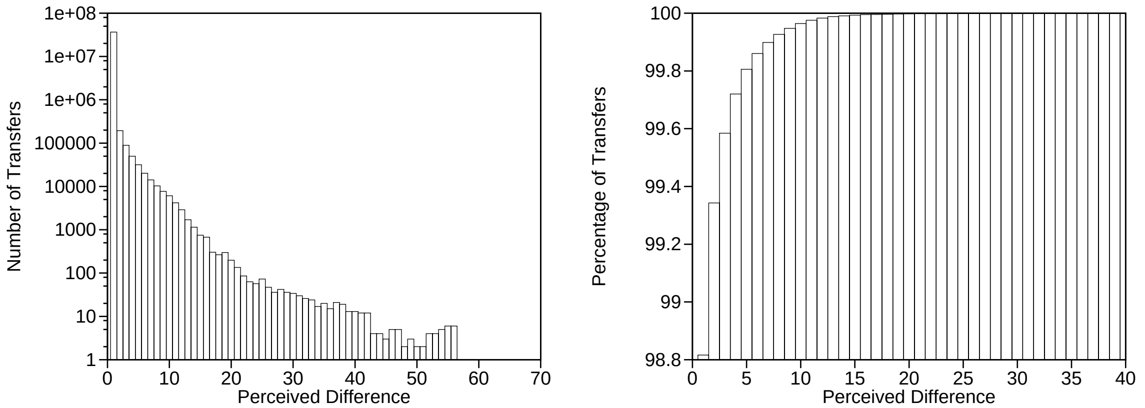


(b) ’lab’ model

Figure 4.10: The perceptually important patches which require accurate visibility estimates for a given source patch for (a) the ‘atrium’ model and (b) the ‘lab’ model.



(a) ‘atrium’ test scene, 1000 iterations



(b) ‘lab’ test scene, 1000 iterations

Figure 4.11: Histograms and cumulative frequency distributions for the perceived change in luminance during individual radiosity transfers.

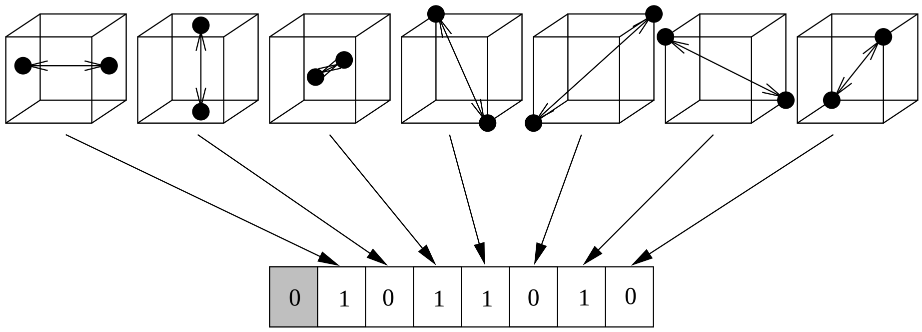


Figure 4.12: Representation of visibility information for a single voxel.

more compact than the 'atrium' model, more than 98.8% of the transfers were still classified as being perceptually insignificant.

4.6.1 Approximating Visibility

Given that such a potentially large percentage of the source-to-receiver radiosity transfers cause an imperceptible change in the radiosity solution when considered individually, performing a fast, approximate visibility computation when calculating their form-factors will greatly accelerate the simulation.

Because an accurate visibility calculation is not required, we propose using a simple visible/invisible result in order to avoid a serious imbalance of energy. This is equivalent to tracing a single ray between the source and receiver patch. For very large environments, tracing this single ray can still be very time consuming, even when spatial subdivision structures such as hierarchical grid-cells or bounding-volume hierarchies are employed, since many ray-polygon intersections must be performed at each voxel.

Our proposed acceleration method utilises a uniform subdivision of the scene into a grid of voxels, and avoids testing each polygon in each voxel for intersection with the ray. As a pre-process, we first identify all objects intersecting each voxel. Figure 4.12 shows a number of directions between faces or vertices of a single voxel, each of which is associated with a single bit of data. We intersect a number of jittered rays running in each direction with all objects in the voxel. If more than half of the rays for a particular direction are found to be unoccluded, the bit associated with that direction is set. After all directions have been processed, the visibility information for that voxel may be stored as a single byte in the grid structure. This storage method will become significant in Chapter 5, when a parallel implementation of this perceptually-driven radiosity algorithm is discussed.

Each visibility test consists of a traversal of this grid structure by a single ray, accessing the visibility information stored with each voxel rather than testing rays for intersection with each polygon that is encountered. As each voxel is visited, the direction of the ray is approximated to one of the seven shown in Figure 4.12, and the corresponding bit is checked. If the bit is set, then traversal continues onto the next voxel, otherwise the transfer is considered occluded. Only if the traversal reaches the end voxel is the radiosity transfer allowed to occur. In order to avoid the situation where transfers are incorrectly classified as occluded due to intersections with the polygons on which the ray starts and ends, the visibility information contained in the first and last voxel in the traversal sequence is ignored.

In Figures 4.13 and 4.14, the final solutions obtained after 1000 iterations using this perceptually adaptive visibility test are compared against those computed using a standard test. Figure 4.13 shows the solutions computed for the 'atrium' model, and Figure 4.14 shows solutions for the 'lab' model. The right hand images were computed using 36 jittered rays for each visibility calculation, as would occur in a typical radiometric algorithm. A two-level hierarchical grid-cell was used to accelerate the ray-casting process. The left images were generated using the approximate visibility test for all transfers that caused a change of $E^*_{uv} \leq 1$, and the standard test for all other transfers. As can be seen, the differences between the solutions are very small, and are mainly due to differences in the jittered positions of rays.

Statistics for each solution are given in Table 4.3. Without the perceptually adaptive visibility test,

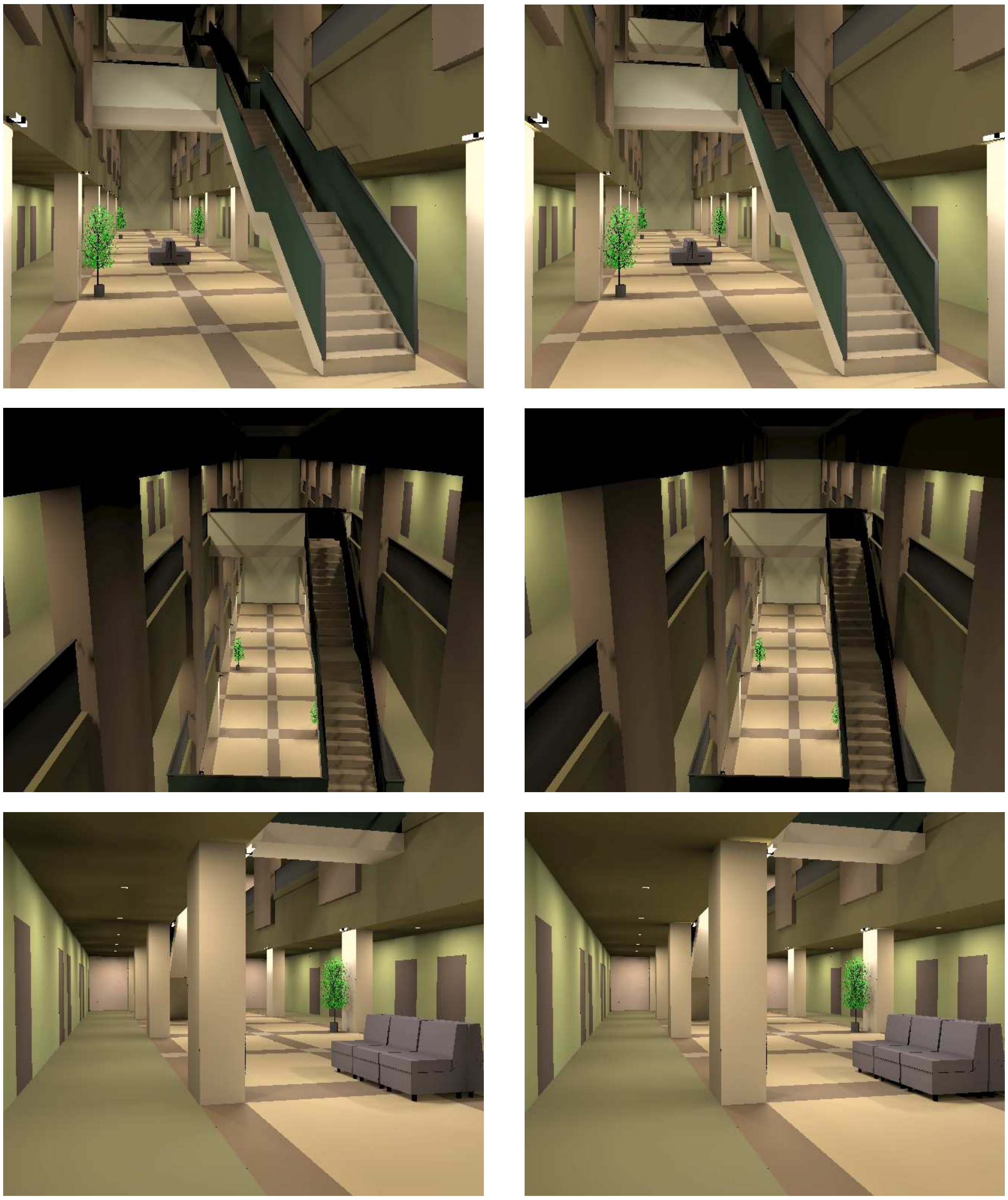


Figure 4.13: Visual comparison of solutions generated for the 'atrium' test scene, (left) with and (right) without perceptually adaptive visibility.

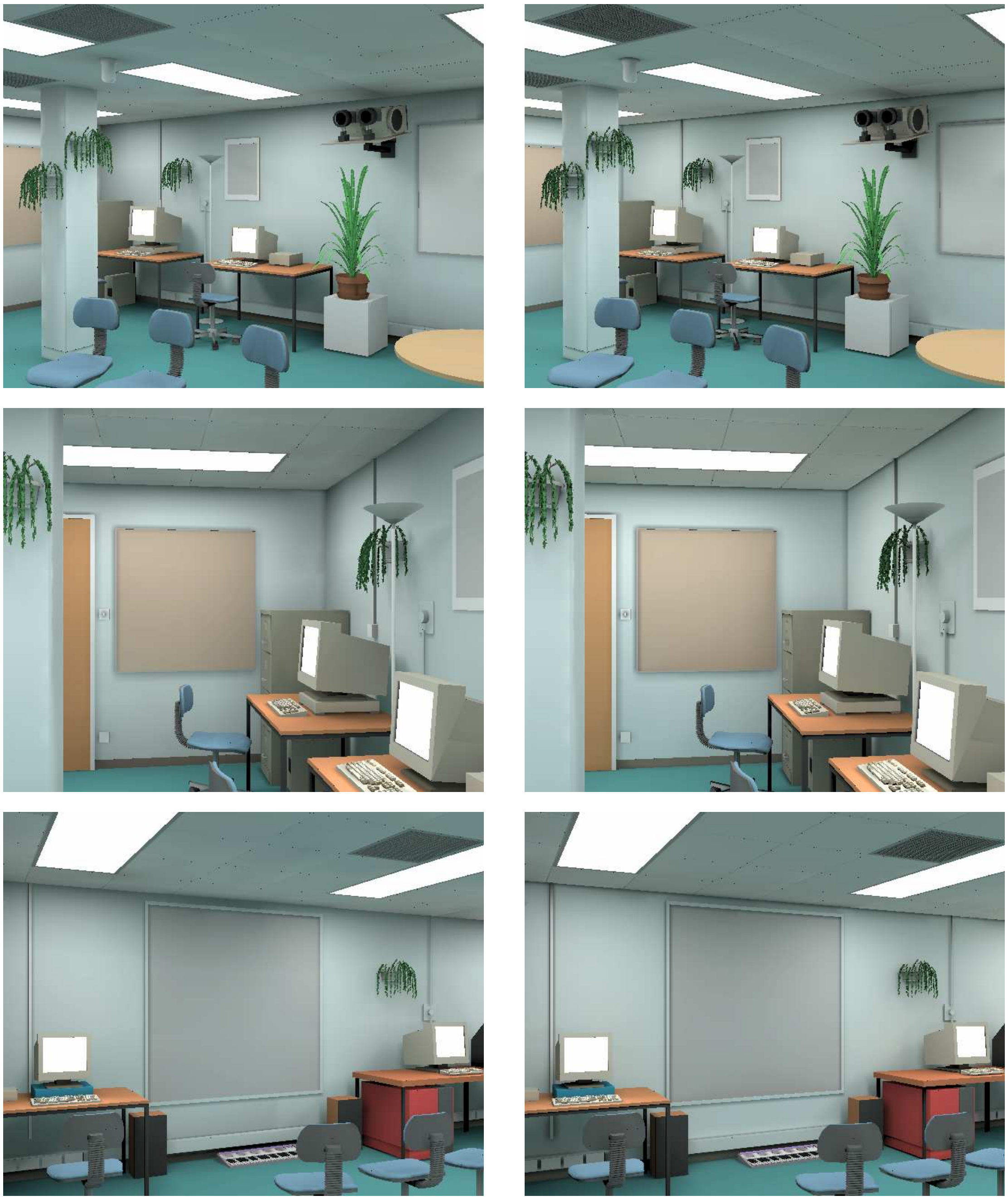


Figure 4.14: Visual comparison of solutions generated for the 'lab' test scene, (left) with and (right) without perceptually adaptive visibility.

Model	'atrium'	'lab'
Non adaptive		
Solution Time (hrs)	45.95	24.94
Total Rays	1.98×10^9	4.03×10^9
Adaptive		
Solution Time (hrs)	0.95	1.07
Speedup	$\times 48$	$\times 23$
Accurate Transfers	1.13%	2.73%
Total Rays	2.08×10^8	1.1×10^8
Reduction	89.47%	97.26%
Polygon Rays	1.87×10^8	7.47×10^7

Table 4.3: Statistics for solutions generated with and without perceptually adaptive visibility test.

1000 iterations were computed in 45.95 hours for the 'atrium' model and 24.94 hours for the 'lab' model, with 1.98×10^9 and 4.03×10^9 rays cast respectively[2]. Using the perceptually adaptive visibility test to compute the same 1000 iterations, solution times fell to 57.4 minutes and 1.07 hours, giving a speedup of over 48 times for the 'atrium' model, and 23 times for the 'lab' model. The total number of rays that were traced during these solutions fell by 89.47% and 97.26% to 2.08×10^8 and 1.1×10^8 respectively. 90% and 68% of those rays were used for intersection with the scene polygons, and the remainder were traced through the uniform grid-cell structure described above.

Figures 4.15 and 4.16 plot the growth in computation time as each solution progresses, and the overall reduction in the percentage of source-to-receiver transfers that are computed using the standard visibility estimate as each solution nears convergence. The saving in computation time due to the reduced number of rays that must be intersected against the scene geometry is of great advantage, especially considering that the 'atrium' model was only 75% converged, and the 'lab' model only 85% converged when the simulation was halted after 1000 iterations. As the number of iterations increases and the distribution of light nears convergence, the visual importance of each transfer is reduced and perceptually-adaptive visibility becomes even more beneficial.

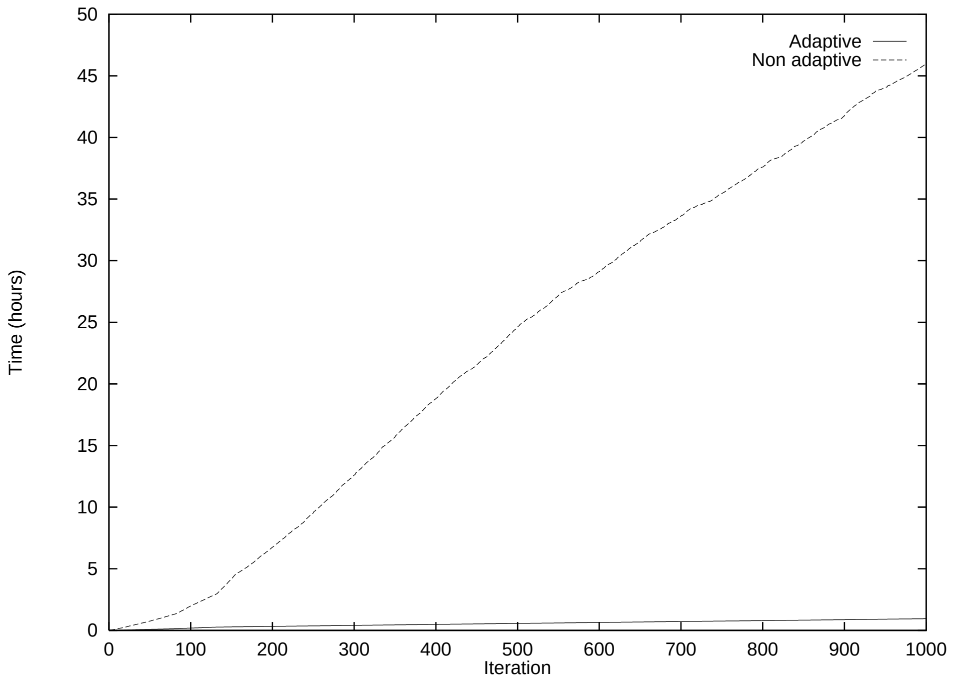
Finally, Figure 4.17 shows images of a solution that has been computed using the perceptually driven algorithm for an environment containing over 1.16 million polygons and 656 light-sources. 2000 iterations were performed in 17 hours and 26 minutes using one 195 MHz MIPS R10000 CPU on a Silicon Graphics Origin2000 with 10 GBytes of main memory. Out of a total of over one billion source-to-receiver transfers, only 0.84% were performed using an 'accurate' visibility calculation, with the remaining 99.16% using the approximate test. We estimate that solution time for this environment would be approximately 20 days without perceptually-adaptive visibility.

4.7 Summary

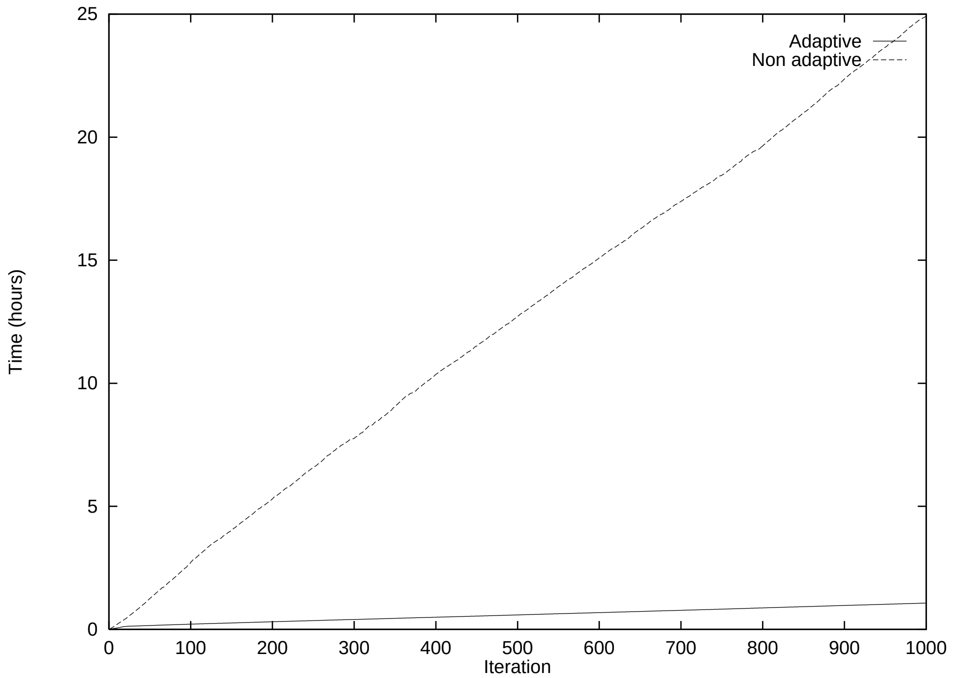
In this chapter, we have shown how perceptual knowledge can be applied to accelerate the computation of visually accurate view-independent radiosity solutions. This was achieved by first producing a tone-reproduction transformation that mapped real-world luminance values to display colours. This tone-reproduction transformation was estimated *a priori*, and enabled the perceived difference between two real-world luminance values to be determined during the solution process.

An adaptive subdivision technique was presented that used the perceived difference between real-world luminance values to halt subdivision once no extra visible detail was introduced by the refinement. This technique was shown to produce visually accurate element meshes. Form-factor calculations were also accelerated by using a fast, approximate visibility test whenever any potential shadow falling across

[2] All computations were performed using a 150 MHz MIPS R10000 processor.

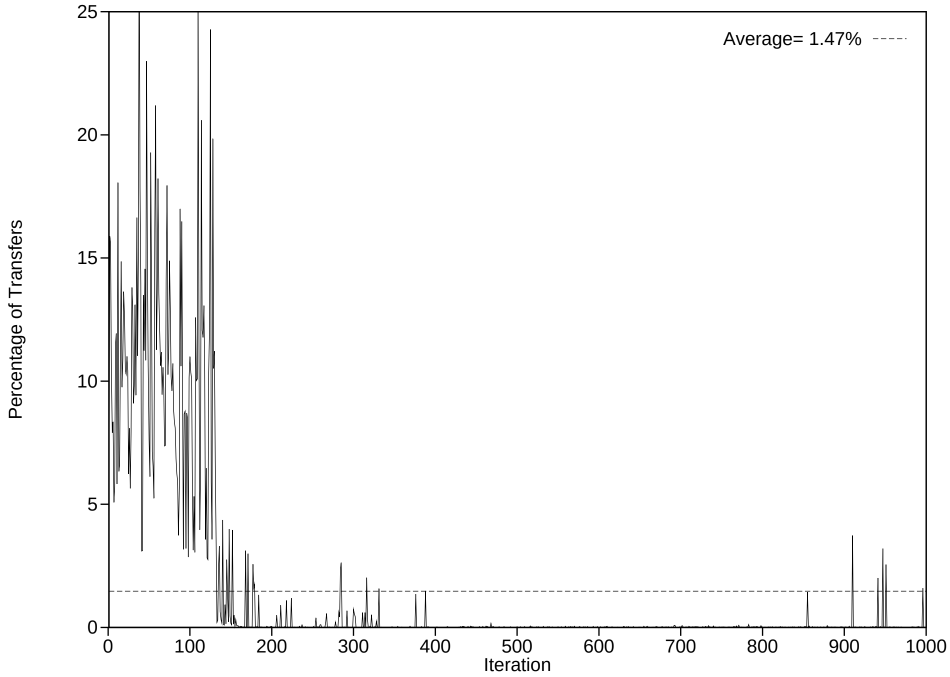


(a) ‘atrium’ test scene

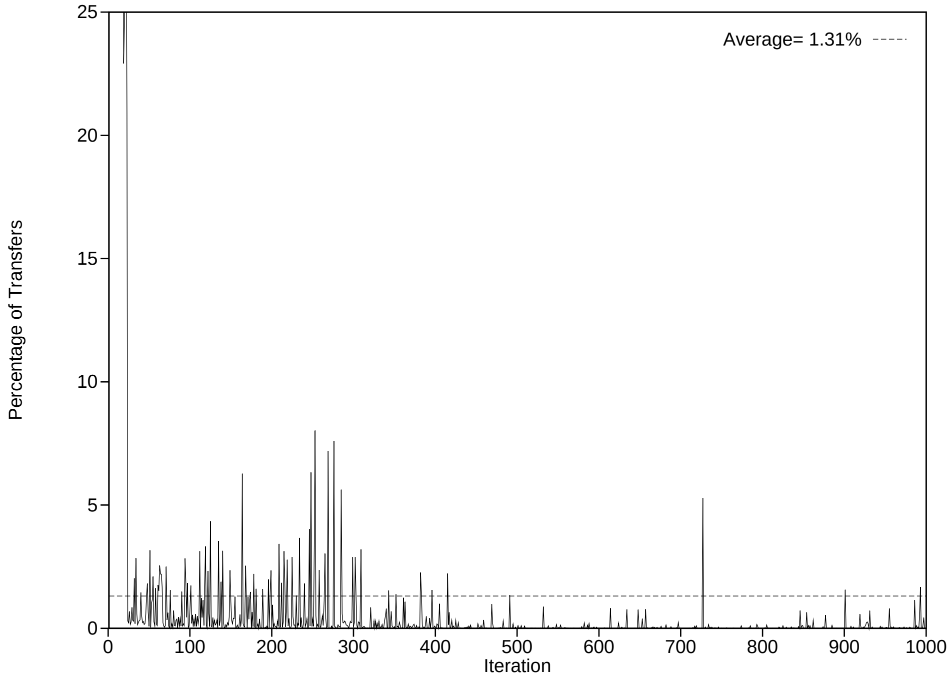


(b) ‘lab’ test scene

Figure 4.15: Computation times with and without approximate visibility for 1000 iterations with (a) the ‘atrium’ scene, and (b) the ‘lab’ scene.



