Virtual Light Fields for Global Illumination in Computer Graphics

Jesper Mortensen

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of the

University of London.

Department of Computer Science

University College London

January 25, 2011

To Charlotte, Vita & Nova

Abstract

This thesis presents novel techniques for the generation and real-time rendering of globally illuminated environments with surfaces described by arbitrary materials. Real-time rendering of globally illuminated virtual environments has for a long time been an elusive goal. Many techniques have been developed which can compute still images with full global illumination and this is still an area of active flourishing research. Other techniques have only dealt with certain aspects of global illumination in order to speed up computation and thus rendering. These include radiosity, ray-tracing and hybrid methods. Radiosity due to its view independent nature can easily be rendered in real-time after pre-computing and storing the energy equilibrium. Ray-tracing however is view-dependent and requires substantial computational resources in order to run in real-time.

Attempts at providing full global illumination at interactive rates include caching methods, fast rendering from photon maps, light fields, brute force ray-tracing and GPU accelerated methods. Currently, these methods either only apply to special cases, are incomplete exhibiting poor image quality and/or scale badly such that only modest scenes can be rendered in real-time with current hardware.

The techniques developed in this thesis extend upon earlier research and provide a novel, comprehensive framework for storing global illumination in a data structure - the Virtual Light Field - that is suitable for real-time rendering. The techniques trade off rapid rendering for memory usage and pre-compute time. The main weaknesses of the VLF method are targeted in this thesis. It is the expensive pre-compute stage with best-case $O(N^2)$ performance, where N is the number of faces, which make the light propagation unpractical for all but simple scenes. This is analysed and greatly superior alternatives are presented and evaluated in terms of efficiency and error. Several orders of magnitude improvement in computational efficiency is achieved over the original VLF method.

A novel propagation algorithm running entirely on the Graphics Processing Unit (GPU) is presented. It is incremental in that it can resolve visibility along a set of parallel rays in $O(N)$ time and can produce a virtual light field for a moderately complex scene (tens of thousands of faces), with complex illumination stored in millions of elements, in minutes and for simple scenes in seconds. It is approximate but gracefully converges to a correct solution; a linear increase in resolution results in a linear increase in computation time. Finally a GPU rendering technique is presented which can render from Virtual Light Fields at real-time frame rates in high resolution VR presentation devices such as the CAVE™.

Acknowledgements

Thanks to my supervisor Professor Mel Slater, secondary supervisor Professor Anthony Steed and fellow VLF'ers: Pankaj Khanna and Insu Yu. EPSRC made this thesis possible by funding the research projects; "The Virtual Light Field" (GR/R13685/01), and "Presence in a Virtual Light Field" (EP/C511824/1).

Thanks to the XVR team; Franco Tecchia and Giuseppe Marino for the support and prompt bug fixes while integrating the methods developed in this thesis into eXtremeVR. Also, thanks to Bernhard Spanlang who supplied the avatars and animation subsystem. Thanks to David Swapp manager of the CAVE ^{TM}setup at University College London for helping with getting the tracking data right and other helpful tips and tricks.

Also, thanks to the people in the VECG group at University College London and particularly those in research lab 6.22 - you know who you are. Thanks to Mette Ramsgaard Thomsen for involving me in a number of likely and unlikely projects and Chris Parker for joining in this effort.

The following people were very helpful reading and commenting on the drafts of this thesis; Anthony Steed, Charlotte Thrane, Pip Bull, Joel Jordan, João Oliveira and Bernhard Spanlang.

Thanks to the external examiner Erik Reinhard and internal examiner Jan Kautz for spending a long day talking about light fields.

Finally, I thank my family Charlotte, Vita and Nova for their patience, unwavering support and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computer graphics is a broad field of disciplines ranging from data visualisation, cartoons, complex phenomena such as crowds, particles, traffic, human motion through to studies and measurements of how light reflects off surfaces. However in order to put the work presented in this thesis in context, focus will be restricted to the subfield of rendering. One of the main endeavours of modern computer graphics is to generate images from abstract 3-dimensional representations of virtual scenes stored on a computer. Since the 1960's, when William Fetter coined the term, this domain has revolutionised the world of entertainment, architecture, archaeology, industrial prototyping, design, virtual reality and many other domains. These very different domains where computer graphics is applied for rendering have very different requirements, but currently research can be crudely divided into two main strands, the synthesis of which have been elusive; *photorealism* and *real-time*. This thesis provides a method that shows how both can be achieved.

Photorealism in computer graphics attempts to render images that look *real*. This term is rather ambiguous and incorporates many facets including modelling, texturing, illumination and image reproduction. One of the most important issues is how the illumination of objects is modelled; after all images that we see are formed by light reflected into our eyes. This issue is also the main topic of this thesis. There are many approximations of the interplay of light between the objects of a scene, *local illumination* captures only the first order effects of illumination namely light falling directly onto objects from other emissive objects (light sources). However, illumination is more complex and *global illumination* captures also important second order effects such as soft shadows, indirect illumination, colour bleeding, and caustics with a varying degree of approximation. The equations governing transport of light in 3-dimensional environments are known and they can be solved numerically albeit at a high cost in terms of storage and / or processing. Since the 1980's much research has gone into developing physically plausible techniques that are also computationally tractable. Still images of complex environments with full global illumination can take from minutes to hours to produce.

Real-time computer graphics on the other hand is concerned with generating images with a strict budget on the time it takes to render a frame. The goal is to be able to render scenes quickly enough that the observer does not realise that the images are a sequence of discrete images. In order to achieve this, frame-rates (how many images are rendered per second) need to be 20-30 frames per second. There

are two main types of real-time computer graphics; *dynamic walkthrough* and *static walkthrough*. In dynamic walkthroughs all objects of the scene can undergo animation independently of other objects and the vantage point of the observer can be moved interactively. Typical examples of this occurs in computer games, interactive virtual reality and special effects. In a static walkthrough the objects in the scene are fixed relative to one another but the vantage point can be moved, this is often used in architecture and design applications. A lot of research and investment has gone into this problem and the most successful technique to date is the hardware implementation of the rasterisation pipeline. The technique transforms polygonal objects into the viewing coordinates system, clips them and projects them onto the screen, (typically) solving the hidden surface problem with a z-buffer [FDFH90]. The illumination in this framework is typically restricted to first order effects such as local illumination applying the Phong [Pho75] or similar local illumination model. Effects such as shadows, reflections and indirect illumination are typically realised by applying ad-hoc techniques and texture mapping.

From this it is obvious how different the constraints are between real-time and photorealism. There exists a clear trade-off between photorealism and real-time rendering and to date very few techniques have attempted to provide full global illumination with a real-time constraint.

1.1 The Computer Graphics Pipeline

In order to put the work in this thesis into context the process of generating an image will be described. The computer graphics pipeline can be separated into four distinct steps; *modelling*, *animation*, *rendering* and *image reproduction*.

1.1.1 Modelling

The modelling step of the computer graphics pipeline is concerned with supplying the input to the rendering step. An unambiguous specification must be given of all aspects of the virtual 3-dimensional environment; the objects it is composed of, the material properties of the surfaces present in the scene and a subset of objects acting as emitters. This excludes participating media such as smoke, fog, water and other volumetric media which is outside the scope of this work.

The geometrical specification of objects in a 3-dimensional environment falls into two categories; *solid representations* and *boundary representations*. Solid representations include abstract primitives such as spheres and cones which subdivide space into two halves; inside (the solid) and outside. This subdivision is typically described by an equation. These simple shapes can be combined in hierarchies with *computational solid geometry* (CSG) or instanced recursively or procedurally to create more interesting objects or groups of objects. The boundary representation on the other hand describes only the surfaces that delimit objects. This is typically achieved through parametric descriptions of the surface (Bézier and NURBS surfaces [FDFH90]) or discretised boundary representation such as *polyhedra*. A polyhedron is a collection of planar primitives (faces) that together describe a surface. A face is formed by a set of vertices that are connected by straight lines. A face has a front and a back and the front is defined by the side that when viewed straight on imposes a counter-clockwise ordering of the vertices. Faces are planar. They can be either convex or concave, but a concave face can be represented by a set

of convex faces.

Polyhedra can similarly be open or closed fully delimiting a volume of space. See Figure 1.1 for renderings of some polyhedra. Today, by far the most popular representation is the boundary representation. The games industry, special effects industry, the virtual reality and CAD community almost exclusively rely on NURBS, subdivision surfaces and polyhedra to describe their geometry. One of the main reasons for this is the implementation in hardware of the rasterisation pipeline which requires polyhedral geometry and also the fact that almost all non-polyhedral representations can be subdivided into polyhedra.

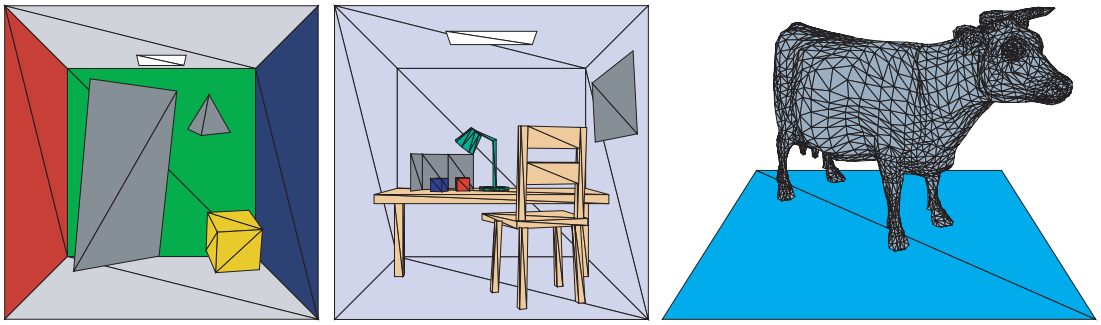


Figure 1.1: Polyhedral representations shown for a variety of scenes.

In order to correctly illuminate an object, its material properties need to be defined. The material properties determine the way light interacts with the surface. There exists a vast amount of research into this area which is one of the most important parts of the global illumination problem. The most general expression of the reflectance of a surface is the *bidirectional surface scattering distribution function* (BSSRDF) [JMLH01] which is an 11 dimensional (or 9 dimensional if points are restricted to surfaces) function relating outgoing radiance at a point x_0 in a given direction ω_0 to the incident irradiance at some other point x_i from a given direction ω_i at a given wavelength λ. This function is overly complex for most rendering purposes [Dut96] and some simplifications are usually introduced to limit the dimensionality of this function. Assuming wavelength independence and disregarding an exact model of subsurface scattering leads us to the simpler *bidirectional reflectance distribution function* (BRDF) which assumes that light entering at a point leaves at the same point. If we also assume that the material is homogeneous for surface elements we arrive at a 4-dimensional function which is more appropriate for our purposes. A number of BRDFs have been proposed that can roughly be classified into *empirical*, *theoretical* and *measured* BRDFs [War92].

A number of the surfaces of the environment need to be emitters, they act as a starting point for illuminating the scene. Self-emitted flux is generated by various physical processes such as heat and chemical reactions and is the source of the initial energy introduced into the environment. In the rasterisation pipeline it is typical to only allow point emitters due to the fact that they are very simple to sample. On the other hand they are an abstract concept not found in nature, and they preclude phenomena such as soft shadows. Most global illumination algorithms assume area light sources and assigning flux to a number of the existing surfaces is a general and elegant way of defining *initial* emitters; at first they are

the only emitters, after they have distributed their energy into the environment surfaces that received flux then in turn become *secondary* emitters and so on. Similarly to the BRDF emitters can be defined with a high dimensionality if they are anisotropic and non-homogeneous over the surface.

The last necessary part of the input is the camera model. Its main objective is to define the vantage point of the viewer with respect to the environment. The most typical camera models are the pinhole camera model and the thin lens camera model. The thin lens model allows for such effects as depth of field whereas the entire scene is in perfect focus in the pinhole camera model [CPC84]. In this work the pinhole camera model will be used exclusively for simplicity.

1.1.2 Animation

The animation step alters the relationship between the components defined in the modelling step. Various types of animation can be envisaged including camera movement, rigid body animation, deformations and unstructured motion. Clearly, a global illumination solution for a specific configuration of an environment becomes partly invalid after animation that alters the relative position of components of that environment and the solution must be recomputed to adapt to the new configuration of objects.

In this thesis interactive walkthrough of globally lit environments is the main objective; thus it is assumed that the environment is static and only the camera pose undergoes animation. However, the implications of supporting dynamic scenes will be discussed in the appropriate sections.

1.1.3 Rendering

The rendering step visualises the virtual model of the scene. A number of different rendering techniques exist from wireframe renderings through non-photorealistic rendering to full global illumination solutions which explore all possible light paths. A classification can be made depending on whether the algorithm is image-based or object-based. Image-based algorithms (also known as pixel driven) compute the illumination relative to a specific camera position whereas object space (also known as finite element) methods compute the illumination in object space and then in a separate pass render an image given a camera vantage point. The advantage (and disadvantage at the same time) of pure pixel driven algorithms is that they recompute the illumination for every viewpoint which allows for dynamic environments in a straightforward manner. However, for larger scenes with complex lighting they are typically too computationally intensive to achieve real-time frame rates. On the other hand finite element methods can typically render images from a given vantage point quickly but moving objects in the scene would require that the illumination solution is (partially) recomputed. Also they often require large data structures in order to store the solution. However with the explosive growth of memory capacities of typical workstations, currently several gigabytes, many techniques hitherto intractable are becoming possible. The work presented in this thesis works in object space and uses a large data structure to store a pre-computed global illumination solution for fast rendering.

1.1.4 Image Reproduction

The output of the rendering step is a collection of radiance values captured on the image plane of the camera model. This raises some fundamental questions about what we are attempting to achieve. Ac-

cording to Hall and Greenberg [HG83]:

> *Our goal in realistic image synthesis is to generate an image that evokes from the visual system a response indistinguishable from that evoked by the actual environment.*

As Peter Shirley correctly points out [Shi90a] this would require an interface directly connecting the computer to the visual cortex of the viewer, a subject of current research [Dob00]. Also the camera model used does not model the eye and the output devices do not have the fidelity and dynamic range of the human visual response system, although research into HDR displays is underway [War08]. The way this problem is circumvented is usually by redefining the problem to the more reachable one; *to generate an image of the virtual scene which is indistinguishable from a photograph of that scene*. In this work this will be modified slightly to a more appropriate one; *to create a matrix of radiation [i.e. colour] values, that is displayed on an output device in an appropriate way* [Dut96].

1.2 Scope and Objectives

In this thesis methods for allowing real-time walkthroughs of fully globally illuminated static scenes are investigated. The methods should be applicable to Virtual Reality (VR) systems driving immersive projection systems such as the CAVE^{TM} [CNSD^{+}92].

The techniques require a pre-processing stage where light transport is computed and stored in a large data structure. They therefore belong to the class of finite element or object space solutions. In this work closed non self-intersecting polyhedra formed by planar convex faces will be used exclusively. The BRDFs used will be restricted to some commonly used theoretical BRDFs namely the lambertian and specular BRDFs or a mixture of the two and the glossy modified Phong BRDF. Transparent surfaces will not be supported. This, however, is not a limitation of the technique and could easily be accommodated. Only area light sources are supported. Scenes with small or point light source would require an initial gathering pass [SKSMT00]. Only isotropic homogeneous emitters are used for simplicity. Again this is not a limitation of the techniques described. Radiance is shot from the light sources outwards and a radiance field is assembled covering the entire scene. In order to produce global illumination solutions efficiently, a number of techniques for light transport from the light sources are investigated. This includes continuous clipping methods, point sampling methods, rasterisation with the z-buffer and incremental methods as well as hardware based techniques exploiting existing programmable graphics hardware.

A number of techniques for real-time rendering from this field of radiance are evaluated and discussed in terms of efficiency and error.

1.3 Contributions

The overall contribution of this thesis can be summarised as a method for storing and the production of a solution to the global illumination problem that allows real-time walkthrough on workstations [SMKY04] and with extensions to make it practical in terms of pre-compute time for moderately complex scenes. The work will focus on *fast propagation* and *fast walkthrough*. More specifically the contributions are:

- Realising an implementation of an abstract method known as the Virtual Light Field [SMKY04] as part of a project[1].
- Providing a literature survey of other work done in the field of real-time global illumination.
- Addition of a low-level radiance transport operator based on point sampling. An adaptive point-sampling algorithm is presented improving propagation speed threefold without loss of precision.
- Addition of a high-level incremental radiance transport operator that yields linear scaling of propagation time in the number of scene polygons [MKS07]. An order of magnitude improvement in propagation speed is presented for a number of scenes.
- Mapping the newly developed point-sampling based propagation algorithm to programmable graphics hardware [MKS07] leveraging the power of recent GPUs gaining two additional orders of magnitude improvement in speed. The algorithm runs entirely on the GPU.
- Mapping the rendering algorithm to programmable graphics hardware [MKYS07] leveraging the power of recent GPUs to attain real-time rendering of large frame sizes at high frame rates. The algorithm runs entirely on the GPU.
- Integrating the VLF-GPU method into a fully featured immersive virtual reality system [MYK+08] supporting dynamic objects and presenting a case study using the system for presence research [YMKS10].

1.4 Organisation of this Thesis

The remaining chapters in this thesis are organised as follows:

- Chapter 2 begins with a short overview of the history of light transport in physics. Then the basic concepts of light transport, the terms and units that will be used and equations that mathematically describe light transport are introduced.
- Chapter 3 presents a taxonomy for global illumination algorithms. Other work in the field with particular focus on global illumination algorithms that allow real-time walkthroughs is surveyed.
- Chapter 4 presents the Virtual Light Field method (VLF) [SMKY04]. It comprises of a data structure particularly suited to the task of fast rendering of scenes with complex illumination and an algorithm to fill in this data structure with physically plausible global illumination using forward propagation from light sources in a pre-compute stage.
- Chapter 5 presents new transport techniques operating on the data structure presented in Chapter 4 which attempt to alleviate the main weakness of the algorithm; pre-computation time. Specifically a technique running entirely on the GPU [MKS07, MKYS07] is presented.

[1] EPSRC research project ”The Virtual Light Field” (GR/R13685/01).

- Chapter 6 presents results for the VLF-GPU method. The transport techniques developed in Chapter 5 are evaluated in terms of efficiency and error.

- Chapter 7 outlines an integration of the VLF-GPU method into the eXtremeVR immersive VR system [MYK+08]. Then a case study using the system for presence research is described [YMKS10]. Lastly, final conclusions are drawn and ideas for future directions of research are presented.

Chapter 2

The Global Illumination Problem

In this chapter, the global illumination problem is formulated in a physically correct way. First a short recap of the history of the theory of light in physics is presented. Then the problem that the thesis deals with is delimited and suitable limitations are clearly outlined and discussed. The mathematical notation adopted is largely borrowed from Philip Dutré's Global Illumination Compendium [Dut96, Dut03] for consistency. See Appendix A for tables of symbols used in the notation.

Finally, in the background sections earlier work in this field is surveyed paying special attention to global illumination algorithms.

2.1 A Short History of Light

Light has been studied from classical times. One of the first recorded studies of this kind is that of Empedocles in the fifth century BC, who believed that Aphrodite created the human eye from the four elements and lit a fire in it enabling sight [OR02]. This view was questioned by many including Euclid in 300 BC and Lucretius in 55 BC since it would enable sight in total darkness.

It was not until 1000 AD that a more advanced model of light was proposed and the beam model was abandoned. This was due to al-Haytham, who argued that sight is enabled only by light sourced from outside the eye, his most persuasive example being that of the camera-obscura (or pinhole camera). He also proposed that light is made up of minute particles travelling in straight lines with a high but finite velocity. He explained that the phenomenon of refraction[1] is caused by light having different velocities when travelling through different substances. His work was not widely available in Europe until the end of the 16th century.

It was not until the beginning of the 17th century that Kepler made significant advances in the field. His work on optics correctly explained the workings of the eye, based on deriving a mathematical theory of the camera-obscura. He correctly showed that upside-down images are formed on the retina and explained shortsight and longsight. One of his most important results though was that the intensity of light observed from a source varies inversely with the square of the distance of the observer from the source. In *Dioptrice* of 1611 he described the telephoto lens and described total internal reflection.

[1] The bending of the normal to the wavefront of a propagating wave upon passing from one medium to another where the propagation velocity is different. The most common example is the refraction of light on passing from air to a liquid, which causes submerged objects to appear displaced from their actual positions.

Kepler's argument that images are formed upside-down on retina in the eye was not widely accepted since the counter-argument that we do not see the world upside down seemed more plausible; it was finally proved experimentally by Descartes who scraped off the backside of an ox's eye yielding an upside-down image. The sine law of refraction of light found by Thomas Harriot in 1601 and Willebrord Snell in 1621 was not published until Descartes did so in 1637 in his work on optics *La Dioptrice*. Diffraction[2] of light was named and discovered by Francesco Grimaldi. His work was improved upon by Isaac Newton who showed that white light is composed of light of different colours by splitting white light in a prism and recombining it in another. Newton believed that light was composed of particles and gave experimental evidence supporting this but in the 1670s this was a cause of much controversy.

Robert Hooke, Christiaan Huygens and later Leonhard Euler and Augustin Jean Fresnel both supported a wave theory of light which correctly explained phenomena such as reflection, refraction, diffraction of which the latter is the most difficult to explain with the corpuscular theory. The opposing corpuscular and wave theories of light hinged on one important property of light; namely whether light travelled faster in a denser medium proposing the former theory or slower proposing the latter. A definite proof that the speed of light is finite was given by the astronomer Ole Römer in 1676 who calculated from observations of Jupiter the speed of light to be 225,000 km per second, remarkably close to the correct value of 299,792 km per second. But at the time experiments involving the speed of light were impossible to carry out. In 1801 Thomas Young published his results on experiments on interference of light and also explained Newton's results in terms of wave theory. He related the colours of light to its wavelength and showed how the amount of refraction depends on this value. Using Newton's experimental data on diffraction he calculated the wavelengths of different colours. He explained that the eye contains receptors sensitive to red, green or blue light and thus colour vision. Malus discovered polarisation of light and published it in 1809, this effect was later theoretically explained by Fresnel's transverse wave theory of light. It was not until 1850 that Foucault experimentally showed that the speed of light slowed in water, thus confirming the wave theory. Around 1862 James Clerk Maxwell unified the theory and described light as an electromagnetic phenomenon. In his seminal paper *Electricity and Magnetism* [Max54] (1873) he gave the four partial differential equations which fully describe the classical electromagnetic theory.

But this is not the end of the story. Between 1880 and 1926 two parallel lines of research merged and gave some startling new insight into the behaviour of light; namely quantum mechanics and the theory of relativity. Quantum mechanics started with experiments with blackbody[3] radiation carried out by Josef Stefan in 1879 which showed that the total radiation energy per unit time emitted from a blackbody is proportional to the fourth power of the absolute temperature of the body. In 1896 Wilhelm Wien discovered that the wavelength at which the maximum energy is radiated becomes shorter as the

[2]Diffraction is a phenomenon by which wavefronts of propagating waves bend in the neighbourhood of obstacles.

[3]The blackbody is a hypothetical entity which absorbs all energy, reflects none and emits energy with perfect efficiency. A blackbody is assumed to satisfy the following ideal conditions; (i) A blackbody absorbs all incident radiation regardless of wavelength and direction. (ii) For a prescribed temperature and wavelength, no surface can emit more energy than a blackbody. (iii) Although the radiation emitted by a blackbody is a function of wavelength, it is independent of direction. A blackbody is defined as a diffuse emitter.

temperature of the blackbody is increased. Wien's distribution law of radiation prompted the Raleigh-Jeans law of radiation distribution, which states that all possible frequency modes can radiate with equal probability. Whereas this law was valid for low frequencies it broke down for high frequencies because an infinitesimally small wavelength would imply infinite radiation, which is of course impossible. In 1900 Max Planck resolved this by realising that radiation energy could only exist in discrete quanta proportional to the frequency. This would imply that the higher frequency modes were less likely and thus avoid the problems with high frequencies inherent in the Raleigh-Jeans law. Parallel to this Heinrich Hertz discovered the photoelectric[4] effect in 1887 and in 1900 Philipp Lenard showed that the effect was caused by electrons being ejected from the metal surface when struck by light. This again gave impetus to a corpuscular theory of light as Albert Einstein showed that this effect could only be described if light was in fact composed of discrete particles or energy quanta; thus the photon was born. In a publication in 1905 Einstein used Planck's work and described electromagnetic radiation of light in terms of the quantum hypothesis. That same year he also presented the theory of relativity that said that physical laws must have the same form in any frame of reference, and as a consequence that the speed of light remains constant in all frames of reference.

The difficulty in explaining for example the light interference phenomenon with a corpuscular theory of light remained. But in 1924 de Broglie put forward the particle-wave duality theory [5], which together with Heisenberg's uncertainty principle [6] of particles of light would come together in what became known as the Copenhagen interpretation of quantum theory. This was due to Niels Bohr and his colleagues. The Copenhagen interpretation says that under observation light waves collapse into particles. One of the best examples of which is Thomas Young's double slit interference experiment re-enacted by Bohr by firing single photons towards the slits; the light depart as particles, behave as waves hence the interference pattern and finally arrives as particles. This clearly has some fascinating philosophical implications, which are outside the scope of this thesis.

2.2 Geometry

In this section the geometry used as input to the methods described in this thesis is described. As discussed in Section 1.1.1 techniques developed in this thesis solve the global illumination problem for three-dimensional environments. These three-dimensional environments O are sets of simple closed polyhedra P that consist of planar convex polygons[7]. Each such set of polygons P describe a simple closed surface (or *solid*) of finite volume that partitions space into three distinct sets; points *outside* the surface P^-, points *inside* the surface P^+ and finally points *on* the surface P^0. The combined set of surface points for the environment is denoted A. We can define an environment O as a set of subsets of Euclidian

[4] He observed that when that ultraviolet light was shone onto metallic electrodes the voltage required for sparking to take place was lowered.

[5] The particle-wave duality theory states that matter has the properties of both particles and waves.

[6] The uncertainty principle states that there is a limit to the precision with which the position and the momentum a particle of light can be known.

[7] The simplest way to enforce this would be to only allow triangles, but in this thesis the framework will also support quadrilaterals under the assumption that they are planar and convex.

3-space as follows:

$$O = \{P_0^+, P_1^+, \ldots, P_n^+\}, \quad \text{where} \quad P_i^+ \subset \mathbb{E}^3$$

It is further assumed that no two solids intersect or more formally:

$$(P_i^+ \cup P_i^0) \cap (P_j^+ \cup P_j^0) = \emptyset, \quad \text{where} \quad P_i, P_j \in O \quad \text{and} \quad i \neq j$$

In practice it is often more appropriate to describe the input as a set of three-dimensional surface points especially when dealing with environments devoid of participating media. The surface points are points that belong to one of the surfaces in the environment. The set of all surface points in an environment is denoted by A and each point x is (parametrically) described by a three dimensional position and a surface normal at that point n_x (see Figure 2.1).

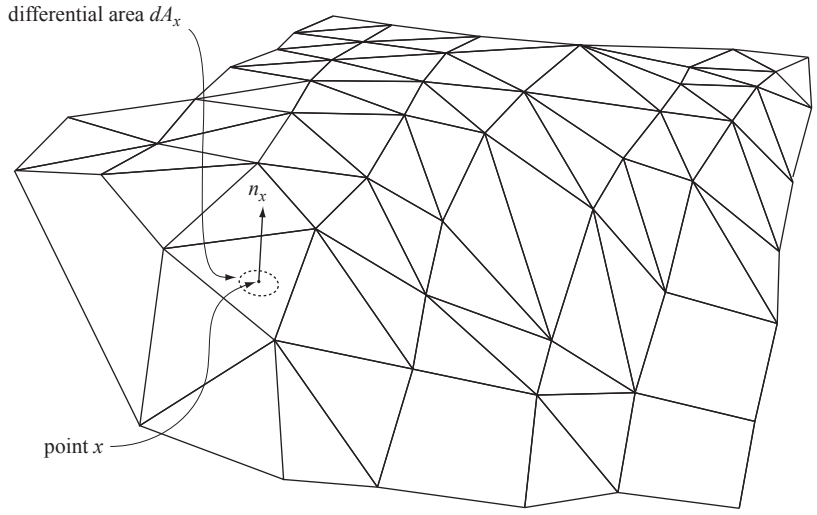


Figure 2.1: A set of polygons and a point x with its normal n_x and a differential area dA_x around x.

Since we are restricting polygons to be planar, all surface points belonging to a polygon will share one single normal (described by the plane equation). This limitation can be lifted and the changes necessary to do so will be described when the techniques are presented. A differential surface area around a point x is denoted by dA_x.

2.2.1 Free Space Simplification

The techniques described in this thesis deal only with energy transport between surface points $(x, y) \in A$. This assumption is also known as the *free space* simplification and it implies that interaction only occurs at the surfaces of the objects in the scene, which greatly reduces the complexity of the task at hand. This assumes the absence of any participating media, which would absorb or scatter energy. This means that absorption and scattering only occur at surface points $x \in A$ and self emission only occurs at surfaces.

This is a reasonable assumption for many indoor and architectural scenes where distances are limited such that atmospheric effects are negligible, however, effects such as smoke or dust particles in the air are not accounted for. Many large scale outdoor scenes require participating media in order to take into account mist, fog, atmospheric attenuation and other effects. In this thesis the focus is mainly on

architectural and indoor scenes and participating media is outside the scope of this thesis and will not be considered further.

Refer to [Bli82, KH84, Max86, NMN87] for previous work involving global illumination in the presence of participating media.

2.2.2 Solid Angles & Directions

In global illumination techniques it is frequently appropriate to work with directions on a sphere surrounding a surface point. This stems from the fact that light transport for a differential surface area is defined as an integral over the sphere around the surface area[8]. The sphere around a surface point, say x, can be partitioned into two sets; the set of directions above the point (H_x^+), or, the *positive hemisphere*, and the set of directions below the point (H_x^-); the *negative hemisphere*.

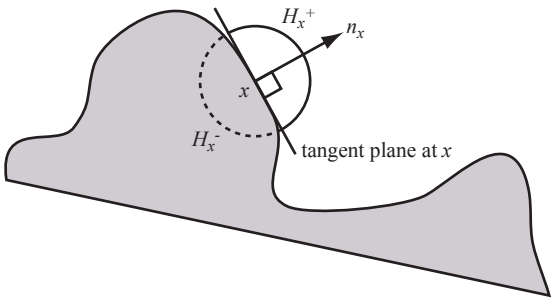


Figure 2.2: The sphere around point x. H_x^+ denotes the upper hemisphere while H_x^- is the lower hemisphere with respect to the tangent plane at x.

Above and below is defined with respect to the tangent plane at the surface point. Figure 2.2 illustrates this concept. More formally these sets can be defined as:

$$\Omega_x \equiv H_x^+ \equiv \{\forall \Theta_x \in \mathbb{S}^2 : \Theta_x \cdot n_x \geq 0\} \tag{2.1}$$

$$H_x^- \equiv \{\forall \Theta_x \in \mathbb{S}^2 : \Theta_x \cdot n_x \leq 0\}, \qquad \text{where } \mathbb{S}^2 \text{ is the unit sphere of directions.} \tag{2.2}$$

The reason for this partitioning is that a large class of (opaque) surface materials only interacts with light in the positive hemisphere of directions (Equation 2.1), in these cases it is wasteful, or even erroneous, to include the negative hemisphere (Equation 2.2) in the quadrature. Since only opaque materials will be considered in this thesis (see Section 1.2) an alternative notation for the positive hemisphere has been introduced for clarity; Ω_x (see also Appendix A for a table listing the notation used).

When an integral over the hemisphere (or sphere) is involved ($\int_{\Omega_x} f(x,\omega)d\omega$) it is useful to be able to treat the sphere (or hemisphere) as a *linear dimension* over which to integrate, which in turn requires a definition of a differential unit area on the hemisphere to serve as the unit for the variable of integration $d\omega$.

The *solid angle* is used for this purpose. Mathematically, it is a unitless quantity but for practical reasons the standard unit *steradians* (sr) is commonly assigned, the sphere comprises 4π steradians. In Figure 2.3 the differential solid angle $d\omega$ around direction Θ is shown. The direction Θ can alternatively

[8] This will be formally introduced in Section 2.4

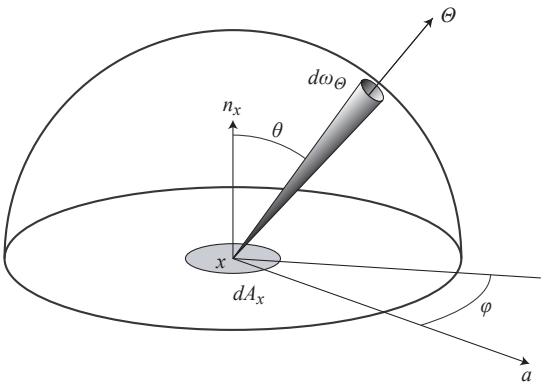


Figure 2.3: Differential solid angle $d\omega$ around direction Θ from surface point x.

be described in spherical coordinates by a pair of angles (θ, φ) defined relative to the orthonormal basis $(a, n_x, a \times n_x)$ originating at x, where $\times$ is a vector cross product (subsequently the nature of such operators will only be stated if it cannot be derived from the context). The solid angle can then be expressed as $d\omega_\Theta = \sin\theta d\theta d\varphi$.

Another quantity that will be used frequently is differential solid angle $d\omega$ originating at a surface point x expressed in terms of a differential surface area dA_y around the surface point y along the direction $\overline{xy}$. This relationship is expressed in Equation 2.3 and illustrated in Figure 2.4.

$$d\omega_\Theta = d\omega_{\overline{xy}} = \frac{\cos\theta_y dA_y}{r_{xy}^2} \tag{2.3}$$

Equation 2.3 is used heavily in hemisphere integration tasks where the area subtended by a surface upon the hemisphere over a point needs to be computed.

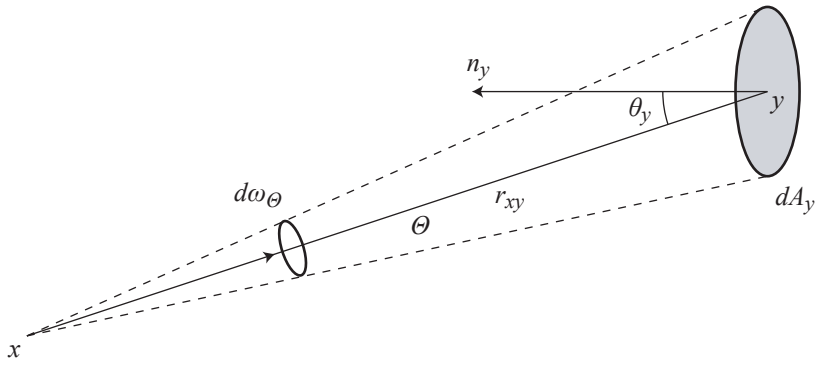


Figure 2.4: Geometric relationship between differential solid angle $d\omega_\Theta$ and differential surface area dA_y.

It can also be useful to perform hemisphere integration tasks in spherical coordinates instead of solid angles especially when working with hemisphere parameterisations (see equation 2.4).

$$\int_\Omega f(\Theta) d\omega_\Theta = \int_0^{2\pi} \int_0^{\pi/2} f(\varphi, \theta) sin\theta d\theta d\varphi \tag{2.4}$$

2.2.3 Visibility & Ray Casting

Visibility determination is an important part of any global illumination algorithm since it determines which surfaces can exchange energy with each other.

Visibility

Visibility for a pair of points is more formally defined as a function $V(x,y)$ that returns 1 if x and y is mutually visible, that is, no other surface obscures y as seen from x and 0 if there is an intervening[9] surface between x and y, this is expressed in Equation 2.5.

$$V(x,y) = \begin{cases} 1, & \text{if } x \text{ and } y \text{ are mutually visible;} \\ 0, & \text{otherwise.} \end{cases} \tag{2.5}$$

In Figure 2.5 visibility queries are illustrated; point x on object P_5 and point y on object P_0 are mutually visible, while point x' on object P_4 and point y' on object P_2 are not since surface points on object P_3 obscures y' from x'.

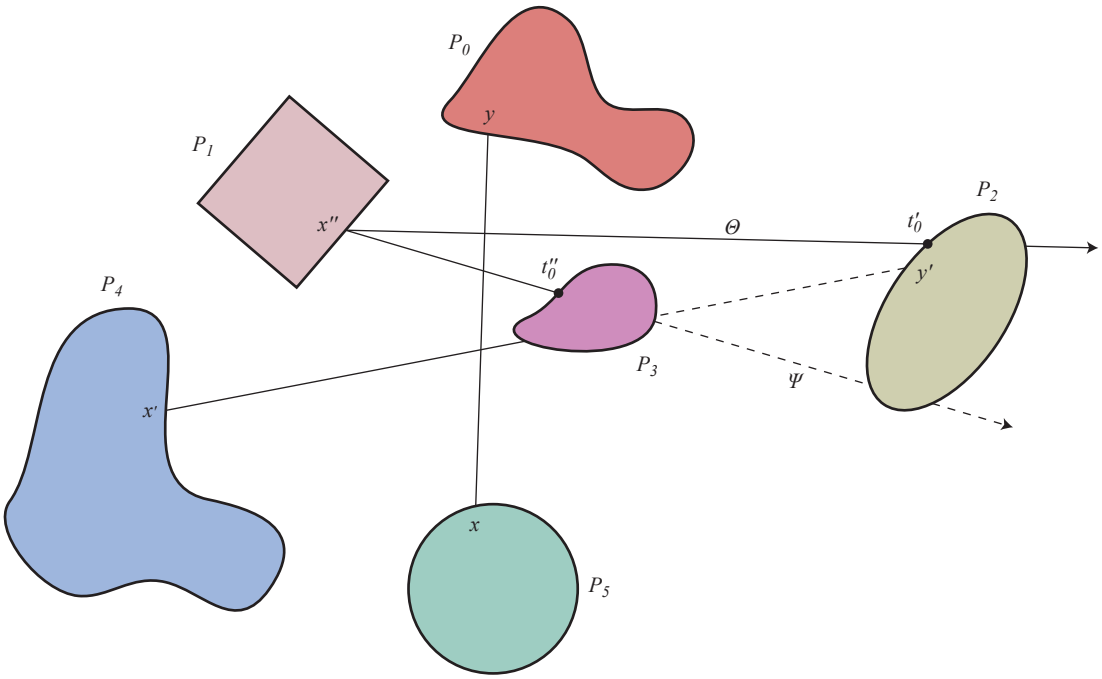


Figure 2.5: Visibility and ray casting in $\mathbb{E}^2$ space.

Ray Casting

It is illuminating to rephrase the visibility concept: two points x and y are mutually visible if the line segment starting at x in direction $\overline{xy}$ and terminating at y does not intersect any object in the scene. This naturally leads to the ray casting problem, which is a more general visibility problem for ray segments in n-dimensional Euclidian space $\mathbb{E}^n$ [Hav01]. Given a point x and a direction Θ the ray casting function determines the closest surface point visible from x along Θ. This is more formally expressed in

[9] Even if the intervening object is (partly) transparent the visibility is still considered blocked as the intervening object may interfere with the transport between the points due to surface scattering, refraction or another such effect.

Equation 2.6.

$$r(x,\Theta) = x + t_{\text{inf}} \cdot \Theta \tag{2.6}$$

$$t_{\text{inf}} = \inf\{t > 0 : (x + t \cdot \Theta) \in A\} \tag{2.7}$$

If no point in A is visible to, say, x along Θ the ray casting function is undefined, $\{t > 0 : (x + t \cdot \Theta) \in A\}$ in Equation 2.7 returns the empty set and consequently *infimum*[10] is undefined and $t_{\text{inf}} = \infty$ [Arv95]. This may happen for open scenes. It is, however, simple to handle this special case in practice by assuming some overall background emission or vacuum. In Figure 2.5 ray casting is illustrated; the nearest visible surface point from x'' on object P_1 along Θ is $(x'' + t_0' \cdot \Theta)$ on object P_2, and the nearest visible surface point from x'' on object P_1 along Ψ is $(x'' + t_0'' \cdot \Psi)$ on object P_3. One obvious property of the ray casting function is its reciprocal nature. If y is visible from x along direction Θ then x is visible from y along the opposite direction $-\Theta$.

$$\begin{aligned} &\text{let} \quad y = r(x,\Theta) \quad \text{and} \quad x, y \in A \\ &\Rightarrow x = r(y, -\Theta) \\ &\Rightarrow x = r(r(x,\Theta), -\Theta) \end{aligned}$$

For a good overview of ray casting algorithms with visibility data structures and their efficiency see [Hav01]. See [Wal04] for an overview of significant optimisation strategies exploiting hardware.

Exchange Lists

Given a scene O which forms a subset of three dimensional Euclidian space $\mathbb{E}^3$ consisting of simple closed polyhedra P the surfaces of which are described by A, and a ray originating at x, where x lies outside of any object P pointing in direction Θ with parametric definition $y = x + t \cdot \Theta$ (see Figure 2.6). If $\{t_0, t_1, \ldots, t_n\}$ is a list of positive parametric t values representing intersection points with the scene, and assume the absence of any grazing intersections[11], this list must be either empty or its cardinality even; this follows from the fact that if the ray enters the inside of an object P over a point $x \in A$ it must also exit the object P over a point $x' \in A$, so every *outside-inside* transition is followed by an *inside-outside* transition.

This partitions the ray into a set of ray segments; $R = \{(0,t_0), (t_1,t_2), \ldots, (t_n,\infty)\}$ that describe the possible energy exchange pairs. Each such ray segment describe an interval in free space over which radiance can be assumed to be constant as no transitions occur. Every such element excepting the first and last is a segment over which bi-directional radiance exchange will take place, ie. mutually visible pairs of surface points along Θ and for every such pair (t_i, t_{i+1}) (except elements 0 and ∞ in the first and last pairs) it follows that $n_{(x+t_i \cdot \Theta)} \cdot \Theta > 0$ and $n_{(x+t_{i+1} \cdot \Theta)} \cdot \Theta < 0$; meaning that the ray at t_i *exits* on the upper hemisphere above the intersection point and the ray at t_{i+1} *enters* on the upper hemisphere above the intersection.

[10] Infimum or inf is the greatest lower bound.

[11] A grazing intersection is an intersection through an edge or vertex, these special cases are easily handled by appropriate sampling in a discrete setting. Or, alternatively, as treating it as a *pair* of intersections.

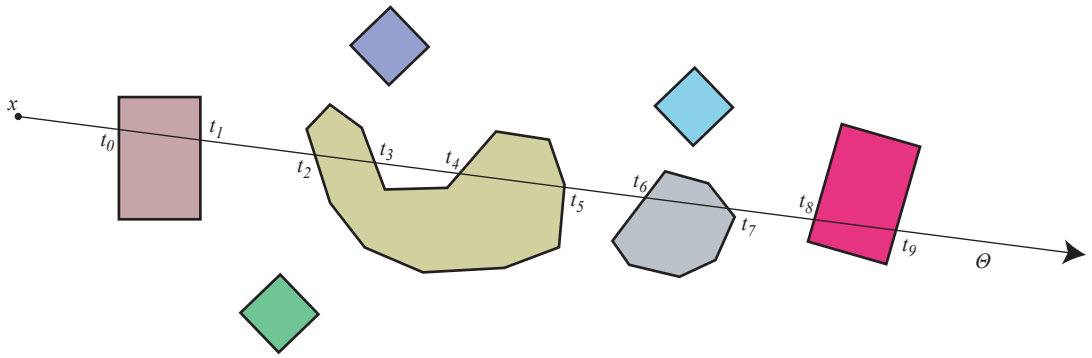


Figure 2.6: Exchange pairs in $\mathbb{E}^2$ space; pairs $\{(t_1,t_2),(t_3,t_4),\ldots,(t_7,t_8)\}$ are mutually visible along the ray originating at x in direction Θ and can exchange energy.

The first $(0,t_0)$ and last (t_n,∞) segments are special cases where only t_0 and t_n are points that belong to the scene thus the radiance travelling over these segments must be handled differently than other exchange pairs. The solution is to either assume that scenes are properly closed such that no such pairs can exist or let radiance enter and leave the scene allowing for open scenes. The techniques developed in this thesis supports both open and closed scenes, and in Chapter 4 and Chapter 5 this is elaborated upon.

2.3 Radiometry

Energy transport will occur in systems of non-uniform energy. This is captured in the laws of thermodynamics:

- 1^{st} law of thermodynamics: The total energy in a (closed) system is conserved.
- 2^{nd} law of thermodynamics: Heat flows spontaneously from a hot body to a cool one.

Energy can be transported spatially via three means; *conduction*, *convection* and *radiation*. In the *conduction* process energy is exchanged by direct inter-molecular collisions in a medium. When there is an energy gradient within a medium, equilibrium is reached by transfer of energy from areas of high energy to areas of low energy. This phenomenon is described by Fourier's law of heat conduction (due to Joseph Fourier 1768-1830). This is a microscopic effect of the 1^{st} law of thermodynamics.

In the *convection* process energy is transferred by movement of the medium itself due to volumetric expansion and pressure. As the volume of a medium increases its density decreases making it buoyant causing displacement of the medium itself. This is essentially a macroscopic effect of the 1^{st} law of thermodynamics.

The *radiation* process deals with inter-medium energy transfer carried by photons of light in the infrared and visible part of the electromagnetic spectrum. Radiative energy transfer, in contrast to the two former modes of energy transfer, does not need an intervening medium.

So which of these conceptual models will best suit our purposes? Global illumination in computer graphics attempts to visualise environments and model light in a way that accounts for the interplay of light in complex environments. The environments in question are assumed to be large-scale in time

and space relative to the frequencies and wavelengths of light respectively. So our task is to acquire a model that captures the phenomenological characteristics of light as it appears to the eye. Physical optics as described by Maxwell's equations is most appropriate at scales the order of the wavelength of light where wave optics effects predominate, whereas geometrical optics ignores wave dependent qualities of light and thus is not sufficient. Radiative transfer deals with light at a macroscopic scale, yet it incorporates into the simulation theories describing what happens at microscopic scales, e.g. local scattering, absorption and use of physically based material properties such as BRDF's [Arv93]. Hence, radiative transfer is appropriate for our purposes.

2.3.1 Terms and Units

Two sets of terms and units are commonly used to describe radiative exchange; they are lent from the sciences of *radiometry* and *photometry*. *Radiometry* deals with measurement of optical radiation (wavelengths 0.01-1000 micrometres) whereas *photometry* deals with measurement of light which is electromagnetic radiation visible by the human eye (wavelengths 360-830 nanometres). The main difference apart from the spectral range is that radiometry employ objective physically based measurements whereas photometry employs measurements weighed by the spectral response of the human eye. In Table 2.1 and Table 2.2 the radiometric and photometric terms and units are shown.

Physical quantity	Photometric name	Photometric unit
Energy	Luminous energy	talbot
Flux	Luminous power	lumen $(lm = talbot \cdot s^{-1})$
Angular flux density	Luminance	nit $(lm \cdot m^{-2} \cdot sr^{-1})$
Flux density	Illuminance	lux $(lm \cdot m^{-2})$
Flux density	Luminosity	lux $(lm \cdot m^{-2})$
	Luminous intensity	candela $(lm \cdot sr^{-1})$

Table 2.1: Photometric terms and units. Where *s* denotes unit time and *sr* is unit steradian (solid angle)

Physical quantity	Radiometric name	Radiometric unit
Energy	Radiant energy	joule $(J = kg \cdot m^2 \cdot s^{-2})$
Flux	Radiant power	watt $(W = J \cdot s^{-1})$
Angular flux density	Radiance	$(W \cdot m^{-2} \cdot sr^{-1})$
Flux density	Irradiance	$(W \cdot m^{-2})$
Flux density	Radiosity	$(W \cdot m^{-2})$
	Radiant intensity	$(W \cdot sr^{-1})$

Table 2.2: Radiometric terms and units. Where *s* denotes unit time and *sr* is unit steradian (solid angle)

Since we are dealing with light transfer, and velocity of light is essentially infinite compared to the velocity of objects in habitable environments, it is often assumed that (dynamic) equilibrium is

instantaneously reached thus yielding a *steady-state* distribution. Thus in the following the distinction between power and energy will be ignored and *flux* will be the fundamental quantity.

2.3.2 Principles of Radiative Transfer

In the following the principles of radiative transfer will be covered and the fundamental quantities that will be the focus of the remainder of the thesis will be derived.

Phase Space

For a function or object with n degrees of freedom, the n-dimensional space which is accessible to the function or object is called its phase space. Phase space is a useful concept for describing the domain in which light transport takes place. It comes from the field of *transport theory*[12] which studies the motion of particles in abstract settings [Vea97]. The phase space for general light transport can be described as a system of N photons, each described by position x, direction ω and wavelength λ, assuming that light is incoherent in all dimensions. Thus this system is $6N$ dimensional and a given static system state would be represented as a point in this space, and the evolution of the system over time would be expressed by a 1D curve in this space. Under the assumption that photons do not interact the phase space can be described for single photons as a 6-dimensional state vector. Then a system of N photons can be described as a set of N points in the 6-dimensional phase space varying with time. This is useful because it yields a natural way of relating radiometric quantities to the density of photons in sub-regions of the phase space, of which the most fundamental quantity is the *photon number* N_p which counts the number of photons in a region of phase space.

In order to illustrate the problem the phase space can be restricted to five dimensions by limiting the scope to gray or mono-energetic radiative transfer with a constant velocity v by leaving out wavelength dependence. The resulting domain becomes $\mathbb{R}^3 \times \mathbb{S}^2$, where $\mathbb{R}^3$ is Euclidian three-space and $\mathbb{S}^2$ is the unit sphere of directions. *Phase space density* is a function defined over phase space and time:

$$n(r, \omega, t) dr d\omega$$

It relates a photon count to the differential volume dr about point $r \in \mathbb{R}^3$ in the differential solid angle $d\omega$. Thus the unit for such a quantity is $m^{-3} \cdot sr^{-1}$ at a time t. Now in global illumination it is more practical to consider the rate of flow across a surface [Arv93]. In Figure 2.7 this concept is illustrated. It shows the flow of photons across the differential surface area dA with directions in the differential solid angle $d\omega$ about the surface normal at dA during time dt. With velocity v in time dt the photons are contained in the volume $dAds$ where $ds = v \cdot dt$. The photon count becomes:

$$n(r, \omega) dA ds d\omega$$

Instead of focusing on the photon count in the volume it is useful to look at the rate of flow of photons across dA:

$$\phi(r, \omega) \equiv v n(r, \omega)$$

[12] Transport theory is the field that encompasses all macroscopic phenomena resulting from the interaction of infinitesimal particles with a medium. The macroscopic behaviours of photons, neutrons, and gas molecules are all within its purview. The central equation of transport theory is known as the Boltzmann equation.

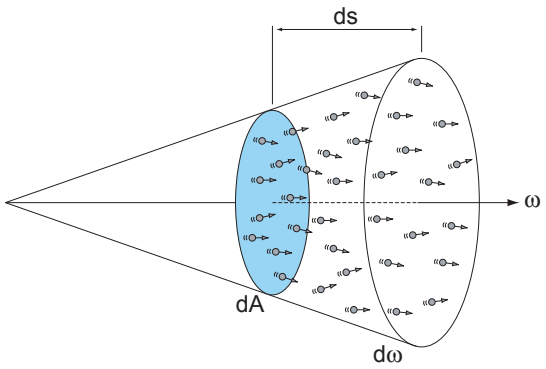


Figure 2.7: Phase space flux as flow over a differential surface dA.

The unit then becomes $m^{-2} \cdot sr^{-1} \cdot s^{-1}$. And the count of photons in $dAds$ is:

$$\phi(r,\omega)dAd\omega dt$$

The quantity ϕ (*phase space flux*) is the fundamental quantity of interest.

The abstract quantity of phase space flux can be translated into the radiometric quantity of radiance by assigning a notion of energy to the photons; the energy of a photon is proportional to its frequency $E = hv$, where h is Planck's constant $6.626 * 10^{-34}$, with unit $j \cdot s$ (joule second). The relationship between phase space flux and radiance then becomes [Arv93]:

$$\begin{aligned} L(r,\omega) &= hv\phi(r,\omega) \\ &= chvn(r,\omega) \end{aligned} \tag{2.8}$$

Where c is the universal speed of a photon in vacuum, and the unit for this quantity is $W \cdot m^{-2} \cdot sr^{-1}$.

2.3.3 Properties of Radiative Transfer

There are a few important properties of energy transport that must be addressed:

- *In a static scene flux is in equilibrium* - When viewing any (static) scene the distribution of light will appear constant, although light is flowing throughout the system. Also this equilibrium is very quickly reached because of the speed of light.
- *Light energy is conserved in a closed system* - This is stated in the 1st law of thermodynamics.

The latter property is especially useful as it can be used to verify that light transport is done in a correct manner.

2.3.4 Throughput *T*

The phase space of a differential surface area yields the following quantities:

$$\begin{aligned} d^2T &= d\omega dA^{\perp} \\ T &= \int\int d\omega dA^{\perp} \end{aligned} \tag{2.9}$$

2.3.5 Flux Φ

Flux is flow, in our case across a surface boundary ($J \cdot s^{-1} = W$):

$$\Phi = \frac{dQ}{dt} \tag{2.10}$$

The principle of energy conservation as defined above applies to this quantity. As noted earlier we will ignore temporal variation and assume that the system is in equilibrium, thus we need not make Φ a function of time.

2.3.6 Irradiance *E*

The following is flux density arriving at a differential surface dA ($W \cdot m^{-2}$):

$$E = \frac{d\Phi}{dA} \tag{2.11}$$

E is normally associated with incident radiation, so the term exitance M was introduced to denote radiation leaving a surface. Radiant exitance is also commonly known as radiosity B.

2.3.7 Radiance *L*

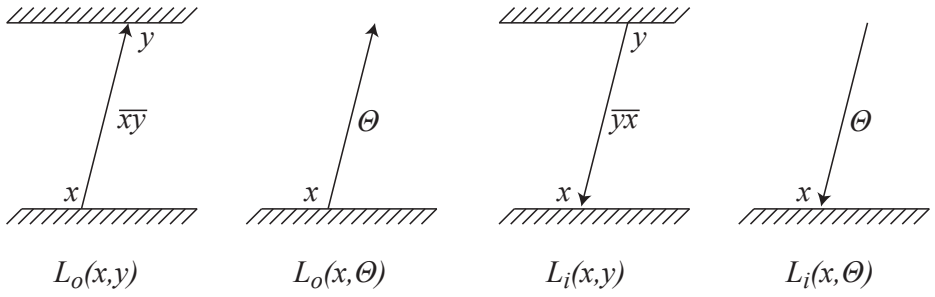


Figure 2.8: Exitant and incident radiance.

Radiance is the radiant power per unit projected area perpendicular to the ray per unit solid angle in the direction of the ray ($W \cdot m^{-2} \cdot sr^{-1}$):

$$L = \frac{dE}{d\omega} = \frac{d^2\Phi}{d\omega dA^{\perp}} = \frac{d^2\Phi}{d\omega dA\cos\theta} \tag{2.12}$$

Radiance is the fundamental quantity used to characterise the distribution of light in an environment. Care must be taken when dealing with radiance and directions; *incident* and *exitant* radiance are two distinct entities entirely. The notation in this thesis is as follows (see Appendix A):

- $L_o(x,y)$ - denotes *exitant* radiance leaving point x in direction $\overline{xy}$.
- $L_o(x,\Theta)$ - denotes *exitant* radiance leaving point x in direction Θ.
- $L_i(x,y)$ - denotes *incident* radiance impinging on point x from direction $\overline{yx}$.
- $L_i(x,\Theta)$ - denotes *incident* radiance impinging on point x from direction Θ.

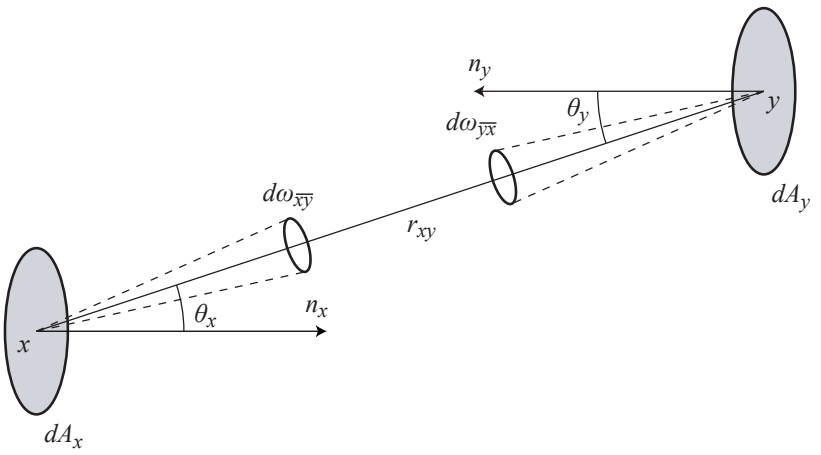


Figure 2.9: Radiance invariance geometry.

The notation above is illustrated in Figure 2.8. Incidentally, there is more than one way to describe the same entities, for example $L_o(x,y) = L_o(x,\overline{xy})$ and $L_i(x,y) = L_i(x,\overline{yx})$ the reason that these duplicate notations have been kept is that they simplify the notation in some equations. Radiance has the following important properties:

- Radiance is invariant along a ray.

- Response of a sensor is proportional to the radiance of surfaces visible to it.

- Radiance is a fundamental quantity. Other quantities describing light are derived from it.

The implications of these properties will now be addressed.

Radiance Invariance

If we apply the free space simplification (eg. there is no energy loss between surfaces see Section 2.2.1), the energy conservation principle tells us that, given two differential surfaces dA_x and dA_y and assuming no other surfaces emit any flux towards dA_y, the incident flux arriving at dA_y must equal the exitant flux at dA_x sent towards dA_y. The geometry of this concept is illustrated in Figure 2.9. This can be used to show that radiance is invariant along straight paths, that is, the exitant radiance at a point x in direction $\overline{xy}$ equals the incident radiance at y arriving from direction $\overline{xy}$ given that y is the closest visible surface point as seen from x in direction $\overline{xy}$, in other words $L_o(x,\overline{xy}) = L_i(y,\overline{xy})$ where $y = r(x,\overline{xy})$. This can be shown using the definition of radiance in terms of flux (Equation 2.12), the notation $d^2\Phi_{x\to y}$ means total

flux leaving dA_x arriving at dA_y, and $d^2\Phi_{y\leftarrow x}$ is total flux arriving at dA_y from dA_x:

$$
\begin{aligned}
d^2\Phi_{x\to y} &= d^2\Phi_{y\leftarrow x} && \text{(energy conservation principle)}\\
L_o(x,y)\cos\theta_x dA_x d\omega_{\overline{xy}} &= L_i(y,x)\cos\theta_y dA_y d\omega_{\overline{yx}} && \text{(Equation 2.12)}\\
L_o(x,y)\cos\theta_x dA_x \frac{\cos\theta_y dA_y}{r_{xy}^2} &= L_i(y,x)\cos\theta_y dA_y \frac{\cos\theta_x dA_x}{r_{xy}^2} && \text{(Equation 2.3)}\\
L_o(x,y)\frac{\cos\theta_x dA_x \cos\theta_y dA_y}{r_{xy}^2} &= L_i(y,x)\frac{\cos\theta_x dA_x \cos\theta_y dA_y}{r_{xy}^2}\\
L_o(x,y) &= L_i(y,x) && (2.13)
\end{aligned}
$$

Sensor Response

So, in the absence of participating media, radiance will not attenuate between surfaces and it is independent of the surface area and the solid angle. Thus, radiance can be thought of as perceived intensity; the quantity has been untied from angular and spatial areas. This is of great utility and together with the ray invariance principle is what makes ray tracing work. This is also why the brightness of a wall is independent of the distance from which you view it; given that the radiance is constant across the wall (see Figure 2.10). Sensor response is proportional to:

$$R = \int\int L\cos\theta dA d\omega = L\int\int \cos\theta dA d\omega = LT$$

Now, dA remains constant since it is the differential area on the sensor and $d\omega$ remains constant because it is projected onto the unit sphere. Hence, the throughput T (see Section 2.3.4) remains constant and the sensor response remains constant.

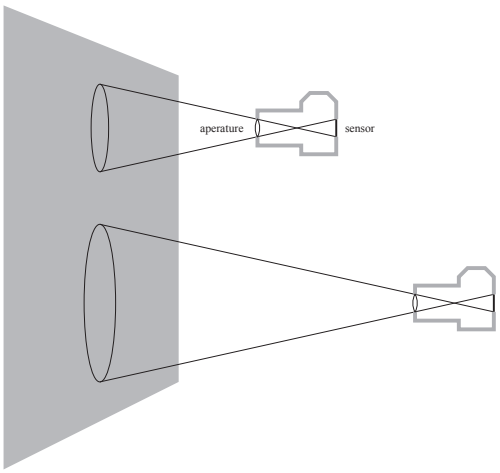


Figure 2.10: Sensors and radiance invariance.

Derivation of Irradiance from Radiance

The incident power arriving at a differential surface area is an oft-needed quantity. In the following irradiance is derived from the fundamental quantity radiance. Intuitively, radiance can be thought of as a function $L(x,\Theta)$ that is integrated over the solid angle $d\omega_\Theta$ subtended by the differential area x (around

direction Θ) and the projected area $dA_x \cos\theta$ yielding the flux leaving that area within that particular solid angle (see Figure 2.7). Now, if we let dA_x and $d\omega_\Theta$ tend to zero we have the flux travelling along a infinitesimally thin ray in direction Θ.

$$\Phi = \int\int L_o(x,\Theta)\cos\theta d\omega_\Theta dA_x \tag{2.14}$$

The incident flux density (power per unit area) at a differential area around x from direction Θ can be found in terms of radiance by using Equation 2.12:

$$E(x,\Theta) = \frac{d\Phi}{dA_x} = L_i(x,\Theta)\cos\theta d\omega_\Theta \tag{2.15}$$

However, the *irradiance* is the flux per unit area incident on a surface ($W \cdot m^{-2}$), so we need to integrate over the hemisphere over the surface:

$$E(x) = \int_{\Omega_x} L_i(x,\Theta)\cos\theta d\omega_\Theta \tag{2.16}$$

2.3.8 The Bidirectional Reflectance Distribution Function

The bi-directional reflectance distribution function (BRDF) provides a mathematical description of how radiance scatters when striking surfaces:

$$f_r(x,\Theta_i,\Theta_r) = \frac{dL_r(x,\Theta_r)}{dE_i(x,\Theta_i)} = \frac{dL_r(x,\Theta_r)}{L_i(x,\Theta_i)\cos\theta_i d\omega_{\Theta_i}} \tag{2.17}$$

The BRDF captures the reflective properties of the material of the surface. A large collection of BRDF models exist for various materials, some of which are analytical and some which are measured [War92]. The BRDF is a 4-dimensional function (given that wavelength dependence is left out) that given an incoming direction Θ_i and a reflected direction Θ_r over a surface point x yields a ratio of the incident irradiance arriving at x from $-\Theta_i$ through a differential solid angle to the differential radiance reflected into the reflected direction Θ_r. This yields a hemisphere of directions above x each emitting a differential radiance value due to the incident irradiance at x from $-\Theta_i$. The geometry of this is illustrated in Figure 2.11.

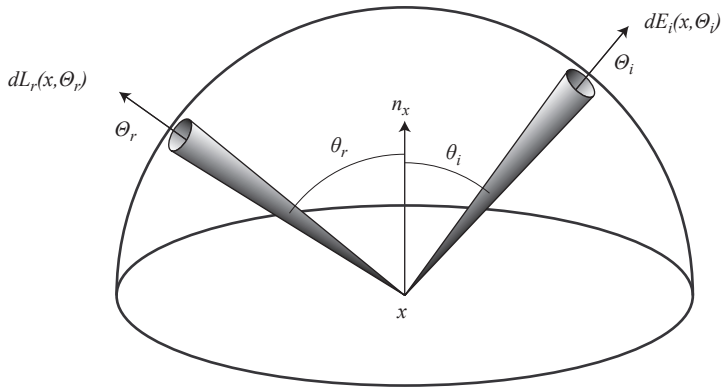


Figure 2.11: Geometry of the BRDF.

It is often useful to be able to quantify the total reflected radiance due to the incident differential irradiance over the hemisphere over a surface point. This can be achieved by integrating Equation 2.17

over the hemisphere; this is also known as the *reflectance* equation:

$$\begin{aligned} dL_r(x,\Theta_r) &= dE_i(x,\Theta_i) f_r(x,\Theta_i,\Theta_r) \\ L_r(x,\Theta_r) &= \int_{\Omega_x} dE_i(x,\Theta) f_r(x,\Theta,\Theta_r) \\ L_r(x,\Theta_r) &= \int_{\Omega_x} L_i(x,\Theta) f_r(x,\Theta,\Theta_r) \cos(n_x,\Theta) d\omega_\Theta \end{aligned} \tag{2.18}$$

A number of variations on the BRDF exist:

- BTDF: Bi-directional transmittance distribution function is used to model transparent surfaces.
- BSDF: Bi-directional scattering distribution function is a more general mathematical description of surface scattering. It is a union of two BRDFs and two BTDFs one each for the upper and lower hemisphere surrounding the surface [Vea97].
- BSSRDF: Bi-directional subsurface scattering distribution function is used to model subsurface scattering where the point of incident radiance may differ from the point where radiance exits [JMLH01].

A few properties, pertaining to BRDFs, need to be addressed. Note, that these properties apply to BRDFs not necessarily BTDFs or BSDFs [Vea97].

Helmholtz Reciprocity

Physical surface reflection is known to be symmetric [NRH+77]. E.g. the reflective properties remain the same no matter which direction light travels. This means that Θ_i and Θ_r can be freely interchanged in Equation 2.17:

$$f_r(x,\Theta_i,\Theta_r) = f_r(x,\Theta_r,\Theta_i) \tag{2.19}$$

Energy Conservation

In order for the BRDF to be physically plausible it needs to adhere to the energy conservation principle. More precisely, the total amount of power reflected over all directions can never exceed the total amount of power incident on the surface. The difference in power is absorbed by the surface and converted into other forms of energy (typically heat). This can be expressed in the following way [Dut96]. The total irradiance over the hemisphere E is given by Equation 2.16, the total amount of reflected power is given by M and depends on the BRDF:

$$M = \int_{\Omega_x} d\omega_\Theta \int_{\Omega_x} d\omega_\Psi f_r(x,\Theta,\Psi) L_i(x,\Theta) \cos(n_x,\Psi) \cos(n_x,\Theta) \tag{2.20}$$

So the energy conservation principle says that $\frac{M}{E} \leq 1$ or:

$$\frac{\int_{\Omega_x} d\omega_\Theta \int_{\Omega_x} d\omega_\Psi f_r(x,\Theta,\Psi) L_i(x,\Theta) \cos(n_x,\Psi) \cos(n_x,\Theta)}{\int_{\Omega_x} d\omega_\Theta L_i(x,\Theta) \cos(n_x,\Theta)} \leq 1 \tag{2.21}$$

Often Equation 2.21 is used to check the validity of a candidate global illumination algorithm. Clearly it is not sufficient to prove that the algorithm solves the global illumination problem accurately, but if the energy conservation condition is not met it is a good indication that the transport is not solved correctly. In this thesis we will use Equation 2.21 as part of the validation.

2.4 Mathematical Framework for Global Illumination

The global illumination problem is governed by a pair of adjoint recursive integral equations; *the radiance equation* and *the potential equation*. They are both Fredholm equations of the second kind and have generally no analytical solution so they must be solved numerically [Dut96]. They are closely related but approach the global illumination problem from different perspectives and thus have very different interpretations. The following sections are devoted to these equations.

2.4.1 The Radiance Equation

The radiance equation is a slightly different form of the rendering equation introduced by Kajiya in his seminal paper [Kaj86]. The radiance equation expresses the equilibrium of flux over points and directions as the sum of emitted radiance and reflected radiance incident over the hemisphere. There are two forms of the radiance equation, one form is defined in terms of radiance incident or exitant over the hemisphere of directions and the other form is defined in terms of visible surface points, each of these can in turn be described in terms of exitant or incident radiance yielding four forms of the radiance equation.

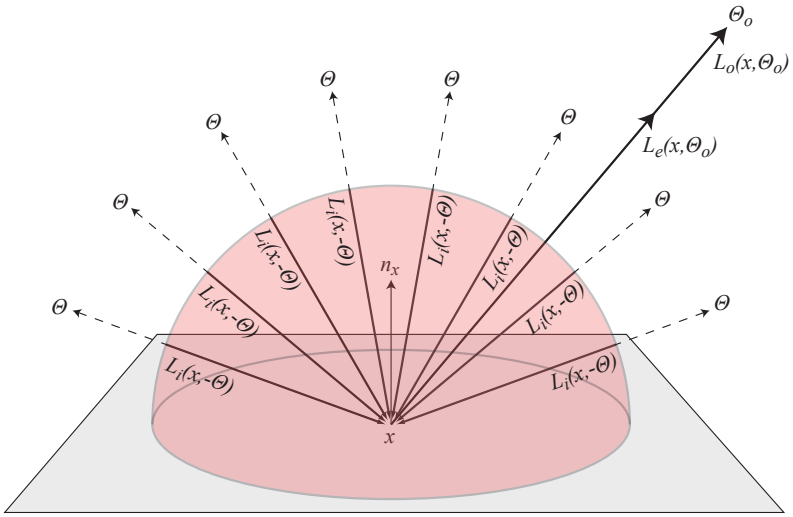


Figure 2.12: Geometry of the rendering equation; hemisphere integration.

Hemisphere Integration

The radiance equation is essentially the reflectance Equation 2.18 combined with self-emitted radiance. The radiance equation can be written as follows:

$$\begin{aligned} L_o(x,\Theta_o) &= L_e(x,\Theta_o) + L_r(x,\Theta_o) \\ &= L_e(x,\Theta_o) + \int_{\Omega_x} L_i(x,-\Theta) f_r(x,\Theta,\Theta_o) \cos(n_x,\Theta) d\omega_\Theta \end{aligned} \tag{2.22}$$

Equation 2.22 expresses the outgoing radiance of a point x in direction Θ_o in terms of self-emitted radiance $L_e(x,\Theta_o)$ and the amount of incident radiance over the hemisphere Ω_x over x that is reflected into Θ_o. This is illustrated in Figure 2.12.

Equation 2.22 includes a mixture of incident and exitant quantities of radiance. It is more convenient to either express the radiance equation in terms of exitant or incident radiance solely. Fortunately, due to the radiance invariance principle (Equation 2.13) this is straightforward. The following equations express the radiance equation in terms of exitant and incident hemispherical radiance respectively:

$$L_o(x,\Theta_o) = L_e(x,\Theta_o) + \int_{\Omega_x} L_o(y,-\Theta) f_r(x,\Theta,\Theta_o)\cos(n_x,\Theta)d\omega_\Theta, \quad \text{where } y = r(x,\Theta) \tag{2.23}$$

$$L_i(x,\Theta_i) = L_e^i(x,\Theta_i) + \int_{\Omega_y} L_i(y,-\Theta) f_r(y,\Theta,\Theta_i)\cos(n_y,\Theta)d\omega_\Theta, \quad \text{where } y = r(x,-\Theta_i) \tag{2.24}$$

Equations 2.23 and 2.24 are illustrated in Figure 2.13.

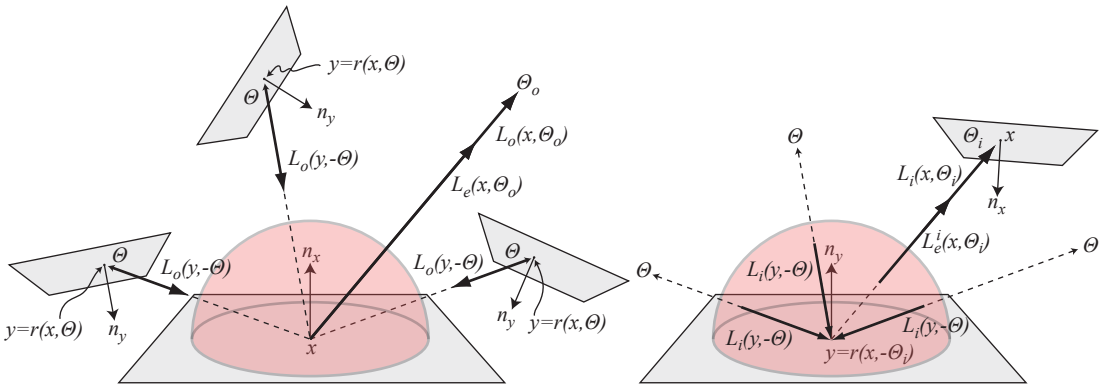


Figure 2.13: Geometry of the rendering equation; exitant and incident hemisphere integration respectively.

Surface Integration

Depending on the application it can be more appropriate to integrate over visible surface points in the scene rather than the hemisphere of directions. This transformation changes the integration domain from the hemisphere of directions to visible surface points exploiting the visibility functions described in Section 2.2.3. Equation 2.22 is rewritten replacing $d\omega_\Theta$ by dA_z where z is the closest visible point from x in direction $\Theta = \overline{xz}$ using Equation 2.3. This transformation yields the following equation:

$$\begin{aligned} L_o(x,y) &= L_e(x,y) + L_r(x,y) \\ &= L_e(x,y) + \int_{A_x} L_i(x,z) f_r(x,\Theta,\overline{xy})\cos(n_x,\Theta)\frac{\cos(n_z,-\Theta)}{r_{xz}^2}dA_z, \qquad \text{where } z = r(x,\Theta) \end{aligned}$$

By using the visibility function (Equation 2.5) we can avoid the dependence of the integration domain on x and arrive at the more general expression, which integrates over the entire surface area A:

$$\begin{aligned} L_o(x,y) &= L_e(x,y) + \int_A L_i(x,z) f_r(x,\Theta,\overline{xy})\cos(n_x,\Theta)\frac{\cos(n_z,-\Theta)}{r_{xz}^2}V(x,z)dA_z \\ &= L_e(x,y) + \int_A L_i(x,z) f_r(x,\Theta,\overline{xy})G(x,z)dA_z \end{aligned} \tag{2.25}$$

$$\text{where} \quad G(x,z) = \frac{\cos(n_x,\Theta)\cos(n_z,-\Theta)V(x,z)}{r_{xz}^2}$$

In Equation 2.25 the geometric function $G(x,z)$, which quantifies to what extent x and z are able to exchange energy, is introduced for clarity.

As was the case for integration over the hemisphere Equation 2.25 includes a mixture of incident and exitant quantities of radiance. The following equations express the radiance equation in terms of exitant and incident radiance respectively:

$$L_o(x,y) = L_e(x,y) + \int_A L_o(z,x) f_r(x,\overline{xz},\overline{xy}) G(x,z) dA_z \tag{2.26}$$

$$L_i(x,y) = L_e^i(x,y) + \int_A L_i(y,z) f_r(y,\overline{yz},\overline{yx}) G(y,z) dA_z \tag{2.27}$$

Equations 2.26 and 2.27 are illustrated in Figure 2.14.

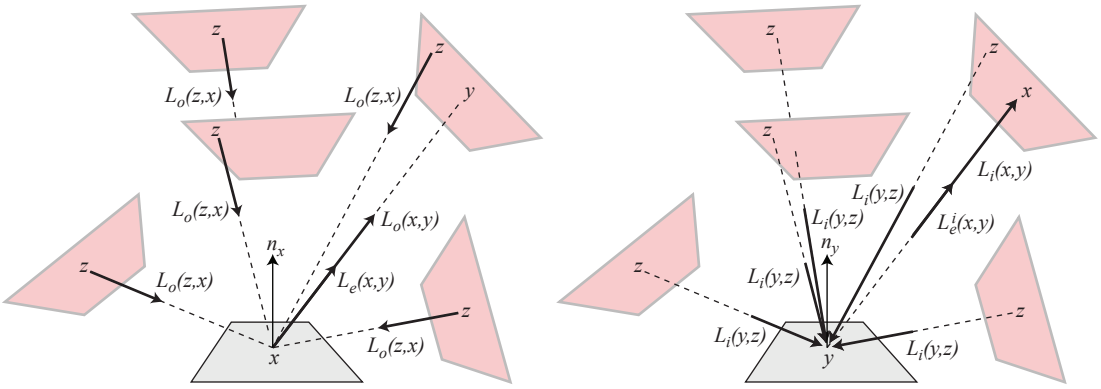


Figure 2.14: Geometry of the rendering equation; exitant and incident surface integration respectively.