(a) ‘atrium’ test scene



(b) ‘lab’ test scene

Figure 4.16: Percentage of patch transfers computed using the standard visibility estimate for (a) the ‘atrium’, and (b) the ‘lab’ scene.

Figure 4.17: Images of a radiosity solution computed for an environment containing over 1.16 million polygons.

an element was predicted to be unnoticeable. Since a large percentage of the radiosity transfers have very little visual effect on the solution, this reduced the number of rays that were traced (and therefore the overall solution time) by a significant amount.

The techniques presented have several advantages over previous methods for accelerating radiosity computations. Most importantly, the decrease in solution time does not incur a decrease in the visual quality of the solution – the acceleration comes from computing those aspects of the solution that are not visible to the human eye with less accuracy than those aspects that are visible. Secondly, the process of choosing thresholds and tolerances for the simulation is greatly simplified, since the parameters controlling the perceptually-driven radiosity algorithm are based on just noticeable differences in a perceptually uniform colour space, rather than radiosity values. Other possible applications of the proposed approach will be given in Chapter 6.

Chapter 5

Scalable Parallel Radiosity using Distributed Shared Memory

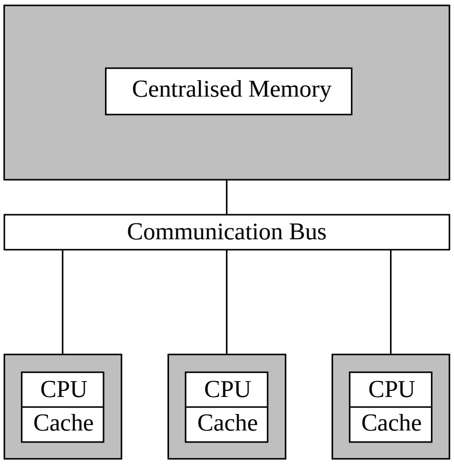
5.1 Introduction

Regardless of the algorithmic improvements made to radiosity methods, computing very large solutions can still take several hours. For situations when rapid computation is required, such as when feedback on changes to the geometry or surface characteristics of a complex model are needed, parallel processing has been proposed as a technique to decrease solution time. In this chapter, we will discuss previous parallel radiosity algorithms, and describe a novel approach that has been developed to allow radiosity solutions to be generated efficiently using multiple processors.

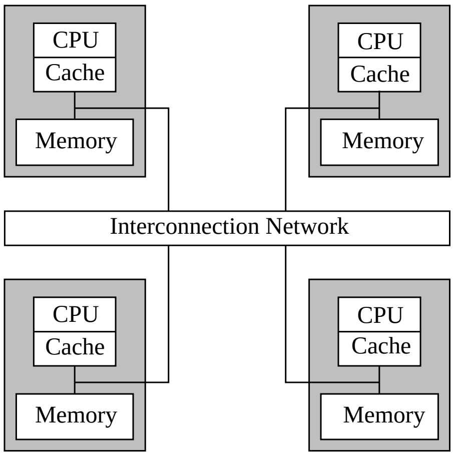
Parallel processing architectures can be divided into classes based on their types of instruction and data streams, and on the way that their CPUs access memory [Fly72]. Single Instruction Multiple Data (SIMD) systems execute the same stream of instructions on each processor using different streams of data. These systems usually have very large numbers of relatively slow processors, each with a small amount of local memory. Data residing on other processors must be accessed by passing messages along the communication paths between the CPUs. For simple programs with a single path of execution and predictable memory access patterns, efficient parallel execution is achievable on such systems. For more complex programs, such as the radiosity algorithms described in previous chapters, many conditional branches are executed, and implementation on SIMD architectures becomes very difficult and inefficient.

Multiple Instruction Multiple Data (MIMD) systems are more flexible, as they allow different instruction streams to be executed simultaneously on different processors. In *shared memory* systems (an example of which is shown in Figure 5.1(a)) a centralised area of main memory is connected to each processor by a high speed bus, and each processor has access to all the memory in the system. A cache is usually provided on each CPU in an attempt to reduce the amount of communication traffic to and from the memory, although the limitations in speed of the communication bus restricts the scalability of these systems to relatively few processors [Fie96].

Distributed memory MIMD systems have individual memories located with each processor (Figure 5.1(b)), and each CPU can only access its own local memory. When data items that reside in a remote CPU's memory are needed, a request must be sent over the network and the data transferred back to the processor. In purely distributed memory systems, this is done explicitly by the application using message passing. Generally, these systems provide better scalability than bus-based shared memory architectures, due to the reduced amount of communication traffic. Alternatively, *distributed shared memory* (also known as virtual shared memory) may be employed, effectively providing a global address space for the physically distributed memory associated with each processor. Distributed shared memory may be implemented using specialised hardware (as in the Silicon Graphics Origin2000 machine [LL97]), or using a software layer, such as the KOAN system [LP92] with a distributed memory machine. This hides the need for explicit message passing from the application programmer. Virtual shared memory



(a) Shared memory



(b) Distributed memory

Figure 5.1: Example shared and distributed memory MIMD parallel architectures.

systems attempt to combine the scalability of a distributed memory, with the ease of programming and flexibility of shared memory machines.

5.2 Previous Work

The fundamental problem which must be addressed is the efficient parallelisation of radiosity simulations for large environments, containing hundreds of thousands of surfaces. There are many aspects of the radiosity process that must be examined when attempting to design such a parallel algorithm, and by which previous work may be characterised. Firstly, good parallel efficiency is essential – solution time should decrease (ideally, in a linear fashion) as extra processors are used to perform the simulation. At worst, solution time should never *increase* when more processors are employed. The use of parallel programming constructs such as mutual exclusion locks and synchronisation barriers, and attention to locality of reference when accessing data will affect performance. The equal partitioning of work amongst processors must also be achieved, so as to ensure that individual processors are not left idle. Any parallel radiosity algorithm must therefore be designed with these factors in mind, if acceptable parallel efficiency is to be achieved.

The underlying method used to solve the linear system should also be efficient, otherwise solution times will still be excessive, even if the parallel implementation scales well. This requires either a progressive or hierarchical solution method, as the full-matrix method has been shown to be inefficient in both its time and memory usage. In order to obtain visually accurate solutions, techniques such as adaptive (or hierarchical) element refinement must be included, without detracting from the efficiency of the parallel simulation. Form-factor accuracy is also essential if accurate solutions are to be obtained. The parallel algorithm must therefore not be limited to projection methods such as hemicube-based techniques discussed in Chapter 2, as these have been found to introduce noticeable artifacts into the solution [BRW89].

Finally, in order to be able to generate radiosity solutions for very large environments, the memory available to the parallel system must be used efficiently. For distributed memory systems, the database required to perform the radiosity simulation must not be replicated in each processor's local memory, as this would limit the size of environment which could be processed.

Table 5.1 summarises the most notable prior efforts that have been made towards parallelising progressive radiosity simulations. The table identifies:

[1] KOAN SVM was used to simulate a virtual shared memory on the distributed memory iPSC/2 hypercube.

Author	Memory	Form-Factors	Loop	Database	Efficiency	Scene Size
[Che89]	DM	Hemicube	Outer	Replicated	Near-linear (8 CPUs)	6,000 patches
[BW90]	SM	Hemicube	Inner	Shared	50% (8 CPUs)	38,000 patches
[PSV90]	DM	Hemicube	Outer	Replicated	68% (6 CPUs)	1000 patches
[RGG90]	DM	Single-plane	Outer	Replicated	50% (12 CPUs)	13,200 patches
[CP91]	DM	Hemicube	Outer	Circulated	31% (42 CPUs)	448 patches
[FP91]	DM	Hemicube	Outer	Circulated	60% (9 CPUs)	4,500 patches
[DS92]	DM	Ray-casting	Inner	Distributed	1:3.8 (8K:32K CPUs)	5,700 patches
[VP92]	DM	Single-plane	Inner	Distributed	Non-given	4,000 patches
[BMP93]	DSM[1]	Hemisphere	Outer	Shared	82% (32 CPUs)	50,500 patches
[cAO93]	DM	Hemicube	Outer	Circulated	86% (16 CPUs)	8,000 patches
[SSJ95]	DM	Ray-casting	Inner	Distributed	26% (256 CPUs)	60,000 patches
[Š96]	DM	Hemicube	Outer	Replicated	90% (8 CPUs)	9,800 elements
[YIY96]	DM	Ray-casting	Inner	Replicated	78% (32 CPUs)	300 polygons
[RAPP97]	DSM	Hemisphere	Both	Distributed	50% (32 CPUs)	1.2×10^6 patches

Table 5.1: Summary of previous work on parallel progressive radiosity methods.

- Memory model of the parallel system – DM (distributed memory), SM (shared memory) or DSM (distributed-shared memory);
- Method of form-factor computation – hemicube, hemisphere or single-plane projection, or ray-casting;
- Method of parallelisation – inner or outer loop of the progressive radiosity algorithm (see below);
- Distribution of the scene database amongst processors – replicated in each memory, shared, circulated between processors, or distributed between processors;
- The resulting efficiency of the parallel algorithm;
- The size of database with which performance was measured.

One of the most common approaches to parallelisation has been to designate a single processor in the system as the 'master', and allow this master processor to select and distribute items of work to the other processors (denoted as 'workers'). The selection of work depends upon the method by which the simulation has been parallelised. A simple approach was initially developed, in which the outer loop of the progressive-refinement algorithm (Figure 2.10) was parallelised, with multiple source patches selected by the master and distributed to the workers [Che89, RGG90, PSV90, Š96]. Each worker then computed the form-factors from its assigned source patch to all of the receiver patches in the environment in parallel. The database describing the environment was replicated in the memory of each processor to avoid large amounts of inter-processor communication during these form-factor calculations, and workers transferred vectors of form-factors back to the master once they had been completed. Once the master received a packet of form-factors, it updated all patch radiosities and selected new source patches to continue the computation.

The performance of these algorithms was adequate with small environments and relatively few processors. As larger environments and more processors are used, however, the efficiency of this approach will be reduced, due to the limited amount of bandwidth available for communication between the master and worker processors. Processors were also at risk of being left idle once they had completed an iteration in the algorithms described by Chen and Recker *et al.*: the time required to perform a single iteration of the progressive radiosity algorithm can vary, depending upon the geometric relationships between patches, and the master processor was required to wait for all workers to finish before selecting a new set of source patches. Puech *et al.* attempted to reduce this idle time by assigning an idle processor a new source patch as soon as its form-factor vector had been completed [PSV90]. They also noted that the difference in the selection order of source patches affected the convergence rate of the simulation, although no indication as to the severity of the change was given.

Related approaches were also proposed by Feda and Purgathofer [FP91] and Chalmers and Paddon [CP91] for distributed memory transputer systems, with the patch data describing the environment distributed between the available processors. Workers were each assigned a source patch by the master, and computed a partial hemicube for that patch. Receiver patches were then transferred between workers, in order to complete the form-factor calculations. Once the complete vector of form-factors was obtained by a worker, messages were sent to other workers indicating an update of each patch radiosity was required. By avoiding the replication of the scene database at each processor, this approach could perform radiosity calculations for larger environments, although Chalmers and Paddon noted that the inter-processor communication network quickly became congested with messages as larger numbers of processors were employed [CP91]. Overall, approximately 31% efficiency was observed for a 42 processor simulation.

Çapin *et al.* implemented a similar progressive radiosity algorithm on a distributed memory iPSC/2 multicomputer [cAO93]. A ring topology was employed, with each processor in the ring assigned a subset of patches. An efficient synchronous global patch selection scheme was proposed that selected a source patch for each processor. Processors then computed form-factors from their assigned source patches to the receiver patches each was responsible for. In order to complete each hemicube calculation, receiver patch data was again circulated around the ring. Each processor then performed the energy distribution from all current source patches to its receiver patches. Because of the synchronous patch selection scheme, processors could remain idle whilst others completed their work. An attempt at static load-balancing was therefore made by assigning receiver patches to processors using a 'scatter' distribution [cAO93]. Efficiency using a ring topology was around 86% for 16 processors with a relatively simple environment containing 8,000 patches. Similar results were obtained by Aykanat *et al.* when using a hypercube network [AcO96]. Although relatively successful in terms of efficiency, this approach is limited by the increase in complexity for implementing important techniques such as adaptive subdivision. This would require dynamic load-balancing in order to equalise the work load for each processor, and the communication of increasing amounts of data as patches become subdivided would limit efficiency.

An alternative to the master/worker paradigm was proposed by Bouatouch *et al.* to parallelise progressive radiosity simulations on a distributed memory machine [BMP93]. The KOAN virtual shared memory system was used to provide a virtual address space that was shared by each processor [LP92]. Each processor was again responsible for a subset of patches, but was provided access to all patches in the environment by the virtual shared memory. Multiple iterations were performed in parallel, with an initial source patch being selected by each processor from those it was assigned. Once a processor had finished computing the form-factors from its current source patch to the rest of the environment, a message was sent to all other processors consisting of the form-factor values to the other processors' patches. When a processor received such a message, it would interrupt its work, and update the radiosities of its receiver patches. During this update, a new source patch was also selected, and given back to the original processor, which examined all new source patches that were sent back, and chose the one with the greatest amount of unshot power for the next iteration. No synchronisation was required between processors to select new source patches using this approach, and good efficiency was achieved for a moderately complex environment of $50,000$ patches arranged in a simple pyramidal structure of 107 primitives. No acceleration techniques were implemented for the hemisphere form-factor calculations, however, which increased the time spent in each iteration. Because each processor accessed all patches in the environment during each iteration, the amount of network traffic would increase as the size of environment is increased. As acceleration techniques are introduced, and iteration times fall, the effect of this large amount of inter-processor communication would become important, thereby limiting the scalability of the algorithm for large environments.

A different approach to parallelising progressive radiosity was proposed by Baum and Winget for a bus-based shared memory machine with polygon rendering hardware [BW90]. Instead of performing multiple iterations in parallel, the inner loop of the simulation was parallelised. This corresponds to performing the iteration for a single source patch in parallel. The master processor was charged with the selection of the source patch, and accelerated graphics hardware was employed to produce hemicube item buffers. Worker processors were then dynamically assigned blocks of these buffers. Each worker calculated form-factors for its blocks, and wrote the results to a local copy of the form-factor vector. Again, load-balancing was required, since the time required to process a block depended upon the relative numbers of analytic and simple delta form-factors the block contained [BW90]. Once all blocks had been processed, the master summed the local copies of the form-factor vectors, and energy was distributed to the receiving patches. The production of the hemicube item buffers for the next iteration, and the computation of form-factors for the current iteration was overlapped using a buffering system [BW90]. Although adequate performance for small numbers of processors was reported, given a fixed hardware rendering performance, the efficiency of this algorithm quickly drops as the difference in execution time between the overlapped master and worker computations increases.

All of the algorithms discussed previously have employed a hemicube, or related technique, to calculate form-factors, and therefore suffered from the problems associated with such projection methods (Section 2.5). The more accurate ray-casting techniques of Wallace *et al.* [WEH89] were first used by Drucker and Schröder for form-factor computation during a parallel progressive radiosity algorithm on a massively parallel distributed memory SIMD machine [DS92]. Their algorithm was also the first to discuss the implications of including adaptive element refinement in a parallel radiosity algorithm – another technique essential for producing accurate results. Patch data was distributed amongst the available processors, and a processor allocation scheme was used to parallelise a single iteration of the progressive radiosity algorithm, as well as to perform visibility calculations in parallel, with new processors being allocated for each potential ray-object intersection [DS92]. Similar processor allocation was employed to perform adaptive subdivision, with the resulting algorithm scaling almost linearly. For an environment containing over $5,700$ patches, a speedup of 3.8 was observed using $32,000$ processors, when compared to the solution time obtained for $8,000$.

Adaptive element subdivision was also included in a recent parallel progressive radiosity algorithm proposed by Yu *et al.* [YIY96]. Patches were statically distributed amongst the available processors, although geometric data describing the scene and ray-tracing acceleration structures were replicated in each processor's memory. The globally brightest patch in the environment was selected, and the form-factors from this single source patch to each receiver were then computed in parallel, with each processor distributing energy to those patches is was responsible for. During this process, adaptive refinement was performed. Dynamic load-balancing was not employed because it was considered to incur too great a communication overhead. In order to avoid a load imbalance occuring when patches were subdivided, the static assignment of patches to processors was therefore performed in an attempt to minimise the effect of this imbalance. Due to the fact that each processor only required access to its local patches, and the geometric data required for visibility calculations was replicated at each processor, the algorithm scaled well, resulting in 78% efficiency on 32 processors, albeit for a simple environment containing 300 initial polygons. A 10% load imbalance was reported for the simulations.

The need for dynamic load-balancing during parallel progressive radiosity employing adaptive subdivision of receiver elements was also identified by Stürzlinger *et al.* [SSJ95]. In their approach, the scene patches were distributed evenly amongst the processors, and at each iteration, each processor selected from those assigned to it the patch with the greatest unshot power. The master then selected the globally brightest patch, and communicated its geometry and associated unshot radiosity to each worker. Each worker calculated form-factors from the source patch to its assigned receiver patches using ray-casting, and updated its receiver radiosities. For visibility information that required geometric data residing on other processors, the sample points associated with each ray were transferred over the network to the appropriate processors, which were identified using a 'global BSP-tree' [SSJ95], replicated in each CPU's memory. Dynamic load-balancing algorithms were discussed that attempted to correct for the changes in computational load introduced by adaptive subdivision. The geometric data associated with the scene polygons was replicated on a number of processor rings, enabling packets of sample points to be transferred between rings once processors became idle. Efficiency of the algorithm was 26% on 256 CPUs for an environment containing $60,000$ patches, with the combination of static and dynamic load-balancing giving the best performance. No indication was given, however, as to the reason for the low overall speedup.

Finally, Renambot *et al.* developed a parallel progressive radiosity algorithm for distributed shared memory computers that attempted to limit the size of the working set of data required by each processor, thereby improving cache usage [RAPP97]. The patches in the scene were partitioned into sub-environments, and one or more sub-environments assigned to each processor. A 'virtual interface' [APRP96, APV91] was constructed to model the transfer of energy between each sub-environment.

A source patch was selected by a processor to distribute its energy to its local sub-environment using a hemisphere form-factor calculation, and this patch was then propagated through virtual interfaces and into adjacent sub-environments where it was added into a queue. A visibility mask, corresponding to the set of elements in the hemisphere was also maintained for each source patch. This mask indicated which areas of the hemisphere were occluded by polygons in a previous sub-environment, and so did not require further form-factor computations. Visibility masks were transferred between sub-environments along with each source patch. The aim of this approach was to allow a processor to perform energy transfers to a single group of receiver patches, without the need to access data stored on other processors. Results showing the reduction in cache misses indicated that this approach improved performance with both serial and parallel implementations, although the scalability of the parallel algorithm was limited to 50% on 32 processors for a large environments containing over 1.2 million patches. No indication of the cause of the drop in performance was given, although the large amount of communication required to exchange visibility masks, and the lack of dynamic load-balancing could reduce scalability.

5.2.1 Summary of Previous Work

To summarise, there have been several factors that have limited the performance of previous parallel radiosity algorithms. Firstly, many early approaches replicated the data describing the environment in the memory of each processor. Although this reduces the amount of inter-processor communication required for form-factor computations, the size of environment that can be processed is limited to that which will fit in a single processor's memory. If radiosity values are maintained at each processor as well, any updates must be broadcast over the network, increasing communication costs for large environments and large numbers of processors.

Secondly, approaches that employ the master/worker paradigm are often limited by communication to the master becoming a bottleneck. This occurs in particular when workers transfer vectors of form-factors or updated element radiosities to the master. As the size of the environment (and hence the size of each vector) increases, or the number of worker processors increases, the bandwidth of the network connection to the master processor will be quickly reached.

Load-balancing has also been identified as an important factor in achieving good performance, as the time processors are left idle is reduced by assigning further items of work. The exact form of load-balancing must be considered carefully, as the amount of inter-processor communication that is incurred by the transfer of work must not affect performance significantly. Aside from performance, the algorithm must also not be limited in the application of techniques such as ray-tracing based form-factors or adaptive subdivision – an efficient algorithm that produces inaccurate results is of as little use as an algorithm that does not scale well.

The algorithm proposed in this chapter has been designed to exploit the characteristics of distributed shared memory architectures such as the Origin2000, in an attempt to provide scalability for geometrically complex environments up to moderate numbers of CPUs (the Origin2000 machine is available in configurations with up to 128 processors). In order to avoid communication bottlenecks, we have avoided the master/worker paradigm. We also attempt to ensure that the tasks performed by each processor remain as local to that processor as possible. Given the memory hierarchy of the Origin2000 machine on which the algorithm has been implemented (see Section 5.3), this not only reduces inter-processor communication, but should provide good scalability to larger numbers of processors.

Our novel approach to parallel progressive radiosity is based on the notion of a *sub-iteration* being the transfer of radiosity from a single source patch to a subset of the scene's receiver patches (see Section 5.5). As will be described in the next section, the patch data describing the environment is subdivided into several partitions, and one or more partitions assigned to each processor. Processors then compute sub-iterations to their assigned partitions concurrently. An efficient queue-based scheduling system is used to transfer source patches between these partitions, enabling the radiosity simulation to be performed without the need for processor synchronisation between sub-iterations. The partitioning of patch and geometric surface data enables each processor to maintain an affinity with the data it requires, and inter-processor communication only occurs during a sub-iteration for visibility calculations. The effects of the perceptually-based visibility calculations introduced in the last chapter on this inter-processor

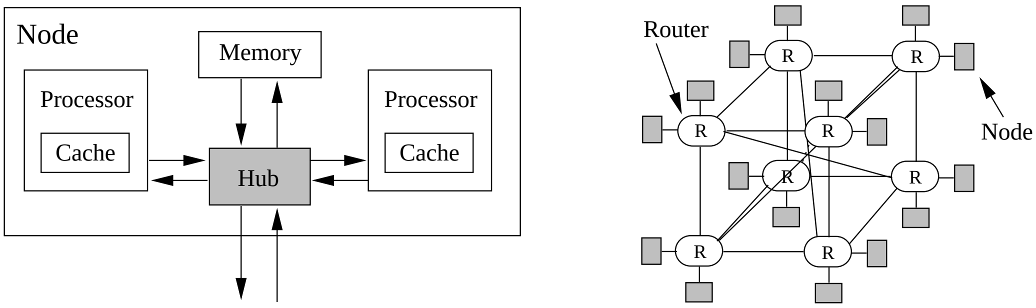


Figure 5.2: Connection of nodes and routers in the Origin2000 to build a 32 processor system.

communication will be discussed later. Finally, dynamic load-balancing is performed to minimise the time a processor is left idle, but is only required after the simulation has been stopped. Although developed independently, our approach could be considered most similar in spirit to the recent work of Renambot *et al.* [RAPP97].