Linear Transport Operator Form of the Radiance Equation

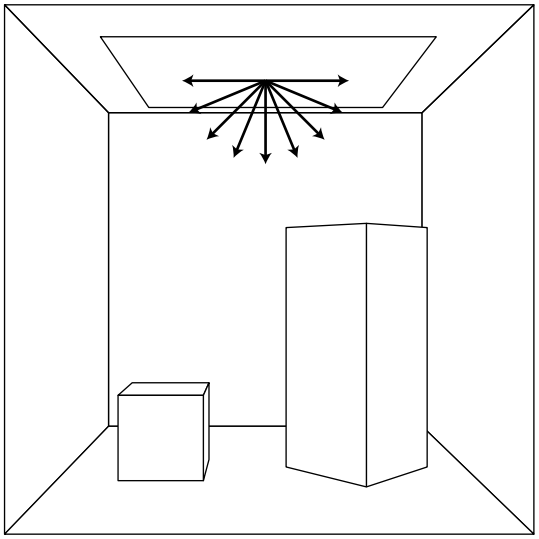
The recursive radiance equation can be expressed more elegantly by introducing an integral operator. The integral transport operator links the emitted and reflected parts of the radiance equation by transforming one distribution of radiance into another distribution that expresses the initial distribution after one reflection. The integral transport operator is T and it introduces a new function TL that is applied to Equation 2.23:

$$\begin{aligned} L_o(x,\Theta_o) &= L_e(x,\Theta_o) + TL(x,\Theta_o) \\ TL(x,\Theta_o) &= \int_{\Omega_x} L_o(y,-\Theta) f_r(x,\Theta,\Theta_o) \cos(n_x,\Theta) d\omega_\Theta, \quad \text{where } y = r(x,\Theta) \end{aligned} \tag{2.28}$$

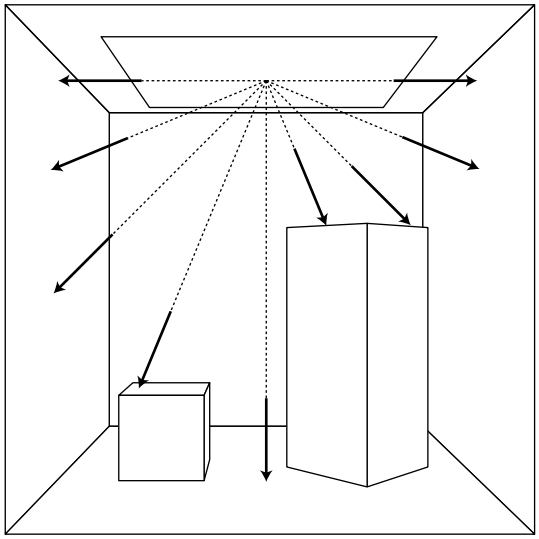
Again the integral can be shifted from the directional domain to surface points by exploiting the geometric function (see Equation 2.25):

$$\begin{aligned} L_o(x,y) &= L_e(x,y) + TL(x,y) \\ TL(x,y) &= \int_A L_o(z,x) f_r(x,\overline{xz},\overline{xy}) G(x,z) dA_z \end{aligned} \tag{2.29}$$

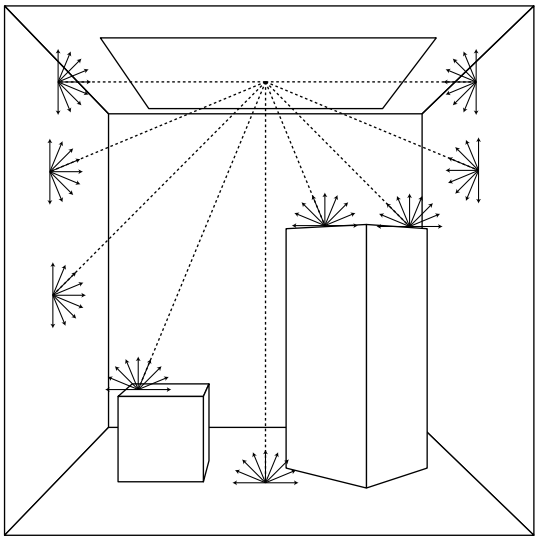
Equations 2.28 and 2.29 serve as powerful and intuitive tools for describing transport algorithms. They are often used as a shorthand by assuming that they can evaluate (x,Θ) or (x,y) pairs defined in the domain $A \times \Omega$. In this shorthand the radiance equation is simply $L = L_e + TL$ and TL describes the component of L that is due to reflection only. Then radiance emitted by emitters is given by L_e and TL_e gives the total reflected radiance which had its origin in the emitters but reflected once (see Figure 2.15(a),



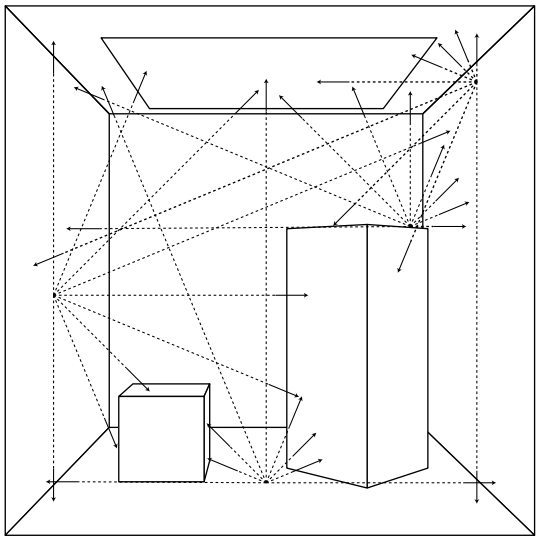
(a) L_e



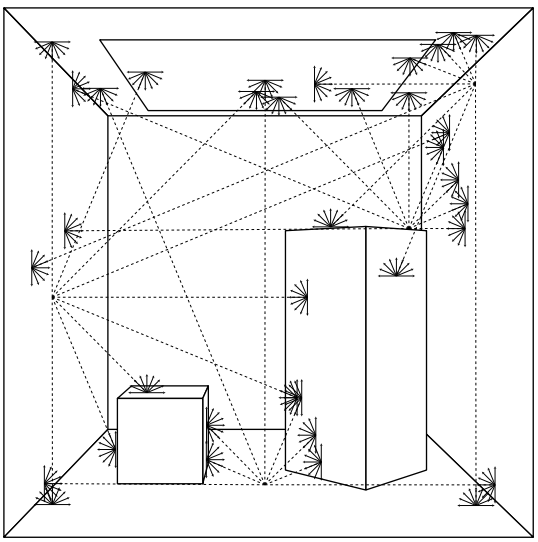
(b) Shoot L



(c) Reflect$\rightarrow TL_e$ (for clarity only bold hemispheres are shot further)



(d) Shoot L

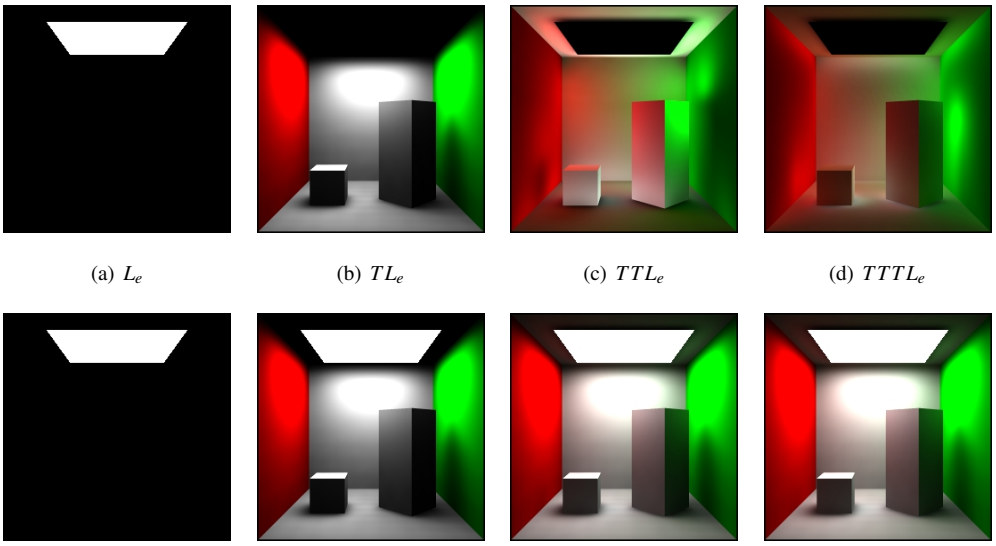


(e) Reflect$\rightarrow TTL_e$

Figure 2.15: Shooting radiance from an emitter. For clarity not all paths in TL_e are shot.

Figure 2.15(b) and Figure 2.15(c)), TTL_e is the emitted radiance reflected twice (see Figure 2.15(d) and Figure 2.15(e)) and T^nL_e is radiance reflected n times describing radiance due to paths of length $n+1$ back to an emitter. The operator can also be expanded out in a power series $L = L_e + TL_e + T^2L_e + T^3L_e + ...$ describing radiance due to paths of any length back to an emitter.

This concept is illustrated further in Figure 2.16 that shows a Cornell box where various radiance distributions due to zero, one, two or three reflections are shown. The bottom row shows operator expansion to a power series. In Figure 2.16(e) only emitted radiance L_e is shown, in Figure 2.16(f) radiance reflected once is added causing the box and objects to be directly illuminated, in Figure 2.16(g) radiance reflected twice is added causing effects such as colour bleeding and finally in Figure 2.16(h) radiance reflected thrice is added the effect of which is very subtle. The upper row of images shows the radiance for paths of length one to four respectively, but the radiance is not summed up; only the contribution due to the particular reflection is shown. In Figure 2.16(a) the initial radiance distribution L_e is shown, Figure 2.16(b) shows the distribution reflected once. In Figure 2.16(c) and Figure 2.16(d) the distributions due to a second and third reflection is shown. Because the albedo of surfaces is normally less than 1, the radiance for longer paths is substantially weaker due to absorption, the images have been scaled for viewing purposes.



(a) L_e (b) TL_e (c) TTL_e (d) $TTTL_e$

(e) L_e (f) $L_e + TL_e$ (g) $L_e + TL_e + TTL_e$ (h) $L_e + TL_e + TTL_e + TTTL_e$

Figure 2.16: Radiance distributions after 0-3 reflections. Image c and d are scaled by 5x and 15x respectively.

2.4.2 The Potential Equation

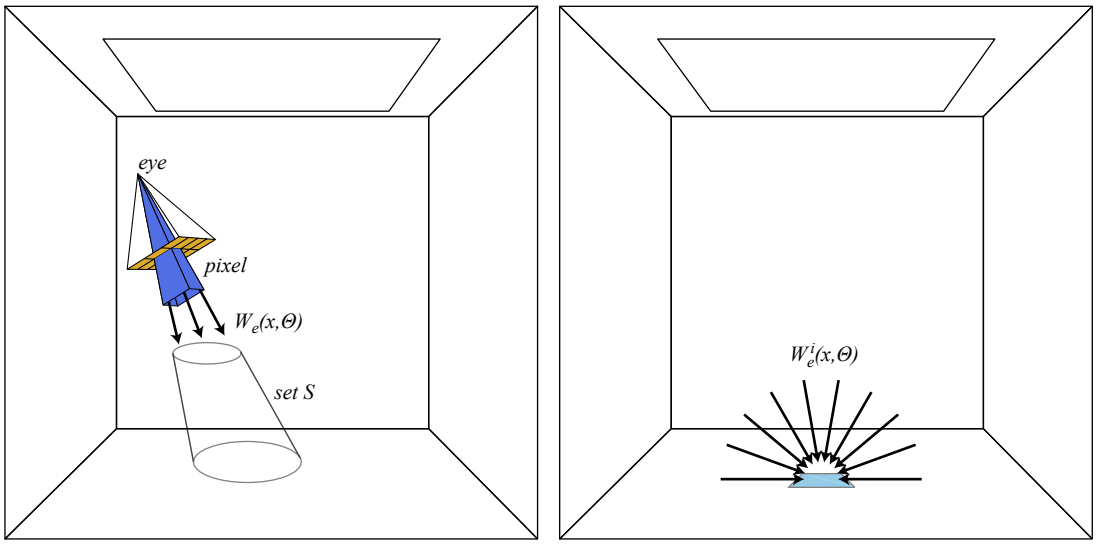
The potential equation was first introduced to the computer graphics community by Smits [SAS92]. In his paper which exploits the duality of the rendering equation and its adjoint in order to improve the effectiveness of a radiosity algorithm the term *importance* is used to denote the quantity distributed by the adjoint transport equation. Later Pattanaik [Pat93b] used the duality to develop a taxonomy classifying

global illumination algorithms and used the term *potential* rather than *importance*. In the following the naming convention introduced by Pattanaik will be used in order to avoid confusion with *importance sampling* used in Monte Carlo methods.

Given a set $S = A_S \times \Omega_S \subset A \times \Omega$ the potential equation expresses the potential influence of any $(x,\Theta) \in A \times \Omega$ on the illumination of S. More precisely, the potential $W(x,\Theta)$ is the flux radiated through the set S as a result of a unit of radiance emitted through a differential volume around point x and direction Θ:

$$\begin{aligned} W(x,\Theta) &= \frac{d^2\Phi(S)}{d^2\Phi(x,\Theta)} \\ &= \frac{d^2\Phi(S)}{L_o(x,\Theta)\cos(n_x,\Theta)d\omega_\Theta dA_x} \end{aligned} \tag{2.30}$$

Potential is a dimensionless quantity. Due to the linearity of reflected radiance it is independent of the magnitude of source radiance, so is merely a function of geometry and material properties. The emitted radiance for (x,Θ) can influence the flux of S in two ways; directly or through one or more surface reflections.



(a) Exitant pixel potential (b) Incident patch potential

Figure 2.17: Potential for a pixel set and a patch set.

If $(x,\Theta) \in S$ then it is clear that the emitted radiance directly contributes to the flux of S; in this case the potential $W(x,\Theta) = 1$. This requires the existence of a function that can determine whether a given point and direction belongs to the set, such function can be expressed as follows:

$$g(x,\Theta) = \begin{cases} 1, & \text{if } (x,\Theta) \in S \\ 0, & \text{if } (x,\Theta) \notin S \end{cases} \tag{2.31}$$

Now, the direct contribution can be formally expressed using $W(x,\Theta) = g(x,\Theta)$. Analogously to the radiance equation this contribution to the potential will be referred to as self-emitted exitant potential $W_e(x,\Theta)$ or self-emitted incident potential $W_e^i(x,\Theta)$. In Figure 2.17 self-emitted potential for two well

known sets is illustrated. The set in Figure 2.17(a) represents one pixel on the image plane of a pinhole camera and the exitant potential for the set emanates from the solid angle of the pixel frustum, this set is typically used with ray tracing and path tracing algorithms. The set in Figure 2.17(b) represents a finite element, the set comprises the area of the patch and the entire hemisphere above it and the incident potential is shown, this set is for example used with radiosity algorithms.

The radiance can indirectly contribute towards the flux of S through one or more surface reflections. Surface reflections are governed by the BRDF of the surface (see Section 2.3.8) and a quantity of radiance $L_o(x,\Theta_o)$ incident on a surface point $y = r(x,\Theta_o)$ results in an outgoing reflected differential radiance value $dL_o(y,\Theta)$ in each direction Θ. See Figure 2.18 for an illustration of this geometry.

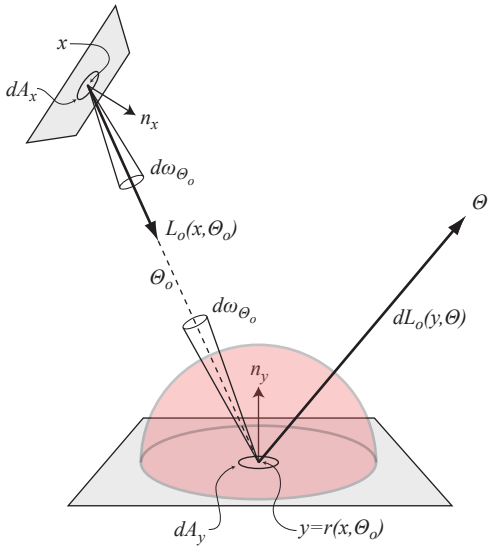


Figure 2.18: Indirect contribution to the flux of S.

Each of these differential radiances can contribute to $d^2\Phi(S)$ and we need to integrate over the hemisphere Ω_y in order to arrive at an expression for the potential for (x,Θ_o) in terms of the potentials over the hemisphere Ω_y. Clearly this expression is recursive since any number of reflections can occur before reaching the set S. The expression for the indirect component of the potential is:

$$W(x,\Theta_o) = \int_{\Omega_y} W(y,\Theta) f_r(y,-\Theta_o,\Theta)\cos(n_y,\Theta)d\omega_\Theta, \quad \text{where } y = r(x,\Theta_o) \tag{2.32}$$

The full expression for the potential adds together the direct (Equation 2.30) and indirect (Equation 2.32) contributions since it is quite possible due to the recursive nature of radiance that the potential value can have both a direct contribution and an indirect contribution that undergoes several reflections. The following expression gives the full potential equation:

$$W(x,\Theta_o) = W_e(x,\Theta_o) + \int_{\Omega_y} W(y,\Theta) f_r(y,-\Theta_o,\Theta)\cos(n_y,\Theta)d\omega_\Theta, \quad \text{where } y = r(x,\Theta_o) \tag{2.33}$$

The potential given in Equation 2.33 is clearly very similar to the radiance equation given in Equation 2.12, although each transport a different quantity they are mathematically identical. Potential as a transport quantity is also invariant along straight lines as is the case for radiance.

$$W_o(x,\Theta) = W_i(y,\Theta), \quad \text{where } y = r(x,\Theta)$$

This allows us to give the potential equation in different forms: one based on purely incident potential and another purely in terms of exitant potential. This is analogous to the radiance equations given in Equation 2.23 and Equation 2.24. The counterparts for potential are given here for completeness:

$$W_o(x,\Theta_o) = W_e(x,\Theta_o) + \int_{\Omega_x} W_o(y,-\Theta) f_r(x,\Theta,\Theta_o)\cos(n_x,\Theta)d\omega_\Theta, \quad \text{where } y = r(x,\Theta) \tag{2.34}$$

$$W_i(x,\Theta_i) = W_e^i(x,\Theta_i) + \int_{\Omega_y} W_i(y,-\Theta) f_r(y,\Theta,\Theta_i)\cos(n_y,\Theta)d\omega_\Theta, \quad \text{where } y = r(x,-\Theta_i) \tag{2.35}$$

Akin to the discussion on the radiance equation, the potential equation can also be expressed in terms of surface points rather than the hemisphere of directions:

$$W_o(x,y) = W_e(x,y) + \int_A W_o(z,x) f_r(x,\overline{xz},\overline{xy}) G(x,z) dA_z \tag{2.36}$$

$$W_i(x,y) = W_e^i(x,y) + \int_A W_i(y,z) f_r(y,\overline{yz},\overline{yx}) G(y,z) dA_z \tag{2.37}$$

Linear Transport Operator Form of the Potential Equation

As was the case for the radiance equation the potential equation can be elegantly expressed as a linear series in terms of the integral transport operator Q:

$$\begin{aligned} W_i(x,\Theta_i) &= W_e^i(x,\Theta_i) + QW_i(x,\Theta_i) \\ QW_i(x,\Theta_i) &= \int_{\Omega_y} W_i(y,-\Theta) f_r(y,\Theta,\Theta_i)\cos(n_y,\Theta)d\omega_\Theta, \quad \text{where } y = r(x,-\Theta_i) \end{aligned} \tag{2.38}$$

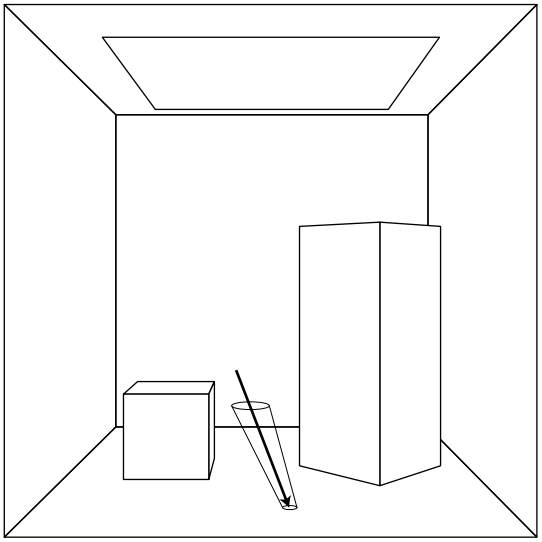
Again the integral can be shifted from the directional domain to surface points by exploiting the geometric function:

$$\begin{aligned} W_i(x,y) &= W_e^i(x,y) + QW_i(x,y) \\ QW_i(x,y) &= \int_A W_i(y,z) f_r(y,\overline{yz},\overline{yx}) G(y,z) dA_z \end{aligned} \tag{2.39}$$

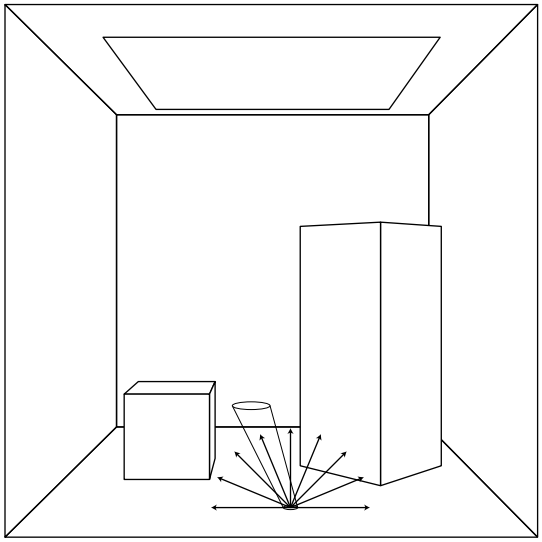
W_e^i is the self-emitted incident potential and serves as a starting point for a potential distribution defining the subset of $A \times \Omega$ that is currently the region of interest. QW_e^i is the incident potential for visible surface points $y = r(x,-\Theta_i)$ from x that has undergone one surface reflection at x. This provides a shorthand for describing distributions of incident potential, $Q^n W_e^i$ expresses the distribution of potential n reflections away from the initial incident distribution, and the power series $W_i = W_e^i + QW_e^i + Q^2W_e^i + Q^3W_e^i + \ldots$ describes the potential as the sum of potential for paths of any length. Figure 2.19 illustrates this concept for up to two reflections.

2.4.3 Solutions to the Global Illumination Problem

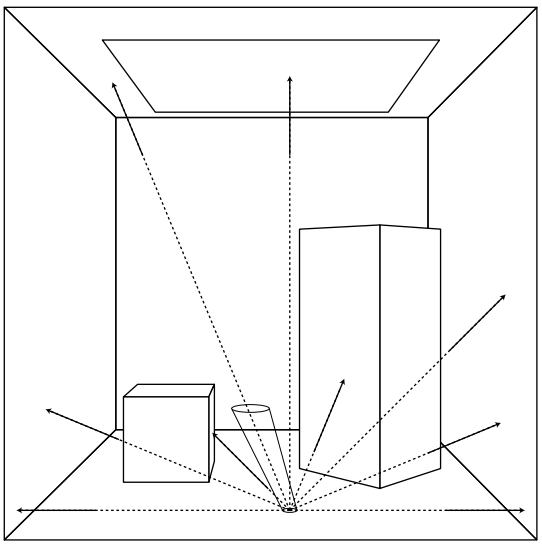
The radiance equation and the potential equation are both Fredholm equations of the second kind. Typically, this kind of recursive integral equation has no analytical solution and has to be solved numerically. In order to solve either of these equations in any one of its forms, hemispherical or surface integration, exitant or incident, would require finding radiance values or potential values for all surface points and all directions relative to those points $(x,\Theta) \in A \times \Omega$. Due to the recursive nature of the equations, each such point in the 5-dimensional space is (potentially) dependent on all other points making this problem



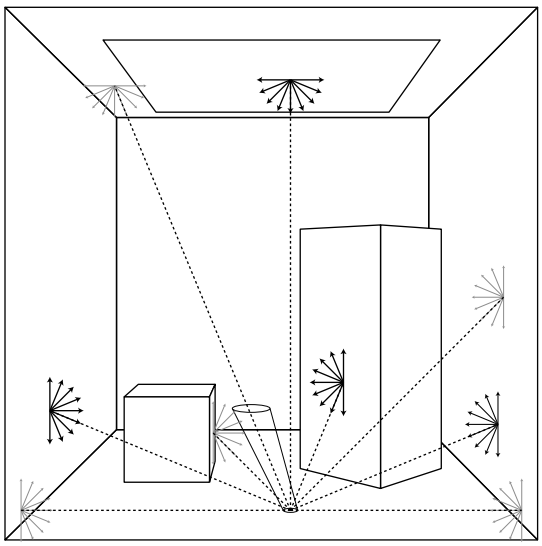
(a) W_e^i



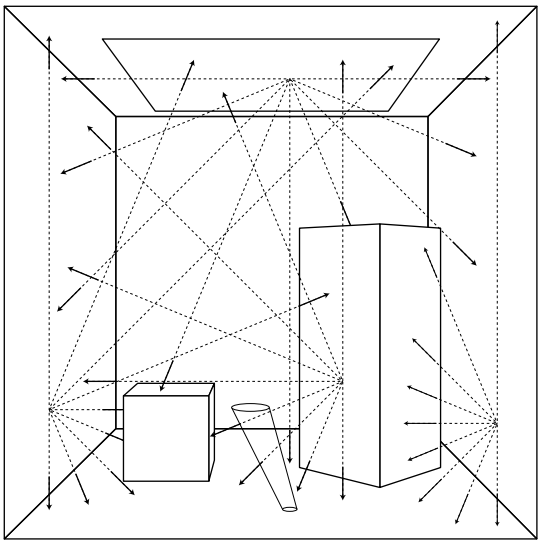
(b) Reflect back



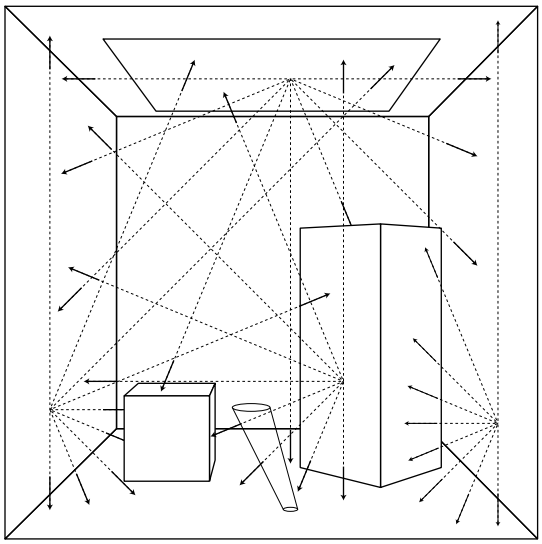
(c) Shoot$\rightarrow QW_e^i$



(d) Reflect back (for clarity only bold hemispheres are shot further)



(e) Shoot$\rightarrow QQW_e^i$



(f) Reflect back

Figure 2.19: Shooting potential for a set. For clarity not all reflected paths in QW_e^i are shot.

difficult to solve for the entire domain. Taking this route is intractable in practice due to its computational complexity.

However, by assuming that radiance changes slowly for some *collections* of points and directions the problem can be made viable by splitting the domain into a finite number of sets $S = A_S \times \Omega_S \subset A \times \Omega$ of points and directions for which an average radiance value is computed. This is achieved by computing the flux leaving the set $\Phi(S)$ and dividing it by the total surface area and total solid angle (throughput) of the set in question:

$$L_{avg}(S) = \frac{\Phi(S)}{T(S)} \tag{2.40}$$

The global illumination problem then reduces to computing the average radiance for the finite selection of sets chosen to represent the problem. The number of such sets and their relative size depends on the geometry of the environment, the material attributes and the desired accuracy of the solution. For example irradiance and radiance cache algorithms compute fluxes with a sparse sampling of $A \times \Omega$ under the assumption that for diffuse and glossy surfaces values can be predicted by interpolating between the average values represented by the chosen sets [WH92, KGPB05].

Computing Flux Sets with the Radiance Equation

More formally, $S = A_S \times \Omega_S \subset A \times \Omega$ is the set of points and directions for which an average radiance value is desired. The radiance equation expresses the outgoing radiance for a point in $A \times \Omega$, hence, the radiant flux can be computed by integrating radiance over the points and directions in set S in question. The flux leaving the set S can then be expressed as a double integral over points and directions belonging to the set by exploiting the set dependency function (see Equation 2.31):

$$\begin{aligned}\Phi(S) &= \int_A \int_{\Omega_x} L_o(x,\Theta) g(x,\Theta) \cos(n_x,\Theta) dA_x d\omega_\Theta \\ &= \int_A \int_{\Omega_x} L_o(x,\Theta) W_e^i(x,\Theta) \cos(n_x,\Theta) dA_x d\omega_\Theta\end{aligned} \tag{2.41}$$

Intuitively, this method starts from S and works its way backwards towards emitting points for which the emitted radiance is non-zero by evaluating the radiance equation.

Computing Flux Sets with the Potential Equation

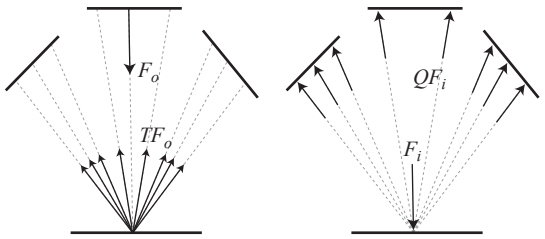
Similarly, the flux leaving a set S can also be evaluated by applying the potential equation. Recall that the potential equation provides a measure of the influence of an emitting point (x,Θ) on the flux of a set. Conveniently, all self-emitting points in the environment are known; that is, all points for which $L_e \neq 0$. These *initial* radiance values account for the illumination of the entire environment in question. So, in order to compute the flux for a set S only the potential of emitting points needs to be evaluated. The flux in terms of the potential equation can be expressed as follows:

$$\Phi(S) = \int_A \int_{\Omega_x} W_i(x,\Theta) L_e(x,\Theta) \cos(n_x,\Theta) dA_x d\omega_\Theta \tag{2.42}$$

Intuitively, this works opposite to Equation 2.41 in that it starts from emitting points and works its way towards points and directions in the set S for which the potential is non-zero by evaluating the potential equation.

Unified Solutions to the Global Illumination Problem

In order to more clearly see the symmetry hinted at in the previous two subsections the linear transport operator form of the radiance equation and the potential equation must be revisited. Earlier, two operators T and Q were introduced. The transport operator T can transform an exitant distribution to another exitant distribution whereas the operator Q transforms an incident distribution to another incident distribution by reflection. Figure 2.20 illustrates both operators.



(a) Operator T (b) Operator Q

Figure 2.20: Exitant transport operator T and incident transport operator Q.

In Section 2.4.1 the operator T was used to linearise the radiance equation in its exitant form $L_o = L_e + TL_o$, and in section Section 2.4.2 the operator Q was used to linearise the potential equation in its incident form $W_i = W_e^i + QW_i$. However, since the radiance equation and potential equation are mathematically identical other combinations are possible. The remaining two combinations are transport of radiance in its incident form $L_i = L_e^i + TL_i$ and transport of potential in its exitant form $W_o = W_e + QW_o$. This provides a set of four distinct possibilities for transport in three-dimensional environments.

The symmetry between radiance and potential distributions is striking. In [Pat93a] Pattanaik argued that the operators T and Q are in fact adjoint with respect to the following inner product[13]:

$$\langle F_1, F_2 \rangle = \int_A \int_{\Omega_x} F_1(x,\Theta) F_2(x,\Theta) \cos(n_x,\Theta) dA_x d\omega_\Theta \tag{2.43}$$

This inner product is designed to be able to capture all possible variations for integrating radiance over a set S; it is essentially the "skeleton" of Equation 2.41 and Equation 2.42. It contains the two functions F_1 and F_2, which can be suitably replaced by a combination of radiance and potential equations. Mathematically the adjointness of T and Q with respect to the inner product above requires that $\langle TF_1, F_2 \rangle = \langle F_1, QF_2 \rangle$ is satisfied for all F_1 and F_2. The adjoint of T is denoted T^*, so $Q = T^*$.

In order to "design" an expression for the flux of a set, a number of choices exist: Firstly, radiance can be exitant and potential incident or radiance can be incident and potential exitant. Secondly, radiance can be the transport quantity combined with an initial known potential distribution or potential can be the transport quantity combined with an initial also known radiance distribution. This yields exactly four mathematically equivalent formulations for the global illumination problem:

[13] A proof of this can be found in [Dut96].

	Radiance transport	**Potential transport**
Exitant transport	$\Phi(S) = \langle L_o, W_e^i \rangle$	$\Phi(S) = \langle L_e^i, W_o \rangle$
	$L_o = L_e + TL_o$	$W_o = W_e + TW_o$
Incident transport	$\Phi(S) = \langle L_i, W_e \rangle$	$\Phi(S) = \langle L_e, W_i \rangle$
	$L_i = L_e^i + T^* L_i$	$W_i = W_e^i + T^* W_i$

In order to arrive at a solution for the flux of a set starting from an inner product the unknown function is substituted with its transport equation for either radiance or potential. An example of this is shown in Equation 2.44 where exitant radiance is propagated against a known initial potential distribution defining the set for which the flux is required.

$$\begin{aligned}
\Phi(s) &= \langle L_o, W_e^i \rangle \\
&= \langle L_e + TL_o, W_e^i \rangle \\
&= \langle L_e, W_e^i \rangle + \langle TL_o, W_e^i \rangle \\
&= \langle L_e, W_e^i \rangle + \langle TL_e, W_e^i \rangle + \langle TTL_o, W_e^i \rangle \\
&= \langle L_e, W_e^i \rangle + \langle TL_e, W_e^i \rangle + \langle TTL_e, W_e^i \rangle + \langle TTTL_o, W_e^i \rangle
\end{aligned} \tag{2.44}$$

The first substitution produces two terms one of which contains only known terms $\langle L_e, W_e^i \rangle$ and one that must be expanded further $\langle TL_o, W_e^i \rangle$ because it contains the unknown function TL_o. Any term of the expanded inner product can contribute to the flux of the set provided that there are (x, Θ) pairs with a non-zero value for both the radiance and potential distributions. Physically, the first term in Equation 2.44 is the contribution due to self-emittance of the set this is only non-zero for emitters in the environment. The second term contributes to the flux by way of direct lighting. This is radiance arriving directly from emitters and is reflected once into (x, Θ) pairs that belong to the set thereby contributing to the flux leaving the set. Finally, the third contribution is radiance from emitters reflected once and then reflected into (x, Θ) pairs belonging to the set; one well known effect from this type of illumination is colour bleeding.

Due to the fact that the operators are adjoint T can be shifted to T^* in the inner product at any stage. This leads to many possible solutions which are mathematically equivalent. They are best illustrated as a binary tree of possible series of the inner product starting at $\langle L_e, W_e^i \rangle$:

$$\Phi(S) = \langle L_e, W_e^i \rangle + \begin{matrix} \langle L_e, T^* W_e^i \rangle \\ \langle TL_e, W_e^i \rangle \end{matrix} + \begin{matrix} \langle L_e, T^* T^* W_e^i \rangle \\ \langle TL_e, T^* W_e^i \rangle \\ \langle TTL_e, W_e^i \rangle \end{matrix} + \begin{matrix} \langle L_e, T^* T^* T^* W_e^i \rangle \\ \langle TL_e, T^* T^* W_e^i \rangle \\ \langle TTL_e, T^* W_e^i \rangle \\ \langle TTTL_e, W_e^i \rangle \end{matrix} + \ldots \tag{2.45}$$

This tree illustrates how the propagation can be shifted from radiance to potential or the other way around at any stage in expanding the inner product. The example illustrated in Equation 2.44 can be found in Equation 2.45 by following the lower branch from $\langle L_e, W_e^i \rangle$ to $\langle TTTL_e, W_e^i \rangle$ and beyond. The upper branch describe transport of potential and internal branches use a combination of both radiance and potential transport by exploiting the adjointness of the transport with respect to the inner product.

Some of the solutions to computing flux for a set are more intuitive than others. Although mathematically equivalent it is easier to work with exitant radiance than incident radiance in a transport setting for the simple reason that emitters do not have self-emittance L_e^i. Incident self-emitted radiance is only defined (have a non-zero value) on points directly visible from emitters, which is rather awkward conceptually. However, when working with exitant radiance L_e is defined over the surface of the emitter itself. Similarly, exitant potential only have non-zero values at points visible from the set not the set itself. Thus, it is much more convenient to phrase solutions in terms of exitant radiance and incident potential. In the following, solutions to the global illumination problem will generally be phrased in this way. This leaves two main solutions; one based on transport of exitant radiance using the operator T and the other based on transport of incident potential using the operator T^*.

Finally, algorithms can be conveniently categorised by the way they expand the inner product. Equation 2.45 nicely illustrates this by considering the lower branch, the upper branch and all the inner branches separately.

The lower branch of the tree reaches a solution using transport steps for radiance given an initial potential distribution defining the set. This amounts to finding paths of increasing length from the set back to the emitters that carry non-zero radiance, intuitively this can be thought of as a *gathering* algorithm since it gathers radiance values for the set.

On the other hand the upper branch of the tree reaches a solution using transport steps for potential given an initial distribution of self-emitted radiance. This amounts to finding paths of increasing length from emitting surface points to the set. Intuitively this can be thought of as a *shooting* algorithm since it shoots radiance outwards from the emitting surfaces along paths of non-zero potential towards the set.

The remaining internal branches use a combination of transport steps for both radiance and potential and can thus be thought of as bi-directional or hybrid methods.

2.5 Summary

In this chapter the principles of radiative transfer were outlined. The focus was restricted to three-dimensional environments consisting of closed polyhedra made up of flat convex polygons devoid of participating media.

In this context the radiance and potential equations were outlined. The radiance equation starts at a given set and *gathers* radiance from emitters via paths from the set back to emitters. The potential equation on the other hand *shoots* radiance outwards from emitting surfaces along paths of non-zero potential toward the set. Both transport equations are recursive integral equations and are known as Fredholm equations of the second kind. Typically, this kind of recursive integral equation have no analytical solution and have to be solved numerically.

From those equations four ways of computing flux for an arbitrary set S were derived in terms of an inner product of linear transport operators and an initial distribution of potential and emitted radiance. Of those, transport of exitant radiance with an initial incident potential distribution and transport of incident potential with an initial distribution of exitant emitted radiance were emphasised since they are intuitively simpler to visualise. These were further shown to be mathematically equivalent since incident

and exitant transport with respect to the inner product of radiance and potential are adjoint.

Finally, the methods were categorised as *shooting*, *gathering* or *bi-directional* based on their behaviour in terms of transport.

Chapter 3

Global Illumination Methods

Over the last five decades a vast amount of research has been carried out in the field of global illumination ranging from Appel's hidden line drawings [App67] to light field rendering [GGSC96, LH96], and more recently methods for dynamic environments running at interactive rates [DSDD07, RGKS08, ML09]. Any thesis in this field would be incomplete without a summary of past work, but clearly any such summary will be imperfect due to the amount of work that deserves mention. In the following sections only the most relevant work is discussed and summarised.

A taxonomy for global illumination algorithms is developed and past work is summarised in this context. Emphasis is placed on techniques that offer real-time walkthrough and work well with indoor architectural scenes typical in VR scenarios and exhibiting complex light paths. This is followed by a separate section that discusses the field of global illumination on graphics processing units (GPUs), since the work presented in Chapter 5 relies on concepts from this field.

3.1 A Taxonomy for Global Illumination Algorithms

There are many properties of global illumination algorithms that could be emphasised and used as a basis for a taxonomy. A classification could be based on whether the method is pixel based or finite element based, whether it is stochastic or deterministic, whether it solves the global illumination problem fully or only partly, or whether it is real-time or used for high-quality rendering. Most algorithms, however, fall into more than one of these categories. In this chapter a more fundamental property of the algorithms are used to provide the initial classification; namely whether the algorithm uses *gathering* or *shooting* light transport or a combination of the two.

This division was formalised in Section 2.4, which illustrated that all global illumination algorithms can be described by an inner product of a potential distribution and a radiance distribution. An *initial* known potential distribution defines the set of points and directions for which a measure of flux is required. At the other 'end' an *initial* known distribution of radiance defines emitted flux at surfaces. The goal is to compute the quantity of flux that reaches the set through any number of reflections; integrating all possible paths between the set and the emitting surfaces. The distinction is then based on which of these *initial* distributions transport steps are applied to.

A *gathering* algorithm applies transport steps to emitted radiance given an initial known potential

distribution that describes the set for which the flux is required; this process can be thought of as finding paths of non-zero radiance from the set towards the emitters. Essentially gathering radiance from the environment surrounding the set.

Conversely, a *shooting* algorithm applies transport steps to potential given an initial known radiance distribution that describes the flux at the emitting surfaces; this can be thought of as finding paths of non-zero potential from the emitter towards the set. Essentially shooting radiance outwards towards the set.

There is a third group of algorithms - known as *hybrid*, *multi-pass* or *combined* methods - which use a combination of transport steps for radiance and transport steps for potential. These methods seek to combine the strengths of both approaches while avoiding the weaknesses.

The exposition will show that from an algorithmic standpoint certain modes of light transport, that are difficult to handle with the gathering approach, are more easily handled by the shooting approach. Conversely, there are certain parts of a global illumination technique that greatly benefits from using a gathering approach. The following will clarify the origins of the VLF [SMKY04] approach; its data structures, global illumination simulation technique and rendering methods. It will become apparent that the VLF approach spans all three categories of this classification in its attempt to provide a solution to globally illuminated walkthroughs of architectural scenes.

In the following the Heckbert light transport notation [Hec90] is used to describe the types of transport supported by the techniques. See Appendix A.5 for a description of the notation.

3.2 Gathering Methods

In the following gathering global illumination methods are summarised.

3.2.1 Gathering Radiosity Methods

Radiosity was among the first algorithms to produce globally illuminated renderings. The technique was inspired by research in thermal engineering in the 1950s and was introduced to the graphics community in the mid 1980s [GTGB84, NN85].

It is based on the assumption that diffuse illumination changes slowly across a surface such that it is possible to subdivide each surface into a (relatively small) set of patches for which the flux can be assumed constant, but yet still approximates the illumination across the surface well. An amount of initial flux stored in (some) of these patches is transported to other patches visible to them. Relative visibility is captured by the form-factor between a pair of patches; this is a geometric quantity that gives the fraction of total outgoing flux from one patch reaching another patch. Due to the directional independence of reflected radiance from a diffuse surface, the surface reflectance function f_r can be moved outside the radiance integral Equation 2.22, greatly simplifying the resulting equation by allowing interactions over large solid angles to be performed in a single step.

In an environment comprising N patches, the method explicitly creates a global system of N linear equations to describe energy equilibrium between the patches in the environment. The equilibrium radiosity values are obtained by solving this system of equations.

The radiosity algorithm stores a single radiosity value for each patch because of the directional

independence of radiance from a diffuse surface; thus it only accounts for *diffuse* interactions $L(D)^*E$. However, it supports area light sources intrinsically and exhibits important effects such as soft shadows and colour bleeding. The method is interesting not only historically but also because it is the first method to allow interactive walkthrough of a globally lit environment. This is due to the fact that diffuse reflections are view-independent making it possible to pre-compute a set of textures capturing the diffuse interreflections for use during interactive walkthrough.

In [GTGB84] only unoccluded environments were supported, in [NN85] support for approximate occlusions was added based on vertex visibility, but the complexity of the method restricted its use to simple environments. Gathering radiosity was made practical for more complex scenes by introducing the hemi-cube that allows fast computation of form factors that take into account occlusion [CG85]. In that paper it was also noted that for a static environment it is possible to change lighting conditions and re-use the form-factor matrix to compute a new radiosity solution quickly. However, the method requires computation on the order of $O(N^2)$ in the number of patches because all form-factors are computed before the matrix solution is started. This also means storing on the order of $O(N^2)$ form-factors, which given the memory capacities at the time quickly became overwhelming for even relatively simple environments. The form-factor computations by far dominate the computational cost of the method.