In the following sections, we will present the details of our approach. Section 5.3 first describes the architecture of the distributed shared memory Origin2000 on which the algorithm has been developed and tested. Sections 5.4, 5.5 and 5.6 then outline how data is partitioned amongst processors, how sub-iterations are performed in parallel, and how dynamic load-balancing is achieved. Two different scheduling strategies are given, one attempting to maximise the convergence of the simulation, and the other attempting to reduce remote memory accesses. Following that, the impact that visibility calculations have on the data-locality of the algorithm is then discussed in Section 5.7. Finally, results and performance statistics are presented in Section 5.8.

5.3 The Silicon Graphics Origin2000

The Silicon Graphics Origin2000 [LL97] is a cache-coherent non-uniform memory access multiprocessor (ccNUMA). The basic building block of each system is a *node*, which is composed of one or two MIPS R10000 processors, up to 4 GBytes of local memory, and a *hub* that manages each processor's access to both local and remote memory, and i/o devices. Multiprocessor systems are built by connecting multiple hubs together using a *router*, which can handle connections with to up to six hubs or other routers. Systems with up to 32 processors are built by connecting routers together in a hypercube topology (Figure 5.2). Beyond 32 processors, routers are connected to form a fat hypercube (up to 64 processors), or a hierarchical fat hypercube (up to 128 processors). In theory, the system is able to scale up to 1024 processors by replicating 128 CPU systems in a 5-D hypercube [LL97].

Each R10000 processor contains a two-level cache hierarchy – a 64 KByte 2-way set associative level-1 (L1) cache split evenly between instructions and data, and a 4 MByte 2-way set associative unified level-2 (L2) cache. The cache line size for the L1 data cache is 32 bytes, for the L1 instruction cache it is 64 bytes, and the L2 cache utilises a line size of 128 bytes. All caches employ a least-recently used replacement policy. In addition to the L1 and L2 caches, the R10000 also maintains a translation look-aside buffer (TLB) that stores mappings between virtual and physical addresses. Latencies for typical memory access times on the Origin2000 are given in Table 5.2 [LL97]. As the table shows, access to a processor's L1 and L2 cache are relatively cheap when compared to its local memory, and the cost of access to remote memory rises gradually due to the hypercube topology of the system.

When a process requests an allocation of memory, the operating system attempts to obtain that memory from the same node as the process is running on. When this is not possible, the memory is allocated from a nearby node. Remote memory pages (16 KByte) that are used heavily by one process may be *migrated* into its local memory. This can be done by the operating system, or explicitly by the application by using a software library. Additionally, software tools are provided to allocate groups of nodes on which to run parallel applications. An application may request a number of nearby node

Memory Level	Latency (ns)
L1 cache	5.1
L2 cache	56.4
Local Memory	310
4 Processor Remote	540
8 Processor Remote	707 (avg.)
16 Processor Remote	726 (avg.)
32 Processor Remote	773 (avg.)
64 Processor Remote	867 (avg.)
128 Processor Remote	945 (avg.)

Table 5.2: Latency for memory accesses on the Origin2000. This represents the time from when the L1 cache is accessed.

memories, and ensure that the parallel processes it runs will be placed on those nodes [Fie96].

In order for the parallel radiosity algorithm to exhibit good scalability, use must be made of this memory hierarchy. The partitioning and scheduling algorithm that will be described below have been designed to maintain an affinity to the data that a single processor requires.

5.4 Data Partitioning

In order to facilitate the processing of very large models (typically hundreds of thousands, or millions of polygons), the memory available in the parallel system must be used efficiently. In distributed memory systems with a relatively small amount of memory on each processor, the size of environment that can be processed will be limited if the data set for that environment is replicated in each processor's memory. The data describing structures such as element quadtrees, surface geometries, and grouping hierarchies must instead be partitioned between each processor. There are two competing requirements to any data partitioning performed, namely attempting to maintain *load-balance* between processors, and *data locality* for the work a single processor performs.

The environment must be partitioned in such a way as to balance the computational load on each processor as evenly as possible. Any imbalance will cause processors which have completed their allocated tasks to sit idle whilst others finish, thereby reducing the efficiency of the algorithm.

Given that the scene data is distributed between processors, the global nature of the radiosity simulation means that any processor may need access to the data that is stored in the memory of another. On distributed shared memory systems this remote access is more expensive than local accesses to a processor's main memory or cache, and each remote data access reduces the efficiency of the simulation. It is therefore necessary to partition the scene amongst the available processors in such as way that each processor requires access to as little remote data as possible. This will mean that a processor will more likely be able to re-use data that has already been transferred into its local memory or cache.

Unfortunately, obtaining a partition that satisfies both these requirements *a priori* is not possible because we do not know the access patterns or workloads until after the solution has been computed. This is especially true when acceleration techniques such as those discussed in Chapter 4 are employed, since the time required to perform a single iteration of the progressive radiosity algorithm will vary depending on the geometry of the scene and the radiosity of the source patch.

To produce a partitioning of the environment amongst processors that attempts to maintain both load-balance and data-locality, a recursive subdivision of the scene's patches is performed using a KD-tree. A root node is first constructed that contains all the patches in the environment, and this node is split recursively. In order to split a node into computationally equivalent parts, a measure of the estimated work for a node is required. The standard progressive radiosity algorithm is known to have time complexity of $O(n)$ for a single iteration, where n is the number of elements in the scene. The

Processors	Min	Max	$\bar{x}$	s
1	100.00%	100.00%	100.00%	0.00%
2	50.00%	50.00%	50.00%	0.00%
4	25.00%	25.00%	25.00%	0.00%
6	12.27%	25.00%	16.67%	6.14%
8	12.27%	12.73%	12.50%	0.20%
10	5.98%	12.66%	10.00%	3.14%
12	5.98%	12.36%	8.33%	2.96%
14	5.27%	12.30%	7.14%	2.22%
16	5.22%	7.07%	6.25%	0.51%

Table 5.3: Statistics for the partitioning of the scene shown in Figure 5.4 between processors.

total computation time for a node will therefore depend upon the number of elements it contains, and the number of source patches that distribute energy to those elements. If we assume that during each iteration, approximately the same amount of energy is transferred to all patches, and each patch is subdivided into approximately the same number of elements, then we can use the number of patches contained in a node as a measure of the computational load.

A splitting plane is therefore placed along the longest side of the node, and its position along that direction set to the averaged centre of all patches contained within that node. Patches contained entirely on either side of the split are then placed in the appropriate child node. Once all of these patches have been assigned, those that straddle the splitting plane are assigned to a child node in order to make the number of patches in each as equal as possible. This partitions a node into two sections that are compact, and will require approximately equal amounts of computation.

There are some limitations to this estimate of computational cost, especially when acceleration techniques such as perceptually adaptive shadow testing (Section 4.6) are introduced, since the time taken to perform the radiosity transfer between a source patch and a receiver can vary enormously. Because of this, additional *dynamic* load-balancing methods that correct for any outstanding cost differences will be introduced later.

As the KD-tree is constructed, a list of leaf nodes is maintained, as it is these nodes that are assigned to processors. Whilst the length of this list is less than the number of partitions required, the node with the largest number of patches is removed from the list, split into two parts, and its children added to the list. Once the correct number of partitions has been obtained, each processor is assigned a node to work on. The data structures corresponding to the patches, elements and vertices associated with each KD-tree partition are migrated into the local memory of the processor to which it has been assigned. The sections of the hierarchical grid-cell that intersect the partition are also migrated, along with the geometric polygon information that will be used for intersection testing.

Figure 5.3 shows how four partitions may be obtained using this subdivision technique. Adjacency information (shows as dotted links between the nodes) is also maintained during the construction of the KD-tree, indicating which partitions share faces (or parts of faces) with other nodes. These links are placed between nodes that share a significant fraction of a face, and will be used in the next section when a novel scheduling algorithm is described.

The partitioning of a typical environment containing over $50,000$ patches is shown in Figure 5.4 for 1, 2, 4, 8, 12, and 16 processors, with the patches colour-coded according to the processor to which they have been assigned. The minimum, maximum and average percentage of patches assigned to each processor, along with the standard-deviation are also given in Table 5.3. The partitioning algorithm performs well for 2^n processors ($n = 1, 2, 3 \ldots$), as every partition is split the same number of times. For other numbers of processors, several partitions will be subdivided more than the others, and so an imbalance in the number of patches occurs. The dynamic load-balancing algorithm introduced in Section 5.6 will attempt to correct for this.

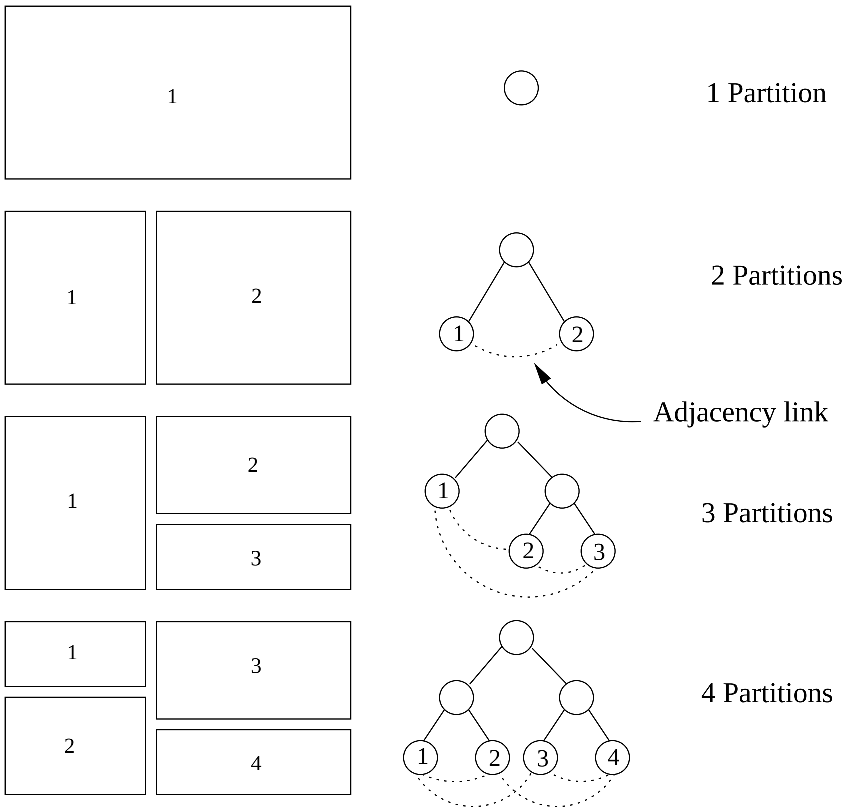
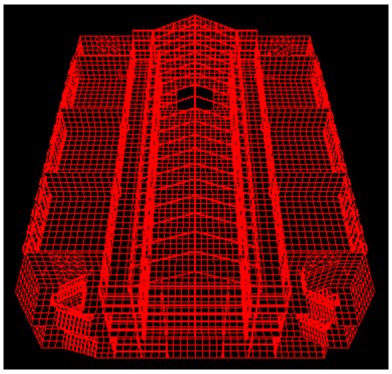
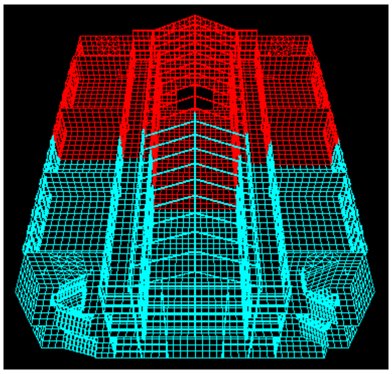
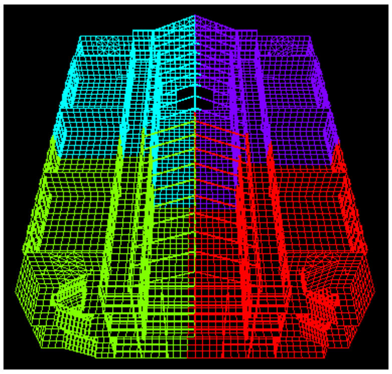
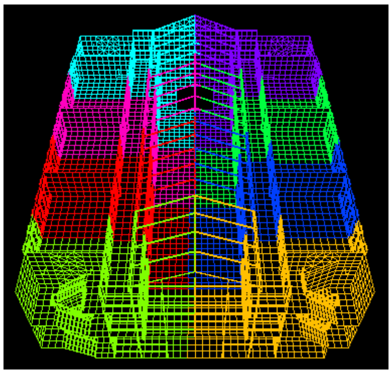


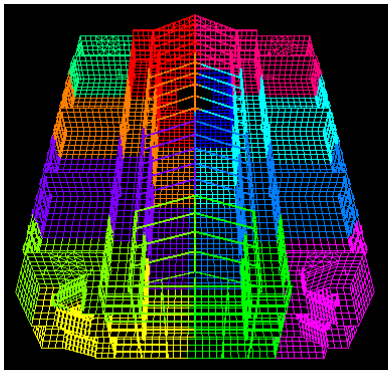
Figure 5.3: Recursive subdivision of the scene into a set of partitions to be assigned to processors.


(a) 1 Processor


(b) 2 Processors


(c) 4 Processors


(d) 8 Processors


(e) 12 Processors

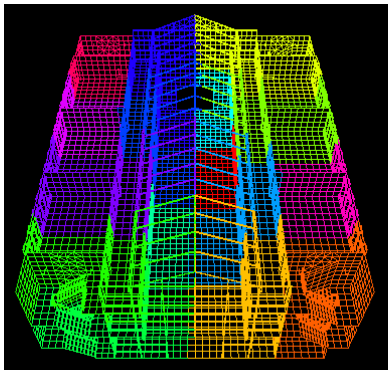

(f) 16 Processors

Figure 5.4: Partitioning of a typical environment between processors.

```
1  Repeat {
2      for each partition
3          transfer pending sources into ready queue
4
5      select a partition
6      if (partition's ready queue is not empty) {
7          dequeue a source patch
8          export patch to pending queues in adjacent partitions
9          perform a sub-iteration
10     }
11     else {
12         test stopping criteria
13         if (not stopped) {
14             select new source patches
15             place patches in processor's pending queues
16         }
17         else finish
18     }
19 }
```

Figure 5.5: Pseudo-code for the scheduling of sub-iterations.

Our algorithm allows for any number of partitions to be assigned to a single processor. When multiple partitions are being assigned, the KD-tree is first subdivided into n nodes, where n is the number of processors. Each of these nodes is then subdivided further, until the required number of partitions for each processor is obtained. This ensures that the partitions assigned to a single processor are located together in one branch of the KD-tree hierarchy, which will be important for achieving good cache usage during visibility computations (Section 5.7).

5.5 Sub-iteration Scheduling

Instead of considering the iterations of the progressive algorithm as being an ordered set of mutually-exclusive operations, each to be completed before the next begins, we take an alternative approach by introducing the concept of a *sub-iteration*. One sub-iteration distributes radiosity from a source patch to the receiver patches contained within a single KD-tree partition assigned to a processor. One complete iteration is therefore equivalent to performing all sub-iterations for that source patch. Sub-iterations for source patches may be performed in any order, and so once a processor finishes working on one sub-iteration, it can continue with another for the same group of receiver patches, meaning data-locality can be preserved with respect to the set of receiver patches in a processor's node.

Figure 5.5 gives pseudo-code for the scheduling of sub-iterations. Each processor executes the *Repeat()* procedure, with the set of KD-tree partitions it has been assigned. Two queues of source patches are maintained for each partition: the *ready* queue stores those source patches that are ready to be used to perform sub-iterations, whilst the *pending* queue stores source patches that need to be inserted into the ready queue. During initialisation, each processor examines the patches in each of its partitions, and places those with non-zero emission values in that partition's pending queue.

The first thing a processor does before performing a sub-iteration is to examine all its assigned partitions, and take all pending source patches and insert them into the partition's ready queue (lines 2 and 3). The order in which the source patches are placed in the ready queue is determined by the current scheduling strategy (see below), and will affect the convergence rate and data locality of the simulation. A partition is then selected using a method that also depends upon the scheduling strategy. Provided the

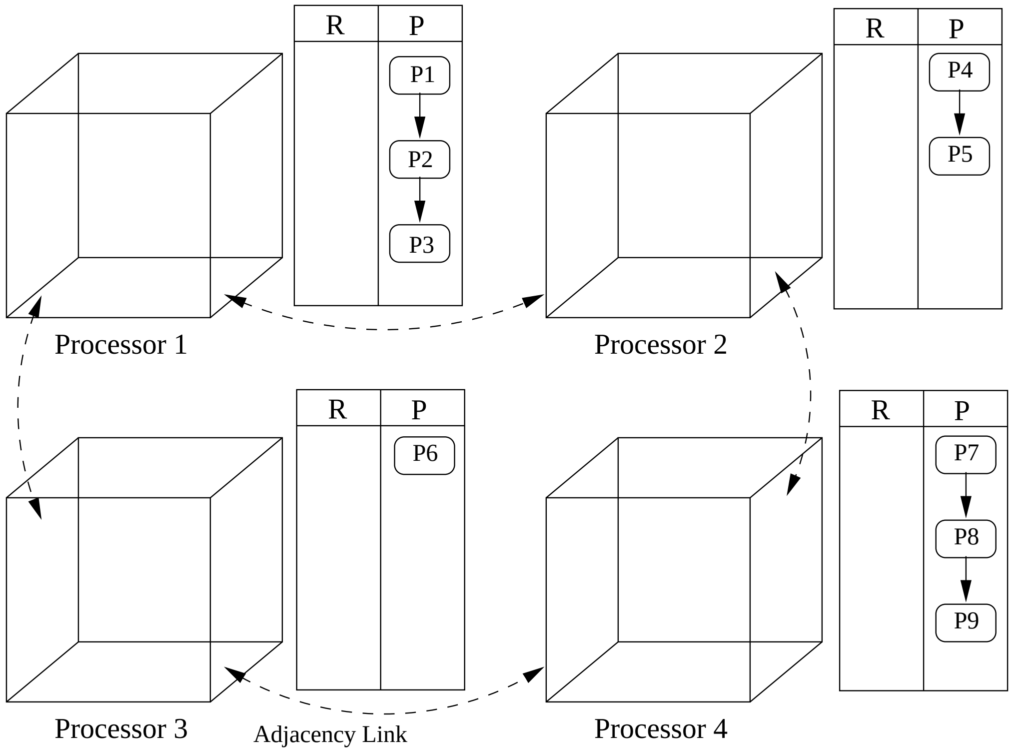


Figure 5.6: Four processors are assigned KD-tree partitions, and the pending queues of each is filled with primary light-source patches.

ready queue of the selected partition is not empty, a patch is removed from the head of the queue (line 7). Because each source patch will initially be present in only a single queue, the dequeued patch must then be placed in the pending queues of the other partitions (line 8) so that it will eventually distribute its energy to the entire scene. The adjacency links that are maintained whilst the KD-tree is built are used for this purpose – each source patch is added into the pending queue of those partitions that are adjacent to the selected partition. The processor then performs a single sub-iteration, distributing the unshot energy of the source patch to the receiver patches in the selected partition (line 9).

Due to the locations of light-sources in the environment, some processors may initially have empty pending queues, and would therefore be unable to perform any useful work at the start of the simulation. To avoid this situation, patches that are placed in the pending queue during initialisation are immediately added into the queues of all other partitions.

This scheduling technique may be implemented efficiently only because each partition has two queues. Consider what would happen if only a single queue was used: access to this queue must be limited to a single processor at a time using a mutex (*mutual exclusion*) lock, and so inserting a source patch in the correct position requires first locking the queue, traversing its contents to find the insertion position, inserting the patch, and then unlocking the queue. When large numbers of patches are being moved between partitions, access to each queue would become inefficient, with many processors waiting for a queue to become unlocked.

Using two queues reduces the chance of a processor being stalled. When a patch is exported to a pending queue, it is simply appended onto the tail of the queue, meaning the time that the queue remains locked is greatly reduced. Similarly, when the processor moves pending patches into a ready queue, the pending queue must only be locked whilst a single source patch is removed. Because each ready queue is only ever accessed by a single processor, the search for the best point of insertion can always take place without the need for mutex locks.

Figures 5.6 to 5.9 give an example of the execution of the algorithm. Figure 5.6 shows four processors,

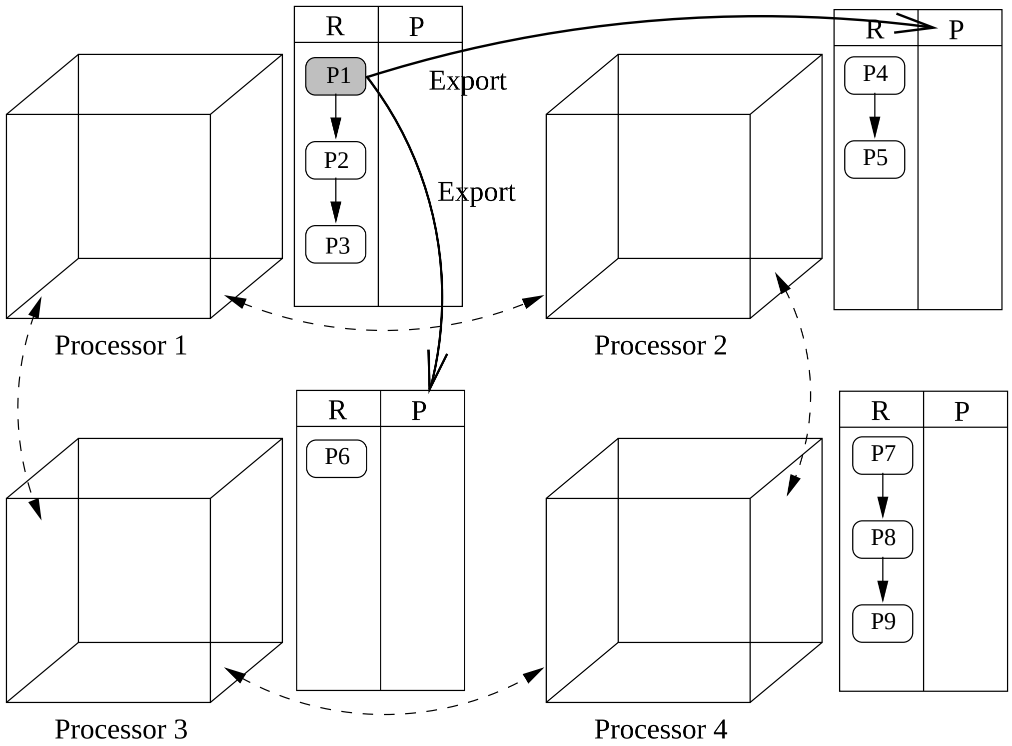


Figure 5.7: Before processor 1 performs a sub-iteration, the selected source patch is exported to the pending queues of adjacent partitions (i.e. those belonging to processors 2 and 3).

each of which has been assigned a single partition (the adjacency links between partitions are shown as dotted arrows). At the start of the simulation, each processor's pending queue (P) is filled with primary light-source source patches, and each ready queue (R) is initially empty.

Once the simulation has begun, each processor examines its partitions, and inserts all of the pending sources into the partition's ready queue (Figure 5.7). Processor 1 then selects patch 1 as its current source patch. In this case, the partitions of processors 2 and 3 have adjacency links to partition 1, and so this patch is placed on the tail of the pending queues of partition 2 and 3. A sub-iteration is then performed by shooting energy from the current source to partition 1's receiver patches (Figure 5.8). For clarity, this example assumes the other processors have remained idle, although in reality, patches from each processor will be exported and have sub-iterations performed with them. Once the other processors have computed a sub-iteration, the source patches in each of their pending queues are inserted into the ready queues.

Whilst the ready queues of a processor's partitions remain non-empty, sub-iterations can continue to be performed independently of the computation of other processors. Once all of the ready queues assigned to a processor become empty, however, no more source patches can be obtained. When this happens, the stopping criteria for the simulation must be tested for (line 12 in Figure 5.5). As in the serial algorithm, the simulation may be halted once a user-specified number of complete iterations have been computed, or when the solution is judged to have converged. The number of complete iterations may be measured by counting the total number of source patches that have been selected (see below). In order to measure convergence, each processor maintains the sum of the unshot power present in the patches in its KD-tree partitions. The sum of each processor's unshot power may be compared against the initial unshot power of the environment. Once it is found that the ratio of these is below a user-specified percentage, convergence has been achieved. Once either of these stopping criteria has been achieved, the simulation must still be continued until the ready queues of all processors are emptied in order to ensure that all source patches have distributed their energy to all receivers.

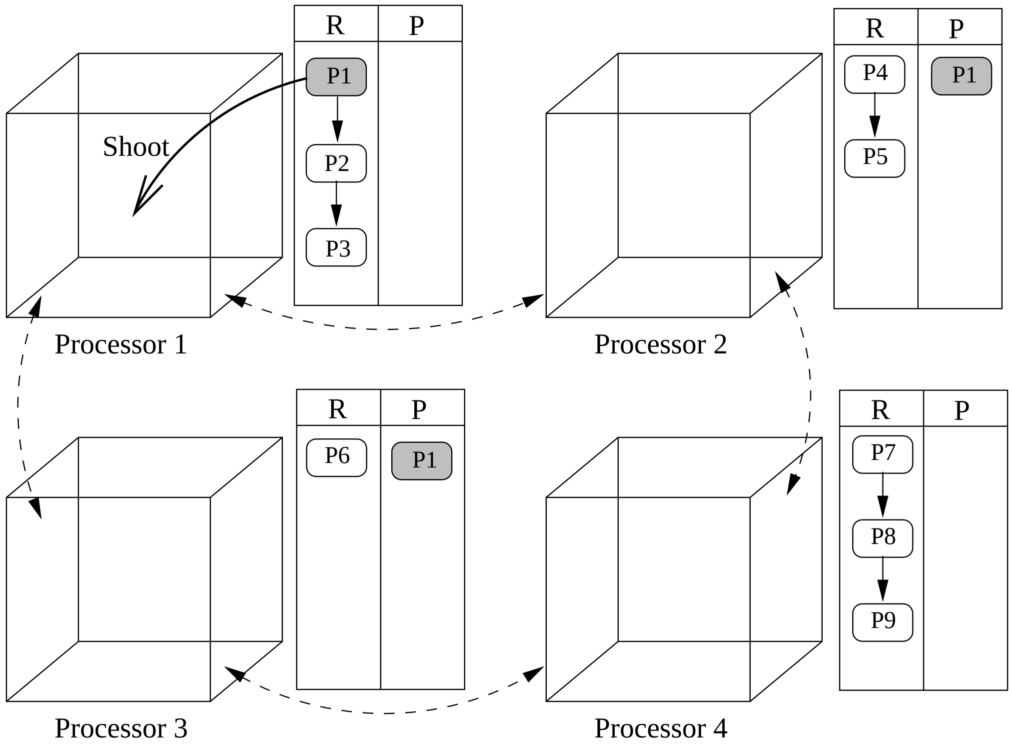


Figure 5.8: Processor 1 performs a sub-iteration, distributing energy from patch 1 to the receiver patches in its partition.

If the stopping criteria have not been met, new source patches must be selected and placed in the processor's ready queues (lines 14 and 15). The number of patches selected is set by the user before the simulation starts, and the effect this number has on performance will be discussed later. Along with each patch, a record of the patch's unshot radiosity is kept in the queue as well. As each new source patch is selected and inserted into the pending queue, its unshot radiosity is set to zero. This ensures that each patch shoots the same amount of unshot energy into each partition.

Until now, no mention has been made of the methods by which current partitions are selected, or by which sources from a pending queue are inserted into the ready queue and new source patches identified. There are two different strategies for performing these operations we have identified, *convergence-sensitive*, which aims to maximise the convergence rate of the simulation, and *locality-sensitive*, which attempts to minimise remote memory accesses.

5.5.1 Convergence-Sensitive Scheduling

In order to maximise the convergence rate of the simulation, the sub-iterations should be performed with source patches in decreasing order of their unshot power. The current partition with which to perform a sub-iteration is therefore selected as the one whose first ready patch has the greatest amount of unshot power. Convergence sensitive scheduling also places patches from a partition's pending queue into its ready queue in order of decreasing unshot power, so the first source patches removed from the head of the ready queue will have a greater effect on the convergence rate than those removed later.

When selecting new source patches, the N patches with the greatest amount of unshot power must be identified. Each processor maintains a list of its N brightest patches. When selecting new sources, the lists of each processor are identified, and the N patches with the globally greatest amount of unshot power found. This avoids each processor examining every patch in the environment, which would increase the amount of inter-processor communication.

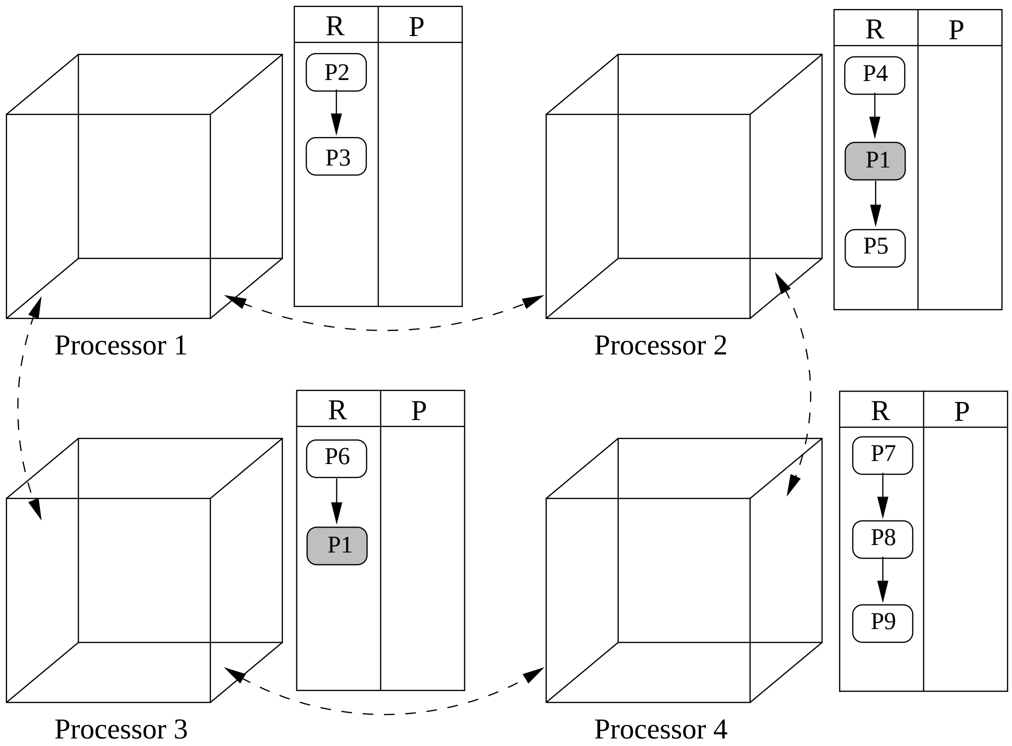


Figure 5.9: Before the next sub-iteration, each processor inserts all pending sources into its partition's ready queue.

5.5.2 Locality-Sensitive Scheduling

When CS scheduling is used to order source patches in the ready queue, data locality may be compromised with respect to the geometric data required for visibility calculations (see Section 5.7), due to the fact that the subsequent source patches may be scattered around the environment. Locality-sensitive (LS) scheduling attempts to minimise this by placing source patches in a ready queue in such a way that they are next to other source patches that originated from the same KD-tree partition.

Affinity for local receiver patch data may also be reduced by the method used to select a processor's current partition. With the LS scheduling algorithm, source patches are always retrieved from the current partition until that partition's ready queue is empty. Only then is a new partition selected.

In order to reduce the amount of inter-processor communication that occurs when new source patches are selected, each processor only selects sources from its assigned partitions. The combination of these techniques should mean that the LS scheduling method improves the data locality of the algorithm, and increases scalability. It may, however, affect the convergence rate of the simulation since sub-iterations are no longer being performed in the same order as for CS scheduling. These effects will be examined in Section 5.8.

5.6 Dynamic Load-Balancing

The methods for scheduling sub-iterations outlined above eliminate the need for load-balancing during the majority of the simulation. For scenes with a non-even distribution of patches, however, load-balancing will still be required once the simulation has been halted because it is impossible to guarantee that each processor will complete computing sub-iterations for all its assigned partitions at the same time.

```
Repeat {
    /* handle load-balancing requests */
    while (steal queue not empty) {
        dequeue requesting processor id
        if (enough work left) {
            give requesting processor some work
            signal STEAL_COMPLETED to requesting processor
        } else signal STEAL_FAILED to requesting processor
    }

    /* perform a sub-iteration */
    for each partition
        transfer pending sources into ready queue

    select a partition
    if (partition's ready queue is not empty) {
        dequeue a source patch
        export patch to pending queues in adjacent partitions
        perform a sub-iteration
    }
    else {
        test stopping criteria
        if (not stopped) {
            select new source patches
            place patches in processor's pending queues
        }
        else {
            /* request load-balancing if necessary */
            find target processor with most work left
            if (a target processor is found) {
                issue a steal request to target processor
                wait for signal
                if (signal is STEAL_FAILED)
                    finish
            }
            else finish
        }
    }
}
```

Figure 5.10: Pseudo-code for the scheduling of sub-iterations with dynamic load-balancing.

Pseudo-code for the sub-iteration scheduling algorithm with dynamic load-balancing is given in Figure 5.10. Because we have no 'master' processor, each processor must handle its own load-balancing. This is achieved by having a *steal-queue* associated with each processor. Once it is found that no more source patches are present in a processor's ready queues (line 16), and the user has requested to stop the simulation but other processors are still computing (line 22), the processor must issue a request for extra work.

The *requesting processor* identifies the *target processor* that has the most work left to perform. As a measure of work, we use the total number of receiver patches in a processor's KD-tree partitions, multiplied by the number of source patches present in the queues. A *steal-request* is then issued to the target processor by placing the requesting processor's identification number in the target's steal-queue. The requesting processor then sits and waits for a signal from the target processor indicating whether or not the steal request has been successful.

A processor always examines its steal-queue before each sub-iteration is computed (lines 3 to 9), and an appropriate chunk of work is given to each requesting processor. If more than one partition has been assigned to each individual processor, an entire partition is given. This involves migrating those areas of memory associated with that partition's patch and element data into the memory of the requesting processor. All information associated with the partition, such as its ready and pending queues, are also moved. The requesting processor will then 'own' that partition, and continue processing its sub-iterations. If only one partition has been allocated to each processor, or only one remains due to previous load-balancing requests, that partition must be split into two halves, and one half given to the requesting processor. The KD-tree node associated with the partition is therefore subdivided, a new partition is created for the requesting processor with one half of the receiver patches, and the pending and ready queues are copied. Once an appropriate chunk of work has been handed over, a *STEAL_COMPLETED* signal is given to the requesting processor. If the KD-tree node associated with a partition is too small to be split, a *STEAL_FAILED* signal is sent.

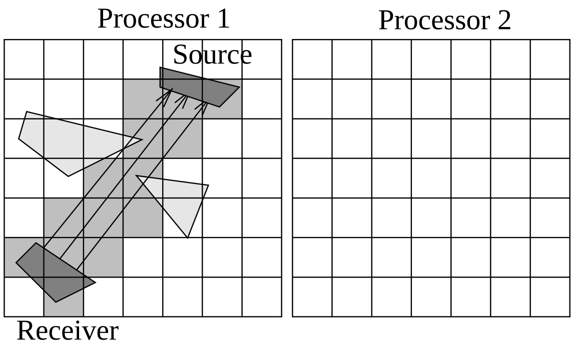
5.7 Visibility Testing

Scheduling sub-iterations using the approach described above ensures that each processor need only work with a subset of receiver patches. For a given size of environment, adding extra processors will reduce the number of patches each must manage, and, we expect, will increase cache performance. The only non-local data that is required to perform a sub-iteration is the geometric data needed for visibility calculations.

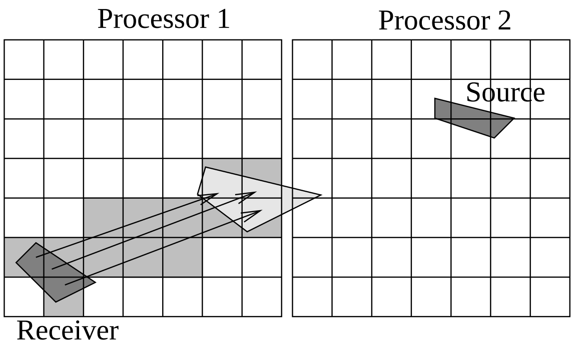
Using the perceptually-adaptive approach to visibility testing that was described in Chapter 4, several rays may be cast through the hierarchical grid-cell in order to obtain an 'accurate' visibility estimate for a form-factor (Section 4.6). When partitions are initially assigned to processors, blocks of the hierarchical grid-cell, and all geometric data contained within, are migrated to that processor's local memory. During visibility calculations, three situations can occur (Figure 5.11).

For source patches that are local to a partition (Figure 5.11(a)), all of the geometric data needed for intersection testing should already be present in that processor's local cache or memory, and access to this data will be relatively fast. When shooting radiosity from a non-local source patch, however, a ray can be occluded inside the bounds of the processor's set of partitions (Figure 5.11(b)). The direction that rays are traced now becomes important – since rays are traced from the receiver towards the source, the occlusion will be detected before the ray leaves the bounds of the grid-cell in the processor's local memory, and so no remote data access will be necessary.

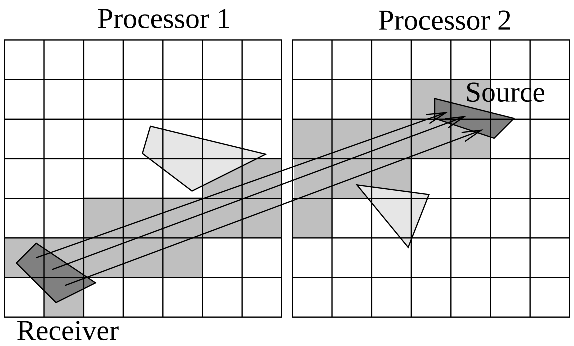
Alternatively, a ray may leave the part of the grid-cell that bounds the processor's partitions (Figure 5.11(c)). In this case, it must be intersected with geometric data that is not present in the processor's cache or local memory. In order to minimise the amount of inter-processor communication that occurs, each processor caches the last polygon with which an ray intersection was found, and tests each ray against this polygon before tracing through the hierarchical grid-cell. This simple caching technique has been found to reduce the number of ray-polygon intersections by approximately 40% for typical scenes. This reduction in intersection tests will cause a corresponding reduction in the number of inter-processor



(a) Source and receiver local to processor



(b) Source non-local, occlusion occurs locally



(c) Source non-local, no occlusion occurs

Figure 5.11: Visibility testing between source and receiver patches.

Scene	Polygons	Patches	Sources	Final Elements
Atrium	13,397	186,880	180	378,204
Building	218,401	425,282	4,594	1,590,802

Table 5.4: Test scenes used to evaluate the performance of the parallel radiosity algorithm.

communications that are required for visibility testing.

For radiosity transfers that do not require an accurate visibility calculation, a fast approximation was introduced in Section 4.6.1. This involved producing a uniform grid-cell containing one byte of visibility information for each voxel, and tracing a single ray through the grid, accumulating visibility information along the way. The approximate visibility test is most often used for radiosity transfers to distant patches, and, for the reasons outlined above, it is these transfers that would require the most remote memory access when using standard ray-casting techniques. Because the memory requirements for this grid-cell are small (a 100x100x100 grid-cell takes less than 1 MByte), this structure is replicated in the local memory of each processor. To improve cache usage when tracing a ray through the grid-cell, the data is grouped into 5x5x5 sub-cubes, allowing the visibility information associated with each sub-cube to be stored in a single 128 byte cache line. With the replication of this data structure and the use of the polygon cache, we hope to be able to limit the amount of inter-processor communication that occurs during the computation of each sub-iteration, thereby improving the efficiency of the algorithm.

5.8 Results

In this section, performance results and analysis of the proposed parallel radiosity algorithm are presented. Statistics showing the scalability of the algorithm with increasing numbers of processors, increasing environment size, and increasing length of simulation will first be given. We will then examine the effect the two different scheduling strategies have on convergence rate, and the improvement in performance brought by dynamic load-balancing.

During all reported tests, library calls to *dplace* [Fie96] were used to select node memories and run processes on the CPUs associated with those memories. Since each node in the Origin2000 system contains 256Mb of local memory and two R10000 CPUs, $n/2$ memories were selected for n processors, and one process was placed on each CPU. This meant that the memory bandwidth from each processor to its local memory was only half the maximum bandwidth of 800 $MB\ s^{-1}$. For a number of processors less than the available number of nodes in the system, the results reported below could be improved by running each process on a separate node. This was not done during these experiments, so that a valid comparison could be made regarding the typical scalability of the algorithm.

Single user access to the Origin2000 was not possible, and so all experiments were run whilst the machine was moderately loaded. The operating system also attempted to balance the computational load of the system whilst simulations were being run, and so processes were sometimes temporarily migrated to different CPUs. This affected the cache performance and data locality of the algorithm slightly. The results presented in this section may therefore be considered as being indicative of the performance that is obtained during typical use of the parallel radiosity system, rather than ideal results obtained from an otherwise idle machine. All reported statistics were obtained by averaging the results of four simulations. The variability in solution time for all simulations was, on average, around 5%, but never rose above 10%.

Due to the configuration of the batch queuing system on the Origin2000, it was not possible to obtain results for very large environments (more than 500,000 patches), or when using more than 16 processors. Performance statistics for such simulations will be obtained when appropriate access becomes available, and reported in a later publication.

Two test scenes were used to evaluate the performance of our algorithm, and these are summarised in Table 5.4. The 'atrium' scene is shown in Figure 4.13, and is a moderately complex model containing

Figure 5.12: Images from the radiosity solution of the 'building' model.

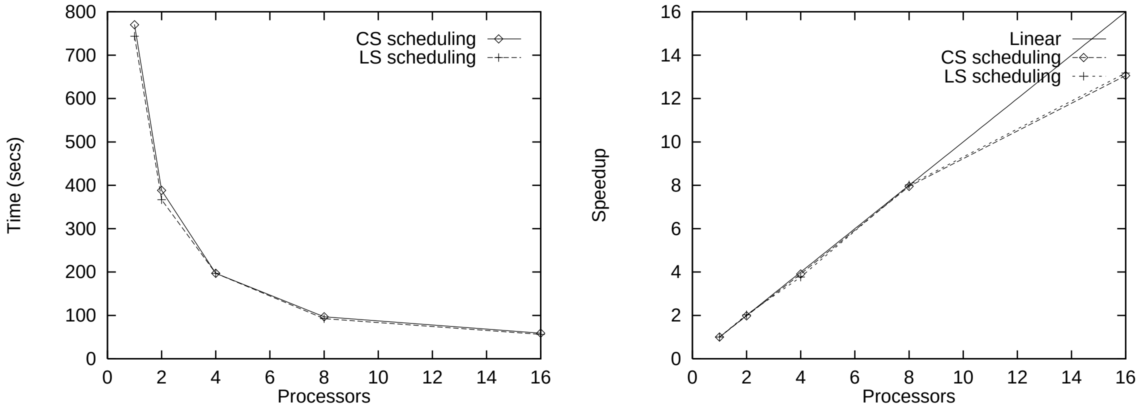


Figure 5.13: Solution time and speedup for the 'atrium' scene, direct illumination only (180 iterations).

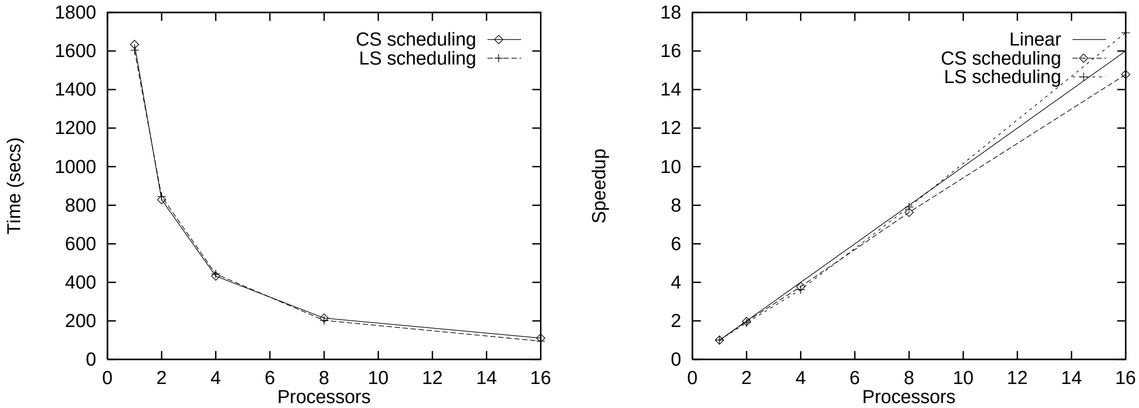


Figure 5.14: Solution time and speedup for the 'atrium' scene, 1000 iterations.

13,397 polygons. The 'building' scene is more complex, and contains 218,401 polygons. Images of this model are shown in Figure 5.12. Both of these scenes are typical of the architectural interiors with which radiosity-based lighting simulation is usually performed.

5.8.1 Speedup and Efficiency

Figures 5.13, 5.14 and 5.15 give execution times and speedup for the 'atrium' test scene. Speedup is defined as the ratio of solution time on a single processor (T_1) to solution time on N processors (T_N)[2]

$$\text{Speedup} = \frac{T_1}{T_N} \tag{5.1}$$

Efficiency is defined as the percentage of linear speedup that has been achieved on N processors

$$\text{Efficiency} = \frac{100 \times \text{Speedup}}{N} \tag{5.2}$$

When computing the direct component of illumination (Figure 5.13), similar performance results were obtained for both CS and LS scheduling, with good scalability up to the number of processors tested. Overall, an efficiency of 82% was achieved using 16 processors, with solution time falling from over 740 seconds to around 55 seconds. During all simulations, unless stated otherwise, 8 new source patches were selected once a processor's ready queue became empty.

[2]Solution time, as used here, does not include the time required to read in the model, or generate the initial patch mesh.

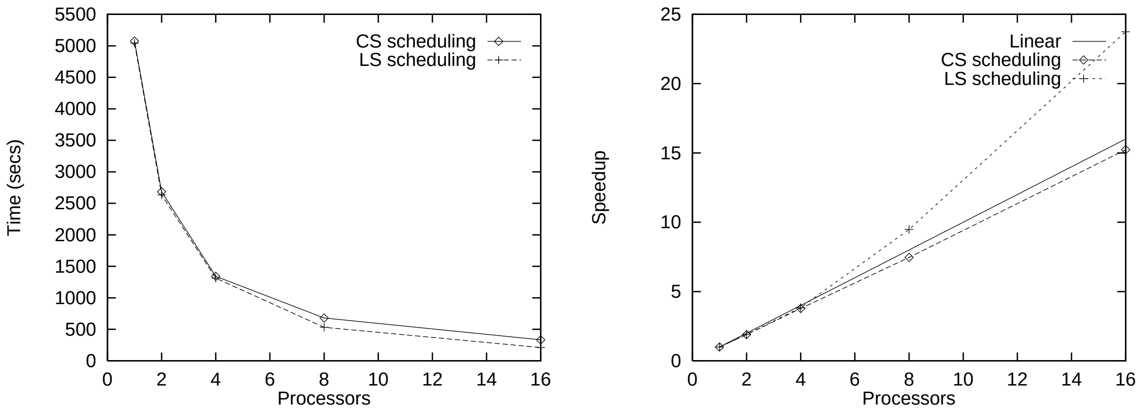


Figure 5.15: Solution time and speedup for the 'atrium' scene, 4000 iterations.