Further work on radiosity is largely restricted to progressive or shooting radiosity algorithms, which are described in Section 3.3.1.

3.2.2 Ray Tracing and Path Tracing

Ray tracing was the first algorithm to produce images with global illumination effects. Also, it is one of the few algorithms that accounts for the complete set of light paths; $L(S|D|G)^*E$. This section summarises early and more recent research in ray based global illumination.

Ray Casting and Direct Lighting

In [App68] Appel presented a novel algorithm capable of generating views of solids with appropriate shadows due to a number of point emitters in an attempt to add chiaroscuro[1] to computer generated images. The most important finding was, that the intensity of a visible point in a novel view of a three-dimensional scene - lit by arbitrarily placed point emitters - could be computed by integrating the visibility from the point to these emitters with respect to the geometry of the environment. This solves the direct lighting problem and accounts for $L(D|G)E$ light paths. The method was greatly hampered by hardware limitations, especially that of display devices.

Recursive Ray Tracing

In Turner Whitted's seminal paper [Whi80] ray casting was extended to include reflections and refractions in addition to shadows by introducing a shading model that *uses global information to calculate intensities*. The fundamental idea was to trace rays from the eye of the observer through a pixel on the image plane of a camera model into the environment, in order to *sample* the intensity of light reaching the eye from that pixel. A tree of such rays is recursively constructed by tracing a ray into the scene

[1] An artistic technique developed during the Renaissance, referring to the use of exaggerated light contrasts in order to create the illusion of volume. (source: Wikipedia, http://en.wikipedia.org/wiki/Chiaroscuro)

and calculating the nearest intersection with a surface, where a shader is executed spawning new rays towards other surfaces and emitters. Finally, the tree is traversed bottom-up and a radiance value is produced for the pixel. The method only supports specular surfaces and point emitters because they can be easily evaluated by a single ray; thus the method accounts for $L(D|G)^?(S)^*E$ light paths. However, the method produces important effects such as (sharp) reflections, (sharp) refraction and (hard) shadows. Also a deterministic adaptive anti-aliasing approach is applied providing sub-pixel accuracy. Diffuse interreflections are replaced by a single global ambient term. The method is elegant, supports polygonal, parametric, instanced and procedural geometry as well as constructive solid geometry [Rot82] and is easily implemented. As was the case for radiosity, the visibility computations by far dominate the computational cost; they can account for as much as 95% depending on the environment. The computational cost of the technique generally restricts it to off-line use.

Distribution Ray Tracing

Some of the fundamental weaknesses of the method were solved in [CPC84] by stochastically sampling integrals over time, pixel area, lens area and reflectance. An important aspect is that the sampling does not introduce extra rays beyond those used for spatial oversampling required by the Nyquist-Shannon sampling theorem (addressed specifically in [Coo86]). Since the integrals are nested, rays that are spawned at each step lead to a combinatorial explosion that will quickly overwhelm the computational resources available. The ray tracing approach is seen as a point sampling method subject to aliasing that, however, can be filtered by distributing rays over the dimension defined by the integral at hand. It is shown that the apparent aliasing in many of the earlier ray based methods is a consequence of using regularly spaced samples, and that frequencies above the Nyquist limit, if sampled non-uniformly, are subject to noise that is much more visually acceptable than aliasing artifacts. Sampling in time produces *motion blur*, sampling the lens produces *depth of field*, sampling the solid angle subtended by an area light source produces *soft shadows* and sampling the reflectance function produces *gloss* and *translucency*. Still the method is not suitable for environments with diffuse surfaces. The method assumes that the reflectance functions can be importance sampled and the most significant contributions can be sampled with only a few samples. With a diffuse BRDF, all directions contribute equally[2] to the integral and very many samples are required to avoid noise. The method accounts for $L(D)^?(S|G)^*E$ light paths. Also, coherence between rays is not exploited; the algorithm starts from scratch for each pixel. The algorithm is not real-time and is mainly used for high-quality rendering.

Path Tracing

The distribution ray tracing algorithm naturally led to the study of Monte Carlo methods for solving the global illumination problem, which dominated research in the field in the 1990s. In [Kaj86] Kajiya introduced path tracing, which differs from distribution ray tracing in some significant ways. Instead of using a ray tree the method estimates the radiance of a pixel with a number of *paths*. The motivation for this is that first generation rays and light rays contribute most to the pixel integral in terms of variance.

[2]Cosine weighted sampling can be used to sample the area near the equator less densely, since it is less likely to contribute much to the integral.

Conventional ray tracing spends the vast bulk of computation on a widening ray tree that increasingly contribute less to the variance of the image; later generations of rays are overrepresented. By employing ray paths instead each generation has the same number of rays. The importance of sampling densely in interesting parts of the domain and sparsely in parts where the integrand is nearly constant was also noted, and a number of criteria for selecting such samples were presented (although not used to compute the results). The method accounts for all possible light interactions $L(S|D|G)^*E$, although it has difficulties handling caustics adequately. In [AK90] Arvo and Kirk introduced *russian roulette* to truncate paths in an unbiased manner. Shirley [Shi90b] proposed to send a single shadow ray to a random point chosen from all the light source surfaces, rather than a random point on each. The fundamental problem with Monte Carlo integration is that the variance of an estimate approximated by N samples is proportional to $1/N$, thus the standard deviation is proportional to $1/\sqrt{N}$, which means that in order to *halve* the error N must be *quadrupled*. Often traditional path tracing requires on the order of hundreds or thousands of samples per pixel to compute a noise free image requiring hours to render a frame.

Fast Ray Tracing for Global Illumination

Recently ray tracing performance has been improved by several orders of magnitude [Wal04, RSH05, FCM09]. Parker et. al. [PMS+99] introduced a Whitted-style ray tracer running interactively on a high-end parallel architecture with soft shadows and an improved method for representing the ambient term. However, the additions of soft shadows and diffuse lighting is achieved in an ad-hoc manner and is not physically based, also only a limited number of dynamic objects were supported. [RSH00, WBS03] introduce systems running on large clusters supporting scenes with many dynamic objects requiring updates to acceleration structures, however the lighting models used were simplistic. In [WKB+02] many global illumination effects such as soft shadows, colour bleeding and caustics were supported at a couple of frames per second on a cluster with up to 24 CPUs. However, due to the low number of samples per pixel (20-30) higher order illumination effects were ignored and direct caustics were simulated with only 500-1500 photons per light source. In [WDB+06] frame rates were improved and support for BTFs and HDR environment lighting was added, still the low ray budget for each pixel preclude representing high order global illumination. Nevertheless, the results are impressive considering that many global illumination effects are accounted for and computed nearly from scratch for each frame.

3.2.3 Caching Methods for Global Illumination

One of the main limitations of ray based methods for global illumination is that they often recompute the radiance integral from scratch for each image sample. No use is made of the possible coherence between nearby samples. The section covers techniques that address this issue.

(Ir)radiance caching

One of the first systems using a caching approach to speed up a Monte Carlo technique was devised by Ward et. al. [WRC88]. The method separates the illumination computation at a surface intersection into a direct component, specular reflection and an indirect component. The first two are solved in the traditional way (see [Kaj86]). The fact that indirect illumination is slowly changing is exploited and

because diffuse illumination is expensive to compute it is sparsely sampled. The indirect component is either reconstructed from existing samples nearby, or, if no such samples are available, a new sample is computed and stored in an octree for later reuse. *Irradiance gradients* [WH92] are used to assign a weight to a sample based on the inverse of the estimated error of that sample. The error estimate is proportional to distance and angle between the normals. Only samples with an error less than a user-specified threshold is used to reconstruct a sample. However, the system only accelerates computation of irradiance on perfectly diffuse surfaces, low-frequency glossy surfaces for example are not supported leaving the Monte Carlo technique to sample these at a high computational cost. With some complex geometries the error heuristic is not appropriate and artifacts may result. Also, searching the octree for appropriate candidate samples can be expensive.

The technique has been extended [KGPB05] to add support for low-frequency glossy BRDFs by storing view-dependent information using (hemi)spherical harmonics. Greger et. al. [GSHG98] have proposed a similar method that pre-compute and store a volumetric approximation to the irradiance function, that can be used for semi-dynamic environments. However, the method would require a very dense volume to support accurate irradiance at surfaces.

Ray caching

Since path and ray based global illumination methods may require hundreds of samples per pixel, thus requiring millions of rays per frame, interaction is often not possible. However, decoupling the rendering from the radiance sampling process can facilitate interactive rates. Consequently, the sampling and display processes run asynchronously requiring an intermediate caching method for storing the global illumination samples and making them available to the display process. An image is then formed by reprojecting and/or interpolating available cached samples. These methods are useful when the cost of computing new samples is sufficiently higher than the cost of the caching and interpolation step.

A system based on this approach is Ward. et. al.'s "Holodeck Ray Cache" [Lar98]. Global illumination samples computed with the *Radiance* software [War94] are stored together with their ray in a large data structure stored on disk and cached in memory. Coherent rays are stored together as beams for efficient access. When forming an image the display process requests beams that pass near the eye point within the view frustum, these are partly read from the data structure and re-computed by ray tracing processes. Any re-computed beams are stored in the data structure for later reuse. The image is formed by reprojecting appropriate rays to get correct image positions for the samples and using a quadtree representation to fill the screen. The method suffers from disturbing artifacts near silhouette edges caused by occlusion errors due to rays not passing exactly through the viewpoint but near it. The approach, however, supports full global illumination with arbitrary BRDFs since the illumination is stored in a view-dependent data structure. The method was extended in [WS99] where OpenGL rendering of the original geometry was used to improve the appearance of silhouette edges and local dynamic objects was supported by deriving local lighting from the Holodeck environment as in [WAL+97]. An interesting observation is that the Holodeck is essentially lazily building a 4D light field.

Simmons and Séquin's "Tapestry" [SS00] is an extension to the Holodeck; it constructs an im-

age using mesh reconstruction building a 2D Delaunay triangulation from projected samples which is touched up during viewpoint motion. This method also lacks sharp edges at discontinuities but because of the 3D mesh representation samples have a longer cache life.

Render cache

Walter et. al.'s "Render cache" [WDP99] is very similar to the Holodeck approach. It projects cached samples to the image plane and uses depth culling rejecting samples significantly different than the average of a small local neighbourhood. A 3x3 weighted filter is used to fill in pixels to which no samples were projected. In order to guide the sampling process to compute samples that yield maximum benefit, a priority image is generated using a heuristic that assigns a high priority to old samples, pixels that have no samples associated and pixels on dynamic objects. Error diffusion dithering is used to select samples to compute from the priority image providing both a good spatial distribution and concentration of samples in high priority regions. Rigid body transformations applied to objects can also be applied to samples lying on them in the projection step to improve tracking of such samples. The cache used is relatively small and samples are replaced in a fast approximate LRU fashion. This approach can achieve interactive frame rates with ray tracing for modest image resolutions, but suffers from artifacts especially during motion. It is also sensitive to noise in computed samples (i.e. when using Monte Carlo techniques) causing samples to be discarded prematurely. The technique was extended in [WDG02] adding improved filtering capable of filling larger holes, predictive sampling, improved memory access coherence based on tiling, an improved sample eviction scheme removing stale samples faster and SIMD optimisations.

Bala et. al.'s "Edge and Point Images" [BWG03] is an extension to the Render cache that addresses the problem that complex illumination can not be interpolated across discontinuities without causing blurring. The method computes sparse samples with the Render cache and interactively detects image plane discontinuities such as silhouettes and shadow edges. Pixels are reconstructed by interpolating nearby samples that *are not* separated by a discontinuity edge.

Shading cache

A similar approach is Tole et. al.'s "Shading cache" [TPWG02]. Instead of caching samples in image space it is an object space approach that caches global illumination samples at the vertices of a hierarchical subdivision mesh which can be rendered with graphics hardware. Using a priority map the mesh is lazily refined by selecting locations in the image plane that either requires more accuracy or have stale samples. Subdivided patches no longer in view are deleted to reduce the mesh complexity and maintain interactivity. Dynamic objects are given high priority in order to resolve shading errors due to motion. This method avoids the edge artifacts plaguing the Holodeck and Render cache but is still subject to potentially disturbing artifacts due to stale samples. Also because object motion is instantaneous using graphics hardware the latency of the shading can cause effects such as shadows not being able "to keep up".

A similar object space approach [SHSS00] uses a set of "corrective textures" to update global illumination effects on objects rendered with graphics hardware. Textures are projected onto composite

non-convex objects using point projection from the current viewpoint, this, however, causes artifacts when the depth range of objects is large. The projection during viewpoint motion is also subject to texture flow.

3.2.4 Light Fields

An alternative representation of a global illumination solution is to capture and store the 5D *plenoptic function* [MB95] for an environment. The function represents flow of light (radiance L) at any point in space (x,y,z) in any direction (θ,ϕ) for any given time t and any given wavelength λ. We assume time independence since we are working with static scenes, and also assume wavelength independence. In order to sample this function $L(x,y,z,\theta,\phi)$ a discretisation of (x,y,z,θ,ϕ) must be chosen and the radiance computed for each sample in the discrete representation. Novel views can be generated from a light field by resampling a 2D slice from the 5D data structure. Rendering time is independent of the scene geometry and illumination complexity and interactive walkthroughs of a static scene can be achieved.

The first techniques using this approach was Gortler et. al.'s "Lumigraph" [GGSC96] and Levoy and Hanrahan's "Light Field" [LH96]. They both assume a single bounded object so that the plenoptic function can be reduced to 4D. The parameterisation is a box bounding the object consisting of six light slabs. A light slab consists of a pair of 2D planes (u,v) and (s,t), each oriented line between the planes (u,v,s,t) stores a single radiance value $L(u,v,s,t)$. The methods can build such a representation from both synthetic objects and real objects by capturing images and sampling the 4D data structure from them. Rendering from the representation can achieve interactive rates. The data structure can be large (gigabytes) but compression rates of 200:1 are possible. The main limitation of these techniques is the restriction to a single bounded object. Also, the methods show noticeable artifacts when the camera crosses the boundary between two light slabs due to non-uniformity of the line density.

Ihm et. al. [IPL97] used a different parameterisation for a 4D light field based on a *positional* sphere parameterised by (θ_p,ϕ_p) enclosing the object. Each point (θ_p,ϕ_p) is the origin of a *directional* sphere parameterised by (θ_d,ϕ_d), yielding a global line density based on $(\theta_p,\phi_p,\theta_d,\phi_d)$. Rendering is performed in object space by rendering the triangles of the polyhedron that discretised the positional sphere. The vertices are assigned a radiance value resampled from the directional sphere and graphics hardware performs interpolation. The spherical light field is compressed using wavelets.

Surface light fields were introduced by Miller [MRP98] and extended by Azuma [Azu99] and Wood [WAA+00]. The surface light field uses a 4D parameterisation, where the first two parameters (u,v) describe a point on the surface of the object and the last two (s,t) define the orientation relative to the tangent coordinate frame at the point, essentially defining a direction leaving the surface point. Radiance is sampled for a discretisation of this parameterisation from real or synthetic objects and rendering can be done at interactive rates. The methods are still limited to single objects and the rendering complexity is no longer independent of the geometric complexity of the object.

Uniform Light Field Parameterisations

Camahort et. al. [CLF98] introduced two additional parameterisations; the two-sphere parameterisation (2SP) and the direction-point parameterisation (DPP) and generalised existing parameterisations evaluating them in terms of features and rendering artifacts [CF99, Cam01]. The DPP parameterisation indexes the light field first by direction (θ, ϕ) then by position (u, v) on a plane orthogonal to the direction. DPP-based models expect the light-field data to be stored and accessed in that order. Discretisation uses a (nearly) uniform tessellation of the sphere, then for each direction a uniform grid is imposed on an orthogonal plane. It was noted [Cam01] that the disparity artifacts suffered by the light slab (2PP) parameterisation is caused by the non-uniformity of the line density and is intrinsic. As the angle β between the line connecting the two planes and the plane normal increases, the line density decreases with $\cos^2\beta$, because the planes are uniformly subdivided. The problem is that the parameterisation is not statistically uniform in its continuous form, and thus cannot be discretised simply by sampling the parameters uniformly. This can only be fixed by using a non-uniform subdivision of the planes that account for the $\cos^2\beta$ directional discrepancy. Such non-uniform representations are unwieldy and impractical to implement.

The continuous DPP parameterisation on the other hand is provably statistically uniform [Cam01] and a uniform discretisation can be obtained by uniformly sampling in (θ, ϕ) and (u, v). Further, a 5D parameterisation can be obtained by sampling along the lines originating on the orthogonal plane. If the free-space simplification is assumed the sampling can be limited to surface points only without loss of generality. In spirit this is similar to the *layered light field* [LR98], which stores view-dependent information in a collection of *layered depth images* [SGwHS98] corresponding to directions uniformly distributed on the unit sphere. In this thesis a similar 4.5D DPP parameterisation have been chosen to form the basis for the main data structure introduced in Chapter 4.

3.2.5 Summary

Of the methods discussed in this section only light fields and path tracing account for all possible light paths. The caching methods mostly use path tracing as their underlying 'engine' and can be seen as optimisations to the method.

The light field methods are attractive because they can produce images in constant time, but they only apply to single objects and thus can not be applied to the type of environments targeted in this thesis. However, in Chapter 4 the idea of using a view-independent data structure will be utilised and extended upon so as to support scenes with many objects. Path tracing is attractive due to its per screen pixel $O(\log N)$ complexity but on the other hand it converges slowly; the convergence rate with N samples is $O(1/\sqrt{N})$. It works best with mostly specular scenes with small emitters and even when a caching scheme is used it will struggle to maintain interactive rates in an indoor mostly diffuse scene with large area emitters and complex light paths. Small secondary emitters are also difficult to handle with eye based approaches [WRC88]. Also, it is difficult to *guarantee* a certain frame rate since the computation time is tied to the *visible* geometric and illumination complexity and this can vary greatly between frames.

3.3 Shooting Methods

In the following Section shooting global illumination methods are summarised. This will cover progressive radiosity methods and Monte Carlo based techniques, which have traditionally been applied to solve the integrals in popular shooting algorithms such as *Photon Mapping*. Global line methods, which are related to the methods developed in this thesis, are also described here.

3.3.1 Shooting Radiosity

In [CCWG88] Cohen et. al. introduced the first shooting radiosity technique. The algorithm exploits the fact that patches which radiate the most energy will typically have the greatest effect on the illumination of the environment. Reordering the algorithm to shoot energy from one patch updating all the others and displaying the result of this update immediately, will provide an approximate solution much earlier than previous methods. It was noted how this approximates the way light propagates through an environment starting at the emitters. This also eliminates the need for pre-computing all the n^2 form-factors and storing them before the radiosity process can begin. The form-factors are computed on-the-fly with the hemi-cube [CG85] as needed. In the early stages of the solution an ambient term based on the current estimate of radiosities and the reflectivity of the environment is added to account for unshot radiosity. The algorithm produces a 'useful' estimate in time linear to the number of patches. Wallace et. al. [WEH89] used ray tracing for vertex form-factor calculation to avoid aliasing inherent in the hemi-cube approach. Chen [Che90] extended the method to support dynamic changes to the scene by incrementally updating the radiosity values after a scene change.

Hierarchical radiosity [HSA91] reduces the number of needed interactions to $O(N)$ by observing that form-factors between patches far apart can be computed at a coarse level of detail with error similar to computing it at a finer level of detail. The reason is that the magnitude of the form-factor is proportional to $1/r^2$ (r=distance), so the mutual effect of well separated groups can be approximated with a single interaction. The method starts with few large patches and subdivides them into a hierarchical representation where links are formed at appropriate levels of detail when the error to the form-factor falls below a preset threshold. Occlusion is exploited to prune out unnecessary refinement. During radiosity propagation a *push-pull* technique is applied to ensure that levels in the hierarchy that are not directly affected by the update remain consistent. This was extended upon in [SAG94] which added clustering allowing grouping of small patches into larger ones and thus creating links at a lower level of detail than the initial set of patches.

Monte Carlo techniques can be applied to solve the radiosity problem. A solution to a Neumann series expansion can be estimated directly by stochastically sampling paths from emitters. This assumes a particle model of light and random walks are formed carrying quanta of light – or *photons* – reflecting off surfaces until they are probabilistically absorbed. With this technique form-factors are not computed explicitly and the radiosity is estimated by the density of photon hits. More formally the random walk is a sequence of states given a starting state and a transition probability function $T(s \rightarrow s')$. From a current state the next state is chosen by sampling T. The random walk will have a finite number of steps

if $\int T(s \rightarrow s')ds' < 1$[3] and convergence is assured. Shirley [Shi91] gave an algorithm approximating radiosity with $O(N)$ paths for an environment with N zones. Keller [Kel96] used deterministic Quasi-Monte Carlo techniques based on low discrepancy sequences yielding superior performance compared to random sampling. In [BNN+98] Monte Carlo and hierarchical radiosity was combined. A weakness of these early Monte Carlo methods for radiosity is that they depend on an external technique (such as discontinuity meshing [HW91]) for adaptively refining the mesh in areas of high-frequency gradients. Also, they are not easily adapted to support non-diffuse illumination effects. Furthermore, due to the direct visualisation of the photon density results are visibly noisy, unless very many samples are used.

3.3.2 Non-diffuse Shooting Radiosity

Immel et. al. proposed extending a radiosity system with view-independent information for non-diffuse surfaces [ICG86]. Non-diffuse patches store directional radiance information in a "global cube", which is also used to solve for visibility. The "global cube" is essentially a full cube surrounding a vertex aligned to the global coordinate system allowing for fast lookup of reciprocal cells along a ray connecting two vertices. A progressive solution is employed shooting radiance outwards from emitters initially, then from secondary emitters and so on until convergence. Rendering of non-diffuse surfaces applies interpolated view-dependent vertex intensities extracted from the "global cube" from which pixel intensities are bilinearly interpolated. The method unfortunately requires meshing and inherits all the aliasing problems inherent in using the hemi-cube for visibility. Also, surfaces with mirror reflectance alias when rendered directly from vertex intensities unless they are very densely subdivided. Further, using a depth buffer approach solving for visibility from each surface results in $O(N^2)$ complexity.

Historically, the parallel to Kajiya's paper [Kaj86] published at the same time is interesting; where Kajiya extends ray tracing with diffuse reflection, Immel et. al. extends radiosity with non-diffuse reflection.

3.3.3 Particle Tracing

Inspired by the Monte Carlo approaches to solving the radiosity problem, the technique was soon found to be applicable to solving the full global illumination problem supporting not only diffuse interreflections. However, very few techniques exist which are purely shooting methods due to the noise incurred by directly visualising photons. Dutré et. al. [DLW93] presented a particle tracing algorithm using Monte Carlo techniques to form random walks from the emitters to the pixels on a view plane. At each surface intersection of the random walk rays were traced through the pixels on the image plane; representing direct illumination, indirect illumination reflected once, twice and so on. In order to limit noise object-space gaussian filtering was added *smearing* incident power over the surface of the object. Although this technique samples all possible paths from emitter to receiver, high-frequency effects such as specular reflections and sharp shadows are difficult to reproduce and the technique is view dependent and inappropriate for interactive walkthrough.

In Section 3.4.2 techniques are presented that also shoot particles from the emitters but use a sep-

[3] Global illumination problems are normally subcritical but some assumptions may violate this e.g. a closed environment consisting of perfect mirrors, in such cases external termination criteria are used [AK90, PM92].

arate pass to reconstruct illumination through the pixels on the image plane. This is based on the observation that the eye-scene interactions are more appropriately sampled from the eye since the set of rays potentially contributing to a pixel (W_e) is known in advance. This avoids shooting rays towards the eye for which $W_e = 0$ and thus will not contribute towards the illumination of the pixel. Also, with such methods it is possible to store flux densities for reuse during interactive walkthrough.

3.3.4 Local and Global Lines

Monte Carlo techniques can be characterised based on the line density they use for sampling ray space. Traditionally, *local* line densities have been used which sample a (hemi)sphere aligned to a coordinate system formed by the normal and surface tangents at a point on a surface. A cosine distribution is normally used for such sampling taking into account that rays near the equator contribute proportionally less. An alternative to this is a *global* line density that is independent of any surface orientation. A pre-defined set of global lines is used for all surfaces in the environment regardless of orientation. It can be shown [CS98] that if a uniform density of global lines is used a cosine distributed line density is imposed on each surface. The benefit is that energy transfer can be performed bidirectionally for each pair of mutually visible intersection points along a global line as given by the *exchange list* for a line (see Section 2.2.3). Note also that data structures for such uniform line densities exist as described in Section 3.2.4.

In Buckalew and Fussel's "Illumination Networks" [BF89] a global line density was used to find a set of bidirectional links between surfaces in a pre-processing step. Then light was emitted from the light sources and propagated outwards on the global links arriving at surfaces where the light is accumulated in *in-buffers*. The light available in the in-buffers is passed through a reflectance function and accumulated in *out-buffers* before it is propagated outwards again in a recursive manner. When no light is available in the in-buffers, images can be rendered directly by sampling and interpolating radiance available in the out-buffers corresponding to rays emanating from the eye. The pre-process exploits coherence by using an incremental calculation of ray-object intersections by projecting each object to a plane orthogonal to a given slope and applying a scan-conversion algorithm. Clearly, this only works for planar objects. Also, the global line density used is non-uniform and weights have to be used to correct for this during light distribution, but the density remains biased possibly causing undersampling in certain areas. Moreover, too few details about the data structure are given to be able to judge whether it is appropriate for compression.

Global line densities were used in [Sbe93, Pel95] to compute form-factors with a Monte Carlo approach. This was based on interpreting the form-factor as the probability that a line exiting one patch lands on another. Integral geometry was used to show that given a global uniform line density the form-factor F_{ij} is given by $\frac{r_{ij}}{r_i}$, where r_i is the number of lines intersecting i and r_{ij} is the number of lines intersecting both i and j. Intuitively, much work is wasted if lines are cast from each patch outwards in order to compute form-factors for a single patch to all others since only the nearest intersection is used and all others discarded. Rather, all intersections along a global line can be sorted and used to form pairs of visible patches along the line, which can in turn be used to derive the form-factors. It was shown that

an order of magnitude improvement in efficiency over the hemi-cube could be achieved.

Neumann introduced the ”Transillumination Radiosity Method” [Neu95], which was the first solution to global illumination using a uniform global line density. The method selects a number of random points on the unit sphere that are the direction vectors for the global lines. It was, however, noted that obtaining directions from subdivision of some (semi)regular polyhedron would result in lower variance. Orthogonal to each global direction a gridded plane covering the entire scene is stored. Scene patches are orthogonally projected to this ”transillumination plane” and the patch identifier and depth is stored at the grid points in depth sorted order. This effectively forms an *exchange list* for each grid point along the global direction, which can be used for pairwise bi-directional energy transport. The uniform global density of lines implicitly accounts for the cosine term in energy transport. Coherence is exploited on the transillumination plane when projecting the patches by using a rasterisation-like algorithm making the method efficient. The method was only realised for the perfectly diffuse case, but it was noted that it could be extended to the non-diffuse case. The method works well in scenes with large self-emitting patches or sky illumination. It was extended in [SKTNB97] using a continuous inner integral for surface interactions along the transillumination direction and a discrete outer directional integral evaluated using quasi-Monte Carlo techniques.

In [SPP95] a random uniform global line density is used for radiosity. Some interesting observations were made about global line methods, namely, that under certain constraints[4] the number of objects in the scene can be increased without having to increase the number of global lines. Also, the relative efficiency over local methods is proportional to the average number of intersections a global line makes with the scene. Finally, the efficiency is dependent on having large area sources – a possible solution was proposed using a non-global technique to ”kick-start” the global method by converting surfaces visible to sources into secondary emitters [CMS98]. The approach is progressive, the work is divided into a number of subproblems which are solved separately, then merged making possible a parallel solution where many workstations share the work. In [CS98] various global line densities are compared in terms of error.

Szirmay-Kalos [SK98] extended the transillumination method to the non-diffuse case. A similar bundling of parallel rays was used but the method progressively formed parallel random walks from emitters to the eye. A direction is selected quasi-randomly and irradiance for all surfaces is computed in this direction for all the parallel rays. This irradiance is then reflected into a new direction and solved for all parallel rays in this direction, repeated a few times and finally reflected into the eye. A number of such random walks are averaged on the image plane accounting for all possible light paths. Many methods are discussed for efficiently solving the radiance transport along a single direction. These include both continuous and discrete methods and methods requiring initial sorting. Pre-sorting in the transillumination direction and using a modified scan-conversion algorithm yields $O(NlogN)$ computational complexity. However, the method does not store intermediate results and would need to recompute the random walks when the camera moves, making the method unsuitable for interactive walkthrough.

[4] Some fixed minimum area for a surface is assumed.

In [SKP98] a technique similar to depth-peeling is accelerated using z-buffer hardware for solving transport in a single direction. In [SKSMT00] the method was extended to support point and small light sources by adding a pre-processing step using the hemi-cube approach to shoot out radiance from the light source. Also, it was noted that finite-element methods, which can differentiate first and other bounces, can treat the direct illumination separately and use a first shot approach that converts non-emitting surfaces into large secondary emitters using a gather approach.

3.3.5 Summary

Shooting methods can be very competitive in certain environments. Early shooting radiosity methods inspired very efficient Monte Carlo random walk solutions to form-factor computation and radiosity. Efficiency can be improved by using global line densities that exploit all intersections along a line. Further efficiency can be gained by using a global line density with bundles of parallel rays by exploiting coherence between nearby parallel rays. Global line densities with bundles of parallel rays closely match the data structures (DPP) described in Section 3.2.4 and in Chapter 4 this will be exploited. In Chapter 5 efficient solutions to solving radiance transport in a transillumination direction will be developed making use of graphics hardware. This allows for very effective ways of computing light transport for many rays simultaneously and combined with an appropriate data structure full global illumination can be pre-computed and stored allowing interactive walkthrough by rendering from this data structure.

3.4 Combined Methods

This section summarises methods that combine transport of potential and radiance.

3.4.1 Bi-directional Path Tracing

Bi-directional path tracing was developed independently by Lafortune and Willems [LW93, LW94a], and Veach and Guibas [VG94, VG95, VG97]. The basic idea is to assign equal importance to paths starting at the eye and paths starting at emitters. Monte Carlo random walks are performed simultaneously from the eye and a selected light and are merged. Lafortune and Willems join each intersection point on the eye path with each intersection point on the light path, whereas Veach and Guibas produce a single path by concatenating an eye path and a light path. The ”Metropolis” approach [VG97] mutates existing paths reusing significant parts thereof thus gaining efficiency. It also concentrates computational effort on paths with great importance to the final image without introducing bias. These methods are mainly used for high quality rendering in scenes with particularly difficult light paths. However, with rendering times of hours they are not applicable to interactive walkthrough.

3.4.2 Hybrid and Multi-Pass Methods

A number of methods have tried to combine the strengths of gathering with the strengths of shooting simultaneously avoiding their weaknesses. One group of methods attempt to combine radiosity and ray tracing, whereas others discard radiosity altogether and combine a light tracing pass with an eye tracing pass using the information from the light tracing pass to render images.

A Synthesis of Radiosity and Ray Tracing

Wallace et. al. [WCG87] proposed a two-pass solution to the rendering equation. Light interreflections affecting diffuse surfaces are solved independently by a modified radiosity technique in a pre-pass[5]. The radiosity approach is extended to account for mirror reflections from planar surfaces by calculating extended form-factors for specular to diffuse transfers. Extended form-factors via planar perfect mirrors can be computed using the traditional z-buffer algorithm by rendering the environment, reflected about the mirror plane, onto the "back" of the mirror plane. The mirror acts as a window into a "virtual world" which is a reflected version of the environment. In the viewing stage specular reflections seen directly by the eye are solved recursively using a reflection frustum. The diffuse component is calculated by interpolation from the vertex intensities determined in the pre-pass. The algorithm, however, is not appropriate for scenes with more than a few mirrors as the "virtual world" can become very complex after several recursive reflections. Also, it requires meshing for the radiosity pre-pass. Sillion and Puech [SP89] lifted the restriction to planar mirrors by computing the extended form factors using ray tracing.

Shao et. al. [SPL88] extend upon Immel's approach [ICG86] (see Section 3.3.2) using a similar approach also storing a view-independent radiance distribution for non-diffuse surfaces. The computational complexity is reduced by iteratively estimating delta form-factors for non-diffuse surfaces from an increasingly accurate radiance distribution using a radiosity solution as a starting point. Also, a final ray tracing pass is used for image generation, this however, is relatively fast since no shadow rays are needed and only specular geometry needs to be ray traced. It effectively reconstructs geometry seen via one or more specular reflections, which is inadequately represented by a discrete representation when the reflectance is highly specular $\rho_s \approx 1$. The method requires computing *all* form-factors and storing them on disk before the iteration scheme can commence making it impractical for complex scenes.

So far the methods rely on using the traditional hemi-cube along with z-buffering to resolve form-factors. This is computationally expensive on the order $O(N^2)$ and is possible for only simple scenes. Furthermore, appropriate meshing of the scene in order to capture high gradient illumination adequately can be difficult. Shirley [Shi90b] uses a light tracing pass to account for LS^+D paths stored on diffuse surfaces in textures and ray traced progressive radiosity with mirror reflections for $(L|D)S^*D$ paths, adding a final pass removing all direct lighting LD and direct specular lighting LS^+D from the maps. Images are generated using distribution ray tracing for calculating direct lighting and using the radiosity and caustics maps for the indirect lighting. Chen at. al. [CRMT91] extended the technique using Monte Carlo path tracing for eye passes reducing artifacts due to the radiosity meshing. However, the approach requires hundreds of rays per pixel to avoid noise and is very computationally expensive. Heckbert [Hec90] introduced an alternative technique similar to Shirley's using adaptive radiosity textures or *rexes* to avoid the tesselation required by the radiosity step and decouple illumination from geometry. The adaptive nature of the textures allows for adequate sampling of high gradient illumination. However, when nodes are split light rays are shot for each child node discarding the information in the parent node wasting computational resources.

[5] An in-depth discussion about the implementation and theory behind this pre-pass is presented in [RT90].

Sillion et. al. [SAWG91] added arbitrary BRDFs to a two-pass approach that uses a ray traced progressive radiosity approach with surfaces storing directional intensity information using spherical harmonics. Directional diffuse transfers are sampled at receiving vertices requiring adaptive meshing of the scene. A final ray tracing pass evaluates $ES^*(D|G)$ paths looking up the directional distribution when a directional or diffuse surface is struck by the eye ray.

Photon Mapping

Although some of the multi-pass techniques using radiosity were quite powerful producing full $L(S|G|D)^*E$ global illumination, the tesselation required by the radiosity pass introduces artifacts and limits the kind of objects that can be used with the technique. Arvo [Arv86] proposed using illumination maps generated by a light tracing pass to assist a ray tracer in a second pass thus decoupling illumination from the geometry. But this approach was limited to point light sources and simple reflectance distributions, also caustics are only formed on ideal diffuse surfaces. In [PM92] it was shown that Monte Carlo particle tracing can adequately simulate even perfectly diffuse reflections if the number of particles is high enough. Photon paths in non-participating mediums can be evaluated using ray tracing with russian roulette termination to shorten paths and avoid bias. The benefit is that Monte Carlo techniques can easily be used with analytical surfaces, complex surface reflectances and complex light sources. Texture maps were used for accumulating photon weights on diffuse surfaces and a final ray tracing pass was used to generate images using the textures. Only ideal specular or perfectly diffuse surfaces were supported in the eye pass though. Jensen and Christensen [JC95] extended this technique using irradiance maps and high resolution caustic maps for diffuse surfaces and *photon maps* for complex objects where texture maps are inappropriate. The photon map is a kd-tree storing all photon interactions within the environment, photons which are part of a caustic are flagged. When estimating flux at a surface point a sphere positioned at the point is extended until it contains n photons, a disc is used as an estimate of the area holding those photons. Simple rejection rules are used to avoid using the wrong photons. During rendering Monte Carlo path tracing is used in conjunction with the photon and irradiance maps. The direct illumination is determined by sampling the light sources, caustics are visualised directly from the caustics maps, secondary diffuse reflections are fetched directly from the irradiance maps and non-diffuse reflections are treated using standard Monte Carlo path tracing. However, the path tracing step causes long rendering times and the method is inappropriate for interactive walkthrough. The method was extended in [Jen96] where a high resolution photon map is used to store caustics and a lower resolution global photon map estimates the general flux within the scene. Shadow photons are used to accelerate the visualisation pass by using them to limit the number of shadow rays necessary to compute the direct illumination. The indirect illumination is computed using the global photon map and the BRDF to generate optimised sampling directions used with irradiance gradients [WH92]. Specular and glossy reflections are computed with Monte Carlo path tracing. Although this approach greatly improves rendering times they are still too high for interactive walkthrough.

Density Estimation

Density estimation schemes [SWH+95, WHSG97] are similar to photon mapping schemes in that they share the particle tracing pass. Density estimation, however, use the particle distribution to build a display mesh that capture global illumination effects on diffuse surfaces. The display mesh thus adapts to high gradients in the illumination such as shadow boundaries and can be displayed at interactive rates. However, expensive rendering techniques must be used to capture non-diffuse appearance during visualisation.

In [GDW00] hierarchical radiosity is combined with particle tracing to compute a view-independent solution displayed using density estimation techniques. The hierarchical nature of the light transport step increases efficiency, non-diffuse effects are still elusive and the technique resorts to the *Render Cache* [WDP99] (see Section 3.2.3) for interactive walkthrough.

3.4.3 Summary

Many hybrid and multi-pass techniques successfully exploit the orthogonal advantages of using shooting and gathering. The most successful techniques achieve full global illumination $L(S|D|G)^*E$. Shooting is generally used in a pre-pass and gathering is used during visualisation to collect illumination incident on the eye trough the pixels making up the image. The shooting pass can be estimated by using radiosity extended to include non-diffuse effects, but such techniques are limited by the need to tesselate the environment into patches. Other techniques use particle tracing and storage decoupled from the geometry such as irradiance maps and photon maps. However, most techniques reconstruct at least some of the illumination paths during visualisation due to the fact that representing highly specular illumination in a resolution adequate for direct visualisation is very difficult.

In Chapter 4 the irradiance map will be used and reconstruction during visualisation will be exploited to improve the appearance of specular objects during rendering from the VLF.

3.5 Global Illumination on the GPU

Over the last decade a number of global illumination algorithms have been mapped to the GPU and more recently methods have been developed specifically for the GPU. Historically, however, GPU acceleration can be traced back further. Cohen et. al. [CGIB86] sped up a radiosity algorithm using the GPU to calculate form-factors by rasterising a hemi-cube. Also, Keller [Kel97] used the GPU to simulate radiosity by computing a number of indirect lights using a quasi-random walk and rendering them with shadow-mapping accumulating the results. In a similar vein Stürzlinger and Bastos [SB97] used the GPU to splat a photon mapping solution to screen-space enabling also non-diffuse BRDFs. Despite these early efforts the field was only established as a separate GI niche in the mid 00's when the typical GPU matured to the point where it was programmable, had full IEEE compliance for data types such as floating point numbers and integers and where stable cross platform development tools and languages were available. At this point it became tractable to implement more "exotic" general purpose (GP-GPU) algorithms that were not just exploiting the high performance when rasterising or lighting with the fixed-function pipeline. See Chapter 5 for a more detailed examination of the GPU architecture and capabilities.

3.5.1 GPU Radiosity Methods

Keller's *instant radiosity* [Kel97] method is a GPU friendly method for computing radiosity. It is based on performing quasi-random walks from the light sources and depositing *virtual point lights* (VPLs) in the scene. Lighting the scene with shadow mapped VPLs provides a fast radiosity solution. It maps directly to GPUs since the bulk of the work is rendering shadow maps and performing per-pixel lighting computations for the VPLs. This is also the main problem of the algorithm. Rendering hundreds of shadow maps in a frame is not currently viable. Adding non-diffuse BRDFs requires using many more VPLs.

This method was optimised in [LSK+07] by using frame coherence, updating only a small number of VPLs per frame and reusing the majority from the previous frame. This results in real-time frame rates for single bounce radiosity. However, the scene needs to be re-meshed since parabolic shadow maps are used.

Nichols et. al. [NW09] suggest a hierarchical image space variant of instant radiosity with reflective shadow maps, which clusters coherent pixels in image space. This allows using more samples in areas with high frequency lighting. The approach is single bounce without indirect visibility.

In [CHH03] radiosity was computed by explicitly solving the N^2 form-factor matrix with Jacobi iteration. It is noted that in terms of matrix operations the GPU overtakes the CPU when using more than 2000 elements. However, they were limited by a maximum texture resolution of 2048x2048 and assume unoccluded environments. Although this is an interesting experiment, such an approach will not scale well and requires N^2 memory.

Progressive refinement radiosity was realised on the GPU by Coombe et. al. in [CHL04]. It uses the fragment processing power to compute visibility by rendering the scene in false colour to a parabolic map from the point of view of the current shooter texel. During this pass an occlusion query is also issued for each face limiting the number of possible receivers. Subsequently, each face in the scene that passed the occlusion test is rendered orthographically and a fragment program back-projects each fragment into the shooter item buffer. If the face id matches, energy is received using a differential area-to-area form factor between the shooter and the texel. Shooter selection is performed by rendering each face into a 1x1 frame buffer using residual energy as depth. Due to the nature of the form factor computation, texel sizes must be small near adjoining surfaces to avoid artifacts. The approach requires fine tessellation of the scene since the parabolic projection distorts geometry. The approach is essentially $O(N^2)$ in the number of texels and thus assumes that equilibrium can be achieved using relatively few shots. If the scene contains many large emitters this method will perform poorly.

3.5.2 GPU Particle Tracing Methods

In [SB97] the results of a particle tracing phase (photon mapping) is used to render globally illuminated scenes quickly. The particle tracing is performed in a pre-processing phase and for each surface hit a triangle, which encloses the non-zero parts of a kernel support function, is formed in the plane of the struck surface and stored. During rendering the contributions of photons is splatted to visible surfaces in the framebuffer by rendering the stored *particle triangles* applying the BRDF with respect to the

viewing direction and textured with a map containing the discretised kernel function. This is optimised only taking into account particles with more than one bounce and rendering the direct lighting with shadow mapping and phong illumination. The approach cannot handle area light sources and introduces significant overdraw in the framebuffer limiting the efficiency of the approach.

A variation on the above technique was suggested by McGuire and Luebke in [ML09]. The approach renders the first particle bounce with an approach similar to shadow mapping in light space and traces the surviving particles further through the scene using CPU ray tracing. The final photons are rendered into screen space via splatting. An icosahedron is used to tightly bound the 3D filter kernel and invokes all visible pixels that lie within it. Each such pixel gets a contribution weighted by the surface BRDF and filter kernel. The method is real-time for *one* point light but speed will degrade linearly with the number of lights in the scene. It cannot handle area lights and requires the use of shadow maps for direct lighting. The photon tracing requires eight CPUs to run at real-time rates (for a single light) and the method requires transferring a significant amount of data to and from the GPU. The amount of this data is proportional to the number of lights.

At the other end of the spectrum Hachisuka et. al. [HOJ08] propose a progressive photon mapping approach that can produce renderings with *any* desired accuracy. It captures complex illumination including caustics and reflections of caustics. The method progressively computes photon tracing passes increasing the accuracy of the result without having to store the full photon map before rendering. Although, the approach can produce high quality results for scenes with complex lighting, rendering times for 640x480 images are on the order of many hours, and, due to the nature of the algorithm the camera is static.

3.5.3 GPU Hybrid Methods

In [DS05] Dachsbacher and Stamminger introduced the Reflective Shadow Map (RSM). The RSM can provide real-time single bounce radiosity by computing an extended shadow map that, in addition to the standard depth map, includes extra attributes such as world space coordinates, normal and flux. Since the RSM is rendered from the point of view of the light source all single bounce indirect illumination caused by this light source must originate from surfaces visible in the RSM. Using the attributes stored in the RSM indirect illumination can be computed by considering the pixels in the RSM as secondary emitters. In order to make this interactive a fixed number of indirect pixel lights are used. First the gathering pixel is projected into the RSM. Then pixel lights are selected using a fixed sampling pattern with a density that is inversely proportional to the distance to the projected points under the assumption that the distance in the RSM is a reasonable approximation to the distance in world space. Furthermore, for the indirect illumination a low resolution image of the camera view is computed and during rendering the illumination is either interpolated based on pixel normal and position or recomputed in full if no suitable samples are available.

The benefits of this approach are its simplicity and that it allows fully dynamic environments. Also, it is simple to integrate into an existing framework based on rasterisation such as a game engine. On the other hand it is a coarse approximation as it only applies one bounce of radiosity with no visibility,

which would lead to severe light bleeding in typical scenes. The approach uses a coarse screen-space resolution (32x32) that could cause light bleeding in scenes with significant variation in lighting. Also, it does not work for area light sources unless these are approximated by multiple lights, which would make the approach non-interactive. The complexity is linear in the number of point lights and screen lighting resolution. The results at low resolution with a single spot light are 5-27 frames per second. Maintaining an interactive frame-rate with more than a single light would be intractable. Finally, the light bleeding is alleviated by applying ambient occlusion [Bun05] during rendering but the cost of this is not accounted for in the results presented in the paper.

In [DS06] the gathering step is replaced by a splatting step requiring fewer indirect pixel lights and allowing for glossy materials. However, the approach would not be interactive with more than a handful of lights. In [NW09] the overdraw caused by the splatting is reduced by using multi-resolution splat buffers in screen-space and hierarchically subdividing these splats in areas with discontinuities.

A variation on hierarchical radiosity using implicit visibility was presented in [DSDD07]. Explicit visibility queries were avoided by using an iterative process that emits *antiradiance* from the negative hemisphere on surfaces. This allows an unoccluded global transport operation followed by a number of local iterations using negative radiance to cancel out extraneous light. The approach achieves single digit frame rates and uses traditional shadow mapping for direct lighting computations ruling out area lights. The approach is also hampered by the need for highly tesselated surfaces in order to support indirect shadows. Furthermore, the hierarchy takes $>$9-13 seconds to build for all but trivial scenes. Thus it is not rebuilt during dynamic changes to geometry so tesselation and links must be created in support of a worst-case scenario or a priori knowledge of the dynamic changes is required. Finally, the efficiency is proportional to the depth complexity of the scene, which they set at 3-4. For more complex scenes this will not be sufficient and may cause artifacts.

In a similar vein Dong et. al. [DKTS07] propose a global illumination method that uses a hierarchy to determine visibility implicitly. A predefined number of links over the hemisphere above a vertex are hierarchically constructed by always keeping the *shortest* link in a directional bin when a new candidate link is available. This avoids explicit evaluation of visibility. It does however require an initial hierarchical clustering of the triangles of the scene elements such that the link creation stage has a starting point. For dynamic scenes it is assumed that this initial per object hierarchy is unchanged, essentially limiting the animation to rigid body animation and/or slight deformative animation. Due to the data structure it is suitable for low-frequency BRDFs only. The method requires fine uniformly tesselated scenes to prevent artifacts. For modest scenes with a single bounce they achieve 4-10 fps.

In [RGK$^+$08] Ritschel et. al. presented a global illumination method for dynamic scenes based on *Instant Radiosity* [Kel97]. Virtual point lights are used with low resolution parabolic shadow maps (ISMs) for indirect illumination. This is based on the assumption that accurate visibility is not required for indirect lighting. The ISMs are created by splatting a coarse point based representation of the scene into a depth buffer and filling in any remaining holes. Since the implementations build on VPLs it supports only point or spot lights. A Cornell box scene with a single spotlight can be rendered at 5.5 fps

with 3 bounces and 14 fps with a single bounce.

A variation on this scheme was proposed in [RGKM07]. This uses coherent shadow maps (CSMs) for the objects in the scene. A number of orthographically projected depth maps outside the hull of each object are precomputed, heavily compressed and stored. A monte carlo renderer then uses this information to perform visibility queries against the objects. This allows high quality direct illumination with rigid body animation, arbitrary BRDFs and light sources but no indirect lighting. Precompute times range from seconds to over half an hour depending on resolution.

In [RGKS08] this was combined with coherent surface shadow maps (CSSMs) and used to answer general visibility queries in a global illumination renderer based on hierarchical radiosity [HSA91] and *lightcuts* [WFA+05]. In order to resolve visibility queries from surface points on the objects and not just from outside its hull, cube depth maps were swept across the object surfaces in a precompute process. As with CSMs these can be compressed and stored. When used in combination they can answer general point to point visibility queries effectively. This method achieves single digit frame rates for global illumination with two bounces. Although the method could in theory be used for direct lighting also the reported results were using direct lighting from spotlights with traditional shadow maps.

The many-light problem is a generalisation of the VPL approach introduced by Keller [Kel97]. Rendering from thousands of indirect lights yields realistic indirect lighting from area light sources and supports arbitrary BRDFs. However, brute-force rendering from many lights is prohibitively expensive, mainly because of visibility computations. Walter et. al. addresses this with the lightcuts method [WFA+05], which clusters lights together and builds a binary light tree. For rendering, cuts in this tree are adaptively selected based on an error metric and each cluster is treated collectively as a single (larger) point light. This gives a sublinear complexity in the number of lights and reduces the number of visibility queries needed, and allows rendering of globally lit images in a few minutes of computation. In [CPWAP08] interactive frame rates are achieved (2-4 fps) by assuming a static scene and precomputing visibility cuts (13-40 minutes). The light tree is also improved by introducing a visibility term in the metric. In [HPB07], Hašan et. al. reformulate this into a matrix problem where the rows are sample points and the columns are point lights. The method computes entire rows and columns in this matrix using GPU shadow mapping. Realistic images can be computed in a few seconds by intelligently selecting and computing a small set of rows and columns of the matrix. Wang et. al. [WWZ+09] compute a clustering for both the input lights and the output shading points. Input lights are computed with photon mapping and illumination cuts are computed from the resulting kd-tree. During final rendering pixels are interpolated from the shading clusters using irradiance caching [WRC88]. The light transport is computed with two indirect bounces using ambient and perfectly specular materials only. However, in the final bounce the radiance fields are approximated with a 4-th order SH representation (16 coefficients) [RH01] such that low-frequency glossy BRDFs can be approximated in the final bounce. Interactive walkthrough is possible at a few (1.5 - 4.2) frames per second.

3.5.4 Summary

The majority of the GPU approaches attempt to solve the global illumination problem in some form or other for dynamic scenes at interactive rates ($<$ 15 frames per second). This is, however, inappropriate for VR applications that require stable real-time frame rates [BH95]. The reflective shadow map (RSM) techniques are single bounce with no visibility and support only point light sources. Also, they, along with VPL based approaches and particle tracing methods, use traditional shadow mapping techniques for direct lighting, which can lead to poor performance in scenes with multiple lights. Thus, only a few of these approaches truly support area lights including [CHL04]. Even this approach is quite sensitive to the number and cumulative area of light sources.

3.6 Discussion

In the context of immersive VR applications there is definitely a need for a solution, which can render full global illumination at stable real-time frame rates on high resolution displays. Immersive VR displays are upwards of 1024x1024 pixels rendered in stereo. At 30 frames per second this yields $\sim$60MPixels per second (MPix/sec) or in terms of a ray tracing solution $\sim$60MRays per second (MRays/sec) just in order to form an image on the screen let alone illuminate it.

Although radiosity scenes can be displayed at real-time frame rates after reaching equilibrium [GTGB84, NN85, CG85, CCWG88, HSA91, CHL04] the aim is to support the full spectrum of global illumination including specular effects which are difficult to achieve with a radiosity based algorithm for even moderately complex scenes. Attempts at adding specular effects to a radiosity method do not scale well and are only appropriate for simple scenes [WCG87, ICG86, SPL88].

To our knowledge, the fastest ray tracing method to date is *AEPSA* [FCM09] reporting $\sim$21MRays/sec for a scene with 10k faces. This is for bundles of *coherent* rays shot from a pin hole camera. The efficiency for *incoherent* rays such as secondary rays for illumination or shadow rays are bound to be significantly lower than this. Wald [WKB$^+$02] notes that in order to simulate global illumination effects at least 20 rays are needed per pixel. Using ray tracing will require many CPU cores to even just render the primary rays in the VR use-case above. Adding shadow rays to this and rays for secondary illumination will be impractical even when using (ir)radiance caching methods [WRC88, WH92, KGPB05]. A similar argument applies to *distribution ray tracing* [CPC84] and *path tracing* [Kaj86, AK90, Shi90b].

The *ray cache* [Lar98, WS99, SS00], *render cache* [WDP99, WDG02, BWG03] and *shading cache* [TPWG02, SHSS00] methods can provide global illumination at interactive rates but suffer from variable frame rates and disagreeable artifacts especially near silhouettes and shadow boundaries due to re-projection errors or stale samples. Immersive VR applications using stereo imaging need stable frame rates and a stable image in order to not induce discomfort in a participant.

At the other end of the spectrum are *precompute* methods. These aim to compute and store a global illumination solution in an appropriate data structure. *Photon mapping* [JC95, Jen96] stores a large number of photons in a kd-tree and use these to compute indirect illumination. The method is

typically used with ray tracing and final gathering since the photon map is not appropriate for directly rendering shadow boundaries for example. The method is not real-time and is mainly used for high quality renderings. More recently photon mapping has been reformulated as a rasterisation method using splatting to illuminate surfaces [SB97, WS03, KBW06, ML09] achieving near real-time frame rates. These methods, however, typically do not support area light sources and need shadow mapping or ray traced shadows in order to resolve direct illumination. Additionally, the method performs much screen-space overdraw especially for dense photon maps, which could render the method fill-rate limited.

Light fields comprise another class of methods that precompute and store a global illumination solution. Traditional light fields could only represent a single object [GGSC96, LH96, IPL97, MRP98, Azu99, WAA+00], but for some parameterisations such as DPP [CLF98, CF99, Cam01] multiple objects can easily be supported without loss of generality by adding depth layers in the representation. The VLF method presented in this thesis makes use of a variation on this data structure.

Recently, many interactive GPU based solutions have been offered. These methods are generally interactive (<15 frames per second) [DSDD07, DKTS07, RGK+08, RGKS08], a number of them only provide a coarse approximation of global illumination [DS05, DS06, NW09] ignoring visibility for indirect light, and most do not support area light sources and resort to shadow maps for direct illumination [DS05, DS06, DSDD07, RGK+08, NW09]. These issues make the approaches inapplicable to real-time VR scenarios.

Table A.4 in Appendix A.4 lists the features of a selection of algorithms presented in this Chapter. Of these only a handful can provide global illumination at stable frame rates above 30 frames per second. Those that do cannot simulate non-diffuse transport or have severe limitations such as supporting only a single bounce or the lack of visibility in the bounces.

The VLF can fill this void. It can render full global illumination with arbitrary BRDFs and area lights only adding small constant time to a traditional rasterisation solution, essentially getting constant time *illumination*. Also, it plugs in well with existing frameworks not requiring special CPU-GPU synchronisation or run-time updates of complicated data structures. Although the geometry must be static it can easily relight dynamic geometry such as avatars [MYK+08]. Propagation is fast; linear time in the number of input faces, and gracefully converges towards physically correct solutions. Additionally, it handles direct lighting for the static geometry only requiring shadowing for dynamic geometry if such is desired [MYK+08].

Chapter 4

A Virtual Light Field Approach to Global Illumination

This chapter describes an algorithm that provides real-time walkthrough of globally illuminated static virtual scenes consisting of convex polyhedra in non-participating media with a mixture of diffuse, glossy and specular surfaces. The algorithm decouples illumination from geometry and can render a frame in nearly constant time. The algorithm makes use of a light field data structure that records radiance values on all surface points in all directions. The light field data structure is discrete and uniformly samples the 5D radiance function $L(\theta,\phi,x,y,z)$ on surfaces only. In an iterative pre-processing step the algorithm propagates light outwards from emitters accounting for all possible $L(S|G|D)^*$ paths. The resulting *virtual light field* (VLF) can be used as a basis for many rendering algorithms. Radiance values can be directly resampled and rendered, geometry can be used to guide rendering when using low sampling densities or the light field can be used for incident illumination in a final gather type algorithm.

The VLF was initially introduced in [SMKY04] but the algorithm was greatly limited by the computational complexity involved in the pre-processing step. Propagating even simple scenes would take days or weeks and scaling was quadratic. Secondly, real-time rendering could only be achieved for low resolution images. In this Chapter the first issue is dealt with. The propagation algorithm will be reformulated and various alternatives will be presented reducing propagation time by several orders of magnitude and scaling to linear complexity. The data structure used is similar to that used in [SMKY04] but all other elements of the system are novel.

4.1 Overview

This Chapter starts with a conceptual overview of the data structure and the proposed light transport algorithm in order to make it easier to digest the following Chapters. The overview is followed by an in-depth presentation of the separate elements of the technique.

4.1.1 Overview of the Data Structure

In order to support *precomputed* global illumination a data structure for storing radiance emitted from the surfaces in the scene is required. Since radiance is defined as radiant power *per unit area* and *per unit solid angle* (see Equation 2.12) the data structure becomes a 5-dimensional representation of

the function $L(\theta,\phi,x,y,z)$. In order to work with this in a computer the function is discretised into a collection of finite elements that can be indexed by spatial position (x,y,z) and direction (θ,ϕ). In this work participating media such as smoke will be ignored so that the spatial sampling can be restricted to the surfaces of the objects in the scene.

The traditional way of representing such data structures is to subdivide the surface area into patches and then store a hemisphere of directions for each patch. The hemisphere is aligned to the tangent plane of the patch. Radiance can then be indexed by selecting a patch and looking up the desired direction in the hemisphere. This is illustrated in Figure 4.1.

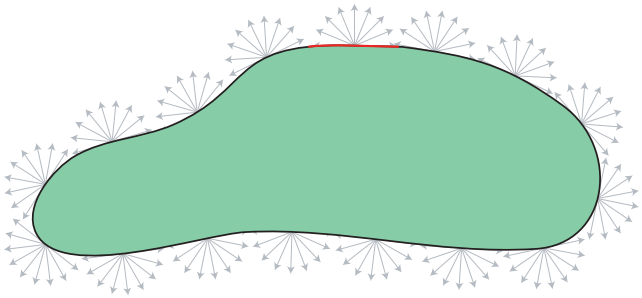
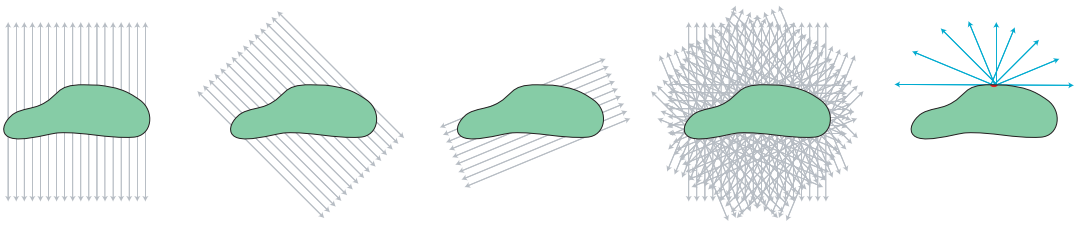


Figure 4.1: Surface based hemisphere data structure. A surface patch is highlighted in orange.

A different way of storing the same information is to use a *global line distribution*. There are many types of these but the central idea is that all surfaces in the scene share a global set of lines where each line intersects the entire scene. If the distribution is sufficiently dense and uniformly distributed the net effect will be similar to a distribution of surface hemispheres. One such distribution is based on the *direction-and-point* parameterisation (DPP) [CLF98]. The DPP distribution samples the domain using a collection of parallel subfields (PSF). A parallel subfield is a collection of parallel lines that intersect the entire scene. The collection of PSFs is chosen such that the directional domain is sampled uniformly. It can be useful to think of the lines in a PSF as originating from some "virtual" plane that is orthogonal to the direction of the lines. The lines in a given PSF are uniformly distributed across this "virtual" plane. This is illustrated in Figure 4.2.



(a) PSF_0 (b) PSF_1 (c) PSF_2 (d) Union of many PSFs (e) PSF samples

Figure 4.2: VLF data structure using parallel subfields. (a)-(c) show example PSFs and (d) illustrates the union of 8 PSFs. (e) shows PSF samples intersecting a small surface area on the object.

Given this global line distribution a data structure for storing radiance can be easily devised. Given

an object store outgoing radiance on all the lines that intersect the object. If the scene is composed of convex polygons then the intersection of a polygon and a PSF will form an orthogonal 2D "image" or "layer" of radiance values; one value for each line intersecting the polygon. Thus each polygon stores such a layer on each PSF. Indexing into this data structure is in a sense opposite to the surface hemisphere approach. First a direction is selected, then a surface position can be indexed by selecting a sample in the layer of the object or polygon in question. One important optimisation must be noted. For diffuse objects it is clearly wasteful to store outgoing radiance as layers in each PSF as it will clearly be the same in all directions, so for diffuse objects radiance is only stored once in a texture map. When querying a diffuse face for its radiance along a line in a PSF, the point is reprojected into the texture map and a radiance value is returned.

The benefit of using this data structure is that the inherent coherence in the sampling of (θ, ϕ, x, y, z) is exposed. Neighbouring lines in the PSFs are likely to intersect a similar set of objects and the objects share a global set of lines so multiple surface intersections along the lines can be reused.

Furthermore, if the polygons are assumed to be non-intersecting they can be sorted by depth along a given PSF. If the layers for the polygons are generated and stored using this ordering (in each PSF), a unique ordering of radiance samples along each line from near to far (or vica versa) is guaranteed. This makes it simple and fast to determine mutually visible samples by simply traversing the samples linearly requiring no separate visibility queries.

4.1.2 Overview of the Light Transport Algorithm

Propagation using the VLF is described next. Radiance is shot outwards from each sample in the 5-dimensional data structure. PSFs are handled in turn exchanging radiance between mutually visible pairs of samples. In order to make the technique feasible mutual visibility between pairs of samples must be resolved efficiently.

The depth sorted set of PSFs described in the preceding Section represents all the visibility needed to perform global illumination transport in linear time since each sample only needs to be visited once and the sample visible to it is just the next one in the list. This way it is possible to maintain an "overall" image or radiance interface (*RI*) of the radiance transported so far. This image can then be projected onto receivers. A fraction of the radiance received from the *RI* is accumulated in the diffuse texture map of the receiver and a fraction is reflected into other PSFs depending on the BRDF assigned to the receiver. This reflection maps the radiance layer for the receiver into the radiance layer of a another PSF. Transport is done twice for each PSF once in each direction. Each face only needs to be projected once into the *RI* and the *RI* is projected once into the face when transporting in the opposite direction yielding $O(n)$ complexity, where n is the number of faces. This is illustrated in Figure 4.3. Here face r (yellow) is only projected once into RI_1, and later (not shown) the *RI* is projected into face r when transporting from left to right.

A sample in a PSF is represented by a square pixel the area of which depends on the spatial resolution of the PSF plane. Thus when transport is computed with these pixels a given face may only partially cover the pixel. In order to compute the correct contribution a face will make to such a pixel the

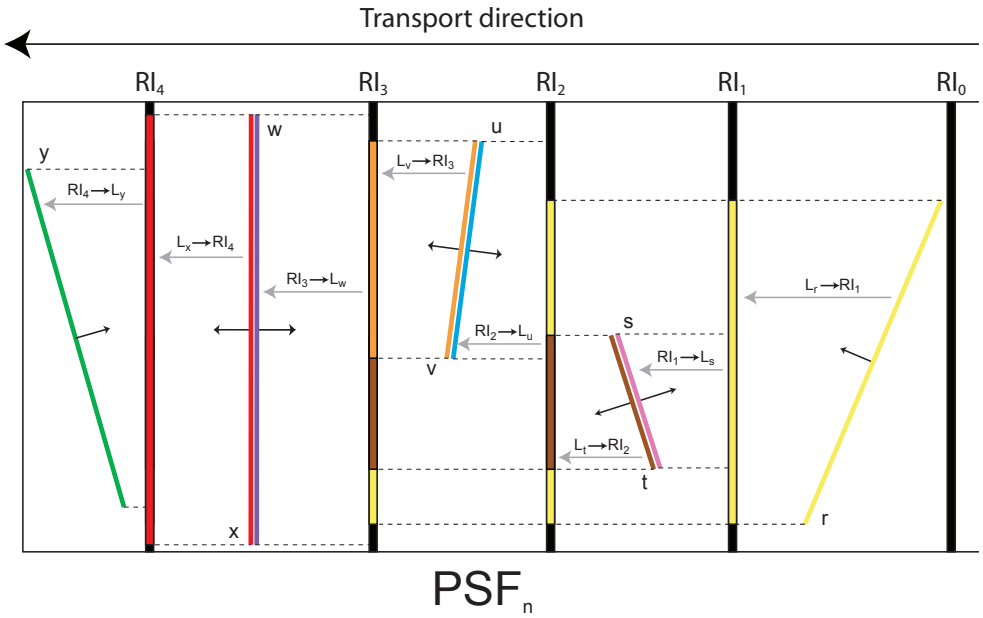


Figure 4.3: Incremental approach for radiance transfer with pre-computed sorting. Transfer takes place from right to left. Faces are incrementally rendered into a radiance interface. For clarity the radiance interface is shown multiple times. RI_{n+1} is the same interface as RI_n with additional faces rendered onto it.

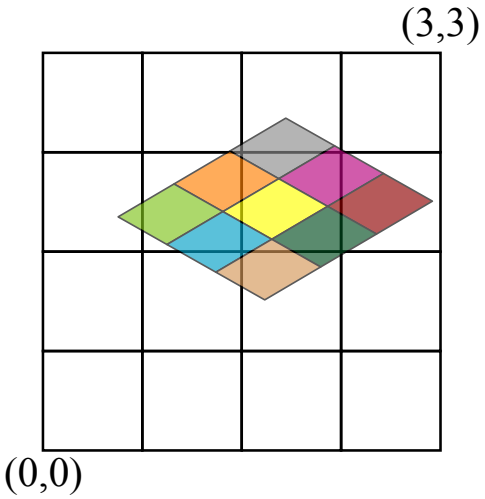


Figure 4.4: Irradiance map projection to a 4×4 PSF grid.

fraction of overlap between the pixel and the face must be computed. This is particularly an issue when transporting from a diffuse surface into a PSF, since the pixels will be unaligned with the PSF pixels when projected from the texture map on the face. This is illustrated in Figure 4.4.

In summary, the aim is to present a fast incremental technique that is realised using transport operations that map well to graphics hardware such that the foundation is laid for Chapter 5 where the technique is presented in terms of rendering operations. In particular the following items will presented in this Chapter:

- In Section 4.2 the data structure is described in detail. The properties of the DPP parameterisation and its benefits will be discussed. An approach for tiling the radiance layers in order to save memory is presented.

- In Section 4.3 the propagation stage is described. This is arranged using a *bottom-up* approach. The stages are as follows:

  - Section 4.3.1 describes the low-level finite element interactions. A technique using point sampling that map well to graphics hardware will be presented.

  - Section 4.3.2 describes the PSF propagation algorithm for exchanging radiance among faces along a PSF. Various options involving unsorted and sorted face sequences will be described. An incremental approach will be presented that is capable of performing the transport in linear time.

  - Section 4.3.3 describes the high level VLF transport operation. The use of ”virtual” jittered PSFs in order to reduce artifacts is described.

- In Section 4.4 methods for rendering from the VLF data structure are presented.

4.2 Data Structure

The data structure used for the VLF was inspired by Levoy and Hanrahan's *light field* [LH96] and Gortler et. al.'s *lumigraph* [GGSC96]. However, the light slab approach used in their work only supports a single object and is nonuniform causing biases and disparity problems resulting in rendering artifacts. The light field representation employed in this thesis is constructed by choosing a regular point subdivision over a hemisphere, to give a set of directions, and then corresponding to each direction there is a rectangular grid of parallel rays covering the scene. Each object is orthographically projected in each direction onto the grid of rays and represented by the rays intersecting it, storing information such as radiance, object identifier and depth with the ray intersection. Projecting multiple objects forms conceptual layers in the grid of rays. This data structure is similar to Lischinski's *layered light field* [LR98] that uses *layered depth images* [SGwHS98] uniformly distributed over the sphere. Camahort et. al. [CLF98] also presented a similar data structure; the direction and point parameterisation (DPP) and used it for light fields with occlusion.

4.2.1 Uniformity of Representation

The goal of a light field representation is to capture the 5D radiance function $L(\theta,\phi,x,y,z)$. In a uniform light field a uniform line density is induced by a uniform random sampling of the individual line parameters (θ,ϕ,x,y,z). Such a light field model, including the DPP, is statistically invariant under rotations and translations. This property, *statistical uniformity*, is very important as it allows the user to freely navigate a model without experiencing aliasing due to changes in resolution. This property is broken in the light slab parameterisation where sampling density varies across the slab, which is the cause of view-dependent aliasing when rendering from it. Statistical uniformity also provides the added benefit that the model exhibits guaranteed constant error bounds in all dimensions [Cam01].

In [Cam01] Camahort presented a theoretical foundation for light field uniformity. A continuous light field parameterisation is said to be *statistically uniform* when it has the following property:

> ***Statistical uniformity*** *– For any set of uniformly distributed light-field parameters, the set of oriented lines represented by those parameters is uniformly distributed over the space of all oriented lines.*

In order to represent the light field on a computer it typically needs to be discretised and for a discrete light field representation the parameters defining it are sampled separately. In a finite context, a set of oriented lines is called a *pencil* of lines and its *measure* is the number of lines contained in the set. In the discrete domain a light field representation is said to be *uniformly sampled* when it has the following property:

> ***Sampling uniformity*** *– A uniform sampling of the parameters of the light field induces a uniform sampling of the set of oriented lines in the light field support.*

These properties can be related; given a statistically uniform parameterisation, a uniformly sampled discrete light field representation can be obtained by uniformly sampling each of the continuous parameters individually. The light slab representation is not statistically uniform so even though the parameters are sampled uniformly the set of lines in the light field support is not uniform. In this thesis a statistically uniform light field parameterisation has been chosen to avoid issues with view-dependent biases. The parameters are sampled uniformly across their domains and this ensures that the light field samples are uniformly distributed in their five-dimensional domain.

4.2.2 Directional Subdivision

In order to facilitate the construction of a uniformly sampled light field representation, a uniform sampling of the sphere bounding the scene represented by the light field must be obtained. This requirement produces a uniform sampling of the set of oriented lines on the sphere. But in principle only a hemisphere is needed if storage and transport are solved bi-directionally. Given any oriented line its ”opposite” line will lie on the opposite hemisphere, so by sampling only the upper hemisphere and letting these samples represent a line in each direction a complete sphere of oriented directions will be available.

A uniform sampling of the *directional domain* represented by (θ,ϕ) where $\theta \in [0,2\pi]$ and $\phi \in [0,\frac{\pi}{2}]$ is then sought. This is normally obtained by subdividing the sphere into l spherical polygons or triangles and letting either the centre or vertices represent the single directional sample $\omega_i = (\theta_i,\phi_i)$,

$i \in [0,1,\ldots,l-1]$. The single directional sample in fact represents the *pencil* of lines passing through the spherical triangle. An ideal subdivision would produce l spherical triangles with identical shape and area $\frac{4\pi}{l}$ or $\frac{2\pi}{l}$ for the hemisphere case. Actual tessellations are often based on platonic solids [Fek90, GMN94, SS95], but since the most complex platonic solid, the icosahedron, has only 20 faces, they are normally recursively subdivided, each subdivision increasing the number of samples fourfold. The subdivision splits a spherical triangle into four smaller triangles and projects their vertices onto the sphere (see Figure 4.5), this has the unfortunate side effect that the centre triangle becomes larger relative to the remaining triangles.

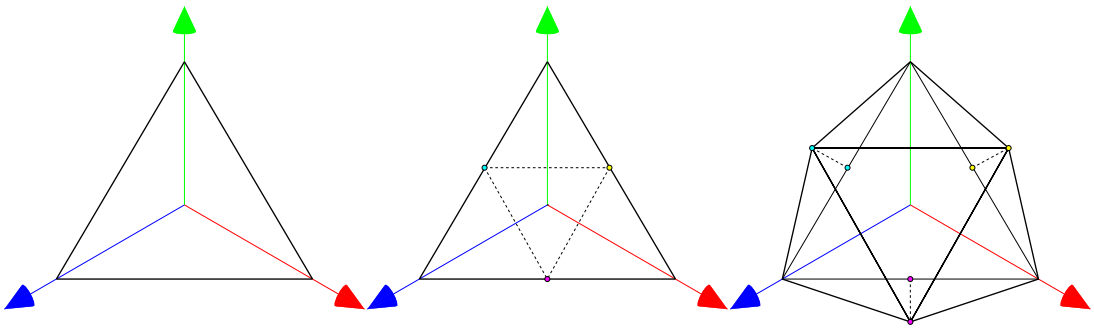


Figure 4.5: Subdivision of the positive quadrant of the hemisphere, edges are split medially and projected onto the sphere creating four sub-triangles.

This problem can only be dealt with by introducing correction factors, that correct light transport and lookup from the light field representation by accounting for the relative variation in area. A weight $\alpha_i = A_i \frac{l}{4\pi}$ is assigned to each sample ω_i with actual area A_i accounting for its variation from the "ideal" area such that $\Sigma_{i=0}^{l-1} \frac{A_i}{\alpha_i} = 4\pi$. This variation is large, on the order 65% difference between the largest and smallest spherical triangle. This of course affects the uniformity of the representation since the measure of the pencils represented by the samples ω_i differ, but in practice this bias does not introduce noticeable artifacts when appropriately accounted for in the rendering and propagation stage. Figure 4.6 shows how the error is distributed over the hemisphere.

The spherical subdivision scheme used in this thesis is based on subdivision of the octahedron. This platonic solid can be represented by a single octant since the remaining are symmetric and can be referenced by an appropriate rotation into the positive octant and using an offset based on the source octant. It is essential to choose a parameterisation over the hemisphere that does not require searching in order to find closest directions – since such directional lookup is a critical operation during both propagation and rendering. Slater [Sla02] gave an algorithm for efficient constant time lookup of directions on the hemisphere in order to find the closest stored direction, this is suitable for the VLF since this operation is performed frequently.

In summary, the sphere is represented by l directions $(\omega_0, \omega_1, \ldots, \omega_{l-1})$, where $\omega_i = (\theta_i, \phi_i)$, each is associated with a solid angle correction factor $(\alpha_0, \alpha_1, \ldots, \alpha_{l-1})$. The directions are restricted to the upper hemisphere of a recursive subdivision of the octahedron and each direction is assumed bi-

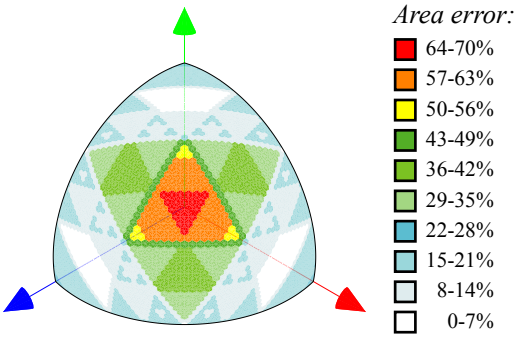


Figure 4.6: Temperature plot of distribution of area error on the positive quadrant of the hemisphere subdivided to level 6 with roughly 8K directions.

directional such that the whole sphere is implicitly represented. This obviously requires that care is taken to avoid representing directions along the equator twice, this is easily achieved by restricting the azimuthal spherical coordinate θ to $[0,\pi)$ when representing directions lying on the equator. Any equatorial directional accesses to directions (θ,ϕ) where $\phi \approx 0$ and $\theta \geq \pi$ are rotated into the represented range by using $\theta - \pi$ as azimuth. The orders of directions supported due to the nature of the recursive subdivision are multiples of four. Figure 4.7 illustrates such a hemisphere subdivision.

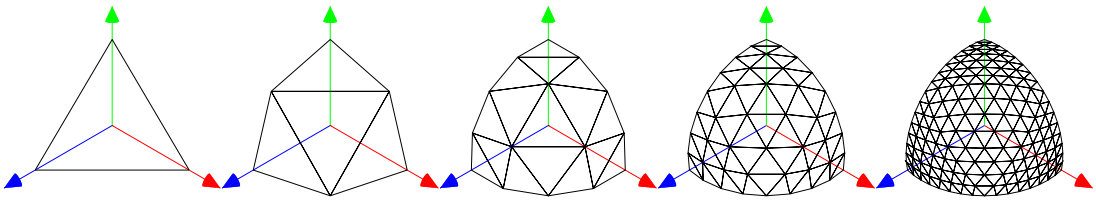


Figure 4.7: Subdivision of the positive quadrant of the hemisphere with 0 to 4 subdivision levels (only positive quadrant shown for clarity).

4.2.3 Spatial Subdivision – Parallel Subfield Representation

Given a nearly uniform sampling of (θ,ϕ) a uniform sampling scheme for the remaining parameters (x,y,z) is required. A volumetric approach is the most intuitive candidate uniformly sampling x, y and z separately forming a uniform voxelation of the space occupied by the scene. However, such a method would use excessive amounts of storage, much of which would be wasted when assuming the absence of participating media and a scene composed of polyhedra. In such a setting it would be far less costly to only store intersection points with scene geometry as these are the only places where light transfer is interrupted and changes state due to absorption or scattering.

For simplicity assume that a transformation $M_{WC\to VLF}$ is initially applied to the scene that centres it at the origin and scales it such that it resides within the unit sphere; this coordinate frame is dubbed *the VLF coordinate system*. In practice this transformation is stored along with its inverse $M^{-1}_{WC\to VLF} = M_{VLF\to WC}$ and applied as needed. Given a directional sample $\omega_i = (\theta_i,\phi_i)$ an orthonormal basis formed

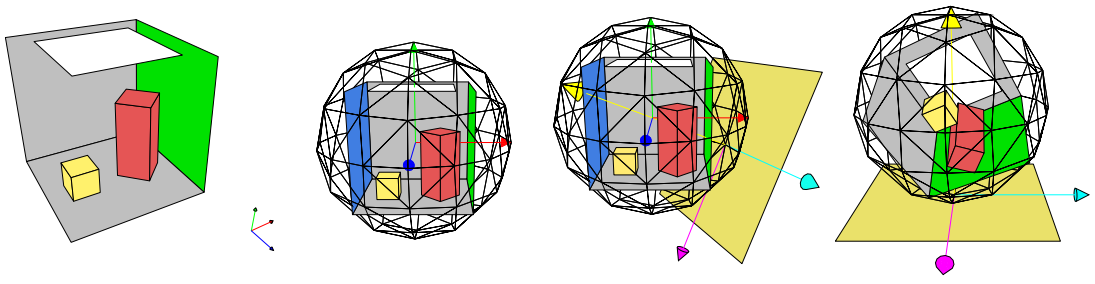
by the triple of vectors $(\overrightarrow{u_i}, \overrightarrow{v_i}, \overrightarrow{n_i})$ exists where $\overrightarrow{n_i}$ is a vector aligned with ω_i:

$$\begin{aligned}
\overrightarrow{n_i} &= (\cos\theta_i \sin\phi_i, \sin\theta_i \sin\phi_i, \cos\phi_i) \\
\overrightarrow{u_i} &= \frac{\overrightarrow{n_i} \times \overrightarrow{e}}{\|\overrightarrow{n_i} \times \overrightarrow{e}\|}, \quad \text{where } \overrightarrow{e} \neq \overrightarrow{n_i} \\
\overrightarrow{v_i} &= \overrightarrow{n_i} \times \overrightarrow{u_i}
\end{aligned}$$

A numerically stable method for computing $\overrightarrow{e}$ in the equation above can be found in [HM99]. The plane formed by $\overrightarrow{u_i}$ and $\overrightarrow{v_i}$ is by definition orthogonal to the vector formed by (θ_i, ϕ_i) and since the scene has a finite extent, a finite range can be imposed upon (u,v) in the plane $\overrightarrow{u_i} \times \overrightarrow{v_i}$ based on the projection of the bounding sphere surrounding the scene. Since the scene resides within the unit sphere due to the aforementioned transformation the range for (u,v) is $[-1,-1] \times [1,1]$. Each such (u,v) forms a bi-directional line parallel to $\overrightarrow{n}$ and the collection of such parallel lines in a given direction is dubbed *a parallel subfield* (PSF). Clearly, a unique parallel subfield exists for each directional sample ω_i with its own local coordinate system defined by $(\overrightarrow{u_i}, \overrightarrow{v_i}, \overrightarrow{n_i})$. Each PSF_i stores a transformation $M_{VLF \to PSF_i}$ along with its inverse $M^{-1}_{VLF \to PSF_i} = M_{PSF_i \to VLF}$ that aligns PSF_i with the canonical PSF where $\overrightarrow{u_i} = (1,0,0)$, $\overrightarrow{v_i} = (0,0,1)$ and $\overrightarrow{n_i} = (0,1,0)$ such that $(\overrightarrow{u_i}, \overrightarrow{v_i}, \overrightarrow{n_i})$ is aligned with (X,Z,Y). Thus, the canonical PSF coordinate system is a left-handed one whereas the world coordinate system is right-handed. However, given this convention points on the PSF plane may be indexed by (u,v) and depth as n. Mapping points from the scene into a given canonical PSF (and back) is simple given these matrices:

$$\begin{aligned}
P_{WC} * M_{WC \to VLF} * M_{VLF \to PSF_i} &= P_{PSF_i} \\
P_{WC} &= P_{PSF_i} * M_{PSF_i \to VLF} * M_{VLF \to WC}
\end{aligned}$$

When a point (x,y,z) has been transformed to the canonical PSF (u,v,n) its position on the PSF plane can be found by simply dropping n. Aggregate matrices $M_{WC \to PSF_i}$ and $M_{PSF_i \to WC}$ are also stored with each PSF, because it is often useful to be able to map a point in WC directly into/from a given PSF_i. Figure 4.8 illustrates the various coordinate frames used.