Processors	Time (secs)	Final Convergence	Speedup	Efficiency
1	5042	73.8%	1.00	100.0%
2	2634	72.6%	1.91	95.7%
4	1313	71.9%	3.84	96.0%
8	532	71.1%	9.48	118.4%
16	212	70.3%	23.74	148.4%

Table 5.5: Statistics for the 'atrium' test scene (4000 iterations, LS scheduling).

When the length of the simulation was increased to 1000 and 4000 full iterations[3] (Figures 5.14 and 5.15) scalability improved, due to the increase in benefit of a reduction in the size of each processor's data set (statistics showing data cache misses will be presented below). Over 1000 iterations, a speedup of almost 15 times was observed for the CS scheduling algorithm with 16 processors, reducing solution time from 27 minutes to only 110 seconds. When LS scheduling was employed, scalability became *super-linear*. Super-linear speedup can result in systems with a tiered memory, when reducing the problem size causes increased cache usage when compared to a single processor, provided the decrease in computation time out-weighs the usual sources of parallel inefficiency [Gus90]. In this case, a speedup of almost 17 times occurred when running on 16 processors.

As the simulation length was increased to 4000 iterations, the benefits of locality-sensitive scheduling became even more apparent, with a speedup of nearly 24 times observed for 16 processors. The CS scheduling algorithm performed less well, but still scaled almost linearly, running at over 95% efficiency. More detailed statistics for the 4000 iteration simulations are given in Tables 5.5 and 5.6.

Figure 5.16 shows solution time and efficiency for 8000 iterations with the 'building' model. Again, an almost linear speedup was seen, with both the CS and LS scheduling algorithms giving similar levels of performance. Overall, a speedup of almost 15 times occurred when running on 16 processors. Statistics for these simulations are given in Tables 5.7 and 5.8.

In order to measure the success of the parallel algorithm at maintaining data locality for each processor, accesses to the first and second level caches were examined. The R10000 processors used in the Origin2000 system contain two on-chip performance counter registers that are able to count up to 32 events such as instruction and data cache misses. Each register may be programmed (using the *perfex* library [Fie96]) to count one of 32 events, or both registers may be multiplexed over all 32 events, and an approximate count of all obtained.

[3]A full iteration corresponds to a standard iteration of the serial progressive refinement algorithm. The simulation was halted after 1000 full iterations by signaling convergence once 1000 source patches had been selected and placed in pending queues.

Processors	Time (secs)	Final Convergence	Speedup	Efficiency
1	5075	73.8%	1.00	100.0%
2	2683	73.7%	1.89	94.6%
4	1342	73.1%	3.78	94.5%
8	679	72.7%	7.47	93.3%
16	333	72.0%	15.23	95.2%

Table 5.6: Statistics for the 'atrium' test scene (4000 iterations, CS scheduling).

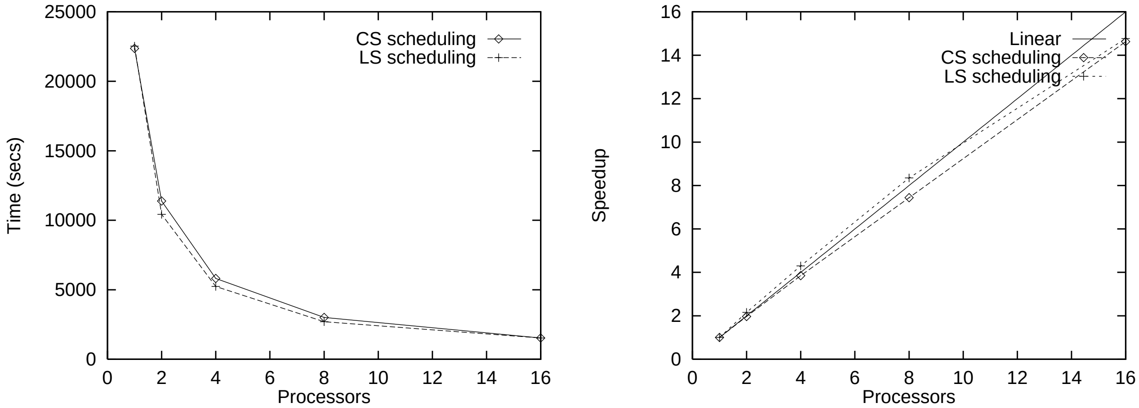


Figure 5.16: Solution time and speedup for the 'building scene, 8000 iterations.

Processors	Time (hours)	Final Convergence	Speedup	Efficiency
1	6.26	55.0%	1.00	100.0%
2	2.90	54.4%	2.16	108.1%
4	1.45	53.9%	4.30	107.6%
8	0.75	53.1%	8.35	104.4%
16	0.42	52.4%	14.77	95.3%

Table 5.7: Statistics for the 'building' test scene (8000 iterations, LS scheduling).

Processors	Time (hours)	Final Convergence	Speedup	Efficiency
1	6.21	55.0%	1.00	100.0%
2	3.16	54.9%	1.97	98.2%
4	1.61	54.6%	3.85	96.2%
8	0.83	54.2%	7.44	93.0%
16	0.42	53.9%	14.63	91.4%

Table 5.8: Statistics for the 'building' test scene (8000 iterations, CS scheduling).

Recalling Table 5.2, a significant latency occurs between accessing a processor's 64KByte L1 cache and accessing the larger, 4MByte L2 cache, and between accessing the L2 cache and the 256MByte local memory. For this reason, exact counts of the L1 and L2 data and instruction cache misses were obtained. The perfex library reported this data in terms of the average number of misses per processor over the entire simulation. These values were therefore multiplied by the number of processors used to give the total number of cache misses during the simulation.

Figures 5.17 and 5.18 show the total number of L1 and L2 data cache misses for each length of simulation. As expected, the number of misses in the L1 data cache increased as longer simulations were performed, and was reduced as more processors were used (Figure 5.17(c)). The number of L2 instruction cache misses also increased with the length of the simulation, although remained relatively unaffected by the size of the processor set.

In Figure 5.18, the effect on the number of L2 data cache misses is shown. As the number of processors increased, the total number of cache misses was greatly reduced, and the LS scheduling algorithm clearly out-performed the CS algorithm, as expected. Overall, the total number of L2 data cache misses was reduced to 42% of its original value on 16 processors using the CS scheduling algorithm, and 16% using LS scheduling. L1 and L2 instruction cache misses were also measured, although no discernible pattern could be found in the results.

Similar graphs are given in Figure 5.19 for 8000 iterations with the 'building' model. Again, the drop in L2 data cache misses can be clearly seen.

5.8.2 Convergence

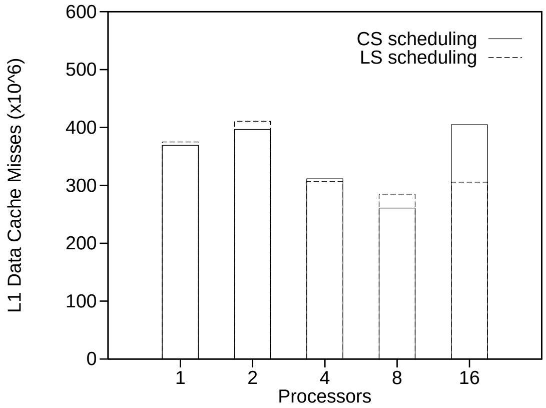
When parallelising the radiosity simulation using the algorithm proposed in this chapter, one observed effect is the slight reduction in convergence rate (as shown in Tables 5.5 and 5.6).

There are several factors that affect the convergence rate of the parallel algorithm. Firstly, the current scheduling strategy (Sections 5.5.1 and 5.5.2) determines how source patches are selected for each sub-iteration: CS scheduling attempts to maximise the convergence rate of the simulation by selecting the source patch that has the largest amount of unshot radiosity, whilst LS scheduling chooses source patches for which the required data will most likely already be present in the processor's cache. The rate of convergence will also be affected by the number of patches that are selected as new sources once the ready queues assigned to a processor become empty (Section 5.5).

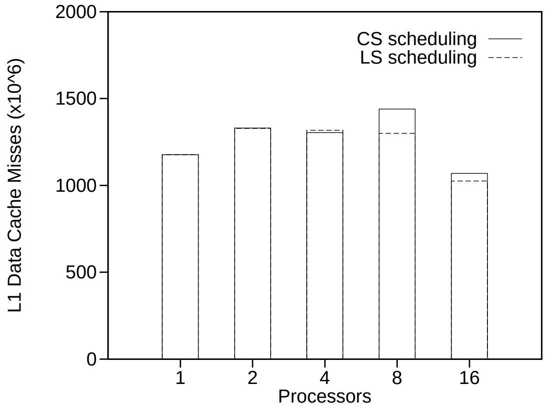
Figure 5.20(a) shows the rate of convergence using the 'atrium' model and 8 processors, with both convergence and locality-sensitive scheduling. Convergence was measured using the ratio of total unshot power in the environment to the initial amount of unshot power:

$$\text{Convergence} = 100 \times \left(1.0 - \frac{\text{Total unshot power}}{\text{Initial unshot power}}\right) \tag{5.3}$$

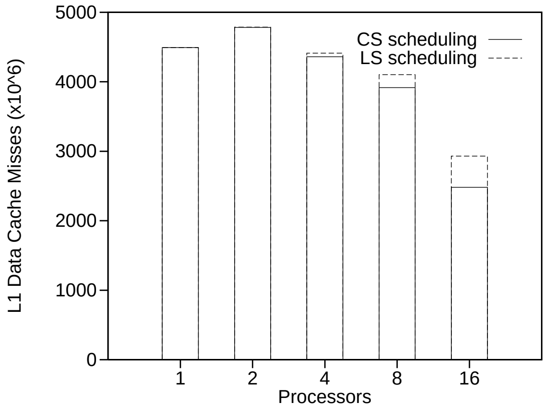
Although the CS scheduling algorithm was designed to increase the convergence rate of the simulation, it actually appears to converge slower than when LS scheduling was employed. This is due to the fact that the total unshot power in the environment (shown in Figure 5.20(b)), with which convergence is calculated, is decreased by the unshot power of a source patch as soon as that patch is selected by a processor and placed into a pending queue. Unlike the serial algorithm, where a fraction of this unshot power is immediately distributed back into the environment, these patches may not have their sub-iterations performed until later, when their energy is distributed into the other partitions and the total unshot power in the environment increased to the correct level. With CS scheduling, source patches may be selected from any task. LS scheduling, however, always selects new source patches from the processor's own set of partitions. The effect these local patches have on the processor's receiver patches will be greater than the effect of the (possibly non-local) patches selected under CS scheduling. The total unshot power in the environment will therefore be increased to the correct level at a slower rate when CS scheduling is employed. As an alternative measure of convergence, the total power *received* by patches is given in Figure 5.20(c). This graph clearly shows that although convergence appears to be faster with LS scheduling, the CS scheduling algorithm actually deposits power onto receiver patches at



(a) Direct illumination

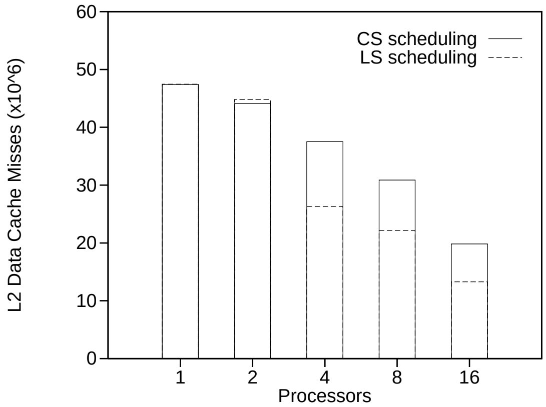


(b) 1000 iterations

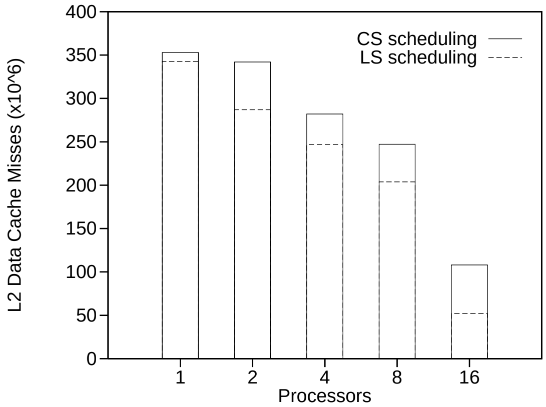


(c) 4000 iterations

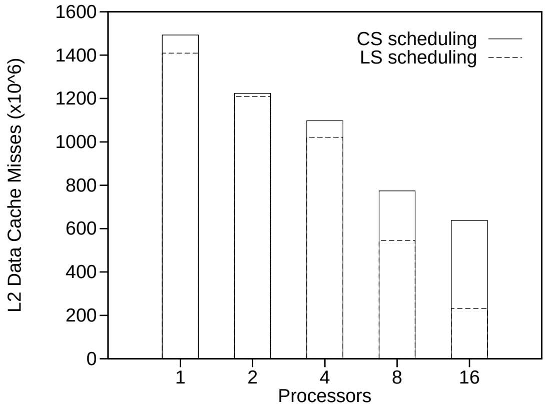
Figure 5.17: L1 data cache misses for the ‘atrium’ scene.



(a) Direct illumination

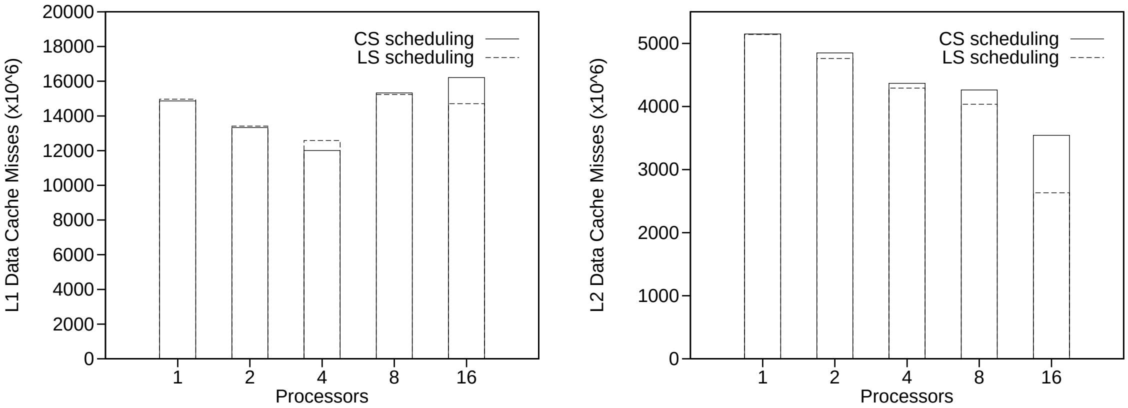


(b) 1000 iterations



(c) 4000 iterations

Figure 5.18: L2 data cache misses for the 'atrium' scene.



(a) L1 data cache

(b) L2 data cache

Figure 5.19: Cache misses for the 'building' scene, 8000 iterations.

a faster rate. The final convergence of the simulations after 1000 iterations was 70.4% for CS scheduling, and 69.5% for LS scheduling.

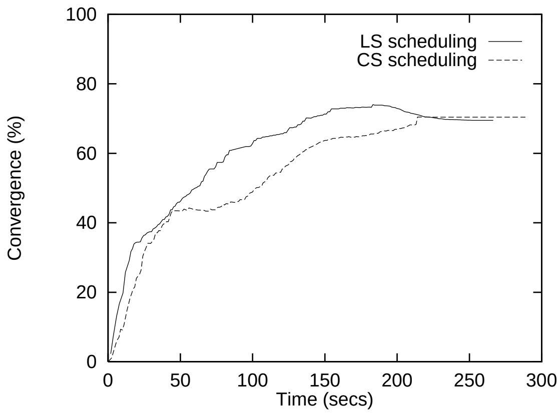
The overall reduction in convergence that was observed during the 4000 iteration simulations in Tables 5.5 and 5.6 is shown visually in Figure 5.21. The reduction is caused by the non-optimal selection of source patches, as with any other progressive radiosity algorithm that performs multiple iterations in parallel. Image (a) shows the solution after 1000 iterations on a single processor. Comparing image (b), which was obtained with the CS scheduling algorithm on 16 processors, the drop in convergence (from 73.8% to 72.0%) can be seen as a slight overall darkening of the image. Image (c), which was obtained with the LS scheduling algorithm, and was only 70.3% converged is noticeably darker, especially on the underside of the ceilings which receive indirect illumination only.

Figure 5.22 shows the final convergence of the solution after 1000 iterations with the 'atrium' model when different numbers of patches are selected as new sources of radiosity, using both LS and CS scheduling and 8 processors. It was expected that increasing the number of source patches would decrease the convergence rate of the algorithm, and this can be seen in Figure 5.22(a) for the CS scheduling algorithm. With LS scheduling (Figure 5.22(b)), the number of source patches made little difference to the overall convergence of the simulation, possibly due to the roughly even distribution of unshot power in the environment.

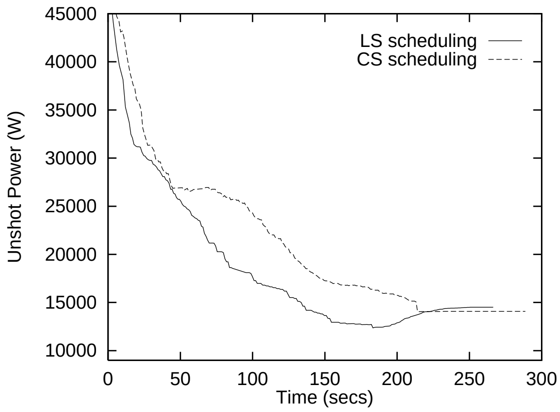
Finally, the solution time observed during these experiments is given in Figure 5.23. When larger numbers of source patches are selected, the time a processor can spend performing sub-iterations for a partition is increased. For the locality-sensitive algorithm, source patches are grouped in the ready queue according to the partition from which they originated. This in turn should increase the possibility of the re-use of data already in the processor's cache, and therefore we expect a decrease in solution time. Solution time for both the CS and LS scheduling algorithms remained relatively unaffected by the different number of source patches. This is possibly due to the reduction in remote data access due to the use of the replicated grid-cell and polygon cache described in Section 5.7 far out-weighing any extra reduction due to increased numbers of selected source patches.

5.8.3 Dynamic Load-Balancing

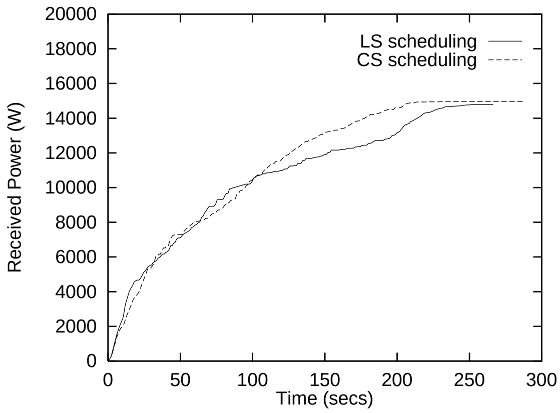
Figure 5.24 shows the difference in execution times (measured in seconds) dynamic load-balancing introduces. (a) shows processor execution times without load-balancing using 8 processors. The time between the first (processor 2) and last processor (processor 6) completing their work was 186 seconds - a 60% load imbalance. Solution times with dynamic load-balancing are shown in (b). The change of shade in each bar indicates from which processor work was taken. With dynamic load-balancing, solution time dropped from 310 to 204 seconds for this 8 processor simulation. Considering the time required on a single processor (approximately 1600 seconds, the dynamic load-balancing algorithm improved overall



(a) Convergence rate



(b) Total unshot power



(c) Received unshot power

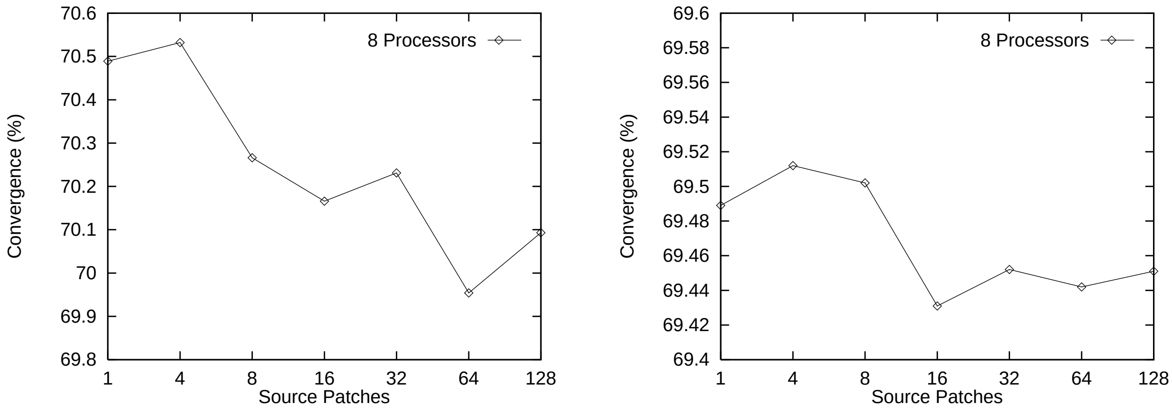
Figure 5.20: Convergence rate, total unshot power, and received power using CS and LS scheduling.

(a) One processor – 73.8% converged

(b) 16 processors, CS scheduling – 72.0% converged

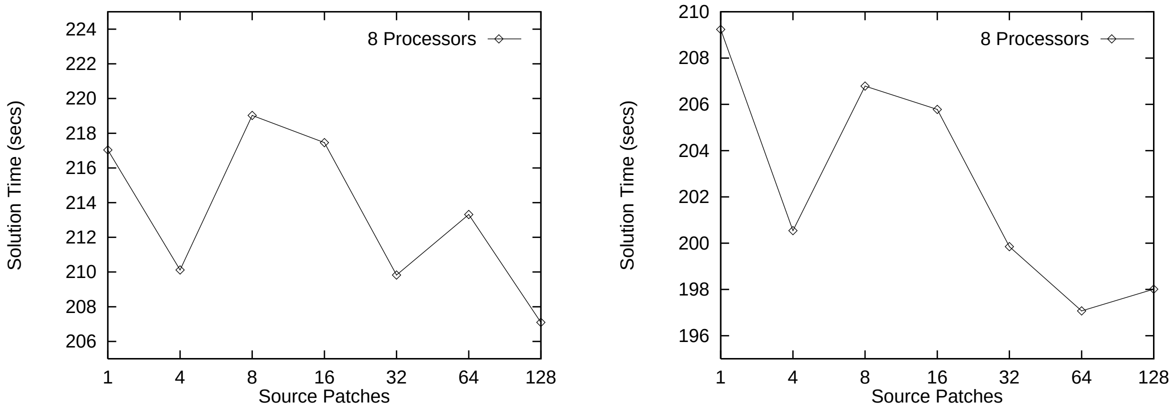
(c) 16 processor, LS scheduling – 70.3% converged

Figure 5.21: Ray-traced images of the atrium model, demonstrating the visual effect of a drop in convergence rate.



(a) CS Scheduling (b) LS Scheduling

Figure 5.22: Final convergences obtained using different numbers of selected source patches.

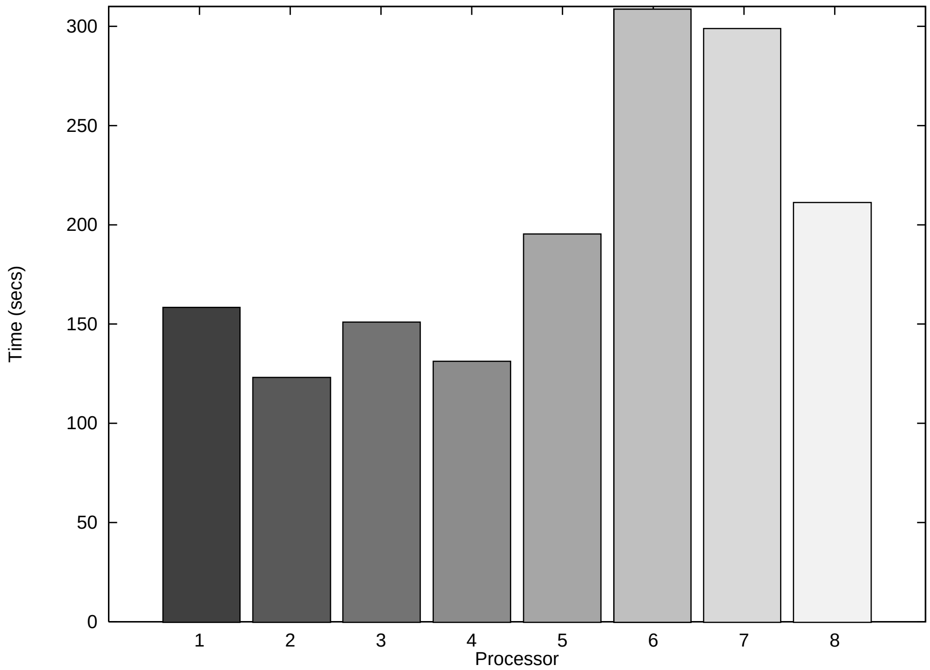


(a) CS Scheduling (b) LS Scheduling

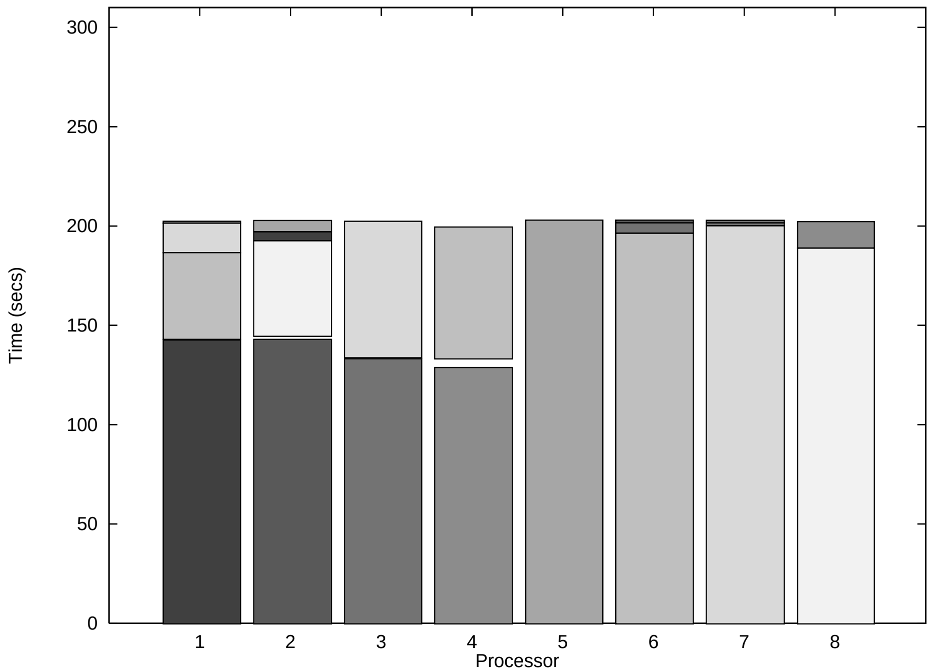
Figure 5.23: Solution times obtained using different numbers of selected source patches.

Processor	Unbalanced		Balanced		
	Time (secs)	Imbalance	Time (secs)	Idle (secs)	Imbalance
1	159.46	48.63%	203.47	1.25	0.61%
2	123.91	60.08%	202.62	2.10	1.03%
3	152.03	51.02%	203.10	1.62	0.79%
4	132.14	57.43%	196.53	8.19	4.00%
5	196.55	36.68%	204.35	0.37	0.18%
6	310.39	0.00%	204.24	0.48	0.23%
7	300.55	3.17%	204.22	0.50	0.24%
8	212.43	31.56%	203.61	1.11	0.54%
Total	1587.46	-	1622.15	15.62	-

Table 5.9: Difference in running times with and without dynamic load-balancing.



(a) Without dynamic load-balancing



(b) With dynamic load-balancing

Figure 5.24: Comparison of processor execution times without (a) and with (b) dynamic load-balancing.

efficiency from only 64% to over 98%.

When a requesting processor requires work from a target processor, it must wait for the target processor to complete its current sub-iteration before the request can be satisfied. The length of time for which each processor was stalled in this way is also given in Table 5.9. The longest total stall time corresponded to 4.0% of the solution time. The stall time will decrease, however, as more processors are used, since the length of time a single processor takes to complete a sub-iteration will also be reduced.

The total time spent doing useful computation with and without load-balancing is 1606.53 and 1587.46 seconds respectively. This means the overhead incurred by dynamic load-balancing is only 19 seconds – approximately 1.2% of the total computation time. This is mainly due to the system overhead that is incurred when memory pages are migrated between processors.

5.9 Summary

In this chapter, a novel approach to parallelising progressive radiosity was presented, and shown to scale well with increasing numbers of processors. After discussing previous attempts at parallelising progressive radiosity simulations, several important aspects of parallel radiosity algorithms were identified. By avoiding the master/worker paradigm used by many previous approaches, we hoped to avoid the communication bottleneck that occurs when using large numbers of workers, thereby improving efficiency. Attention to data-locality was also identified as being an important factor, as it reduced the amount of inter-processor communication.

The algorithm presented was based on the notion of a sub-iteration being the process of distributing energy from a source patch to a subset of receivers. A novel scheduling algorithm was described that enabled these sub-iterations to be performed without the need for processor synchronisation. Two scheduling strategies were identified: convergence-sensitive scheduling attempted to increase the initial convergence rate of the algorithm, whilst locality-sensitive scheduling was designed to increase the possibility of the re-use of data already present in a processor's cache.

The attention to data locality, along with an efficient dynamic load-balancing algorithm allowed the parallel simulation to scale well as increasing numbers of processors were used. Such an approach to performing parallel radiosity simulations, in combination with the perceptually-driven techniques introduced in the previous chapter permits very complex solutions to be generated quickly and efficiently.

There are many aspects of the parallel algorithm that deserve further attention. Although only one partition was assigned to each processor during these experiments, the scheduling and partitioning algorithms presented are general enough to allow multiple partitions to be allocated. The effect of these multiple partitions on scalability should be examined, as the locality-sensitive algorithm is expected to perform even better than with a single partition. This is due to the fact that the LS scheduling algorithm will continue to perform sub-iterations for the current partition until the ready queue of that partition is empty. Only then is another partition selected. This is contrasted by the CS scheduling algorithm, which may select a different current partition after each sub-iteration.

A combination of both convergence and locality-sensitive scheduling strategies should also be investigated, with new source patches selected using the convergence-sensitive approach, but scheduled to maximise data locality. This hybrid strategy should provide a near-linear speedup without a visually noticeable reduction in convergence rate.

The use of a cell and portal-based partitioning scheme as used by [TH93, LG95] could should also be investigated, especially for models such as the 'building' scene shown in Figure 5.12. With these schemes, individual cells could be identified as separate partitions, with adjacency information obtained from the inter-cell visibility structure. Source patches could therefore be transferred between adjacent rooms, and the portal information used to limit the number of receiver patches to which energy was distributed.

Finally, due to restrictions on access to the Origin2000 system used for this work, results could not be obtained for more than 16 processors, or scenes containing more than approximately 500,000 patches. The scalability of the algorithm on larger numbers of processors and with larger environments should be examined, in order to compare the approach to that of Renambot *et al.* [RAPP97] whose algorithm, although developed independently, can be considered the most similar of those published in the literature.

Chapter 6

Conclusion

6.1 Thesis Summary

Ever since the introduction of raster display devices, computer graphics researchers have aimed to generate ever more realistic images of synthetic models and environments. The rise in availability of computer power and memory has enabled ever more complex scenes to be rendered, using increasingly realistic illumination algorithms that model not only the local interaction of light with a surface, but also include important global effects such as shadows and inter-reflections.

The radiosity method, introduced to the rendering community in the mid 1980s [GTGB84, NN85], has become one of the most popular approaches to global illumination, due to its ability to simulate the inter-reflection of energy between surfaces in a view-independent manner. This allows perceptually important features such as soft shadows from area light-sources, and 'colour-bleeding' to be included in the illumination process. The view-independent nature of radiosity methods permits these subtle illumination effects to be pre-computed, and then visualised in real-time using standard Z-buffer rendering techniques.

The aim of the research presented in this thesis has been to develop methods for accelerating the production of visually accurate radiosity solutions. Two contributions have been made. Firstly, a new method for performing progressive radiosity simulations was presented that used perceptually-based metrics to control the solution process. An approach based on *tone-reproduction* was used to determine the perceptual similarity of two radiosity values, and limit the amount of computation required to distribute energy to a patch. This enabled visually accurate solutions to be generated more rapidly than when using standard radiometric measures. Secondly, a novel, scalable parallel radiosity algorithm was developed for distributed shared memory machines that accelerated the solution process even further.

6.1.1 Tone-Reproduction

After previous radiosity methods were discussed in Chapter 2, a detailed examination of the process of tone-reproduction was presented in Chapter 3. Tone-reproduction deals with the techniques used to display a radiosity solution on a display device in such a way that the subjective impression of the image that is formed by the observer is the same as he would obtain when viewing the real scene. Several published approaches were compared, each of which attempted to preserve different features of the human visual system, such as brightness or contrast. Overall, the model proposed by Ferwerda *et al.* [FPSG96] proved most useful, as it included not only the response of the eye to normal photopic conditions, but also the effect of the increased sensitivity of the rod photoreceptors at low levels of illumination.

Tone-reproduction operators, as described in the literature, have only been used to post-process images that have been produced using global illumination algorithms. When view-independent solutions are required, tone-reproduction must be applied to the entire radiosity mesh, rather than the pixels in an image. A method was proposed for performing such tone-reproduction operations that used a *static* real-world adaptation level. This adaptation level was obtained by averaging vertex luminances, and allowed any single-scale tone-reproduction operator to be used to transform those vertex luminances into display intensities. The results of applying static view-independent tone-reproduction were then compared to the more traditional dynamic approach.

6.1.2 Perceptually-Driven Radiosity

All previous radiosity algorithms aim to perform *radiometrically accurate* simulations, so that the final radiosity values that are calculated are as close as possible to the exact solution. For applications such as lighting design in architectural environments, this level of accuracy is necessary. For less stringent applications such as illuminating virtual environments, where the quality of the final image is more important, this numeric precision often becomes redundant when the radiosity solution is displayed on a device such as a monitor screen. This is due to the limitations of these display devices, such as their reduced colour gamut and dynamic range.

In Chapter 4, a framework was proposed for performing progressive radiosity simulations that only attempted to calculate as much detail as will be visible when a human observer views the final solution. This involved first producing a static, view-independent tone-reproduction operator, which required an *a priori* estimate of the final real-world adaptation level. This estimate was produced using a similar calculation to the ambient term for progressive refinement radiosity [CCWG88]. By estimating the final luminance of each patch in the environment, an approximate static adaptation level could be obtained *before* the radiosity simulation was performed.

The fundamental operation involved in the perceptually-based optimisations was the measurement of the perceptual similarity between two real-world luminance values. This was achieved by first applying the estimated tone-reproduction operator in order to determine the actual luminances that are displayed on the screen. These display luminances were then transformed into the perceptually-uniform CIELUV colour space. The perceptual similarity of the two real-world luminances, as would be seen by a human observer when the radiosity solution was displayed on the screen, could then be measured by applying a colour difference formula.

During the course of a radiosity simulation, adaptive refinement of the underlying element mesh must be performed in order to increase the accuracy of the solution. Previous refinement criteria that have been proposed have been based on purely radiometric comparisons, and so do not account for the perception of a human observer. A new adaptive refinement algorithm was presented that determined whether or not the change in radiosity representation over an element would be perceptually noticeable if extra refinement was performed. This technique was shown to be able to produce perceptually accurate radiosity solutions using fewer elements than the radiometric approaches. A refinement simplification technique was also introduced that was able to 'undo' refinement decisions once it was found that the extra subdivision was no longer perceptually significant.

Perceptual measures were also employed to accelerate visibility testing during form-factor computation. It was shown that a large percentage of the radiosity transfers that are performed during over the course of the simulation cause a perceptually insignificant change in the radiosity solution when considered individually. It was proposed that an approximate, but fast, visibility test could be applied during the form-factor calculations for these transfers, as an accurate test would not increase the perceived quality of the solution. This approximate test consisted of tracing a single ray through a uniform grid-cell, where each voxel in the grid stored visibility information computed as a pre-process. This resulted in a speedup of over 48 times for a relatively large test scene (in terms of physical size, rather than numbers of patches), and over 23 times for a more compact scene. The total number of rays that were traced in each case fell by almost 90% and over 97% respectively.

6.1.3 Scalable Parallel Radiosity using Distributed Shared Memory

Even when using acceleration methods such as those described above, simulations for geometrically complex environments can still take many hours to perform. Parallel processing has been proposed as a solution to this problem, and many of the most notable parallel radiosity algorithms have been published were examined in Chapter 5. Most previous algorithms have not shown scalable performance for complex test scenes. A novel algorithm that was designed for distributed shared memory architectures such as the Silicon Graphics Origin 2000 was then presented. Our approach was based on the notion of a *sub-iteration* being the transfer of energy from a single source patch to a subset of the scene's receiver patches. The patch data describing an environment was partitioned amongst the available processors, and an efficient queue-based scheduling strategy was developed to perform these sub-iterations concurrently. The algorithm was designed to exhibit good data locality with respect to the set of patches each processor worked on, and enabled a radiosity solution to be generated without the need for processor synchronisation between iterations. This free-running method of computation eliminated the need for load-balancing until the end of the solution process, and resulted in an almost linear speedup in the time required to generate a solution.

Details of the partitioning algorithm, and an outline of the scheduling process were given. Two different scheduling strategies were identified: *convergence-sensitive* scheduling attempted to maximise the convergence rate of the simulation, possibly resulting in some loss of data locality. Conversely, *locality-sensitive* scheduling attempted to increase the possibility of data re-use, at the expense of overall convergence.

The efficiency of the algorithm was examined for different numbers of processors, and increasing length of simulation, and was found to scale almost linearly in most cases. Due to differences in latency for data access at different levels of the memory hierarchy, a super-linear speedup was observed with the locality-sensitive scheduling algorithm for longer simulations, indicating that the algorithm succeeded in reducing the size of each processor's working data-set. Statistics regarding the number of data cache misses were also presented to confirm this result.

Due to the nature of the parallel algorithm, the order of source patch selection was altered when compared to the serial approach. This in turn affected the convergence rate of the simulation, and the effect of the two scheduling strategies on both initial and overall convergence was examined. Although the method by which convergence was measured indicated that the locality-sensitive strategy appeared to converge faster, examining the amount of power deposited on receiver patches showed that the convergence-sensitive algorithm did indeed converge more quickly.

Finally, the effect of the dynamic load-balancing algorithm was measured. The overall time spent doing useful computation with and without the load-balancing algorithm was obtained, and the overhead of the load-balancing found to be a small percentage of the overall computation time. During an 8 processor simulation, the inclusion of dynamic load-balancing was shown to increase efficiency from only 64% to over 98%.

6.2 Future Work

There are several limitations of the model used to measure perceptual differences in Chapter 4. The tone-reproduction operators that may be used to transform real-world luminances to display intensities are based on results from psycho-physical experiments using single colour targets and backgrounds. As stated by Bartleson *et al.*, the problem of measuring perceptual response in complex fields is not as easily performed, and it is unclear as to whether the results from simple target experiments may be extrapolated to more complex cases [BB67]. The measurement of the perceptual similarity of two real-world luminance values also depends upon the uniformity of the colour space, and the accuracy of the colour difference formula. Although more perceptually-uniform than CIEXYZ, the CIELUV colour space still has a non-uniformity ratio of approximately 6:1 [Poy93]. The simple Euclidean distance formula that was employed to measure the perceptual similarity of colours in CIELUV colour space is also not completely reliable, since the actual colour difference perceived depends upon factors such as the size of the stimulus [Car89], and the colour of the surroundings (so called *simultaneous contrast* [Phi89]).

As shown in Appendix A, the just noticeable difference between two colours, as measured by the Euclidean distance formula, was also seen to vary from $\Delta E^*_{uv} = 1$ to almost 30, depending upon the colours for which the difference was measured. The threshold that was chosen for the comparisons was set small enough to cover all these cases, but greater efficiency could be obtained by varying the threshold according to the chromaticity of the colours being compared. More accurate (and complicated) formulae exist (such as the CMC(1:1) colour difference formula which is better for evaluating small colour differences [Hun87], and the CIE94 model [MS95]), but these are based on the CIELAB colour system for reflected light and so are of no use when comparing colours on monitor screens.

The extension of the perceptually-based adaptive-refinement oracle to ensure Mach bands [Rat72] are not present in the image is a promising area of future work. This could be done by examining changes in the derivative of the display intensities derived by the algorithm. The possibility of an accumulation of error during the application of perceptually-adaptive visibility testing, when many transfers to the same vertex are computed using the approximate test, should also be examined. These artifacts have not proven a problem for the scenes we have examined so far, but they warrant further investigation.

Recent work by Ferwerda *et al.* has shown that individual errors may also be masked by their surroundings [FPSG97]. Their work shows how the presence of one visual pattern (e.g. a texture) can affect the detectability of another (e.g. a radiosity gradient). The application of such a model to the measurement of perceptual error over textured surfaces should allow the just noticeable threshold to be raised when it is known the texture will mask any artifacts.

There are other applications to which the model of measuring the perceptual similarity of real-world luminance values could also be applied. *Final-gather* methods have been proposed to generate high-quality static images of radiosity solutions [Rus88, Rei92, LTG93]. A ray is traced through each pixel in the image, and the elements or patches that contribute energy to the point of intersection are identified. Form-factor and visibility information is then recomputed for each transfer to that point, and the radiosity value for that pixel is recomputed. This results in accurate shadows and radiosity gradients in the final image. For high-resolution images of complex scenes, the time required to perform this final-gather is often far greater than the time spent computing the underlying radiosity solution [LTG93]. Several methods of limiting the number of recomputed form-factors have been proposed, such as only evaluating the direct illumination at each pixel [Rus88, LTG93], or evaluating only the most important sources of energy [KJ92].

All of these approaches attempt to identify the 'most important' sources of radiosity for each pixel, although none provide a reliable way of finding exactly what the most important sources are. Selecting a radiometric comparison threshold (as was done in [KJ92]) will prove difficult, for the same reasons that were given for selecting radiometric adaptive refinement thresholds in Section 4.2. Using the perceptually-based algorithm, source selection is greatly simplified – the form-factor and visibility must only be recomputed if the radiosity from the source patch caused a perceptually significant change at the receiver. This will ensure that all perceptually important shadows and radiosity gradients are rendered accurately.

Another interesting application of the perceptual radiosity comparison is for reducing network traffic during the remote visualisation of the progress of a radiosity simulation. One of the benefits of the progressive refinement algorithm is that it is capable of providing intermediate solutions more quickly than other methods. When the simulation is being run on a remote computer with no direct access to rendering hardware (as was the case for the parallel algorithm discussed in Chapter 5), this benefit is lost. One solution to this problem is to transmit changes in the radiosities of mesh vertices over a network, and render the results on a local workstation [RGG90, SWPW95]. For complex environments, communicating every change in vertex radiosity will quickly saturate the network, especially when the simulation is being run in parallel. By using a perceptually-based measure of the difference between two radiosity values, data may be sent only when a vertex radiosity changes by a perceptually significant amount, thereby reducing network bandwidth requirements.

The interactive manipulation of scene geometry and material characteristics in the setting of progressive radiosity ([Che90, GSG90, MS94]) is also a possible area of application. These techniques treat changes in geometry as the insertion and deletion of objects in the scene, and introduced the notion of shooting 'negative' energy to compensate for changes in light-source emission characteristics. One of the drawbacks with these methods is the large number of visibility calculations that must be made when an object is moved. Interactivity could possibly be improved by being able to identify those radiosity transfers that require an accurate visibility calculation in order to maintain the perceptual accuracy of the resulting images.

Finally, extensions should be made to allow hierarchical radiosity solutions to be computed using these metrics. This requires a hierarchical radiosity algorithm that measures error in terms of the accuracy of the radiosity representation across an element, rather than the error in radiosity transport from a source [LSG94, BW96]. Although surface-based hierarchical radiosity algorithms have proved useful for relatively simple scenes, clustering algorithms such as those discussed in Section 2.6.3 are essential for complex environments containing a large number of initial polygons. It is unclear how current hierarchical clustering algorithms may be extended to compute representation-based error estimates without losing the benefit of clustering. This is a promising area for future research.

Chapter 7

Colour Spaces

Colour is the perceptual response of a human observer to light in the visible region of the spectrum. As discussed in Chapter 3, the presence of three classes of cone photoreceptors in the human eye enables colours to be expressed as the linear combination of three primaries, forming a *colour space*.

The science of *colourimetry* concerns the methods by which three spectral weighting functions may be used to transform a spectral power distribution into tristimulus values that specifies a colour.

Due to the fact that different colour spaces have varying features and attributes, many have been proposed for different applications. In this appendix, some of the colour spaces that have been used in the main body of this thesis are discussed.

7.1 RGB Colour Spaces

Many colour spaces have been proposed for use in computer graphics, the most common of which is the RGB colour space. RGB colour systems are used by all CRT devices, and each tristimulus value independently controls the red, green and blue electron guns that illuminate pixels on the screen. Three spectral weighting functions are defined that resemble the spectral sensitivities of the S, M and L cones in the human eye (Figure 7.1 [WS82]). Each of these functions is used to weight the spectral power distribution of an incoming stimulus to produce tristimulus values representing the colour.

One problem with the RGB colour model is that it is device-dependent – the exact colour that is produced by a triple of numbers is not well defined, and the colour that is actually produced from a given triple of RGB values will vary between monitors.

7.2 CIEXYZ Colour Space

In 1931, the CIE (Commission Internationale de l'Eclairage) defined the device-independent CIE XYZ colour space. The CIE standard observer matching functions for this colour space are shown in Figure 7.2 [WS82]. XYZ space (unlike device-dependent RGB colour spaces) has the benefit of being able to represent all possible colours since its matching functions are positive over the visible spectrum. The Y tristimulus value corresponds directly to luminance, with the X and Z components derived from experimental statistics involving human observers.