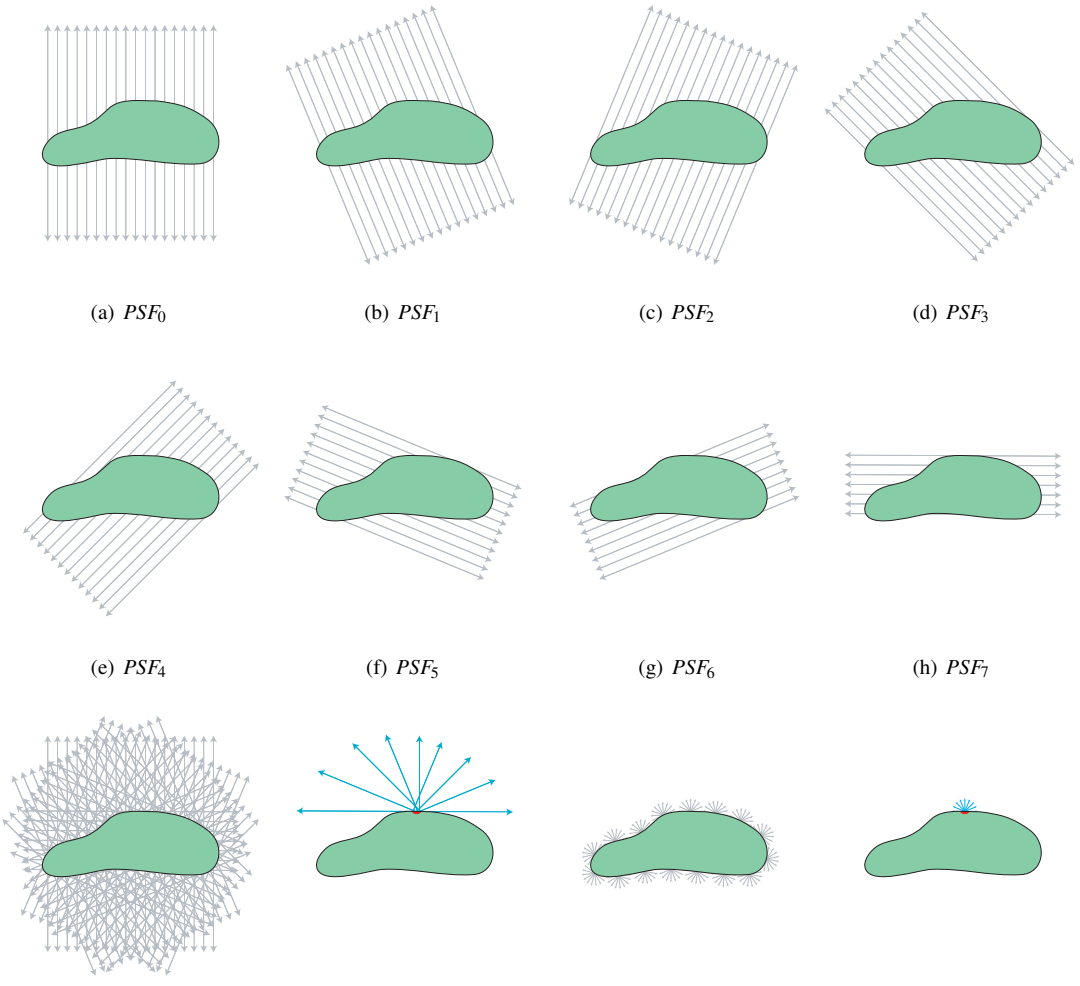
(a) WC (b) VLF (c) PSF (d) canonical PSF

Figure 4.8: Coordinate systems.

Figure 4.8(a) shows a simple Cornell box [GTGB84] residing in the world coordinate system, XYZ axes are shown in red, green and blue respectively. Figure 4.8(b) shows the VLF coordinate system where the scene is embedded within the unit sphere of directions centred at the origin (a level 2 subdivision with

33 directions is used). Figure 4.8(c) shows the PSF coordinate system for a given direction on the sphere, the PSF plane is yellow and the orthonormal basis vectors $(\vec{u}, \vec{v}, \vec{n})$ are shown in cyan, magenta and yellow respectively. In practice the origin of the PSF coordinate frame coincides with the origin of the unit sphere bounding the scene, but for clarity the PSF plane and basis is shifted to lie tangentially to the sphere in the illustration. Also, the lengths of basis vectors are exaggerated for clarity. Figure 4.8(d) shows the PSF from Figure 4.8(c) rotated into its canonical representation.

The data structure differs from traditional surface based data structures in that it samples directional space first. This causes all samples in a single direction to be represented coherently in memory. Figure 4.9 illustrates the difference between the VLF approach and a traditional local lines *hemispheres-on-surface* approach. Although the approaches differ fundamentally in how they sample the space of lines over a surface, they offer a similar line distribution (given similar line densities). Figure 4.9(j) illustrates VLF samples striking a finite surface area. However, the memory access pattern for such an operation would be extremely scattered and ineffective. The VLF data structure is designed to be effective when many samples in the same direction are accessed sequentially.



(a) PSF_0 (b) PSF_1 (c) PSF_2 (d) PSF_3

(e) PSF_4 (f) PSF_5 (g) PSF_6 (h) PSF_7

(i) Union of PSFs (j) PSF cell samples (k) Hemisphere representation (l) Surface cell samples

Figure 4.9: Comparison of VLF data structure to traditional surface based hemisphere data structure.

The notation PSF_{ω_i} or PSF_i where $0 \leq i < l$ denotes the PSF in direction $\omega_i = (\theta_i, \phi_i)$, and, in a continuous context, PSF_ω denotes the PSF in direction ω. In the following the notations PSF_{ω_i}, PSF_i and PSF_ω will be used interchangeably depending on context.

4.2.4 PSF Sampling

In the transillumination method [Neu95] the PSF plane is uniformly subdivided and patches are projected to this grid. The patch identifier with depth is stored in depth sorted order at grid points that fall inside the projection of the patch. Such an approach is efficient in terms of storage since only intersections within projected patches are stored, however coherence between adjacent projected samples is lost and each exchange list at a grid point is sorted separately and radiance exchange is performed for each list. At the other extreme an image with the resolution of the uniform grid storing the depths of projected samples could be maintained for each polygon. This would maintain coherence of samples belonging to the same polygon allowing radiance exchange to be performed for many projected points in parallel, but would waste much storage since small polygons would use only a fraction of the image. Other methods are possible such as quad-trees and edge-tables, but these typically require searching when accessing them and are not efficient when used with graphics hardware.

Tiling

In order to arrive at a suitable compromise between memory consumption and exploitation of coherence a tiled approach to storage has been employed [SMKY04]. The PSF plane is discretised into a regular grid of $N \times N$ samples indexed by (i, j), where $i, j \in (0, 1, \ldots, N-1)$. This sample space is subdivided into *tiles* of resolution $m \times m$, where $1 \leq m \leq N$ and $n = \frac{N}{m}$ is integral. The regular grid of tiles is indexed by (s,t), where $s,t \in (0, 1, \ldots, n-1)$. Each such tile (s,t) is a fully represented map of samples indexed by (u,v), where $u,v \in (0, 1, \ldots, m-1)$.

Consider the projection P' of polygon P into PSF_i. P' will have a tile (s,t) if at least one of the lines in the tile intersects the polygon. This means that the PSF for each (s,t) maintains a list of tiles for projected polygons (partially) overlapping the tile. Figure 4.10 illustrates this concept. The tile $(s,t) = (1,1)$ contains three overlapping polygons, whereas the tiles $(s,t) = (0,1)$ and $(s,t) = (2,0)$ are empty. Consider the tile $(s,t) = (1,0)$ for the light green polygon; this partially overlaps five out of nine samples, potentially wasting 44% on samples falling outside the polygon in that tile map. On the other hand the tile $(s,t) = (2,1)$ for the orange polygon partially covers all samples of its tile.

The benefit of this is that less memory is wasted since only occupied tiles are stored and at the same time coherence for lines intersecting the same polygon is maintained in bundles of $m \times m$ global lines. The tradeoff can be controlled by adjusting m. If $m = N$ then an entire image the size of the PSF is stored for each polygon fully exploiting coherence, on the other hand if $m = 1$ each tile will hold a single sample and memory consumption is minimised.

Looking up a given global line with this data structure uses the following five parameters; (ω, s, t, u, v). When accessing information relating to the surfaces of polygons, the polygon identifier is also supplied in order to locate the corresponding tile; (ω, s, t, u, v, p). The tile can also be accessed using (ω, s, t, p) and the individual samples can be accessed by varying (u,v) over the tile. Finally, a

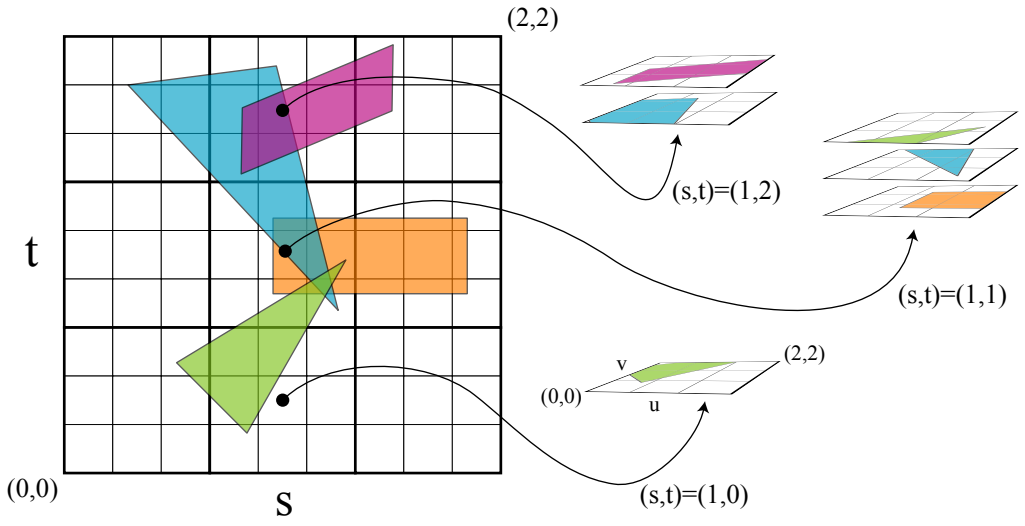


Figure 4.10: Examples of tile lists for four faces projected to a PSF where $n = 3$ and $m = 3$. Bold lines mark the tile boundaries.

"global" lookup, where $i, j \in (0, 1, \ldots, N-1)$, can be converted to a tiled lookup using the following shorthand; $(\omega, i, j, p) = (\omega, \lfloor \frac{i}{m} \rfloor, \lfloor \frac{j}{m} \rfloor, i - m * \lfloor \frac{i}{m} \rfloor, j - m * \lfloor \frac{j}{m} \rfloor, p)$ or conversely a tiled lookup can be converted to a global lookup; $(\omega, s, t, u, v, p) = (\omega, s * m + u, t * m + v, p)$, where $i, j \in (0, 1, \ldots, N-1)$, $s, t \in (0, 1, \ldots, \frac{N}{m} - 1)$ and $u, v \in (0, 1, \ldots, m-1)$.

As described in Section 2.2.3 any ray $r(x, \Theta)$ defined parametrically by $y = x + t \cdot \Theta, t \geq 0$ originating from outside the subset of euclidian three-space occupied by objects will form a list with even cardinality of *t-intersections*, which describe ray-object interactions. If this list is sorted any pair of intersections (t_i, t_{i+1}), where i is even, forms a valid exchange pair. Thus, interactions will always occur between front-facing polygons, one of which is oriented in direction Θ and one which is oriented in direction $-\Theta$. This means that the tiles for a polygon in any PSF_i correspond to the front-facing side of the polygon and that those can be oriented in direction ω_i or $-\omega_i$. Consequently lookups into a tile (ω, s, t, u, v, p) are only defined when $\overrightarrow{n(P_p)} \cdot \overrightarrow{v_\omega} > 0$, where $\overrightarrow{n(P_p)}$ is the outward normal of P and $\overrightarrow{v_\omega}$ is the unit vector in direction ω. Support for transparent objects could be added by storing tiles in both directions.

4.2.5 Data Structures for Radiance Transport

Given a scene and parameters choosing the number of directions, tiling resolution and tile-map resolution (l, n, m) the data structure can be built in a pre-process that cycles through each PSF allocating tiles for each polygon in the scene. Each tile will contain a number of layers storing information required for light propagation such as radiance and visibility information.

Radiance Maps

The VLF stores view-independent radiance. $L(\omega, s, t, u, v, p)$ is the radiance for the line (ω, s, t, u, v) from surface P_p in direction ω. This information is stored as a sequence of tiles $L(\omega, s, t, p)$ covering the projection of P_p into the PSF in direction ω. Each tile stores radiance in two 2D radiance arrays indexed

by (u,v) that are referred to as radiance maps. The *total radiance map* $L_T(\omega,s,t,p)$ refers to total or accumulated radiance, and the *unshot radiance map* $L_U(\omega,s,t,p)$ refers to unshot in-scattered radiance that will be transported in the next iteration.

Visibility Maps

An optional *visibility map* providing information about where P_p is located within a tile can also be stored with a tile. It defines a boolean function $V(\omega,s,t,p)$ for each (u,v) inside the tile. $V(\omega,s,t,u,v,p) = 1$ when polygon P_p is rasterised to cell (u,v) within tile (s,t) for PSF_ω.

The visibility function could also be represented with a real value per sample defining the continuous overlap of P with a given cell (u,v), however this would require much more memory and computation. Using a boolean value requires a single bit per sample and can be efficiently rasterised with graphics hardware. In order to improve the accuracy while retaining the ability to rasterise the maps efficiently a supersampled version VSS of the visibility map V with resolution $sm \times sm$ can be stored in the tile, such that s^2 samples are used to represent the visibility for any (u,v). A continuous value can be extracted from this by summing over the visibility samples representing a given (u,v):

$$V(\omega,s,t,u,v,p) = \frac{1}{s^2}\sum_{u'=0}^{u'<s}\sum_{v'=0}^{v'<s} VSS(\omega,s,t,(su)+u',(sv)+v',p)$$

In some of the propagation techniques presented in this thesis it is more efficient to compute V on the fly than using one stored in the tiles. It will be duly noted when this is the case.

Irradiance Maps

Whereas the radiance maps L_T and L_U clearly represent directional flux, each (partially) diffuse surface P has in addition two associated 2D irradiance arrays referred to as *irradiance maps*. D_U stores unshot irradiance and D_T stores total or accumulated irradiance. These maps are independent of directions since diffuse flux is view-independent and are stored in local texture coordinates of the polygon. $D(u,v,p)$ is the irradiance for any ray originating at (u,v) on polygon P_p. Any ray (ω,s,t,u,v) that passes through a texel of the irradiance map collects a radiance value $L_U^D((\omega,s,t,u,v),p)$, corresponding to the amount of accumulated radiance to be distributed diffusely from the area of the corresponding texel; the radiance is a fraction of the unshot irradiance stored in D_U.

This is a special case optimisation that saves memory when the environment is composed of mainly diffuse geometry. In theory it is not needed and could be replaced by an initial resampling of emitting faces onto the global line discretisation.

Each polygon P stores a transformation matrix $M_{WC\to P}$ along with its inverse $M_{WC\to P}^{-1} = M_{P\to WC}$, which allows transformations to and from its local coordinate system where the irradiance maps reside.

4.3 Propagation

In this section the radiance propagation stage will be described in detail. The goal is to transport radiance amongst polygons and store the results in the radiance maps and irradiance maps available in the data structure. As discussed the organisation of the data structure is such that direction is considered first in order that interactions across polygons in a single direction are handled collectively. The propagation

stage follows this arrangement by exhaustively performing transport amongst polygons in a single direction before moving on to the next. After the propagation stage has completed each finite element in the data structure should hold an average radiance (or irradiance) value that is a *good approximation* to the radiance leaving that finite element.

This Section is arranged using a *bottom-up* approach discussing individual low-level finite element interactions, PSF propagation and, finally, the high level VLF propagation. The stages are laid out in this way:

- In Section 4.3.1 the low-level finite element interactions are described. These are the interactions between the diffuse and non-diffuse finite elements that make up the data structure. Various sampling schemes are discussed and an effective point sampling based technique is presented.
- In Section 4.3.2 the PSF propagation technique is described. This involves transport between the faces in the scene along a PSF and reflections between PSFs. Different techniques for resolving visibility are described and an effective incremental technique is presented.
- In Section 4.3.3 the VLF propagation method is described. This *drives* the overall algorithm by issuing PSF propagation once per PSF for a number of iterations until equilibrium is reached. Additionally, a technique for propagating "virtual" PSFs in order to eliminate artifacts is described in this section.

4.3.1 Low-level Finite Element Propagation

There are two types of finite elements in the VLF data structure. The pixels in irradiance maps D_T and D_U, stored in the local coordinate system of each polygon, and, pixels in radiance maps L_T and L_U stored along directions represented in the VLF. The *fundamental* unit is the radiance sample, the irradiance map concept is added to save storage for mainly diffuse environments as discussed above. This is in contrast to the global line radiosity work of Sbert [SPNP96]. Sbert employs a similar data structure (DPP) to the one used in this thesis, but in that method, however, the *irradiance sample* is the fundamental unit and the data structure is ultimately used to obtain a global line sampling with good coherency that can accelerate radiosity. Diffuse maps only correctly describe perfectly lambertian emitters/reflectors and these are rarely found in a natural scene; most materials have *some* non-diffuse reflection. However, many virtual environments today have mainly diffuse surfaces and using irradiance maps for such scenes offers large savings in storage, making it a useful addition. The propagation stage, however, will centre around the directional radiance sample and avoid simplifications that assume perfectly diffuse behaviour and the presence of the irradiance maps, such that the algorithm is suitable for environments with complex BRDFs.

Each polygon in the scene projects onto each PSF and intersects a number of samples on the PSF grid. The front-facing side of the polygon will send radiance along this set of samples and this outgoing radiance will be recorded with the polygon and deposited on and reflected off other polygons visible to it. (Partly) diffuse faces interact with the radiance samples via their associated irradiance maps whereas (partly) non-diffuse faces interact directly with the radiance samples through their associated radiance

maps. Although the data structure has a different organisation a PSF can be thought of as a collection of beams each containing a list of radiance samples. When sorted along the PSF this forms an exchange list as described in Section 2.2.3. There is one sample for each surface interaction, which is characterised by an intersection between a polygon and the beam. Consider splitting a single beam in two with a plane orthogonal to the PSF direction ω forming a square *cell*, i.e. the cross section of the plane with the beam. For simplicity assume a closed scene and a splitting position inside the scene. Given any (x,y) on the surface of this cell there is a pair of mutually visible surfaces between which transport takes place. The radiance for the cell in direction ω is:

$$L_{cell}(\omega) = \frac{1}{A_{cell}} \int_{p \in cell} L_p(\omega) dA_p \tag{4.1}$$

The radiance for the cell is calculated by integrating over the area of the cell using a visibility function $r(p,-\omega)$ (see Section 2.2.3) to determine which polygons contribute to the radiance of the cell. Alternatively, the same result could be achieved by summing up radiance values weighted by visible area. This would divide the cell into a discrete finite set of continuous territories belonging to the (visible) polygons interacting with the beam:

$$L_{cell}(\omega) = \frac{1}{A_{cell}} \sum_{t \in cell} L_t(\omega) A_t \tag{4.2}$$

Of course the subdivision into territories can be arbitrarily complex; there is no guarantee that a single polygon projects to a single territory. Imagine for example viewing a wall behind a meshed fence.

The first equation naturally suggests a point sampling approach when solved numerically, whereas the second can be solved by calculating the territories employing a continuous clipping algorithm. The latter approach is used in [SMKY04], this thesis, however, introduces a method based on point sampling, which is dramatically more efficient and maps naturally to graphics hardware.

Due to the aforementioned data structure the possible finite element interactions within our domain can be listed as follows:

- Diffuse element interactions
    - Irradiance-radiance cell interactions
    - Radiance-irradiance cell interactions
    - Irradiance-irradiance cell interactions
- Non-diffuse element interactions
    - Intra-PSF radiance-radiance cell interactions
    - Inter-PSF radiance-radiance cell interactions (reflection)

Diffuse Finite Element Resampling

The following sections will deal with the issue of transporting radiance to and from the irradiance maps. The two major issues in this context are visibility computations and choice of sampling strategy. A number of alternatives will be described and their benefits and drawbacks are discussed.

Continuous Transport

Given a set of faces, a method of performing transport between them in a given direction would be to construct a *global visibility map* [SKTNB97, SK98] that subdivides the PSF plane into a number of territories each of which describe a list of faces that project to that territory. This requires that all intersections between all projected edges of all faces are computed in order to build the planar graph from which the territories can be derived. Alternatively, the visibility map can be constructed incrementally by visiting faces ordered by depth along the PSF direction [SKTNB97]. In order to update a receiver surface, the graph of sender surfaces lying in the positive half space with respect to the receiver is clipped against the shape formed by the projection of the receiver onto the PSF plane. This will produce a set of *sender* territories that are visible and overlap the receiver. Transport between the territories and the receiver surface can then be resolved.

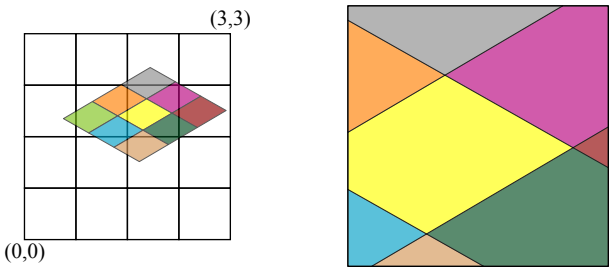
However, this is a *face centric* approach to the transport problem and would still require an additional step to resample and update the radiance maps stored for non-diffuse faces. The immediate solution to this would be to perceive each radiance sample as a square cell that is (already) projected to the PSF and clip each of these against the full graph for receivers (as described above) and just the sender polygon for a sender radiance cell. Of course this would be exceedingly expensive for anything but trivial scenes. Even geometrically trivial scenes could have diffuse texture maps assigned to them causing the clipping to be performed for each texel in the texture map. Furthermore, clipping maps poorly to graphics hardware due to the fact that quite a few conditionals are required.

Semi-Continuous Transport

An alternative approach used in [SMKY04] employs a two-pass semi-continuous approach that avoids the need for examining all other faces for potential overlap by using a discrete supersampled visibility map to guide the clipping step. The approach uses continuous clipping between cell pairs identified by a separate discrete visibility pass. It solves the irradiance-radiance, radiance-irradiance and irradiance-irradiance resampling problems as the algorithm is reversible and "associative", i.e. irradiance-irradiance resampling can be solved by performing an irradiance-radiance followed by a radiance-irradiance resampling stage.

The resampling stage that transports radiance between the irradiance map and the radiance map uses a continuous clipping technique to determine the area of the overlap between a projected irradiance map cell and an axis aligned radiance map cell. Figure 4.11 illustrates a radiance map projected to a PSF grid. In Figure 4.11(b) the continuous contributions made to cell $(2,2)$ are shown, in this example eight different irradiance map cells contribute towards the radiance of the cell and the middle irradiance map cell (yellow) provide the largest contribution.

Recall that the irradiance map is local to the face coordinate system and is subject to an *affine* transformation comprising translation, uniform scaling and rotations to bring it into the PSF coordinate system. After applying an orthographic projection to the transformed irradiance map a grid of parallelograms is formed. The resolution of the irradiance maps are set such that a *many-to-one* mapping



(a) Projected irradiance map (b) Contributions to cell $(2,2)$

Figure 4.11: Irradiance map projection to a 4×4 PSF grid. Continuous contributions made to cell $(2,2)$ are highlighted.

between an irradiance map and a radiance map is guaranteed for all represented directions in order to avoid undersampling artifacts in the radiance samples. During initialisation of the VLF the PSF in the direction most closely matching the face normal, which will receive the largest projection of that face, is used to guide the selection of a resolution ensuring that the mapping criterion will be upheld.

Thus the bulk of the work involved is determining the area of overlap between a radiance cell (quad) and an irradiance cell (parallelogram). The clipping kernel responsible for clipping a parallelogram against a quad is based on the Liang-Barsky polygon clipping algorithm [LB83]. It has been optimised by exploiting the fact that the source and destination shapes are known to be convex.

For a sender polygon updating its outgoing radiance map tiles, visibility is not required and the algorithm simply loops over each irradiance sample, projects it to PSF space and adds area-weighted contributions to any radiance cells the parallelogram intersects. The base parallelogram for the irradiance sample at the origin is computed and the remaining parallelograms can be obtained by adding an appropriate offset to the base parallelogram; this reduces the number of expensive projections involving matrix multiplications.

Dealing with a receiver irradiance map is more complex since it can receive radiance from any number of surfaces depending on their mutual positions, orientations and occlusion. In order to simplify the process the stage is broken down into two passes; one pass computes visibility from the receiving polygon and updates a *temporary* radiance map in PSF space. A second pass performs resampling between this temporary radiance map and the irradiance map. This latter pass is identical to the operation for handling a sender polygon just reversed.

Visibility is computed discretely in PSF space using false colour rendering with z-buffering using rasterisation hardware. Given a PSF_ω and a receiver polygon r that projects to $r' = r * M_{WC \to PSF_\omega}$ in PSF space an orthographic camera is placed at $(0,0,r'_z)$ and points in direction $-\omega$ [1]. Additionally, a clipping plane is set using the plane equation for r' such that no polygon fragments behind r' are rendered. Clipping planes parallel to the PSF direction ω can also be enabled for each edge of the receiver polygon such that only fragments overlapping r' in the PSF plane are considered. Finally, all

[1] For simplicity it is assumed that transport is performed along the PSF direction ω, in practice transport in the opposite direction is handled in a subsequent pass and is performed along $-\omega$ and thus the camera direction is clearly ω.

polygons are rendered coloured with their polygon index with z-buffering and back face culling enabled, producing a discrete visibility map in PSF space where the colour of each pixel encodes a visible polygon index. The resolution of the viewport can be set such that there are one or more pixels for each PSF cell enabling supersampling for more accurate visibility. This is essentially employing the visibility function $r(p,-\omega)$ (see Section 2.2.3) once or more for each radiance cell that intersects the receiver where p is the midpoint of the radiance cell. Supersampling subdivides the receiver PSF cell regularly, creating $U \times V$ smaller PSF cells where U and V define the supersampling resolution. Each of these smaller cells become a target for clipping, and the resultant radiance is the average over all these samples. When a sender surface identifier has been identified for a PSF cell (or sub cell), the corresponding irradiance map is projected to PSF space traversed and each irradiance cell parallelogram is clipped against the radiance cell quad. An area weighted contribution is then added to the temporary radiance map. Each radiance cell (or subcell) only considers the irradiance cells of the polygon identified by the visibility map, making the clipping function *local* and thus constant time; $O(1)$ [2]. Of course this algorithm will perform many unnecessary clipping operations for projected irradiance cells that will clearly produce a null area; this can be avoided by projecting the vertices of the radiance cell quad onto the irradiance map and only traversing those irradiance cells which fall within the axis aligned bounding box formed by the vertices.

This transport technique forms the basis for the VLF algorithm presented in [SMKY04]. However, the algorithm is painfully inefficient and takes hours to days for even relatively simple scenes containing $< 1K$ faces. There are a number of reasons for this.

Since the clipping kernel lives within the innermost loop of the algorithm it is executed very frequently. It is relatively expensive requiring floating point operations and branching. It also requires a per face projection step to PSF space involving multiple 4x4 matrix operations. Mainly due to the branching it maps poorly to streaming architectures such as Cell processors [JB07] or GPUs.

Secondly, the discrete visibility may require high supersampling rates to arrive at the correct radiance values when many small polygons project to a single radiance cell. This is mainly due to the fact that the visibility computed at the centre of the cell may not be representative for the area of the cell and since clipping is performed only against candidates indicated by the visibility, significant error can be introduced by either ignoring contributing candidates or ignoring occlusion of indicated candidates. Each extra (super)sample requires independent execution of the clipping kernel. Furthermore, since the discrete visibility is computed using graphics hardware the pixel data must be fetched from GPU memory to system memory requiring a system bus transfer [3]. These are notoriously expensive since the system bus is about an order of magnitude slower than the internal bus on the GPU even with PCIExpress [PCI02]. The original VLF implementation [SMKY04] fetched a full PSF image of visibility for each receiver polygon, an obvious optimisation is to only fetch the portion of the pixel map which covers the receiver face since this will in many cases be a fraction of the full map. However, this only gave

[2] In fact the clipping function is linear in the number of irradiance cells that is clipped, but this can be assumed to be a constant.

[3] This is not true of all architectures. I.e. games consoles such as the Playstation3 [KBLD08] and Xbox360 [SP07] employ a unified memory architecture allowing the "CPU" to access pixel data directly.

moderate improvements since there is a high per fetch operation overhead. It turns out that in practice, the pixel fetching rate is proportional to the number of pixels in the batch.

Also, the irradiance↔radiance transfers have no notion of *level of detail*. Transfer accuracy is not dependent on the projected area of the polygon in question since every irradiance cell will be projected and clipped independently. This is in contrast to the data structure, which naturally enforces a cosine distribution upon the radiance samples allotted to a polygon. So a transfer to a PSF that is nearly parallel to the polygon will require clipping of each irradiance cell regardless of the fact that they project to a very small number of radiance cells *and* the fact that they contribute relatively little to the end result. Essentially, the contributions made are averaged after projection and clipping in radiance space rather than averaging in polygon (irradiance) space and projecting far fewer and larger areas. It would be desirable to be able to treat groups of irradiance cells in unison for such transfers.

Finally, since visibility is $O(N)$ and must be executed once for each polygon the overall complexity of the algorithm is $O(N^2)$, where N is the number of polygons. This works well for scenes containing few polygons but scales poorly to more complex environments.

To summarise, the problem is two-fold. Practically, the algorithm is difficult (if not impossible) to implement efficiently on current hardware. Theoretically, the complexity of the algorithm - $O(N^2)$ - is such that even if an efficient implementation presented itself, the benefits of any such low-level optimisations would be outweighed by the overall complexity characteristics when increasing the number of faces.

Discrete Transport – Point Sampling

An important contribution of this thesis is applying point sampling to the VLF transport problem forming the core of the approach presented in [MKS07]. A discrete point sampling basis resolves many of the issues that makes the previous approach [SMKY04] so inefficient; in particular it does away with any clipping. Moreover, it makes the algorithm map well to streaming architectures such as the GPU, which will be discussed further in Chapter 5. The technique proposed here solves the irradiance-radiance, radiance-irradiance and irradiance-irradiance resampling problems by introducing a grid↔grid resampling scheme that can transport to and from PSF space.

The problem is to solve Equation 4.1. A straightforward solution to a resampling scheme without continuous clipping is to replace the clipping operator with an operator that estimates the area of overlap by point sampling. In order to determine this area draw a large number of point samples across the receiver quad (radiance cell) and count how many of them fall inside the source parallelogram (projected irradiance cell), the overlap area is then $A_{src \cap rcv} = A_{rcv} \frac{N_{hits}}{N_{total}}$. A black-box replacement like this, however simple, retains many of the problems with the clipping approach. Rather than estimating areas, the goal is to sample the *source signal* appropriately. Provided that enough samples are used and that they are unbiased the answer will be a good approximation to the *true* signal. The minimum number of samples necessary is determined by the upper limit of the frequencies of the source. The *Nyquist limit* is one cycle every two texels, so we need to sample the source with at least 2×2 samples per source texel. In

order to determine the number of samples necessary a source parallelogram is simply projected to PSF space and the area is computed. Given this value A'_{src} and the area of the receiver radiance cell A_{rcv} it is straightforward to set up a sampling grid with $c \times \frac{A_{dst}}{A'_{src}}$ samples, where c is a sampling quality parameter (set $c = 4$ to observe the Nyquist limit).

In many cases the above *fixed* sampling approach will use a high number of samples. As discussed in the previous section, a face nearly parallel to the PSF direction will project all irradiance cells to very few radiance cells. In such cases the technique will use up to four samples per irradiance cell and then average the result to arrive at the final radiance value. This may cause *hundreds* or, in extreme cases, even *thousands* of samples to be used for a single radiance cell. This can clearly be improved upon without sacrificing accuracy by observing some contextual features. The point sampling captures *two* separate features of the signal:

- Radiance integral - average exitant radiance of contributing irradiance cells.
- Area integral - visible area of the irradiance cells overlapping the radiance cell.

The first, a surface integral, samples the irradiance cells of the source polygon[4]. The contents of the irradiance map can be considered relatively *low-frequency* since it has been diffusely reflected at least once[5]. The latter integral is simply the overlapping area between the projected polygon and the radiance cell and is only dependent upon the shape of the source.

In fact the number of samples needed to observe the *Nyquist* limit quoted above only applies for a *high-frequency* source, e.g. a black and white checkerboard. In case of uniform irradiance where only the area integral is needed, the sampling can be independent of the resolution of the irradiance cells and can be performed with much fewer samples. In practice a number of samples in between these poles will be necessary given the relatively low-frequency nature of the signal. Furthermore, it is trivial to use stochastic sampling in this setting either using jittered sample positions or drawing samples from a distribution with blue noise properties [Coo86]. This was successfully applied to *distributed ray tracing* in [CPC84] and the theory underlying stochastic point sampling is discussed in more detail in [Coo86]. It effectively allows undersampling the signal and converting structured aliasing to noise, which is much less objectionable to the human visual system.

Finally, it is possible to use an adaptive sampling scheme that bases the sampling density on the projected area of the source, observing the same cosine distribution for the transport as is used for the data structure itself. The idea is that rather than sampling the irradiance cells individually, especially for the case when many project to the same radiance cell and will be averaged, they can be considered collectively in groups, essentially pre-integrating the radiance values. A different texture representation is needed in order to achieve this. Basically, a lookup function that can produce a texture value given an

[4]The naming convention for the irradiance map may be slightly confusing. Unshot irradiance is converted to exitant radiance by applying the surface albedo prior to use, so in fact, for senders, it contains radiance. Receivers use a different temporary copy to collect radiance into, which then contains irradiance at the end of the iteration.

[5]There is one case where this is not true. That case is textured emitters, which can be arbitrarily high-frequency, e.g. a black/white checkerboard. These can, however, be treated as a special case and sampled with more samples if necessary.

ideal (projected) texel size is needed. An image pyramid [Wil83] serves this purpose well, as it contains a set of progressively downsampled texture layers. Texels in the levels correspond to increasingly larger areas when descending into the pyramid, such that a sample for the desired texel size can easily be retrieved by sampling the appropriate level in the pyramid. This allows the sampling density to be based on the *area integral* only as the *radiance integral* is pre-integrated.

The idea of using adaptive sampling densities for the diffuse transport extends further than this. In the case of very complex scenes with finely tesselated models it would be desirable to be able to transport radiance for groups of faces rather than each one individually. As a thought experiment consider that each texel (in the above scenario) was in fact an individual polygon with a single (ir)radiance value assigned. In such context the adaptive solution would be futile unless the principle is extended to groups of polygons as well as texels. A hierarchical organisation of the model would be needed for this to be effective [HSA91]. In this setting the clustering algorithm would need to produce clusters that are roughly planar and thus minimise self-occlusion in most directions. The algorithm given in [GWH01] has parameters that can be adjusted to minimise various features of the clusters, including planarity and shape, and would be appropriate to this work. Since the techniques in this thesis can in principle support any BRDF, the clustering could with advantage attempt to cluster faces based on the BRDF, such that clusters that are mainly diffuse and are likely to be low-frequency can be sampled more sparsely than highly specular high-frequency clusters. These techniques have not been attempted in this thesis and remain one avenue for future work.

The above technique solves the resampling problem and ignores mutual occlusion. In order to compute incident radiance upon a receiver radiance cell visibility is required. This is essentially employing the visibility function $r(p,-\omega)$ (see Section 2.2.3) for each selected sample point. Any visibility method is applicable. The point sampling technique is particularly appropriate since it can easily take advantage of stochastic sample distribution. Also it integrates nicely with irradiance resampling since the same samples can be used to sample the (ir)radiance, area coverage and occlusion rather than, as in the previous method, solving for visibility with a different set of samples in a separate pass.

This chapter represents a core contribution of this thesis. An alternative point sampling scheme has been introduced that resolves many of the issues with the clipping approach. Namely, the relative inefficiency of the clipping technique and the problems with controlling the sample density used. In particular an optional hierarchical scheme can be used to significantly reduce the number of samples used when the source is low-frequency or it projects to a small area on the receiver. The approach is single-pass and can use the same samples for visibility *and* resampling. Furthermore, stochastic sampling is possible trading noise for structured aliasing when violating the Nyquist limit by undersampling. And finally the inner loop of technique is particularly simple and parallelisable making it appropriate for streaming architectures such as GPUs.

Non-diffuse Finite Element Interactions

The following will deal with the two remaining types of transfer:

- intra-PSF radiance-radiance cell interactions

- inter-PSF radiance-radiance cell interactions (reflection)

These interactions are fundamental to the approach and are tightly bound to the data structure. As described earlier the *cells* of the data structure carry radiance. This is stored in two sets; *unshot* outgoing radiance $L_U(\omega,u,v,p)$[6] and *total* outgoing radiance $L_T(\omega,u,v,p)$. Outgoing unshot radiance is the in-scattered radiance from the previous iteration. When using irradiance maps the in-scattered radiance is only due to non-diffuse reflection and in order to arrive at the radiance from polygon p travelling in direction ω within cell (u,v), an appropriate fraction of the irradiance is added to the unshot radiance using the aforementioned resampling scheme. There is a simple 1-1 mapping between the unshot radiance map and the total radiance map and a simple addition of the images is sufficient to update the total outgoing radiance with the unshot radiance of the previous iteration.

For a receiver face the situation is slightly more involved. Each receiver cell can *see* a number of faces that intersect the beam formed by the cell, occlusion amongst them require visibility to be solved. When a map of incoming radiances has been formed, it is scattered off the receiver face and added to unshot outgoing radiance maps in other PSFs for that face. Based on the BRDF of the surface a fraction of the incoming radiance is deposited in the irradiance map for the face. The remaining radiance is added to n other PSFs. A reflection of incoming radiance $L_c(\theta_i,s,t,a)$ from face a in the PSF direction θ_i into the unshot radiance map $L_u(\theta_o,s,t,b)$ is illustrated in Figure 4.12.