The *chromaticity* of a colour represented using the CIE primaries is defined as the 'true' colour, depending only upon dominant wavelength and saturation. The chromaticity coordinates x, y and z of a colour may be obtained by normalising the tristimulus values X, Y and Z against their sum total:

$$x = \frac{X}{(X+Y+Z)} \tag{7.1}$$

$$y = \frac{Y}{(X+Y+Z)} \tag{7.2}$$

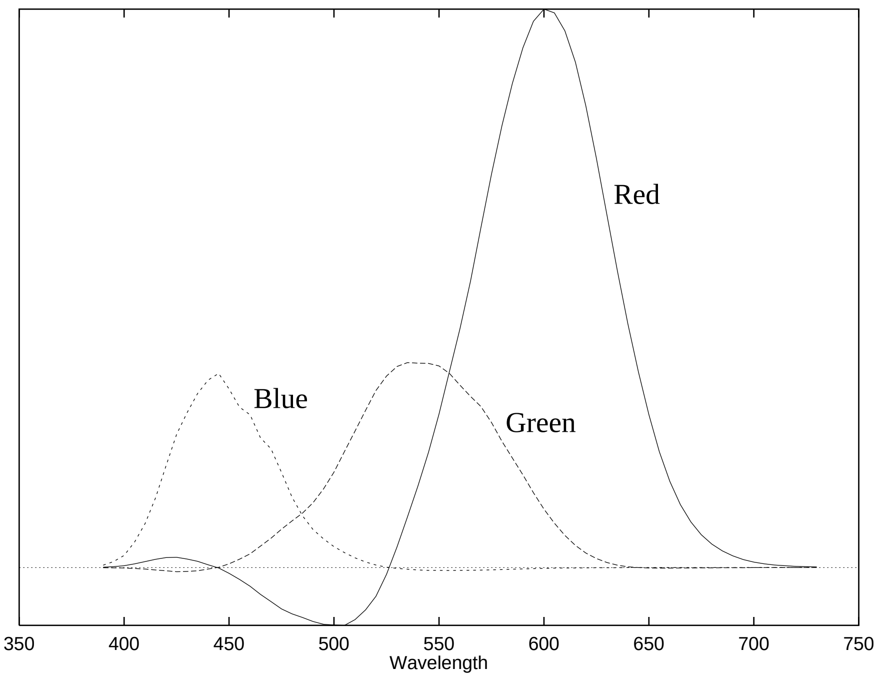


Figure 7.1: RGB standard observer matching functions.

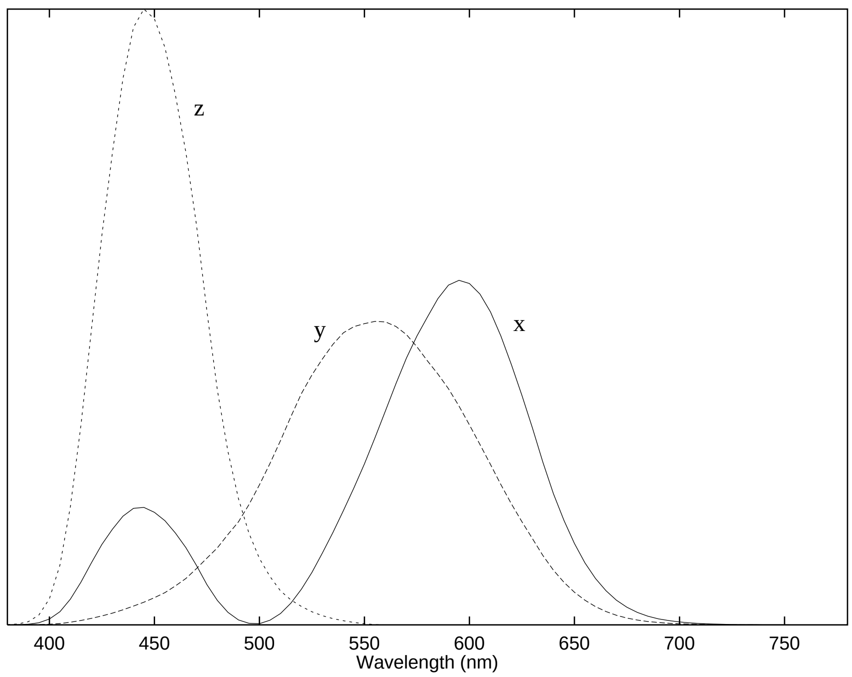


Figure 7.2: CIE standard observer matching functions.

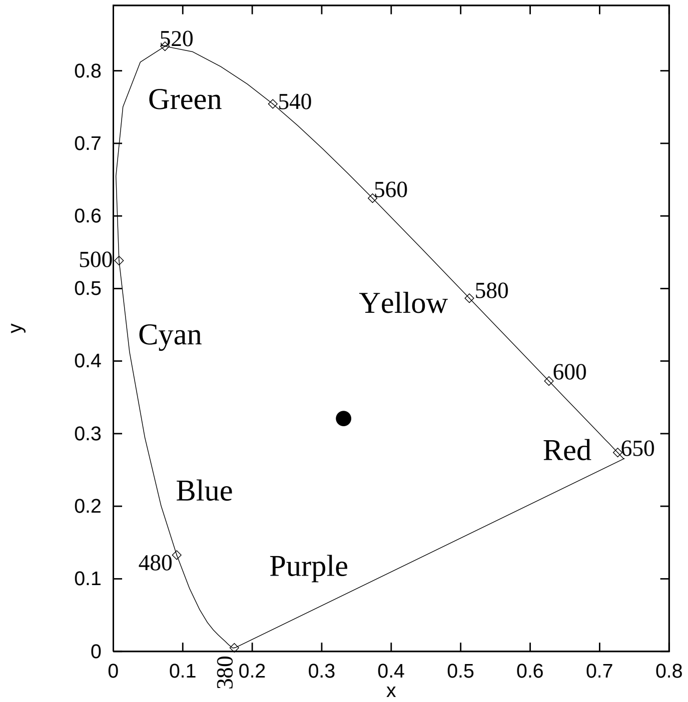


Figure 7.3: The CIE chromaticity diagram.

$$z = \frac{Z}{(X + Y + Z)} \tag{7.3}$$

The chromaticity of a colour is usually represented using the pair of values x and y, since $x + y + z = 1$. The CIE *chromaticity diagram* is given in Figure 7.3, and plots chromaticity coordinates for all values of X, Y and Z. The periphery of the diagram corresponds to the set of spectrally pure colours.

One problem with CIEXYZ space when it is used to numerically compare different colours is that it is not *perceptually uniform* – colours which are the same numerical distance apart may not be perceptually equidistant. This can be seen by overlaying *MacAdam's ellipses* on the chromaticity diagram(Figure 7.4), showing the areas of chromaticity for colours that are observed to be perceptually identical (note that the ellipses have been enlarged by a factor of ten for clarity). The CIEXYZ system, in fact, has a non-uniformity ratio of 80:1 [Poy93].

7.3 Perceptually-Uniform Colour Spaces

In 1978, the CIE introduced the CIELAB and CIELUV colour spaces. The ratios of non-uniformity are approximately 6:1 [Poy93], and although they are not perfect, these systems produce more perceptually-uniform numerical difference measurements. The CIELAB colour space was defined primarily for reflected light, and the CIELUV for additive light (such as CRT screens).

Given an (X, Y, Z) triple represented in the CIEXYZ system, the conversion into CIELAB or CIELUV colour spaces requires the tristimulus values for a standard white, (X_w, Y_w, Z_w), to be known. The transformation from (X, Y, Z) into CIELAB colour space is then given by [CIE78]

$$\begin{aligned} L^* &= \begin{cases} 116(Y/Y_w)^{1/3} - 16 & \text{if } Y/Y_w \geq 0.008856 \\ 903.3(Y/Y_w) & \text{otherwise} \end{cases} \\ a^* &= 500(f(X/X_w) - f(Y/Y_w)) \\ b^* &= 200(f(Y/Y_w) - f(Z/Z_w)) \end{aligned} \tag{7.4}$$

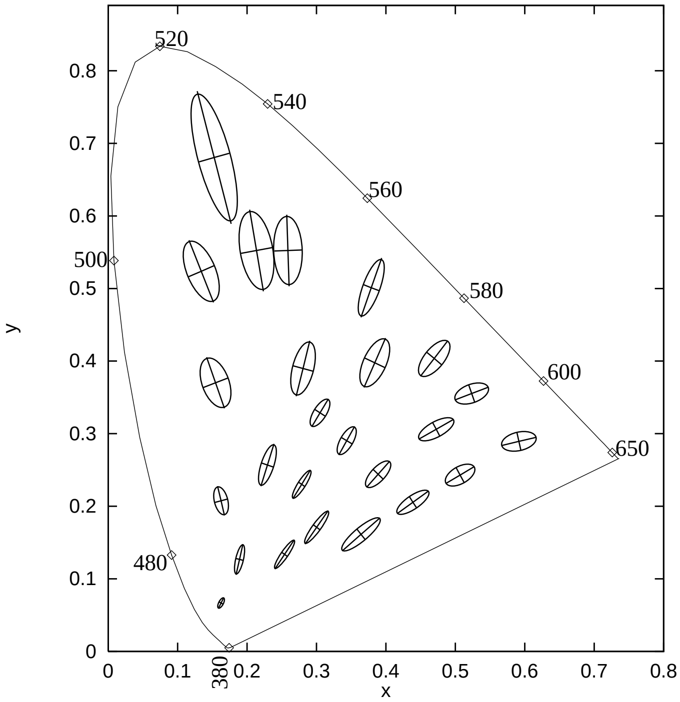


Figure 7.4: 1931 CIE chromaticity diagram showing MacAdam's ellipses (which have been enlarged by a factor of ten).

where

$$f(t) = \begin{cases} t^{1/3} & \text{if } Y/Y_w \geq 0.008856 \\ 7.787t + 16/116 & \text{otherwise} \end{cases} \tag{7.5}$$

L^* represents luminance, and a^* and b^* are respectively red/blue and yellow/blue chrominancies.

The transformation from (X, Y, Z) into CIELUV colour space is given by [CIE78]

$$\begin{aligned} L^* &= \begin{cases} 116(Y/Y_w)^{1/3} - 16 & \text{if } Y/Y_w \geq 0.008856 \\ 903.3(Y/Y_w) & \text{otherwise} \end{cases} \\ u^* &= 13L^*(u' - u'_w) \\ v^* &= 13L^*(v' - v'_w) \end{aligned} \tag{7.6}$$

where

$$\begin{aligned} u' &= \frac{4X}{X+15Y+3Z} \\ v' &= \frac{9Y}{X+15Y+3Z} \end{aligned} \tag{7.7}$$

and

$$\begin{aligned} u'_w &= \frac{4X_w}{X_w+15Y_w+3Z_w} \\ v'_w &= \frac{9Y_w}{X_w+15Y_w+3Z_w} \end{aligned} \tag{7.8}$$

L^* again represents luminance, and u^* and v^* define chrominancy. For the purposes of representing colours on a monitor screen, the CIELUV space is most suitable.

7.3.1 Colour Difference Formulae

The Euclidean distance between two colours (L^*_A, u^*_A, v^*_A) and (L^*_B, u^*_B, v^*_B) in CIELUV space gives a measure of their perceptual difference [CIE78],

$$E^*_{uv} = \sqrt{(L^*_A - L^*_B)^2 + (u^*_A - u^*_B)^2 + (v^*_A - v^*_B)^2} \tag{7.9}$$

More complicated formula for measuring small colour differences have recently been proposed, although these are all based on the CIELAB system. For example, the CIE94 [MS95] formula was defined as

$$E_{94}^* = \sqrt{\left(\frac{L_A^* - L_B^*}{k_L S_L}\right)^2 + \left(\frac{C_A^* - C_B^*}{k_C S_C}\right)^2 + \left(\frac{H_A^* - H_B^*}{k_H S_H}\right)^2} \tag{7.10}$$

where L_A^* and L_B^* and the luminance of colours A and B, C_A^*, C_B^*, H_A^* and H_B^* are the *chroma* and *hue* of the colours

$$C_A^* = \sqrt{(a_A^*)^2 + (b_A^*)^2} \tag{7.11}$$

$$H_A^* = \tan^{-1}(a_A^*/a_B^*) \tag{7.12}$$

S_L, S_C and S_H are weighting functions

$$\begin{aligned} S_L &= 1 \\ S_C &= 1 + 0.045 C_A^* \\ S_H &= 1 + 0.015 C_A^* \end{aligned} \tag{7.13}$$

and finally, k_L, k_C and k_H are parametric factors included to allow for adjustments to be made in order to account for deviations from the rigorously specified viewing conditions [MS95]. Similar colour difference formula that are suitable for measuring small differences in CIELAB colour space, such as the CMC(1:1) and BFD models have also been proposed [Hun87].

7.3.2 Just Noticeable Differences

A *just noticeable difference* is the smallest E_{uv}^* difference between two colours that is distinguishable by a human observer. Various values for the just noticeable difference have been proposed in the literature. Table 7.1 gives values measured during empirical experiments performed by Reddy [Red97]. Note the wide variability in ΔE_{uv}^* for different coloured stimuli.

Initial Colour	Increment Hue	Initial RGB	Final RGB	ΔE^*_{uv}
White	Red	255,255,255	235,255,255	5.10
White	Green	255,255,255	255,246,255	3.43
White	Blue	255,255,255	255,255,242	4.46
White	Cyan	255,255,255	255,247,247	2.30
White	Magenta	255,255,255	243,255,243	4.42
White	Yellow	255,255,255	247,247,255	3.00
Black	Red	0,0,0	11,0,0	29.02
Black	Green	0,0,0	0,7,0	28,20
Black	Blue	0,0,0	0,0,15	16.00
Black	Cyan	0,0,0	0,8,8	22.66
Black	Magenta	0,0,0	11,0,11	26.74
Black	Yellow	0,0,0	7,7,0	27.00
Black	White	0,0,0	7,7,7	19.00
Red	White	255,0,0	255,13,13	28.82
Red	White	128,0,0	134,6,6	16.13
Red	Red	128,0,0	136,0,0	3.93
Red	Red	128,0,0	120,0,0	4.10
Red	Red	255,0,0	244,0,0	3.54
Red	Green	255,0,0	255,10,0	17.38
Red	Green	128,0,0	128,7,0	16.79
Red	Blue	255,0,0	255,0,24	28.86
Red	Blue	128,0,0	128,0,17	28.73
Green	White	0,255,0	33,255,33	25.00
Green	White	0,128,0	5,133,5	4.43
Green	Green	0,128,0	0,134,0	2.33
Green	Green	0,128,0	0,121,0	2.82
Green	Green	0,255,0	0,244,0	2.79
Green	Blue	255,0,0	255,0,24	28.86
Green	Blue	128,0,0	128,0,17	28.73
Green	Yellow	0,255,0	0,255,21	13.04
Green	Yellow	0,128,0	0,128,17	15.65
Blue	White	0,0,255	15,15,255	10.14
Blue	White	0,0,128	8,8,136	11.54
Blue	Blue	0,0,128	0,0,140	4.85
Blue	Blue	0,0,128	0,0,117	4.72
Blue	Blue	0,0,255	0,0,242	3.48
Blue	Yellow	0,0,255	15,15,255	10,14
Blue	Yellow	0,0,128	8,8,128	9.37
Blue	Magenta	0,0,255	16,0,255	6.94
Blue	Magenta	0,0,128	8,0,136	7.50
Cyan	White	0,255,255	41,255,255	12.89
Cyan	White	0,128,128	8,136,136	3.18
Cyan	Cyan	0,128,128	0,134,134	1.67
Cyan	Cyan	0,128,128	0,123,123	1.43
Cyan	Cyan	0,255,255	0,245,245	1.81
Magenta	White	255,0,255	255,20,255	15.72
Magenta	White	128,0,128	124,6,134	5.19
Magenta	Magenta	128,0,128	136,0,136	3.10
Magenta	Magenta	128,0,128	120,0,120	3.21
Magenta	Magenta	255,0,255	247,0,247	2.00
Yellow	White	255,255,0	255,244,45	22.50
Yellow	White	128,128,0	133,133,5	2.84
Yellow	Yellow	128,128,0	134,134,0	2.10
Yellow	Yellow	128,128,0	122,122,0	2.13
Yellow	Yellow	255,255,0	245,245,0	2.23

Table 7.1: Smallest discernible CIELUV colour differences for a variety of coloured stimuli.

Bibliography

[AcO96] C. Aykanat, T. K. Çapin, and B. Ozguc. A Parallel Progressive Radiosity Algorithm Based on Patch Data Circulation. *Computers & Graphics*, 20(2):307–324, 1996.

[Ade82] E. H. Adelson. Saturation and Adaptation in the Rod System. *Vision Research*, 22:1299–1312, 1982.

[AH93] L. Aupperle and P. Hanrahan. A Hierarchical Illumination Algorithm for Surfaces with Glossy Reflection. *Computer Graphics Proceedings, Annual Conference Series, 1993 (ACM SIGGRAPH '93 Proceedings)*, pages 155–162, 1993.

[AK90] J. R. Arvo and D. B. Kirk. Particle Transport and Image Synthesis. *Computer Graphics (ACM SIGGRAPH '90 Proceedings)*, 24(4):63–66, August 1990.

[Alb96] T. D. Albright. An Excellent Lightness. *Science*, 273:1055–1056, August 1996.

[App85] A. A. Appel. An Efficient Program for Many-Body Simulation. *SIAM J. Sci. Stat. Computing*, 6(1):85–103, 1985.

[APRP96] D. Arnauldi, T. Priol, L. Renambot, and X. Pueyo. Visibility Masks for Solving Complex Radiosity Computations on Multiprocessors. *Proceedings of the First Eurographics Workshop on Parallel Graphics and Visualisation)*, page 219.232, September 1996.

[APV91] D. Arnauldi, X. Pueyo, and J. Vilaplana. On the Division of Environments by Virtual Walls for Radiosity Computation. *Proceedings of the Second Eurographics Workshop on Rendering)*, pages 198–205, May 1991.

[ARJ90] J. M. Airey, J. H. Rohlf, and F. P. Brooks, Jr. Towards Image Realism with Interactive Update Rates in Complex Virtual Building Environments. *Computer Graphics (1990 Symposium on Interactive 3D Graphics)*, 24(2):41–50, March 1990.

[Arv86] J. R. Arvo. Backward Ray Tracing. In *ACM SIGGRAPH '86 Course Notes - Developments in Ray Tracing*, volume 12, August 1986.

[Ash95] I. Ashdown. Photometry and Radiometry: A Tour Guide for Computer Graphics Enthusiasts. In *ACM SIGGRAPH '95 Course Notes.* ACM Press, New York, 1995.

[Bak49] H. D. Baker. The Course of Foveal Adaptation measured by the Threshold Intensity Increment. *Journal of the Optical Society of America*, 39:172–179, 1949.

[Bar90] P. G. J. Barton. Evaluation of Subjective Image Quality with the Square-Root Integral Method. *Journal of the Optical Society of America (Optics and Image Science)*, 7(10):2024–2031, 1990.

[Bau75] B. G. Baumgart. A Polyhedron Representation for Computer Vision. *AFIPS Conference Proceedings*, 44:589–596, 1975.

[BB67] C. J. Bartleson and E. J. Breneman. Brightness Perception in Complex Fields. *Journal of the Optical Society of America*, 57(7):953–957, June 1967.

[BG85] V. Bruce and P. Green. *Visual Perception - Physiology, Psychology and Ecology.* Lawrence Erlbaum Associates Ltd, London, 1985.

[BH86] J. Barnes and P. Hut. A Hierarchical $O(N \log N)$ Force Calculation Algorithm. *Nature*, 324:446–449, 1986.

[Bla81] H. R. Blackwell. An Analytic Model for describing the Influence of Lighting Parameters upon Visual Performance. *CIE Technical Committee Volume 1: Technical Foundations*, 19.21, 1981.

[BMP93] K. Bouatouch, D. Menard, and T. Priol. Parallel Radiosity using a Shared Virtual Memory. *Proceedings of the First Bilkent Computer Graphics Convergence on Advanced Techniques in Animation, Rendering and Visualisation*, July 1993.

[BP94] K. Bouatouch and T. Priol. Data Management Scheme for Parallel Radiosity. *Computer-Aided Design*, 26(12):876–882, December 1994.

[BRW89] D. R. Baum, H. E. Rushmeier, and J. M. Winget. Improving Radiosity Solutions Through the Use of Analytically Determined Form-Factors. *Computer Graphics (ACM SIGGRAPH '89 Proceedings)*, 23(3):325–334, July 1989.

[BT75] P. Bui-Tuong. Illumination for Computer Generated Pictures. *Communications of the ACM*, 18(6):311–317, June 1975.

[BW90] D. R. Baum and J. M. Winget. Real Time Radiosity Through Parallel Processing and Hardware Acceleration. *Computer Graphics (1990 Symposium on Interactive 3D Graphics)*, 24(2):67–75, March 1990.

[BW96] P. Bekaert and Y. Willems. Error Control for Radiosity. *Rendering Techniques '96 (Proceedings of the Seventh Eurographics Workshop on Rendering)*, pages 153–164, 1996.

[Cam91] A. T. Campbell III. *Modeling Global Diffuse Illumination for Image Synthesis.* Ph.D. thesis, Technical Report, Dept. of Computer Sciences, Univ. of Texas at Austin, December 1991.

[cAO93] T. K. Çapin, C. Aykanat, and B. Oezguec. Progressive Refinement Radiosity on Ring-Connected Multicomputers. In *Proceedings of the 1993 Parallel Rendering Symposium*, pages 71–76, San Jose, CA, October 1993. ACM.

[Car89] R. C. Carter. Calculate (Don't Guess) the Effect of Symbol Size on Usefulness of Color. *Proceedings of the Human Factors Society 33rd Annual Meeting*, pages 1368–1372, 1989.

[CC83] R. C. Carter and E. C. Carter. CIE $L^*u^*v^*$ Color-Difference Equations for Self-Luminous Displays. *Color Research and Application*, 8:252–253, 1983.

[CCWG88] M. F. Cohen, S. E. Chen, J. R. Wallace, and D. P. Greenberg. A Progressive Refinement Approach to Fast Radiosity Image Generation. *Computer Graphics (ACM SIGGRAPH '88 Proceedings)*, 22(4):75–84, August 1988.

[CDP95] F. Cazals, G. Drettakis, and C. Puech. Filtering, Clustering and Hierarchy Construction: a New Solution for Ray-Tracing Complex Scenes. *Computer Graphics Forum (Eurographics '95 Conference Issue)*, 14(3), 1995.

[CF75] E. C. Carterette and M. P. Friedman, editors. *Handbook of Perception*, volume 5. Academic Press, New York, 1975.

[CG85] M. F. Cohen and D. P. Greenberg. The Hemi-Cube: A Radiosity Solution for Complex Environments. *Computer Graphics (ACM SIGGRAPH '85 Proceedings)*, 19(3):31–40, August 1985.

[CGIB86] M. F. Cohen, D. P. Greenberg, D. S. Immel, and P. J. Brock. An Efficient Radiosity Approach for Realistic Image Synthesis. *IEEE Computer Graphics and Applications*, 6(3):26–35, March 1986.

[Che89] S. E. Chen. A Progressive Radiosity Method and its Implementation in a Distributed Processing Environment. M.Sc. thesis, Program of Computer Graphics, Cornell University, Ithaca, NY, January 1989.

[Che90] S. E. Chen. Incremental Radiosity: An Extension of Progressive Radiosity to an Interactive Image Synthesis System. *Computer Graphics (ACM SIGGRAPH '90 Proceedings)*, 24(4):135–144, August 1990.

[CHS+93] K. Chiu, M. Herf, P. Shirley, S. Swamy, C. Wang, and K. Zimmerman. Spatially Nonuniform Scaling Functions for High Contrast Images. *Proceedings of Graphics Interface '93*, pages 245–253, May 1993.

[CIE78] CIE. CIE Recommendations on Uniform Color Spaces. In *Color-difference Equations and Psychometric Color Terms.* Bureau de la CIE, Paris, 1978.

[CLSS95] P. H. Christensen, D. Lischinski, E. J. Stollnitz, and D. H. Salesin. Clustering for Glossy Global Illumination. Technical Report 95-01-07, Department of Computer Science and Engineering, University of Washington, Seattle, Washington, January 1995.

[Coo86] R. L. Cook. Stochastic Sampling in Computer Graphics. *ACM Transactions on Graphics*, 5(1):51–72, January 1986.

[CP91] A. G. Chalmers and D. J. Paddon. Parallel Processing of Progressive Refinement Radiosity Methods. *Proceedings of the Second Eurographics Workshop on Rendering)*, pages 149–159, May 1991.

[CPC84] R. L. Cook, T. Porter, and L. Carpenter. Distributed Ray Tracing. *Computer Graphics (ACM SIGGRAPH '84 Proceedings)*, 18(3):137–145, July 1984.

[CS94] K. Chiu and P. Shirley. Rendering, Complexity and Perception. *Fifth Eurographics Workshop on Rendering*, pages 21–33, June 1994.

[CSS96] P. H. Christensen, E. J. Stollnitz, and D. H. Salesin. Global Illumination of Glossy Environments Using Wavelets and Importance. *ACM Transactions on Graphics*, 15(1):37–71, January 1996.

[CW93] M. F. Cohen and J. R. Wallace. *Radiosity and Realistic Image Synthesis*. Academic Press Professional, Boston, MA, 1993.

[DD88] R. L. DeValois and K. K. Devalois. *Spatial Vision*. Oxford University Press, New York, 1988.

[DM97] P. E. Debevec and J. Malik. Recovering High Dynamic Range Radiance Maps from Photographs. *Computer Graphics Proceedings, Annual Conference Series, 1997 (ACM SIGGRAPH '97 Proceedings)*, pages 369–378, 1997.

[DS92] S. M. Drucker and P. Schröder. Fast Radiosity Using a Data Parallel Architecture. *Third Eurographics Workshop on Rendering*, pages 247–258, May 1992.

[dV56] L. da Vinci. *Treatise on Painting (Codex Urbinus Latinus 1270)*. Princeton University Press, Princeton, NJ, 1956. A. P. McMahon (Transl.).

[ED59] E. R. G. Eckbert and R. M. Drake. *Heat and Mass Transfer*. McGraw-Hill, New York, NY, 1959.

[Fie96] J. Fier. *Performance Tuning Optimization for Origin2000 and Onyx2*. Number 007-3430-001. Silicon Graphics Inc., 1996. http://techpubs.sgi.com/library.

[Fly72] M. J. Flynn. Some Computer Organisations and their Effectiveness. *IEEE Transactions on Computers*, 21(9):948–960, 1972.

[FP91] M. Feda and W. Purgathofer. Progressive Refinement Radiosity on a Transputer Network. *Proceedings of the Second Eurographics Workshop on Rendering)*, pages 139–148, May 1991.

[FPSG96] J. A. Ferwerda, S. N. Pattanaik, P. Shirley, and D. P. Greenberg. A Model of Visual Adaptation for Realistic Image Synthesis. *Computer Graphics Proceedings, Annual Conference Series, 1996 (ACM SIGGRAPH '96 Proceedings)*, pages 249–258, 1996.

[FPSG97] J. A. Ferwerda, S. N. Pattanaik, P. Shirley, and D. P. Greenberg. A Model of Visual Masking for Computer Graphics. *Computer Graphics Proceedings, Annual Conference Series, 1997 (ACM SIGGRAPH '97 Proceedings)*, pages 143–152, 1997.

[FS93] T. A. Funkhouser and C. H. Séquin. Adaptive Display Algorithm for Interactive Frame Rates During Visualization of Complex Virtual Environments. *Computer Graphics Proceedings, Annual Conference Series, 1993 (ACM SIGGRAPH '93 Proceedings)*, pages 247–254, August 1993.

[FTI86] A. Fujimoto, T. Tanaka, and K Iwata. ARTS: Accelerated Ray Tracing System. *IEEE Computer Graphics and Applications*, 6(4):16–26, 1986.

[FvDFH90] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics, Principles and Practice*. Addison-Wesley, Reading, Massachusetts, 2nd edition, 1990.

[GCS93] S. J. Gortler, M. F. Cohen, and P. Slusallek. Radiosity and Relaxation Methods: Progressive Refinement is Southwell Relaxation. Technical Report CS-TR-408-93, Department of Computer Science, Princeton University, Princeton, NJ, February 1993.

[GH96] S. Gibson and R. J. Hubbold. Efficient Hierarchical Refinement and Clustering for Radiosity in Complex Environments. *Computer Graphics Forum*, 15(5):297–310, December 1996.

[GH97] S. Gibson and R. J. Hubbold. Perceptually-Driven Radiosity. *Computer Graphics Forum*, 16(2):129–140, June 1997.

[Gib95] S. Gibson. Efficient Radiosity for Complex Environments. M.Sc. thesis, University of Manchester, Department of Computer Science, 1995.

[Gla84] A. S. Glassner. Space Subdivision for Fast Ray Tracing. *IEEE Computer Graphics and Applications*, 4(10):15–22, 1984.

[Gla95] A. S. Glassner. *Principles of Digital Image Synthesis*. Morgan Kaufmann, San Francisco, CA, 1995.

[Gou71] H. Gouraud. Continuous Shading of Curved Surfaces. *IEEE Transactions on Computers*, C-20(6):623–629, June 1971.

[Gre88] L. Greengard. The Rapid Evaluation of Potential Fields in Particle Systems. MIT Press, 1988.

[GS87] J. Goldsmith and J. Salmon. Automatic Creation of Object Hierarchies for Ray Tracing. *IEEE Computer Graphics and Applications*, 7(5):14–20, May 1987.

[GSG90] D. W. George, F. X. Sillion, and D. P. Greenberg. Radiosity Redistribution for Dynamic Environments. *IEEE Computer Graphics and Applications*, 10(4):26–34, July 1990.

[GTGB84] C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Battaile. Modelling the Interaction of Light Between Diffuse Surfaces. *Computer Graphics (ACM SIGGRAPH '84 Proceedings)*, 18(3):212–222, July 1984.

[Gus90] J. Gustafson. Fixed Time, Tiered Memory, and Superlinear Speedup. *Proceedings of the Fifth Distributed Memory Computing Conference (DMCC5)*, October 1990.

[Hec24] S. Hecht. The Visual Discrimination of Intensity and the Weber-Fechner Law. *Journal of General Physiology*, 7, 1924.

[Hec34] S. Hecht. Vision II: The Nature of the Photoreceptor Process. In C. Murchison, editor, *A Handbook of General Experimental Psychology*, Worchester, Massachusetts, 1934. Clark University Press.

[Hot54] H. C. Hottel. *Radiant Heat Transmission.* McGraw-Hill, New York, NY, 3rd edition, 1954.

[HS67] H. C. Hottel and A. F. Sarofim. *Radiative Transfer.* McGraw Hill, New York, NY, 1967.

[HS90] P. Hanrahan and D. Salzman. A Rapid Hierarchical Radiosity Algorithm for Unoccluded Environments. In *Proceedings Eurographics Workshop on Photosimulation, Realism and Physics in Computer Graphics*, pages 151–169, 1990.

[HSA91] P. Hanrahan, D. Salzman, and L. Aupperle. A Rapid Hierarchical Radiosity Algorithm. *Computer Graphics (ACM SIGGRAPH '91 Proceedings)*, 25(4):197–206, July 1991.

[Hun67] R. W. G. Hunt. *The Reproduction of Colour.* Fountain Press, London, 2nd edition, 1967.

[Hun87] R. W. G. Hunt. *Measuring Colour.* Ellis Horwood Ltd., Chichester, UK, 1987.

[HW91a] E. A. Haines and J. R. Wallace. Shaft Culling for Efficient Ray-Traced Radiosity. *Proceedings of the Second Eurographics Workshop on Rendering)*, May 1991.

[HW91b] P. S. Heckbert and J. M. Winget. Finite Element Methods for Global Illumination. Technical Report UCB/CSD 91/643, University of California at Berkeley, Berkeley, CA, 1991.

[ICG86] D. S. Immel, M. F. Cohen, and D. P. Greenberg. A Radiosity Method for Non-Diffuse Environments. *Computer Graphics (ACM SIGGRAPH '86 Proceedings)*, 20(4):133–142, August 1986.

[IES84] IES. *IES Lighting Handbook, 1984 Reference Volume.* Illuminating Engineering Society of North America, New York, NY, 1984.

[Kaj86] J. T. Kajiya. The Rendering Equation. *Computer Graphics (ACM SIGGRAPH '86 Proceedings)*, 20(4):143–150, August 1986.

[KJ92] A. J. F. Kok and F. W. Jansen. Adaptive Sampling of Area Light Sources in Ray Tracing Including Diffuse Interreflection. *Computer Graphics Forum (Eurographics '92)*, 11(3):C289–C298, September 1992.

[KK86] T. L. Kay and J. T. Kajiya. Ray-Tracing Complex Scenes. *Computer Graphics (ACM SIGGRAPH '86 Proceedings)*, 20(4):158–164, August 1986.

[Kok93] A. J. F. Kok. Grouping of Patches in Progressive Radiosity. *Fourth Eurographics Workshop on Rendering*, pages 221–232, June 1993.

[LG95] D. P. Luebke and C. Georges. Portals and Mirrors: Simple, Fast Evaluation of Potentially Visible Sets. In *Proceedings of the 1995 ACM Symposium on Interactive 3D Graphics*, pages 105–106. ACM SIGGRAPH, New York, April 1995.

[LL97] J. Laudon and D. Lenoski. The SGI Origin2000: A cc-NUMA Highly Scalable Server. *Proceedings of the 24th Annual International Symposium on Computer Architecture*, pages 241–251, June 1997.

[LP92] Z. Lahjomri and T. Priol. KOAN: A Shared Memory for the iPSC/2 Hypercube. In *CONPAR/VAPP92*, September 1992.

[LRP97] G. W. Larson, H. E. Rushmeier, and C. Piatko. A Visibility Matching Tone Reproduction Operator for High Dynamic Range Scenes. Technical Report LBNL-39882, Lawrence Berkeley National Laboratory, March 1997.

[LSG94] D. Lischinski, B. Smits, and D. P. Greenberg. Bounds and Error Estimates for Radiosity. *Computer Graphics Proceedings, Annual Conference Series, 1994 (ACM SIGGRAPH '94 Proceedings)*, pages 67–74, 1994.

[LTG91] D. Lischinski, F. Tampieri, and D. P. Greenberg. Improving Sampling and Reconstruction Techniques for Radiosity. Technical Report 91-1202, Department of Computer Science, Cornell University, Ithaca, NY, August 1991.

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

[LTG93] D. Lischinski, F. Tampieri, and D. P. Greenberg. Combining Hierarchical Radiosity and Discontinuity Meshing. *Computer Graphics Proceedings, Annual Conference Series, 1993 (ACM SIGGRAPH '93 Proceedings)*, pages 199–208, 1993.

[MBD97] J. S. Montrym, D. R. Baum, and D. L. Dignam. InfiniteReality: A Real-Time Graphics System. *Computer Graphics Proceedings, Annual Conference Series, 1997 (ACM SIGGRAPH '97 Proceedings)*, pages 293–302, 1997.

[Mey86] G. W. Meyer. *Color Calculations For and Perceptual Assessment of Computer Graphic Images.* Ph.D. thesis, Program of Computer Graphics, Cornell University, 1986.

[Mis94] R. G. Mistrick. Rendering The Future. *Lighting Design and Application*, 24(7):41–44, July 1994.

[MK95] K. Myszkowski and T. L. Kunii. An Efficient Cluster-based Hierarchical Progressive Radiosity Algorithm. In R. T. Chin, H. H. S. Ip, A. C. Naiman, and T. C. Pong, editors, *Third International Computer Science Conference: Image Analysis Applications and Computer Graphics (ICSC '95)*, volume 1024 of *Lecture Notes in Computer Graphics*, pages 292–303, Berlin, Germany, 1995. Springer-Verlag.

[Moe96] T. Moeller. Radiosity Techniques for Virtual Reality - Faster Reconstruction and Support for Levels of Details. *Proceedings of the Fourth International Conference in Central Europe on Computer Graphics and Visualization '96*, February 1996.

[MPT97] I. Martin, X. Pueyo, and D. Tost. An Image-Space Refinement Criterion for Linear Hierarchical Radiosity. *Proceedings of Graphics Interface '97*, pages 26–36, May 1997.

[MRC+86] G. W. Meyer, H. E. Rushmeier, M. F. Cohen, Donald P. Greenberg, and Kenneth E. Torrance. An Experimental Evaluation of Computer Graphics Imagery. *ACM Transactions on Graphics*, 5(1):30–50, January 1986.

[MS43] P. Moon and D. E. Spencer. The Specification of Foveal Adaptation. *Journal of the Optical Society of America*, 33(8):444–456, August 1943.

[MS94] S. Müller and F. Schöffel. Fast Radiosity Repropagation for Interactive Virtual Environments Using a Shadow-Form-Factor-List. In *Fifth Eurographics Workshop on Rendering*, pages 325–342, Darmstadt, Germany, June 1994.

[MS95] R. McDonald and K. J. Smith. CIE94 - a new colour-difference formula. *Journal of the Society of Dyers Color*, 111:376–379, December 1995.

[NN85] T. Nishita and E. Nakamae. Continuous Tone Representation of Three-Dimensional Objects Taking Account of Shadows and Interreflection. *Computer Graphics (ACM SIGGRAPH '85 Proceedings)*, 19(3):23–30, July 1985.

[Phi89] P. L. Phillips. Minimum Colour Differences Required to Recognize Small Objects on a Colour CRT. *Journal of the Institution of Electronic and Radio Engineers*, 56(3):123–129, 1989.

[Poy93] C. A. Poynton. "Gamma" and its Disguises: The Nonlinear Mappings of Intensity in Perception, CRTs, Film and Video. *SMPTE Journal (Society of Motion Picture and Television Engineers)*, 102(12):1099–1108, 1993.

[PSV90] C. Puech, F.X. Sillion, and C. Vedel. Improving Interaction with Radiosity-based Lighting Simulation Programs. *Computer Graphics (1990 Symposium on Interactive 3D Graphics)*, 24(2):51–57, March 1990.

[RAPP97] L. Renambot, B. Arnaldi, T. Priol, and X. Pueyo. Towards Efficient Parallel Radiosity for DSM-based Parallel Computers Using Virtual Interfaces. In *Proceedings of the Third Parallel Rendering Symposium, IEEE ACM/SIGGRAPH*, Phoenix, Arizona, October 1997.

[Rat72] F. Ratliff. Contour and Contrast. *Scientific American*, 226(6):91–101, June 1972.

[Red97] M. Reddy. *Perceptually Modulated Level of Detail for Virtual Environments*. Ph.D. thesis, Technical Report, Department of Computer Science, University of Edinburgh, 1997.

[Rei92] M. C. Reichert. A Two-Pass Radiosity Method Driven by Lights and Viewer Position. M.Sc. thesis, Cornell University, Program of Computer Graphics, Ithaca, NY, January 1992.

[RGG90] R. J. Recker, D. W. George, and D. P. Greenberg. Acceleration Techniques for Progressive Refinement Radiosity. *Computer Graphics (1990 Symposium on Interactive 3D Graphics)*, 24(2):59–66, March 1990.

[RH94] J. Rohlf and J. Helman. IRIS Performer: A High Performance Multiprocessing Toolkit for Real-Time 3D Graphics. *Computer Graphics Proceedings, Annual Conference Series, 1994 (ACM SIGGRAPH '94 Proceedings)*, pages 381–395, 1994.

[RPV93] H. E. Rushmeier, C. Patterson, and A. Veerasamy. Geometric Simplification for Indirect Illumination Calculations. *Proceedings of Graphics Interface '93*, pages 227–236, May 1993.

[Rus88] H. E. Rushmeier. *Realistic Image Synthesis for Scenes with Radiatively Participating Media*. Ph.D. thesis, Cornell University, Program of Computer Graphics, Ithaca, NY, 1988.

[Rus94] H. E. Rushmeier. Tone Reproduction. In *ACM SIGGRAPH '94 Course Notes*. ACM Press, New York, 1994.

[RW94] H. E. Rushmeier and G. J. Ward. Energy Preserving Non-Linear Filters. *Computer Graphics Proceedings, Annual Conference Series, 1994 (ACM SIGGRAPH '94 Proceedings)*, pages 131–138, 1994.

[RWP+95] H. E. Rushmeier, G. Ward, C. Piatko, P. Sanders, and B. Rust. Comparing Real and Synthetic Images: Some Ideas About Metrics. *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, 1995.

[SAG94] B. Smits, J. Arvo, and D. P. Greenberg. A Clustering Algorithm for Radiosity in Complex Environments. *Computer Graphics Proceedings, Annual Conference Series, 1994 (ACM SIGGRAPH '94 Proceedings)*, pages 435–442, 1994.

[SAWG91] F. X. Sillion, J. R. Arvo, S. H. Westin, and D. P. Greenberg. A Global Illumination Solution for General Reflectance Distributions. *Computer Graphics (ACM SIGGRAPH '91 Proceedings)*, 25(4):187–196, July 1991.

[Sch94a] C. Schlick. Quantization Techniques for Visualisation of High Dynamic Range Pictures. *Fifth Eurographics Workshop on Rendering*, pages 7–20, June 1994.

[Sch94b] P. Schröder. *Wavelet Algorithms for Illumination Computations*. Ph.D. thesis, Technical Report, Department of Computer Science, Princeton University, Princeton, NJ, November 1994.

[SFB90] M. Stokes, M. D. Fairchild, and R. S. Berns. Precision Requirements for Digital Color Reproduction. *ACM Transactions on Graphics*, 11(4):406–422, October 1990.

[SH92] R. Siegel and J. R. Howell. *Thermal Radiation Heat Transfer*. Hemisphere Publishing Corporation, New York, NY, 3rd edition, 1992.

[SH93] P. Schröder and P. Hanrahan. A Closed Form Expression for the Form Factor Between Two Polygons. Technical Report CS-TR-404-93, Department of Computer Science, Princeton University, Princeton, NJ, January 1993.

[Sha37] S. Shaler. The Relation Between Visual Acuity and Illumination. *Journal of General Physiology*, 21:165–188, 1937.

[Sil95] F. X. Sillion. A Unified Hierarchical Algorithm for Global Illumination with Scattering Volumes and Object Clusters. *IEEE Transactions on Visualization and Computer Graphics*, 1(3), September 1995.

[SP94] F. X. Sillion and C. Puech. *Radiosity and Global Illumination*. Morgan Kaufmann, San Francisco, CA, 1994.

[Spe43] D. E. Spencer. Adaptation in Color Space. *Journal of the Optical Society of America*, 33(1):10–17, January 1943.

[SS60] S. S. Stevens and J. C. Stevens. Brightness Function: Parametric Effects of Adaptation and Contrast. *Program of the 1960 Annual Meeting, Journal of the Optical Society of America*, 50(11), November 1960.

[SS63] J. C. Stevens and S. S. Stevens. Brightness Functions: Effects of Adaptation. *Journal of the Optical Society of America*, 53(3):375–385, March 1963.

[SSJ95] W. Stürzlinger, G. Schaufler, and J. Volkert Johannes. Load Balancing for a Parallel Radiosity Algorithm. In *IEEE/ACM 1995 Parallel Rendering Symposium (PRS '95)*, pages 39–45, Atlanta, Georgia, October 1995.

[SSS97] M. Stamminger, P. Susallek, and H. P. Seidel. Bounded Clustering - Finding Good Bounds on Clustered Light Transport. Technical Report TR-97-1, Universität Erlangen, IMMD 9, 1997.

[SSZG95] G. Spencer, P. Shirley, K. Zimmerman, and D. P. Greenberg. Physically-Based Glare Effects for Digital Images. In *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 325–334. ACM SIGGRAPH, Addison Wesley, August 1995.

[SWH+95] P. Shirley, B. Wade, P. M. Hubbard, D. Zareski, B. Walter, and D. P. Greenberg. Global Illumination via Density Estimation. In P. M. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, pages 219–230, New York, NY, 1995. Springer-Verlag.

[SWPW95] D. Stuttard, A. Worrall, D. Paddon, and C. Willis. A Radiosity System for Real-Time Photo-Realism. In *Computer Graphics Developments in Virtual Environments (CG International '95 Proceedings)*, pages 71–81, Boston, MA, 1995. Academic Press.

[TH93] S. Teller and P. Hanrahan. Global Visibility Algorithms for Illumination Computations. *Computer Graphics Proceedings, Annual Conference Series, 1993 (ACM SIGGRAPH '93 Proceedings)*, pages 239–246, 1993.

[TL91] F. Tampieri and D. Lischinski. The Constant Radiosity Assumption Syndrome. *Proceedings of the Second Eurographics Workshop on Rendering)*, pages 83–92, May 1991.

[TR91] J. Tumblin and H. E. Rushmeier. Tone Reproduction for Realistic Computer Generated Images. Technical Report GI GVU-91-13, Graphics, Visualization & Usability Center, College of Computing, Georgia Institute of Technology, 1991.

[TR93] J. Tumblin and H. E. Rushmeier. Tone Reproduction for Realistic Images. *IEEE Computer Graphics and Applications*, 13(6):42–48, November 1993.

[TS91] S. J. Teller and C. H. Séquin. Visibility Preprocessing for Interactive Walkthroughs. *Computer Graphics (ACM SIGGRAPH '91 Proceedings)*, 25(4):61–69, August 1991.

[VP91] C. Vedel and C. Puech. A Testbed for Adaptive Subdivision in Progressive Radiosity. *Proceedings of the Second Eurographics Workshop on Rendering)*, pages 93–103, May 1991.

[VP92] A. Varshney and J. F. Prins. An Environment-Projection Approach to Radiosity for Mesh-Connected Computers. In *Third Eurographics Workshop on Rendering*, pages 271–281, Bristol, UK, May 1992.

[Š96] L. Šindlar. Parallel Radiosity on a Cluster of Workstations. *Proceedings of WSCG'96*, 1996.

[Wan92] L. Wanger. The Effect of Shadow Quality on the Perception of Spatial Relationships in Computer Generated Imagery. In *Computer Graphics, Special Issue (Proceedings of the 1992 Symposium on Interactive 3D Graphics)*, pages 39–42. ACM Press, March 1992.

[Wan94] B. Wandell. *Foundations of Vision: Behaviour, Neuroscience, and Computation.* Sinauer Press, Sutherland, MA, 1994.

[War94] G. J. Ward. A Contrast-Based Scalefactor for Luminance Display. In P. S. Heckbert, editor, *Graphics Gems IV*, pages 415–421. Academic Press Professional, Boston, MA, 1994.

[WEH89] J. R. Wallace, K. A. Elmquist, and E. A. Haines. A Ray Tracing Algorithm for Progressive Radiosity. *Computer Graphics (ACM SIGGRAPH '89 Proceedings)*, 23(3):315–324, July 1989.

[WH97a] A. J. Willmott and P. S. Heckbert. An empirical comparison of progressive and wavelet radiosity. In J. Dorsey and P. Slusallek, editors, *Rendering Techniques '97 (Proceedings of the Eighth Eurographics Workshop on Rendering)*, pages 175–186, New York, NY, 1997. Springer Wien. ISBN 3-211-83001-4.

[WH97b] A. J. Willmott and P. S. Heckbert. An empirical comparison of radiosity algorithms. Technical Report CMU-CS-97-115, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, April 1997.

[Whi80] T. Whitted. An Improved Illumination Model for Shaded Display. *Communications of the ACM*, 26(6):342–349, 1980.

[WRC88] G. J. Ward, F. M. Rubinstein, and R. D. Clear. A Ray Tracing Solution for Diffuse Interreflection. *Computer Graphics (ACM SIGGRAPH '88 Proceedings)*, 22(4):85–92, August 1988.

[WS82] G. Wyszecki and W. S. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulas.* John Wiley & Sons, New York, 2nd edition, 1982.

[YIY96] Y. Yu, O. H. Ibarra, and T. Yang. Parallel Progressive Radiosity with Adaptive Meshing. Technical Report TRCS96-29, Computer Science Department, University of California, Santa Barbara, September 1996.

[Zie89] O. C. Zienkiewicz. *The Finite Element Method.* McGraw-Hill, London, 4th edition, 1989.