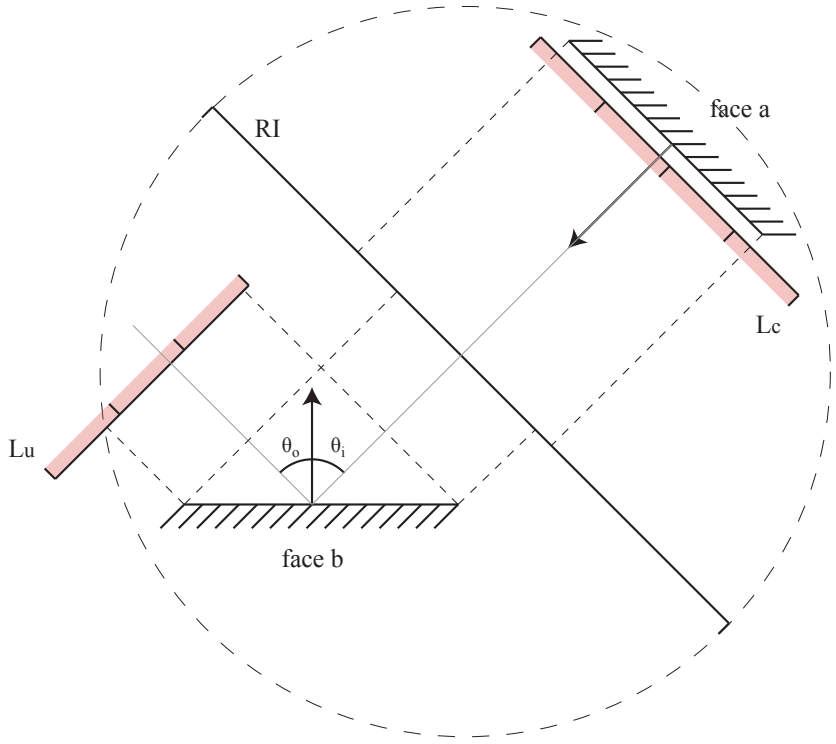


Figure 4.12: Non-diffuse transfer and scattering. Tiles for face a are shown only in the direction of the current PSF, tiles for face b are shown for the current PSF and for the PSF in the reflected direction.

[6] In the following the radiance map indexing will interchangeably be omitting the tile indexing for simplicity.

In theory all PSFs will receive a fraction of the radiance based on the incident direction and the outgoing direction and the BRDF of the surface, but many of those will be zero or very small. For example for an opaque surface all reflections below the hemisphere are zero and can be ignored. Also, specular surfaces have a sharp peak in the BRDF for a narrow band of directions and is nearly zero elsewhere. So in practice n can normally be set significantly lower than the number of PSFs k such that $n \ll k$. In a setting with diffuse and purely specular surfaces (and a mixture thereof) n can be fixed at *one*.

4.3.2 PSF Propagation

The costliest operation by far is solving for visibility when transporting radiance. Each receiver face must compute its visibility to all other faces. Without any extra information such operations must visit each and every other polygon and check whether it is visible. Typically, the number of visible faces, say m, is much smaller than the total number of faces N. This is especially true for the representation in this work since direction is represented explicitly. The visibility of a *beam* formed by a radiance cell is proportional to the depth complexity of the scene when we assume that the cell area is infinitesimal compared to the overall scale of the scene, since in this case the beam can be compared to a *ray*. A main contribution of this thesis is how visibility is computed. By taking into account properties of the data structure and introducing some pre-processing, visibility can be computed in (fast) linear time $O(N)$ for a PSF. Additionally, the proposed method also maps well to a GPU architecture.

For the purpose of discussing the complexity of visibility computation some notation is needed. N is the number of faces in the scene, T is the total number of unique texels on the surface of the faces. Clearly $T \geq N$. For scenes with *high illumination complexity* $T \gg N$. It is assumed that given a global line and a face the intersected texel can be looked up in time O(1). For a given PSF visibility must be solved for each texel in order to compute the light transport.

Unsorted Face Sequence

The two most popular methods for resolving visibility on the fly is rasterisation with the z-buffer [Cat78] and ray tracing [Whi80, Hav01]. In order to solve for visibility for a face along a given PSF all other faces are rendered into a z-buffer aligned to the texture on the face. This essentially solves visibility for all texels on the given face in a single pass. Thus, the complexity of rasterisation with the z-buffer is $O(N \times T)$. On the other hand the complexity of ray tracing with a kd-tree is $O(T \log N)$ since it is assumed that the complexity of ray tracing a subtree is $\frac{N}{k}$, where N is the complexity of the parent and $k \geq 2$. Thus, descending into the tree will, if the tree is balanced, at least halve the number of faces in each step. Because of this it seems obvious that ray tracing should be superior to rasterisation with the z-buffer.

However, fill rates of current hardware are so efficient that in many cases the per pixel cost in the complexity for rasterisation can be ignored such that $O(N \times T) \sim O(N^2)$. This is particularly true for scenes with low to medium complexity, which are targeted in this thesis. In ray tracing traversal of the spatial subdivision structure, which is executed per pixel, is relatively expensive and is only worthwhile when it can cull a significant number of faces. So in practice there is a point where the two scaling curves

cross and this point is dependent on the relative scale of T and N and also on the hardware platform used.

The original VLF method introduced in [SMKY04] was targeted at low complexity scenes with high complexity illumination stored in view-dependent textures. In this scenario rasterisation worked well and the per PSF complexity of $O(N^2)$ was workable when N was on the order of *tens*. However, propagating scenes with *hundreds* or *thousands* of faces would cause excessive running times. Since the aim of this thesis is to target scenes at higher complexities, *tens of thousands* faces, ray tracing would be the superior choice for solving visibility due to superior scaling characteristics of $O(T \log N)$.

Sorted Face Sequence

Due to the organisation of the VLF data structure it is possible to do even better than $O(T \log N)$. The visibility used during transport can be thought of as 2D images of face indices (and possibly depth) under an orthographic projection as *seen* from a receiver face. The projection of a face p onto a PSF_ω intersects a number of cells for which visibility must be calculated. As discussed earlier each such cell corresponds to an infinite *global* line $\bar{l} = (\omega, s, t, u, v)$ and the aim is to find the index, say q, of the nearest face intersected by the ray originating at the intersection of $\bar{l}$ with p, say $x_p = x_{s,t,u,v,p}$, and travelling in direction ω[7]. Assuming p is facing in direction $-\omega$ the resulting index can be found using the ray casting function $q = r(x_p, -\omega)$[8] (see Section 2.2.3).

As noted in the previous section ray tracing or rasterisation with z-buffering can be used to obtain this 2D mapping. However, these two methods compute the surface intersections from scratch for each receiver and global line since there is no inherent ordering of the face sequence with respect to ω. Now, if we assume that the input faces are sorted by depth along ω an incremental algorithm is possible. Thus, partial results can be reused by traversing the faces in depth order updating a shared global set of surface intersections.

During the pre-process the polygons P_p, $p \in (0, 1, \ldots, k-1)$ are rasterised into each PSF allocating tiles and their associated data structures. However, this step produces lists of tiles ordered by their polygon identifier. Given a tile list $(\omega, s, t) = ((\omega, s, t, 0), (\omega, s, t, 1), \ldots, (\omega, s, t, k-1))$, the tile positions are not related to their relative depth along ω. Sorting each tile-list separately is suitable if the propagation step is *tile centric*. However, if operations cross tile boundaries the ordering information contained in several tile-lists must be merged. Alternatively a separate list of polygon identifiers sorted by depth can be maintained for each PSF. The latter was chosen since certain transport steps are performed at a coarser granularity than the tile level.

Intuitively, the aim is to produce a list of polygon identifiers for a given PSF_{ω_i} that guarantees a depth order for the polygons projected to the canonical PSF coordinate system. This is essentially an orthographic projection along the direction ω_i; $P_p^{\perp \omega_i}$, $p \in (0, 1, \ldots, k-1)$ is the set of polygons projected into PSF_{ω_i}. Let $\prec$ be a binary relation defined on the set $P' = P^{\perp \omega_i}$. If $P'_p \prec P'_q$ then there is an overlap between P'_p and P'_q in which P'_p (partially) obscures P'_q. Thus, the aim is to compute an order $(P'_0, P'_1, \ldots, P'_{k-1})$ upon the elements in P' such that $P'_p \prec P'_q$ implies $p < q$. Of course such an order may

[7] Or $-\omega$ depending on the orientation of p with respect to PSF_ω.

[8] For simplicity of notation it is assumed that the ray casting function returns a face index rather than a surface point.

not exist if there is a cycle of overlapping polygons or if a pair of polygons intersect. For simplicity it is assumed that this is not the case for the scenes used in this thesis in order to avoid polygon splitting, which is a relatively trivial problem [NNS72].

This problem is analogous to the hidden surface problem [NNS72, SSS74, dBOS92] as well as depth sorting algorithms for volume rendering applications [Wil92, SBM94, WMS98, CKM+99]. Newell's method is $O(N^2)$ in the worst case since the z-overlap test may have to be performed for every pair of polygons. For scenes with a (nearly) constant frequency of z-overlap it is $O(N \log N)$ in the average case due to the initial rough sorting step by maximum depth.

A possible improvement in the VLF context would be to exploit the fact that the PSF plane is tiled. By applying the sorting algorithm to the individual tile lists followed by a step that merges them into a single ordering could possibly provide a faster algorithm, since for natural scenes the density of objects in each tile is low compared to the total number of objects. A quad-tree like traversal for merging the tiles could be employed. Clearly, the sorting algorithm would need to be super-linear to gain any advantage in efficiency. This is similar in concept to the multi-tiled sort of convex polyhedra given by Williams et. al. [WMS98]. They use the sorting algorithm devised by Stein et. al. [SBM94] as a starting point. An extension to this would be to also subdivide the tiles in depth forming a 3D uniform grid and then sort each volume separately followed by an extended merging operation. Furthermore, the partially sorted 3D uniform grid could be used directly in a tile centric propagation algorithm using the z-buffer, depth-peeling or ray tracing approach to resolve visibility internally in a cell exploiting the low density of objects in each cell.

So far no use has been made of coherence between sorting orders along similar directions. Given similar directions the sorting order of faces is bound to be very similar so devising a fast algorithm that can "fix" a nearly correct sorting order would be beneficial. Upon sorting along one PSF it is trivial to select the closest neighbouring PSF for an initial sorting and it is very likely that such pair of PSFs would have a very similar polygon ordering requiring relatively few operations to repair the sorted order. This is exactly what Govindaraju's *Vis-sort* algorithm exploits [GHLM05]. The GPU is utilised in this approach to implement the comparison operation $P'_p \prec P'_q$. The method can depth sort a *nearly sorted sequence* in linear time. A nearly sorted sequence is defined in terms of its *measure of disorder*, which is the minimal number of elements that need to be removed in order for the rest of the sequence to remain sorted. A sequence with n elements is nearly sorted if its measure of disorder is $k \ll n$. The complexity of the approach has an *upper* bound of $O((4k+2)N)$, which is linear in N if k is constant. The technique does not inherently handle cycles but can detect them and it could easily be extended to deal with them appropriately using a splitting technique.

The sorting algorithm employed for the implementation in this thesis is similar to Newell et. al.'s [NNS72] list priority approach with a BSP splitting and merging step. This was chosen for its simplicity. In order to improve sorting speed, Govidaraju's superior $O(N)$ algorithm should be implemented and integrated. In the results section (see Section 6.1) a prototype implementation of this technique will be applied to the data sets used in this thesis and the results reported. A fully optimised integration with the

VLF framework will be left as future work.

Using the depth sorted set of faces it is trivial to reconstruct an *exchange list* (see Section 2.2.3) for a given global line. Each face is visited in turn and the face indices (and possibly the intersection point) are written into a linked list. Assuming that the scene is closed and composed of non-intersecting closed polyhedra without cycles, this linked list will contain pairs of unoccluded surface intersections between which energy exchange can occur. These mutually visible pairs can then be visited in order and bi-directional energy transport can be computed. This is done once for each global line $\bar{l}$. This algorithm is clearly $O(2 \times T) = O(T)$ for a PSF since each texel is visited exactly twice. The crucial point of the algorithm is that it can reuse intersections along the global line rather than recomputing them from scratch for each receiver.

In practice it is inefficient to compute energy exchange along each global line individually. A single PSF can comprise more than 1 million global lines at high resolutions or high supersampling ratios. Crucially, no use is made of the coherence across surfaces. There are many operations that can be efficiently computed for a bundle of coherent global lines such as BRDF scattering calculations and mapping to/from diffuse maps. As discussed above the VLF data structure itself is organised coherently by tiling the PSFs (see Section 4.2.4). A tile is essentially a coherent bundle of global lines that intersect the same geometry. In order to handle the radiance transport coherently a means of storing intermediate results is needed since the operation is conceptually breadth first. A per tile approach could be taken utilising an intermediate tile for storing radiance. However, a simpler approach is to store an entire PSF image containing the intermediate radiance results. The resolution of this image is the supersampled PSF resolution. This *radiance interface* (RI) is initially black, or alternatively filled with an environment colour[9], each sender, i.e. a face oriented *along* the current direction of transport is rendered into the RI coloured by its unshot outgoing radiance in this direction. This incrementally updates the exchange list for the set of global rays intersected by the sender face. A receiver face, i.e. a face oriented in the *opposite* direction, can simply collect the radiance directly from the RI without calculating any visibility. This is because each pixel in the RI ($\equiv$global line) stores the outgoing radiance of the closest face, i.e. the face most recently rendered into the RI. This process clearly needs to be done twice once back to front and then front to back, so that the faces reverse roles from receiver to sender and visa versa. Figure 4.13 illustrates this concept.

Initially the *RI* (RI_0) is empty. Then face r updates the *RI* (RI_1) with emitted radiance. Face s receives radiance from (RI_1) and face t updates the *RI* (RI_2) partly overwriting the radiance written by r. Face u then receives radiance from (RI_2) which is radiance partly coming from face r and face t. This process is repeated until all faces have been visited. Then the process is repeated in the opposite direction.

The algorithm just described requires a depth sorted set of input faces. As discussed above, k PSFs can be sorted with a complexity of $O((k-1) \times N + N \log N)$, where k is the number of directions used. This makes the overall complexity of the sorting step for a PSF *nearly* $O(N)$ when many directions

[9]This makes applications of skylighting or environment mapping trivial with the VLF.

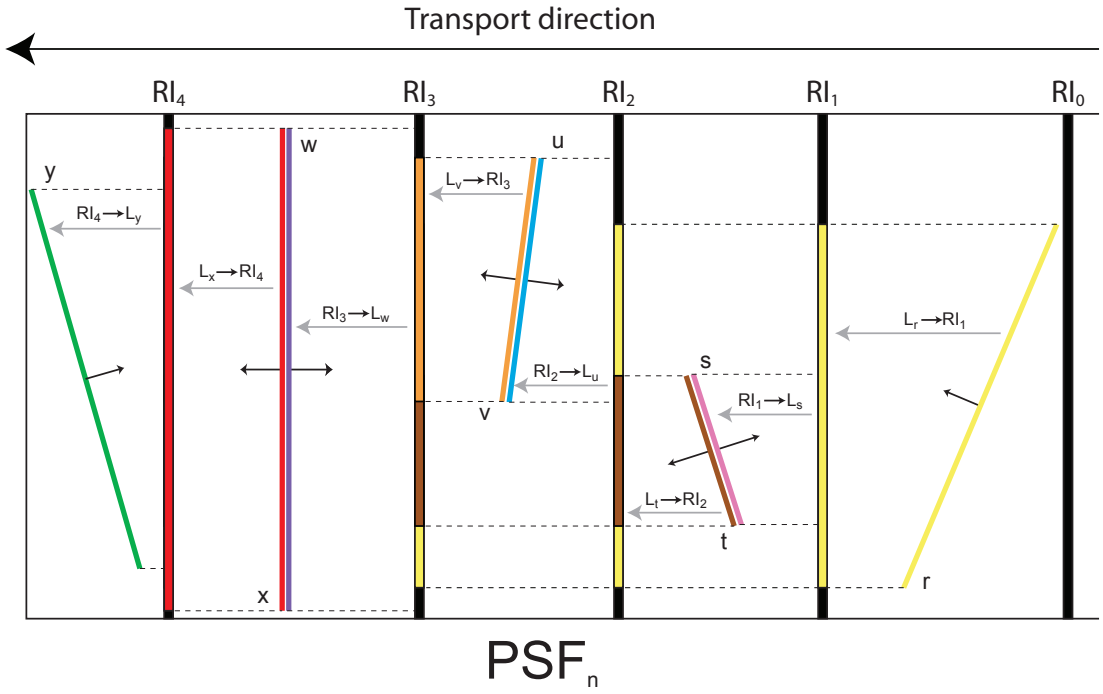


Figure 4.13: Incremental approach for radiance transfer with pre-computed sorting. Transfer takes place from right to left.

are used, since the $O(N \log N)$ term can be amortised over the directions. Consider propagation over I iterations, k directions, N faces and T texels ($T \gg N$). T represents the illumination complexity of the environment, that is, the elements for which visibility must be calculated. Each element is visited twice, once for receiving energy and once for sending accumulated energy. The overall theoretical complexity of propagation is then:

$$\begin{aligned} O^{RI}_{PSF} &= O(2 \times T + \frac{(k-1) \times N + N \log N}{k}) < O(T + N + \frac{N \log N}{k}) \\ O^{RI}_{VLF} &= O(I \times k \times O^{RI}_{PSF}) \end{aligned} \tag{4.3}$$

Comparatively a ray tracing based approach is:

$$\begin{aligned} O^{RT}_{PSF} &= O(2 \times T \log N) = O(T \log N) \\ O^{RT}_{VLF} &= O(I \times k \times O^{RT}_{PSF}) \end{aligned} \tag{4.4}$$

Of course if $T \sim N$, in cases where there is one or very few finite elements per face, ray tracing becomes more effective. This is obvious as the proposed approach would determine the global order and thus visibility and then perform a single transport operation per face. In such a case it would be more efficient to compute the visibility on the fly using ray tracing. However, for cases with moderate scene complexity and high illumination complexity this approach becomes quite efficient, even when purely CPU based, since the finite elements can share the visibility information amongst them. Moreover, in order to compute the light transport without excessive error, texels must be integrated with a number of samples ($\gg 1$). In the proposed approach the visibility complexity is independent of the supersampling

rate, whereas for ray tracing it is not, thus, each extra supersample would need to independently compute visibility.

The approach maps well to graphics hardware and comes into its own on a streaming platform, since the T transport operations can be computed with just N face rendering operations. Each such operation uses the GPU architecture to handle the finite elements across the face in parallel. If it is assumed that a rendering operation is constant time the complexity becomes $O(2 \times N^{GPU} + N + \frac{N \log N}{k})$ for a PSF.

4.3.3 VLF Propagation

In order to achieve equilibrium, radiance is transported until the solution converges. The PSF transport operator is applied for each direction k in the PSF set. Then shot/unshot maps are swapped and a new iteration can commence. For typical scenes, even closed ones, only a few iterations are sufficient as the contribution quickly diminishes due to the repeated application of the BRDF.

As discussed above, each PSF in the set represents a pencil of directions. This solid angle is explicitly represented by the triangle subdivision of the sphere bounding the scenes. Thus, in order to get the correct result the solid angle needs to be integrated so that the radiance stored in the PSF direction is the correct average over the solid angle subtended. In the data structure a single direction is chosen as ”representative” for the PSF. This is the midpoint of the solid angle of the PSF. The reason for choosing the midpoint is that in this direction the projection of the geometry is maximized. However, this does not mean that transport can only be applied in this direction, only that the resultant radiance maps are using this direction for projection. When transporting radiance, a number of samples spread across the solid angle are used and the results are accumulated into the ”representative” direction and weighted appropriately. In the implementation used to generate results for this thesis random sampling and a box filter is used. It would be trivial to replace the sampling with a technique that produces a distribution with blue noise properties (see for example [Bri07]) for faster convergence. As discussed earlier the solid angle subtended by a PSF may vary so the number of samples used for a PSF is proportional to its solid angle in order to ensure uniform sampling over the sphere.

The VLF does not handle very small light sources well. In fact large light sources, which are notoriously hard to sample for *gathering* approaches like path tracing are handled easily with the VLF. The reason for this is that the VLF is a *shooting* approach and thus the angular spread of directions increases with distance and the overlap on a receiver is proportional with the size of the emitter. Figure 4.14 illustrates this effect. The worst imaginable case is a point emitter which will generate a bright point for each direction that intersects a receiver.

Path tracing samples the light sources separately because they are known to contribute significantly to the image. The ideal case (in a gathering approach) is a point emitter that can be adequately sampled with a single ray. The worst case, however, is when the entire scene is an emitter in which case importance sampling the emitters will not work well. A case in point is in scenes with large textured lights. This problem is also pronounced with mostly diffuse scenes where the secondary rays need to sample the diffuse hemisphere in order to estimate the irradiance. A host of complex approaches, like irradiance caching, have been developed to try to remedy this issue in gathering approaches. For the VLF method,

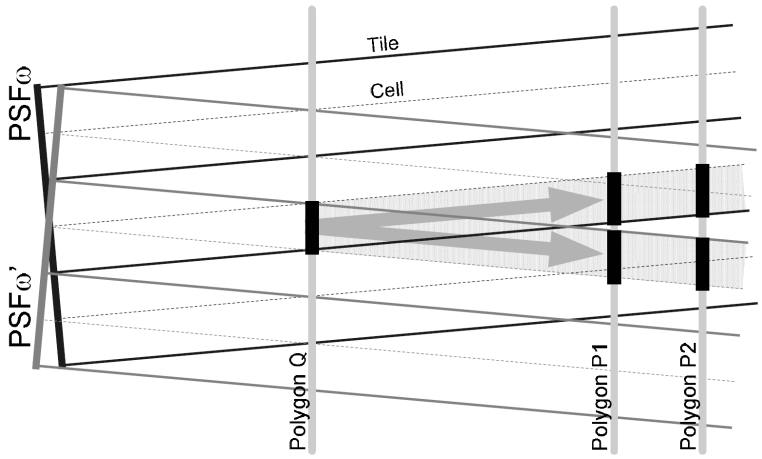


Figure 4.14: Angular spread of fixed directions.

increasing the number of samples used for a PSF can improve the results. It is not necessary to do this for all iterations: typically the first iteration that shoots light from the emitters will cause the largest variance on the receivers and thus needs more samples. After the first iteration emitters will be ”black” and the entire scene will act as an emitter so fewer samples can typically be used. For pathological cases, involving point emitters, it would be more sensible to produce the direct lighting via a different approach like ray casting or even rasterisation with shadow-mapping, which performs well in such cases. Such a step has the effect of turning the geometry of the scene into a big emitter for the second iteration and beyond. This is left as an avenue for future research, in this thesis this problem will be circumvented by avoiding very small emitters and using enough directional supersamples.

4.4 Rendering

To generate novel views from the VLF the directionally dependent radiance stored in the non-diffuse radiance tiles is resampled. As described earlier, the data structure can be formalised as $L_T(\omega,s,t,u,v,p)$. This effectively references a radiance value in direction ω, from a point on p described by the intersection of the canonical ray (s,t,u,v) with p. Due to the discrete representation a PSF matching *exactly* the direction ω is rarely available. The three PSFs $(\omega_i,\omega_j,\omega_k)$ at the vertices of the spherical triangle in which ω falls are used with barycentric weights $(\alpha_i,\alpha_j,\alpha_k)$ for an interpolated value:

$$\begin{aligned} L_T(\omega,s,t,u,v,p) = {} & \alpha_i * L_T(\omega_i,s,t,u,v,p) \\ & + \alpha_j * L_T(\omega_j,s,t,u,v,p) \\ & + \alpha_k * L_T(\omega_k,s,t,u,v,p) \end{aligned} \tag{4.5}$$

When retrieving a radiance value from a radiance map bilinear interpolation is used. This yields a 12 tap filter.

4.4.1 Direct VLF Rendering

Pin-hole and lens based camera models can use Equation 4.5 to retrieve incoming radiance values. Visibility must be determined for each pixel in order to reference the correct radiance map. This can be

achieved with either ray casting or by rasterising an image with z-buffering in false colour using the index of the face, such that each pixel contains the index of the nearest face. This pixel map can then be retrieved from the GPU and used for generating the image on the CPU. This final image can then be presented to the user. Unfortunately, retrieving an image from the GPU is a relatively slow operation even on current hardware.

4.4.2 Irradiance Maps

If irradiance maps are available these can be pre-rendered using hardware texture mapping. A second pass can then fill in the non-diffuse visible pixels using the method above. This is appropriate for moderately sized scenes since rasterisation is typically more efficient than ray tracing for all but very complex scenes. The scenes targeted in this thesis fall in this category. For many viewpoints, depending on the scene, only a fraction of the pixels need to be shaded with the VLF potentially reducing the amount of data that needs to be transferred across the bus to the GPU.

4.4.3 Specular Reconstruction

For VLFs with perfectly specular surfaces such as mirrors, the limited directional resolution will often lead to artifacts, unless, of course, the field-of-view (FOV) of the camera used is very narrow and the image is of a relatively low resolution. But in VR scenarios this is rarely the case, as many immersive projection technologies, such as the CAVE^{TM}, will need a quite large FOV and high image resolutions. In order to support perfectly specular surfaces these can be reconstructed using ray tracing. When an eye ray strikes a non-diffuse surface the ray is reflected and is traced in this manner until it strikes a diffuse surface or some maximum number of reflections has been reached at which point the VLF is looked up in the reverse direction of the incident ray.

For flat specular surfaces this can even be achieved with rasterisation using stencil reflection mapping [Kil02], although this is typically only practical for scenes with very few mirrors, since each mirror requires a separate rasterisation pass of the entire scene. Also, stencil reflection mapping has problems with mutually visible mirrors.

4.4.4 Final Gather

Generalising the above approach for all surfaces, not just perfectly specular ones, yields an approach comparable to the *final gather* approaches developed mainly for photon mapping algorithms [Jen95, JC95]. Presumably, a low resolution VLF would be sufficient for such use making this a quite attractive approach for high quality applications. This is left as a direction for future research.

4.5 Summary

In this chapter several important contributions have been made to the VLF propagation step. At the element level the visibility and computation steps have been integrated and transport involving diffuse surfaces has been made adaptive. The computationally intensive continuous clipping approach has been replaced with a significantly faster point sampling approach yielding indistinguishable quality. At the PSF level a novel incremental approach utilising a *radiance interface* (RI) has been introduced improv-

ing complexity from $O(N^2)$ to *nearly* $O(N)$. The complexity has been compared to an ideal ray tracing algorithm working on the same data structure, and it has been shown that the incremental RI approach is superior to the ray tracing approach for scenes with moderate geometric complexity and high illumination complexity.

The contributions as a whole will in general improve the complexity by several orders of magnitude compared to the VLF approach taken in [SMKY04]. This makes it possible to use the VLF in a VR context operating on scenes with tens of thousands faces and millions of globally illuminated finite elements in minutes rather than days.

Furthermore, the approach developed is suitable for implementation on a GPU architecture, paving the way for an even more efficient approach.

Chapter 5

Virtual Light Fields on the GPU

5.1 Introduction

In the preceding chapter a new method for VLF propagation was introduced, which uses point sampling and incremental visibility computations. The efficiency was shown to be in some cases superior to ray tracing, but the method was carefully designed with parallelism and a streaming architecture in mind and comes into its own on a streaming platform such as a programmable Graphics Processing Unit (GPU) architecture.

In this chapter a novel propagation algorithm running entirely on the GPU is presented. Additionally, a method for rendering globally lit images directly from the GPU data structure is introduced. This can render novel views from a VLF without CPU involvement at real-time frame rates and high resolutions.

An introduction to the GPU architecture is given in the next section. In Section 5.3 the GPU propagation approach is explained. In Section 5.4 GPU rendering from the propagated data structure is described. Finally a summary is given in Section 5.5.

5.2 The GPU Architecture

The Graphics Processing Unit (GPU) has become an integral component of today's mainstream computing systems. Over the last decade it has evolved from an optional add-on for accelerating selected 3D games to a powerful fully programmable streaming engine for solving computationally demanding problems. It has been successfully applied to a diverse set of complex problems achieving order of magnitude improvements over highly optimized CPU implementations [JDO08]. The GPU is designed for problems with large computational requirements that can be mapped to an algorithm with substantial parallelism where latency is not crucial. The GPU is primarily designed for rendering algorithms, which allows for latencies proportional to those of the human visual system operating on a millisecond scale. Consequently, the architecture is based on a deep feed-forward graphics pipeline where thousands of primitives are being processed in the pipeline at any one time greatly improving throughput.

The GPU pipeline is divided into stages, each taking a large number of input elements and the output of each successive stage becomes input to the next stage. Individual tasks map to the stages of the GPU architecture. This results in *two* kinds of parallelism: *Task parallelism* is exposed since multiple

stages can be computed at the same time. *Data parallelism* is exposed since many elements inside a given stage can be computed in parallel. Historically, this has informed the design of the architecture. The architecture divides resources (i.e. die area [1]) among stages and the hardware for a stage can be customised to best exploit the parallelism of the particular task mapped to that stage. The weakness of this approach is that the performance depends on the slowest stage in the pipeline and the fixed architecture prevents dynamic load-balancing. Modern GPUs have addressed this problem by introducing the *unified shader architecture*. This pipeline design has a pool of general purpose programmable compute units that can be dynamically assigned to the stages of the pipeline making dynamic load-balancing possible. Recent GPUs feature a large number of fine grained unified parallel stream processors internally (e.g. *AMD Radeon HD 2900XT* features 320 [JDO08]). The end result is a unit with massive computational capability and memory bandwidth (e.g. *Nvidia 8800 GTX* sports in excess of 330 Gflops and a memory bandwidth of 80+ GB/s [JDO08]).

The graphics pipeline takes as input a stream of geometric world space primitives described by their vertices and some attributes. The primitives are projected to screen space, rasterised and shaded to finally end up on screen as a collection of pixels. This abstraction exposed by the main graphics APIs *OpenGL* and *Direct3D* has with few exceptions [2] remained unchanged. However, suitable stages have evolved from fixed-function to fully programmable ones [3]. Figure 5.1 outlines the Microsoft Direct3D10 pipeline [Bly06].

In the following the pipeline stages are summarised in greater detail:

- **Input Assembler (IA):** Collects vertices with attributes from input streams and converts to a canonical floating-point format. Can also perform *instancing* by replicating a block of vertices a number of times.
- **Vertex Shader (VS):** Fully programmable stage that can transform vertex data. Input and output is a single vertex. Commonly used to transform vertices from object space to clip space.
- **Geometry Shader (GS):** Fully programmable stage that can generate additional geometry. Reads the vertices of input primitives and generates the vertices of zero or more output primitives. Commonly used for subdivision surfaces.
- **Stream Output (SO):** Optional stage that can output data from the GS stage to stream buffers. These can act as input for the IA stage allowing high performance multi-pass geometry processing without the need for CPU involvement.
- **Rasterisation Stage (RS):** Fixed function stage handling clipping, culling, perspective divide, viewport transform, primitive set-up, scissoring, depth offset, and fragment (pixel with attributes) generation. Reads the vertices and attributes of a single input primitive, interpolates them, and produces a series of output fragments.

[1] Area on the chip surface.

[2] The most notable exception is the addition of a *geometry shader* (GS) stage.

[3] The *rasterisation stage* is an exception, this remains a fixed-function stage since it can be implemented very effectively in hardware.

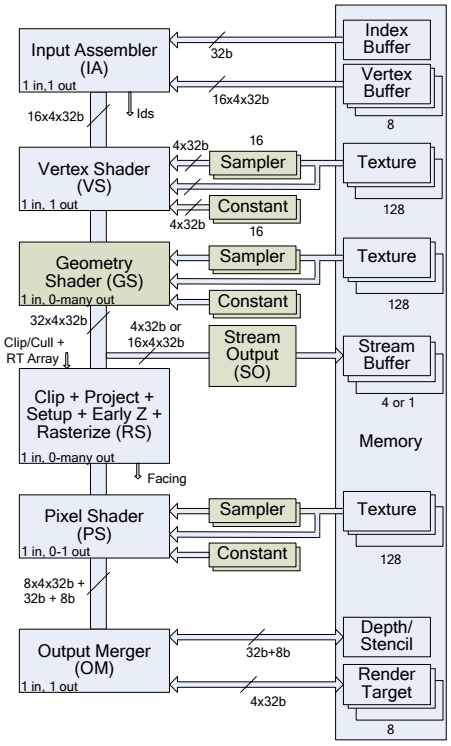


Figure 5.1: Microsoft Direct3D10 GPU rendering pipeline.

- **Pixel Shader (PS):** Fully programmable stage that reads a single input fragment and produces a single shaded output fragment (optionally with depth).

- **Output Merger (OM):** Performs stencil, depth testing, and blending on the fragments. Writes these to one or more *render targets* (colour buffers).

This organisation has proven to be quite efficient. Individual vertices and fragments can be computed in parallel and the stages are independent. While some fragments are being shaded and output, other primitives can be projected and rasterised. This yields a fully saturated pipeline as long as input is being continuously streamed. In the recent past changes to the configuration of the pipeline, *state changes*, such as changing VS or PS programs often caused the pipeline to stall while being flushed resulting in poor throughput. This has been addressed in recent generations of hardware and APIs [Bly06] making it easier to maintain full saturation.

Porting algorithms to run on the GPU was until recently a daunting task due to language, instruction set and data type limitations as well as significant variation in the capabilities of the GPUs. Also, until recently a graphics pipeline abstraction had been imposed on developers making it difficult to map *general purpose* algorithms to the GPU (GPGPU) elegantly. For example, memory on a GPU was restricted to reside in a 2D texture accessed via a texture lookup. Moreover, the APIs used terminology like *vertices*, *textures*, *fragments* and *blending* etc. However, high level GPU languages have matured, instruction sets are now complete, and there is support for general purpose data types such as integers and floating point

values throughout the pipeline. Also, software environments that expose a higher level abstraction of the GPU as a streaming processor are now available making it easier to develop *non-graphics* GPGPU algorithms in an elegant and maintainable manner (see [JDO08] for details). However, in this thesis the graphics abstraction is suitable for the problem at hand so this area of research will be disregarded and the algorithms will be described in terms of a graphics pipeline.

The VLF algorithm certainly satisfies the aforementioned characteristics of algorithms that map well to the GPU: *large computational requirements*, *substantial parallelism*, and *favouring throughput over latency*.

5.3 Propagation on the GPU

This chapter recasts the propagation algorithm in terms of rendering operations. The algorithm is realised on programmable graphics hardware using rasterisation to compute visibility incrementally and texture mapping using render-to-texture functionality to perform the radiance transfer.

5.3.1 GPU Data Structure

In order to perform the propagation on the GPU the radiance data must be available locally on the GPU. Reading and writing to the GPU is still prohibitively expensive: memory access between main memory and GPU memory is still an order of magnitude slower than local memory access on the GPU (even with PCI-Express), and streaming algorithms requiring data from the CPU will in general stall and waste the potential power of the unit.

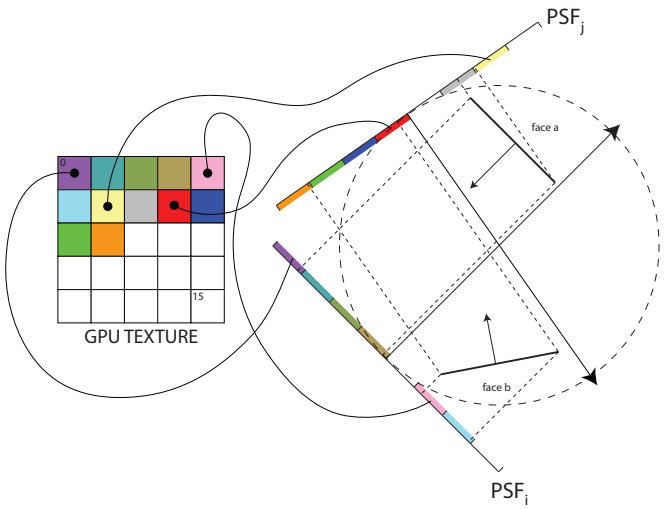


Figure 5.2: GPU data structure for non-diffuse faces is sparsely tiled and linearly stored in a large texture. Here two non-diffuse faces are projected to PSF_i and PSF_j and the tiles needed are shown.

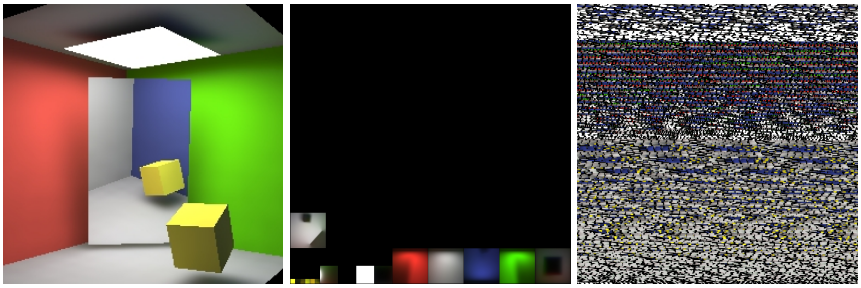
There are two types of data that must be available to the GPU: textures containing radiance caused by diffuse illumination over a polygonal patch, and tiles that contain radiance generated by non-diffuse illumination (see Section 4.2.5). A diffuse surface has a single texture map assigned to it since its emitted radiance is independent of outgoing direction. On the other hand non-diffuse surfaces have radiance maps representing unique outgoing radiance for each direction in the set of global directions with one

texture map for each PSF direction. In order to save memory these maps are tiled, as described earlier, and only tiles overlapping the face (projected to the PSF plane) are stored (see Figure 5.2).

The number of non-diffuse tiles for a scene can be high, for example the scene shown in Figure 6.14(a), which includes specular surfaces has ∼15k tiles and this is a relatively simple scene. It would be inefficient to store these tiles as separate textures since binding a texture to a framebuffer for render-to-texture operations has an associated overhead. The solution is to store many tiles packed in large textures on the GPU. For similar efficiency reasons diffuse textures are also stored in texture atlases. The atlassing for a Cornell box scene is illustrated in Figure 5.3.

In addition to the tiled texture map, an auxiliary data structure is needed that can locate a tile in a texture for a given PSF and face. Figure 5.2 illustrates this; two faces have tiles stored in two PSFs which are linearly stored in a single texture. For example face b has four tiles associated with PSF_j, these are stored at the linear positions 8-11 (assuming position 0 is the upper left hand corner and indexing order is left to right, top to bottom).

During propagation temporary storage is needed for unshot (ir)radiance. Tiles and diffuse textures store three copies of each texture; the *current* map storing the outgoing (ir)radiance in the current iteration, the *next* map storing the received (ir)radiance in the current iteration (to be shot out in the next iteration) and, finally, the *total* map containing the accumulated (ir)radiance over all iterations. In practice two copies would be sufficient but during development it is simpler to work with a purely breadth-first approach. During rendering, only the latter set is needed and the *current* and *next* maps may be discarded.



(a) Scene (b) Diffuse texture atlas (c) Radiance tile atlas

Figure 5.3: Diffuse and tile atlases.

5.3.2 Incremental Radiance Transport

In Section 4.3.2 an incremental approach for solving the necessary visibility was introduced. This requires a sorted sequence of faces in the direction of the PSF, say $S^{\omega}_{1\ldots N}$, which can be made available using one of the methods outlined in Section 4.3.2.

In order to map this algorithm to the GPU the transport operations have been recast in terms of rasterisation operations. Algorithm 5.1 illustrates the overall transport step for a single PSF[4] (see also Figure 4.13).

[4]For clarity the algorithm has been simplified to support only mixtures of specular and diffuse BRDFs.

Algorithm 5.1 Incremental GPU transport operator for a PSF in direction ω

Require: $S^{\omega}_{1\ldots N} \leftarrow$ list of scene faces sorted by depth along direction ω

1: **for all** i such that $1 \leq i \leq 2$ **do** {do two passes, one in each direction}
2: **for all** i and f such that $f = S^{\omega}_i$ **do** {for each face f in the sorted sequence}
3: **if** $\vec{n}_f \cdot \omega > 0$ **then** {is f facing along ω?}
4: $D^f_U \Rrightarrow RI$ {render irradiance to RI}
5: **if** f is specular **then**
6: $L^f_U(\omega, s, t, f) \Rrightarrow RI$ {render unshot radiance to RI}
7: **end if**
8: **else**
9: $RI * w(PSF_{\omega}) \Rrightarrow D^f_{U'}$ {render RI to irradiance texture}
10: **if** f is specular **then**
11: $RI * brdf(\omega, \phi) \Rrightarrow L^f_{U'}(\phi, s, t, f)$ {render RI to reflected PSF_{ϕ}}
12: **end if**
13: **end if**
14: **end for**
15: $\omega \leftarrow \overline{\omega}$ {invert PSF direction for second pass}
16: $S^{\omega} \leftarrow \overline{S^{\omega}}$ {invert sorted sequence}
17: **end for**

Radiance transport

Although such a scheme would be possible to implement on the GPU [Nov05], a much more efficient approach is to use the built-in hardware texture mapping. Current hardware supports floating point textures, and with mip-mapping and anisotropic filtering this is viable as long as enough samples are taken when performing the texture mapping operation. Two such texture mapping operations will be necessary in this context: one that maps a face onto the *RI* and one that maps the *RI* onto a face. By ensuring that the *RI* is supersampled, aliasing can be avoided.

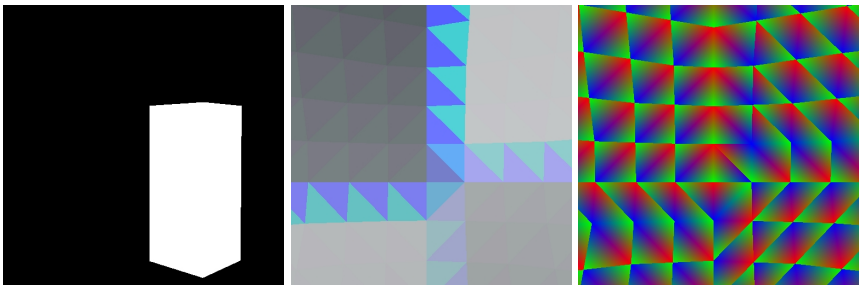
Non-diffuse Scattering

In order to perform the non-diffuse scattering (see Section 4.3.1) a backwards mapping operation is performed. The corners of each tile in the reflected direction are transformed into the PSF of the current transport direction via the plane of the reflecting face. These serve as texture coordinates into the *RI* and the scattering is performed by a texture mapping operation weighted by the barycentric weight and the *BRDF* of the reflector.

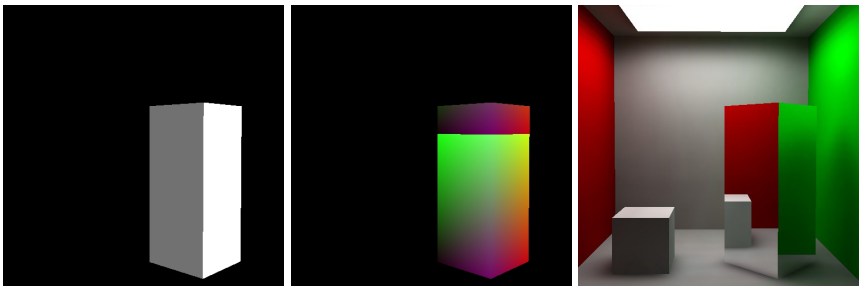
5.4 Rendering from the VLF on the GPU

When the VLF propagation step has converged, the GPU can render novel views from the data structure by interpolating between samples stored in the diffuse textures and non-diffuse view-dependent radiance tiles. Diffuse surfaces can be rendered directly using texturing with the diffuse textures available in the

irradiance texture atlases. The GPU performs interpolation efficiently in this case. Flat specular faces can be rendered with ray-tracing by recursively following a view ray reflected in the specular face until it strikes a diffuse face where the visible radiance can be collected. A similar idea, often used in real-time VR applications, is to use the stencil buffer to render a reflected view of the scene as seen through the specular face and then paste this onto the face with texturing [Kil02]. These methods are only efficient if few specular surfaces are present in the scene and do not apply to, for example, glossy BRDFs.



(a) Non-diffuse stencil (b) *Pass 1:* PSF ids (c) *Pass 2:* PSF weights



(d) *Pass 3:* Face ids (e) *Pass 4:* Hit pos (f) *Pass 5:* Non-diffuse shading

Figure 5.4: Rendering passes.

A more general method is to resample images from the directionally dependent radiance stored in the non-diffuse radiance tiles. As described in Section 4.2.5, the data structure can be formalised as $L_T(\omega,s,t,u,v,p)$. This effectively references a radiance value in direction ω, from a point on p described by the intersection of the canonical ray (s,t,u,v) with p. Due to the discrete representation a PSF matching *exactly* the direction ω is rarely available. The three PSFs $(\omega_i,\omega_j,\omega_k)$ at the vertices of the spherical triangle in which ω falls are used with barycentric weights $(\alpha_i,\alpha_j,\alpha_k)$ for an interpolated value:

$$\begin{aligned} L_T(\omega,s,t,u,v,p) = {} & \alpha_i * L_T(\omega_i,s,t,u,v,p) \\ & + \alpha_j * L_T(\omega_j,s,t,u,v,p) \\ & + \alpha_k * L_T(\omega_k,s,t,u,v,p) \end{aligned} \tag{5.1}$$

In order to compute the values necessary to index into Equation 5.1, four off-screen passes are rendered (see Figure 5.4). A fifth and final pass performs the final shading producing the globally lit image. In order to identify non-diffuse pixels in the image plane an optional stencil image can be produced by rendering the non-diffuse polygons to an off-screen target (see Figure 5.4(a)). This can

serve to limit the computation performed in each subsequent pass to only non-diffuse pixels.

In *pass 1* the camera is placed at the centre of the unit sphere and the spherical triangles are rendered in false colour to a texture. This produces the indices of the three nearest PSFs $(\omega_i, \omega_j, \omega_k)$ for each pixel (see Figure 5.4(b)). This is repeated in *pass 2*, this time setting vertex colours for each spherical triangle to $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$. The GPU interpolates this over each triangle, resulting in a texture with three barycentric weights for each pixel (see Figure 5.4(c)). *Pass 3* serves to determine p, this time rendering the scene geometry in false colour, yielding a texture with a face identifier for each visible non-diffuse pixel (see Figure 5.4(d)). *Pass 4* renders the scene geometry again where each vertex is coloured with its world coordinate position, interpolation across the geometry produces a texture with the world coordinate position of the intersection of the viewing ray for that pixel with the face p (see Figure 5.4(e)). Note that ray casting could easily replace these last two passes. A fifth and final pass renders the final radiances to the image. For each pixel this is achieved by mapping the hit position to each of the three PSFs by applying the respective $M_{WC \to PSF}$ matrix (see Section 4.2.3) to the hit position, producing an (x,y,z) value in canonical PSF coordinates where (x,z) trivially maps to a tile/cell pair (s,t,u,v). The tiled data structure is then looked up and a radiance value for each PSF is weighted by its corresponding barycentric weight and written to the image. This is illustrated in Figure 5.4(f).

Performance is dependent on the time taken to resolve visibility (*pass 3*), the remaining passes and radiance retrieval is small constant time per pixel. Either ray tracing or rasterisation can be used to resolve the visibility, here we use the latter. One of the main points of the VLF approach is that global illumination values can be retrieved directly from the data structure, no further shadow rays or sampling is necessary. This results in stable, predictable frame-rates, which is of great utility in VR applications [BH95].

5.5 Summary

The aim of this work has been to significantly improve propagation performance for moderately complex scenes in the VLF context. The incremental algorithm (see Section 4.3.2) scales linearly $O(N)$ in the number of input polygons. This translates to several orders of magnitude improvement (>2000x) in performance over previous work [SMKY04] for scenes with <200 faces. More complex scenes would be intractable with the original method due to the $O(N^2)$ complexity.

This chapter presented an algorithm that runs entirely on the GPU requiring the CPU only to do minor bookkeeping tasks. The method supports textures that are fully integrated in the propagation stage, replacing the constant K_d term with one that varies across a surface as defined in a high dynamic range (HDR) texture. Textured emitters are similarly supported, see Figure 6.14(b). Also, emitters are not distinguished from other surfaces. Any surface can have emission; propagation and rendering time is invariant to the number and cumulative area of emissive surfaces, which is a desirable property achieved by few global illumination methods. Skylight rendering is also supported without any additional performance penalty.

Rendering from a converged VLF can be done in real-time with very stable frame rates, making this a practical solution for virtual reality applications, where the frame-rate *must* be real-time and con-

stant. Even temporary drops in frame-rate can cause the subject to lose orientation, and maybe cause motion sickness [BH95]. Lack of stable frame-rates is a weakness of many caching algorithms where a sudden change in viewpoint can produce a view that is not fully represented in the cache, causing a temporary drop in fidelity or frame-rate. Similarly, dynamic techniques such as ray tracing for global illumination can also exhibit variable frame-rates when the viewpoint changes from a complex region to a less complex region in terms of illumination.

Chapter 6

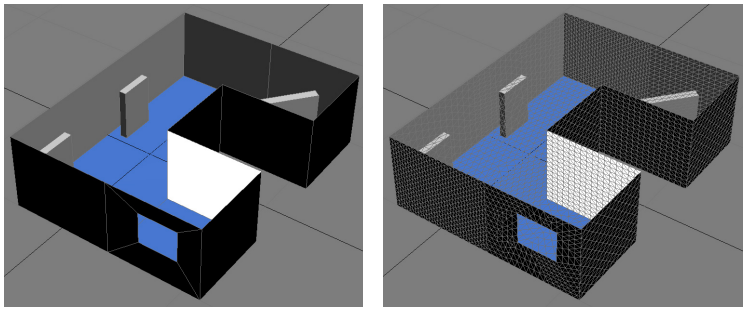
Results

In this chapter results are presented for the VLF-GPU method.

The implementation was written in Cg [MGAK03] and OpenGL on a GeForce 8 series GPU. Recent OpenGL 2.0 extensions were heavily used, such as floating point textures and framebuffer objects for simple render-to-texture functionality. In order to avoid severe penalties for texture state changes, a texture atlas was employed for textures. Further, costly state changes such as setting projection and viewing matrices were avoided by storing matrices for the global set of directions on the GPU and setting them there in a vertex shader; thus exploiting the high internal bandwidth of the GPU rather than sending the matrices from the CPU to the GPU across the (much slower) PCI-Express bus. Currently, diffuse, specular, mixed diffuse/specular and glossy (modified Phong) surfaces are currently supported.

6.1 Sorting Performance

As discussed earlier, the sorting implementation integrated in the VLF framework is based on the Newell sorting with BSP subdivisions, splitting the PSF into volumes that are sorted separately and then hierarchically merged (see Section 4.3.2). This has a complexity of $O(N \log N)$ in the number of faces. While this works well for scenes with predominantly small uniformly distributed faces, scaling is not necessarily guaranteed for large scenes with arbitrary distributions.



(a) Original Maze scene. (b) Maze scene tesselated to 10K faces.

Figure 6.1: The Maze scene was tesselated to varying densities for the Vis-Sort scaling experiment. This shows the original scene and a version tesselated to 10K faces.

The current implementation uses a fixed subdivision scheme that subdivides two levels along the

PSF and an additional split along the width and height respectively, yielding 16 BSP volumes. Split planes are placed at the geometric median of the volume. For scenes of the scale used in this thesis further splits did not significantly improve sorting time. Sorting the Atenea scene in Table 6.1 with 9.4K faces took 126 seconds [1] with no subdivisions and 19 seconds using BSP splits yielding a 6.76x improvement, which is significantly lower than the ideal 16x improvement. This is due to non-uniform distribution and the time taken to merge the lists.

This can, however, be improved upon. As discussed in Section 4.3.2 the Vis-Sort algorithm [GLM04] can sort a nearly sorted sequence in $O(N)$ time. The method achieves this by using GPU occlusion queries to determine correctly sorted subsequences and permuting out of sequence faces until the sequence is sorted correctly. Sorting an already sorted sequence will cause exactly $2n$ queries to be issued. The assumption is that the number of out-of-sequence faces k is low $k \ll n$, such that the number of queries necessary is $\sim 2n$. The number of queries issued when sorting directly determines the running time of the algorithm. The number of queries that can be issued per second depends on the fill-rate of the GPU, to some extent the graphics API used but, mostly, on how the implementation schedules the occlusion queries. It is imperative that the application schedules the queries in batches and works asynchronously with the GPU. Issuing an occlusion query and immediately requesting the result - the so called *stop-and-wait method* - will cause the GPU to stall and perform poorly. It is, however, very simple to implement. For comparison the method implemented in [GLM04] can issue on the order 1274K occlusion queries per second [2] on an *nVidia Geforce FX6800 Ultra* GPU. Scaling this number to the fill-rate of the *nVidia Geforce 8800 GTS* GPU [Rom10] it will be able to issue $\sim 2070K$ occlusion queries per second with the implementation of Vis-Sort given in [GLM04].

By using the sorted sequence from a neighbouring already sorted PSF this can be exploited in the VLF context. In order to demonstrate the suitability of this approach a simple *stop-and-wait* prototype implementation was added in order to illustrate the scaling characteristics achievable. To this end the number of occlusion queries required to sort increasingly complex scenes were recorded and listed in Figure 6.2.

In order to illustrate the scaling a reproduction of the Maze scene used in [DSDD07] was tesselated to between 1K and 65K triangles (see Figure 6.1). Additionally, a variety of other scenes were plotted also. Each scene was sorted along 513 PSFs. When sorting PSF_n the closest PSF_m where $m < n$ was used, this is straightforward to find given the hemispherical subdivision used for the VLF (see Section 4.2). For PSF_0 the (arbitrary) order of the faces given in the scene file was used.

As can be seen in Figure 6.2 scaling is linear in the number of faces. All scenes are clustered near the linear scaling trend exemplified by the Maze scene. The *stop-and-wait* implementation used was able to perform $\sim 30K$ occlusion queries per second due to GPU stalls. This translates to ~ 359 seconds to sort the Atenea scene for example. However, extrapolating to the achievable occlusion query performance of the appropriately batched and optimised implementation given in [GLM04] paints a different picture.

[1] This is *per thread* as the implementation divides the work among 4 threads.

[2] A scene with 91K faces is sorted at 7 frames per second. Since at least $2n$ occlusion queries are required to sort a sequence $91K * 2 * 7 = 1274K$ occlusion queries can be issued per second.

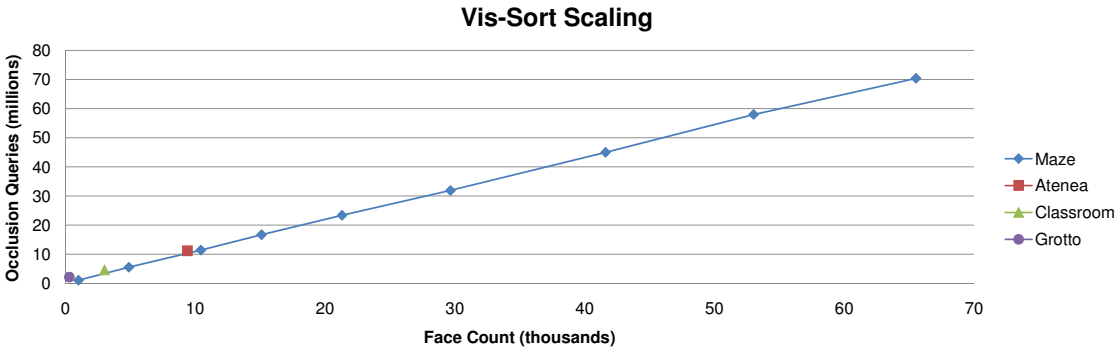


Figure 6.2: Vis-Sort scaling shows number of occlusion queries issued for a selection of scenes when sorting along 513 PSFs. Vis-Sort is $O(Q)$ where Q is the number of queries issued. Actual performance depends on the fill-rate of the GPU. The Maze scene (see Figure 6.1) was tesselated with increasing density illustrating the linear scaling. Notice how the other scenes cluster near this trend.

The query performance is roughly 69x faster [3], which is not unlikely given that *each* occlusion query in the *stop-and-wait* implementation will incur a GPU stall. Using this multiplier the Atenea scene, for example, could be sorted in ~ 5.2 seconds. Furthermore, this also guarantees that scaling is linear and will take advantage of the improvements in fill-rate of future GPUs. For example, the top of the range *nVidia GeForce GTX 295* has a fill-rate in excess of 32 Gpix per second, which comparatively would be able to sort the Atenea scene in 1.7 seconds and the most complex Maze scene (65K faces) in 10.2 seconds.

6.2 Propagation Performance

In order to illustrate the scaling characteristics of the incremental propagation algorithm, the collection of tesselated Maze scenes are used again. One version is purely diffuse and another version has a mixture of diffuse and specular surfaces. For the specular scaling each progressively more complex version of the Maze scene had a proportion of specular/diffuse faces with the most complex version having 1024 such faces. Diffuse element resolution varied between 37K and 4M and specular elements varied between 46K and 8.6M.

As expected, the incremental propagation algorithm exhibits linear scaling (see Figure 6.3). Note that the Atenea scene in particular overshoots the graph slightly. This is due to that scene having a larger proportion of the surface area taken up by specular faces. The slight upward bend in the diffuse scaling graph is due to the diffuse resolution of the most complex instance of the Maze scene requiring *three* 2048x2048 diffuse atlases rather than two. This causes more frequent render target changes and therefore affects the performance slightly.

Figure 6.4 shows the effect of using tiling on the propagation time and memory consumption. A fixed radiance map resolution of 64x64 was used and this was split into tiles at different resolutions. At one end of the spectrum a 1x1 tiling resolution was used with a 64x64 per tile resolution. This results in a

[3] Using the query performance estimate of 2070K queries per second calculated above.

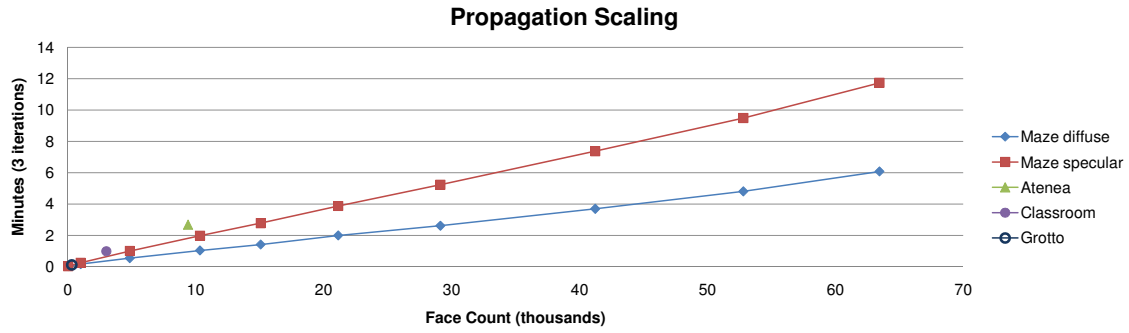


Figure 6.3: Propagation scaling shows how the incremental algorithm scales with the number of input faces. The Maze scene (see Figure 6.1) was tesselated with increasing density illustrating the linear scaling. A purely diffuse version and a version with a mixture of specular and diffuse faces were used. Notice how the other scenes cluster near this linear trend. The radiance interface had 512x512 resolution and 513 PSFs were employed.

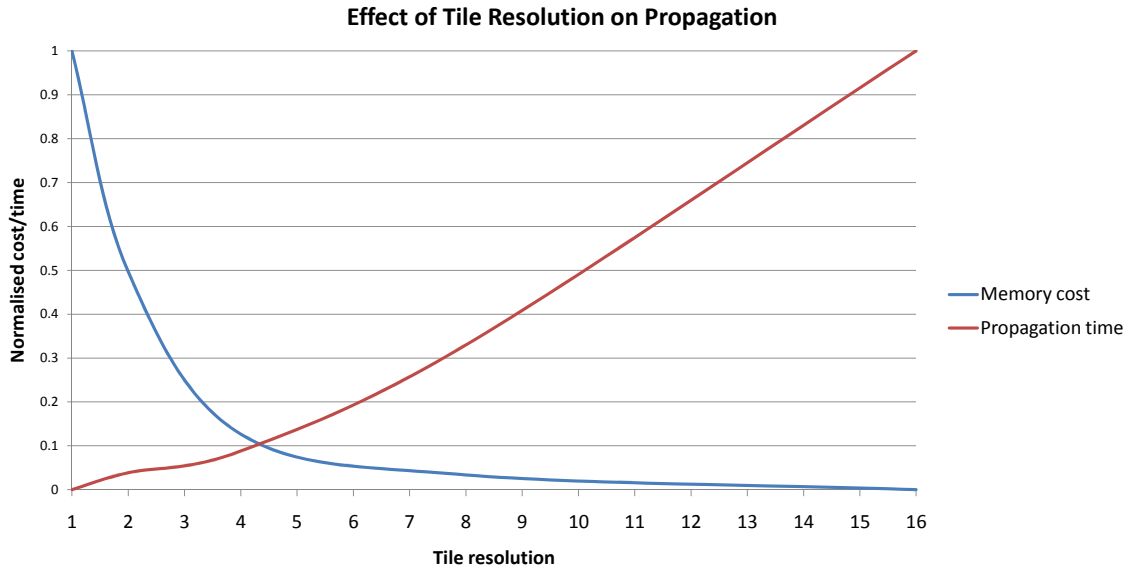


Figure 6.4: This graph shows how tiling resolution affects the propagation time and memory. The Cornell box scene (see Figure 6.14(a)) was propagated with a fixed radiance map resolution of 64x64 and varying tile resolutions between 1x1 and 16x16. 513 PSFs were employed. Propagation time ranged between 1.2 secs. and 3.02 secs. Memory consumption ranged between 217MB and 10MB.

full 64x64 radiance map for each face wasting many texels, but improving the propagation performance since only a single rendering operation is necessary for transport. At the other end of the spectrum a 16x16 tiling resolution was used with a 4x4 per tile resolution. This greatly reduces the memory footprint since only tiles which overlap the face are stored. However, propagation is a 2.5x slower because each face now requires multiple rendering operations for transport since each tile is handled separately and some coherence is lost. At some point further tiling saves little additional memory but adds to the propagation time. A good tiling resolution for this example is 4x4.

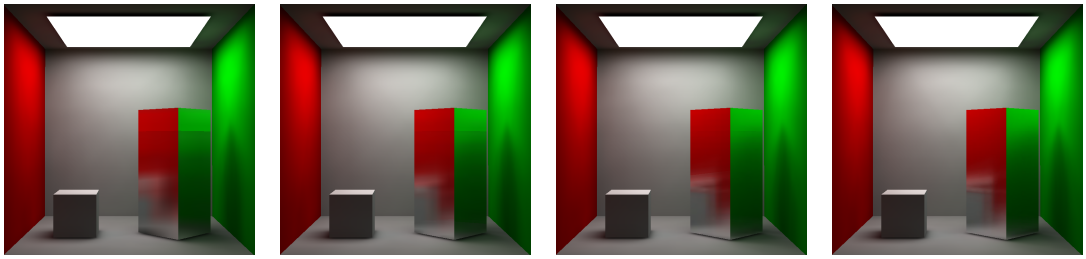
6.3 Analysis of Quality and Correctness

In order to gauge the quality of non-diffuse reflections, a matrix of images are shown in Figure 6.5. The directional resolution is increased from top to bottom and the radiance map resolution is increased from left to right. The scene is a Cornell box where the front of a tall box is using a perfectly specular material. The mirror reflection of the smaller box is clearly visible in the left part of the box. As expected there will be a certain amount of ghosting in a specular reflection when using few directions since the resulting image is a linear interpolation of three orthographically projected radiance maps. The amount of ghosting is inversely proportional to the number of directions and is most clearly discernible in Figure 6.5(d). On the other hand low radiance map resolutions result in blurring of the reflection (left part of the image matrix). There is a clear trade-off between memory use and visual quality of the reflection, but it is obvious that glossy reflections would be possible with relatively low resolution and few directions. Also, if the radiance maps are used for indirect illumination of surfaces with diffuse or low-frequency BRDFs, such as caustic reflections onto diffuse materials, relatively low resolutions would suffice. See Figure 6.14(a) for a comparable image rendered with stencil reflections.

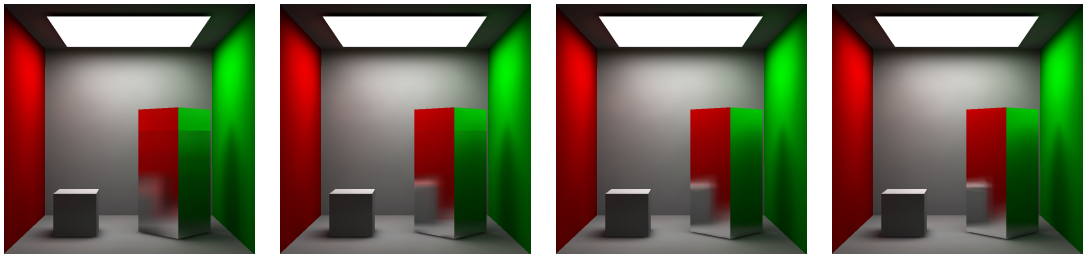
The memory used during propagation assumes *three* copies of the (ir)radiance maps. In fact, *two* copies will suffice, but the implementation uses *three* for ease of debugging such that the propagation is done breadth-first. Also, the memory could be reduced by using Greg Ward's RGBE format. These savings would reduce the propagation footprint of the scene in Figure 6.5(p) to 453MB.

In Figure 6.6 the diffuse quality is evaluated. Again, the directional resolution is increased from top to bottom and the irradiance map resolution is increased from left to right. Diffuse map density is given in texels per world unit and for the scene in question the lowest resolution irradiance map for the floor of the box is 16x16 and the highest resolution one is 1024x1024. During propagation, the irradiance map will typically be filtered with a low-pass filter when they are swapped in preparation for the next light transport iteration. The width of this filter will usually depend on the diffuse map resolutions and directional resolution but in these images the filter width is fixed and relatively narrow [4]. As expected, when using few directions and high resolution irradiance maps, transport artifacts will be visible due to under-sampling. This is most clearly visible in Figure 6.6(d). However, along the diagonal of the image matrix the right balance of directions and irradiance map texels have been struck and the maps are quite smooth, yet especially at the higher resolutions subtle radiosity characteristics are visible. In order to better gauge this the floor irradiance maps for the images in the diagonal are shown in Figure 6.7.

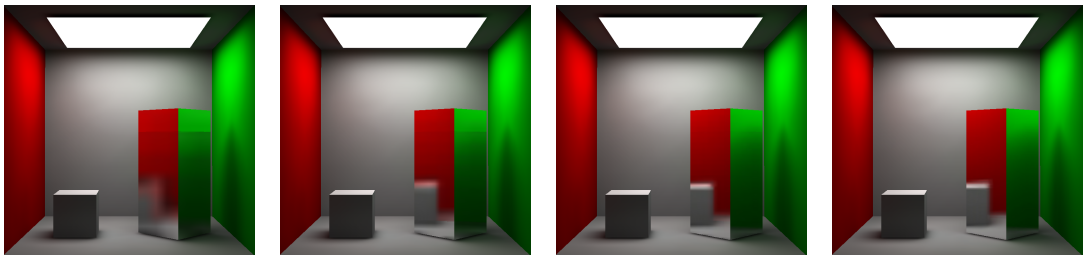
[4] For the images a 5x5 Gaussian filter was used with a sigma of 2.



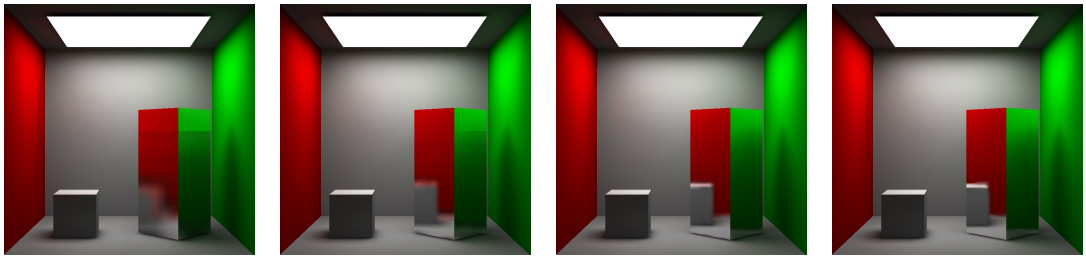
(a) l=129, r=32^2 (b) l=129, r=64^2 (c) l=129, r=128^2 (d) l=129, r=256^2



(e) l=513, r=32^2 (f) l=513, r=64^2 (g) l=513, r=128^2 (h) l=513, r=256^2

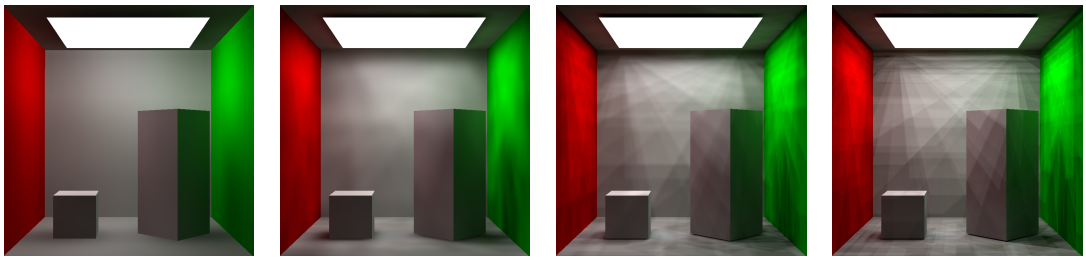


(i) l=2049, r=32^2 (j) l=2049, r=64^2 (k) l=2049, r=128^2 (l) l=2049, r=256^2

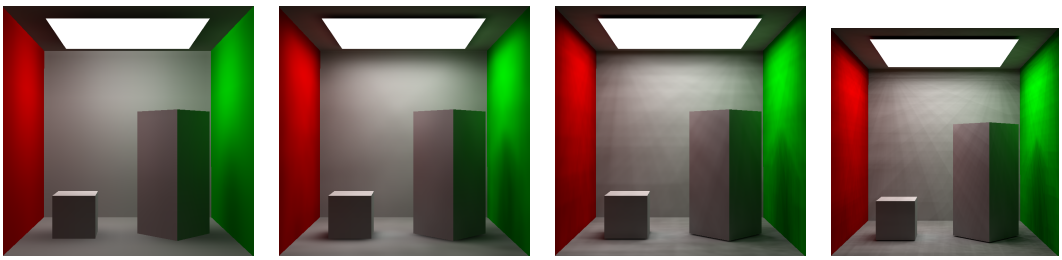


(m) l=8193, r=32^2 (n) l=8193, r=64^2 (o) l=8193, r=128^2 (p) l=8193, r=256^2

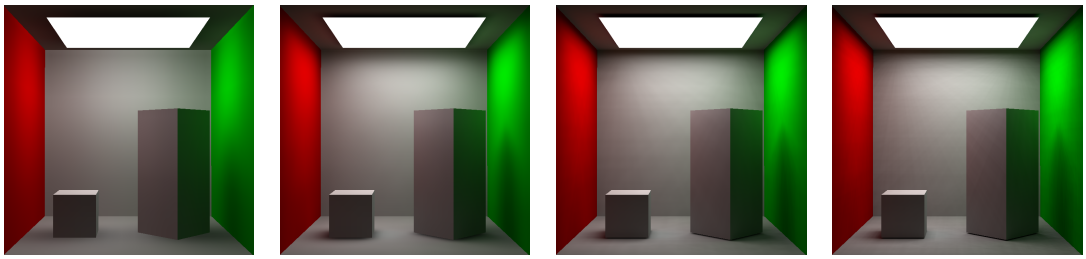
Figure 6.5: Cornell box with specular material; note the reflection of the small box in the tall box. Images (a)-(p) show specular quality when varying the number of directions l and the radiance map resolution r propagated over 5 iterations. Extra jittered directions were added up to 8K to ensure comparable diffuse quality. The scene had 102K diffuse elements and between 128K-56.5M non-diffuse elements. Propagation times ranged between 5.4-26 seconds per iteration. Memory usage during propagation ranged between 4.9MB-1019MB. Memory usage for rendering ranged between 690KB-170MB. See Figure 6.14(a) for a reference image rendered with stencil reflections.



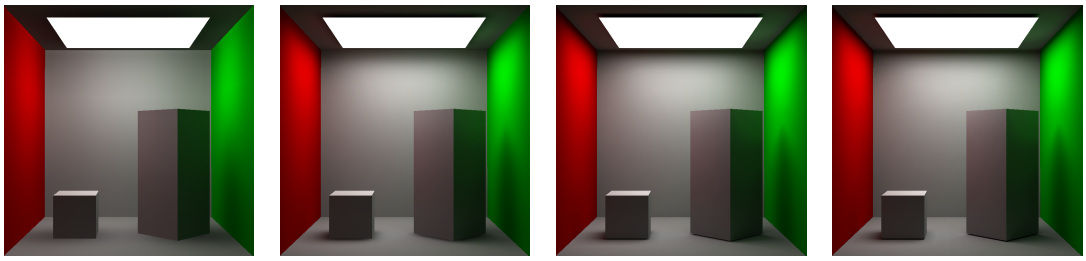
(a) l=129, d=16 (b) l=129, d=64 (c) l=129, d=256 (d) l=129, d=1024



(e) l=513, d=16 (f) l=513, d=64 (g) l=513, d=256 (h) l=513, d=1024

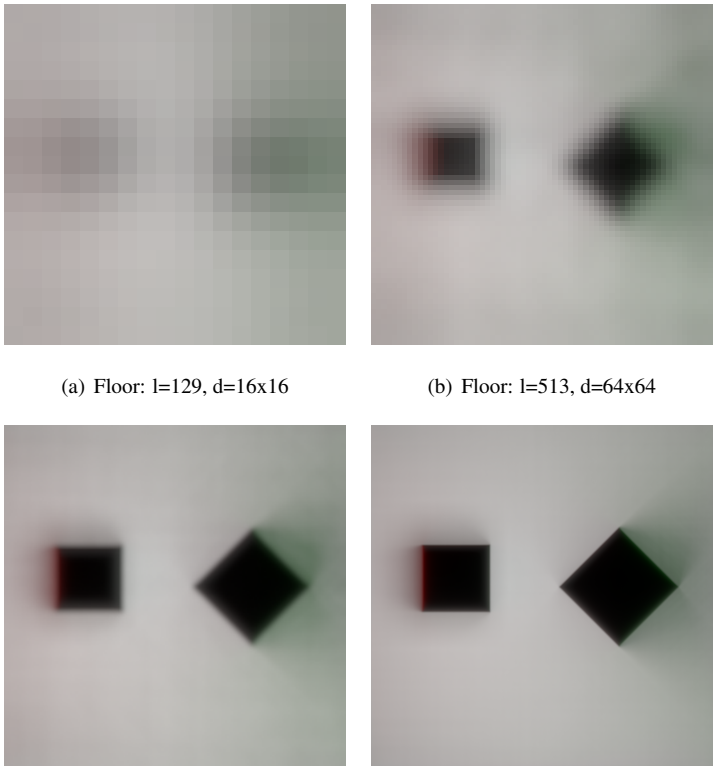


(i) l=2049, d=16 (j) l=2049, d=64 (k) l=2049, d=256 (l) l=2049, d=1024



(m) l=8193, d=16 (n) l=8193, d=64 (o) l=8193, d=256 (p) l=8193, d=1024

Figure 6.6: Cornell box with diffuse material. Images (a)-(p) show diffuse quality when varying the number of directions l and the diffuse irradiance map resolution d per unit propagated over 5 iterations. The dimensions of the box is one unit such that for example (p) has a floor resolution of 1024x1024. The scene had between 2.2K-6.5M diffuse elements. Propagation times ranged between 0.14-30 seconds per iteration. Memory usage during propagation ranged between 39KB-117MB. Memory usage for rendering ranged between 6.6KB-19.5MB.



(a) Floor: l=129, d=16x16

(b) Floor: l=513, d=64x64

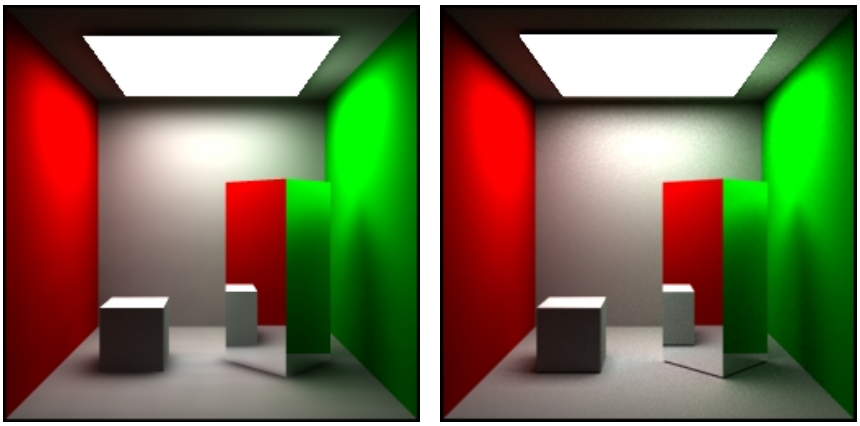
(c) Floor: l=2049, d=256x256

(d) Floor: l=8193, d=1024x1024

Figure 6.7: Cornell box with diffuse material. Images (a)-(d) show diffuse quality of the floor irradiance map when varying the number of directions l and the diffuse irradiance map resolution d per unit propagated over 5 iterations. The images correspond to the diagonal in the image matrix given in Figure 6.6.

In order to objectively assess the correctness of the solution, the results were verified by comparing images computed with PBRT [PH04] a path-tracing system used widely in the field of global illumination. The path tracer used importance sampling and 1024 paths per pixel. The images (see Figure 6.8) were subtracted and the MSE was around $1e^{-3}$ (0.001). The peak errors were around $1e^{-2}$ and concentrated in areas where the low object-space resolution of the ir/radiance was inadequate to efficiently represent high frequency illumination in image space. In particular the shadow at the base of the tall specular box appears blurred with the VLF method. Also, the caustic on the right-hand side of the ceiling is less defined than with PBRT. However, the noise apparent in the path traced image even with 1024 rays per pixel is absent from the image computed with VLF-GPU.

The images took 104 minutes to render with PBRT and 4 seconds with VLF-GPU (including propagation).



(a) Rendered with VLF-GPU. (b) Rendered with PBRT.

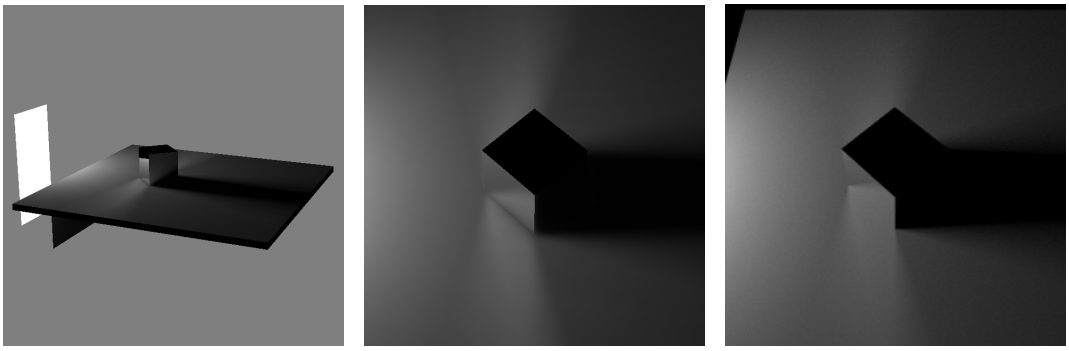
Figure 6.8: Comparison of VLF-GPU and path tracing for a specular Cornell box scene.

6.3.1 Caustic Example

In Figure 6.9 caustic light transport effects are illustrated. The scene is taken from [SJ00] and is part of a suite of test scenes intended for verifying global illumination algorithms. An overview of the scene is shown in Figure 6.9(a). Below the plane is an emitter reflecting light off the vertical diffuse reflector to the left. This diffusely reflected light is deposited on the diffuse floor and reflected off the specular box creating the caustic ”wings” across the floor. Some quite complex light paths are required in this scene. For example looking at the box viewing the reflection of the caustic on the floor represents an *ESDSDL* light path.

6.4 Other BRDFs

In order to demonstrate that the VLF-GPU method can accommodate BRDFs other than specular and diffuse (and mixtures thereof) a prototype glossy reflection kernel using the modified Phong reflectance model [LW94b] has been implemented. The modified Phong BRDF can be written as the sum of a



(a) Scene overview (VLF-GPU). (b) Propagated with VLF-GPU, 50 secs. (c) Smits and Jensen [SJ00].

Figure 6.9: Comparison of VLF-GPU and the Smits and Jensen caustic test scene described in [SJ00]. The scene was computed using 8196 jittered directions over 3 iterations. The data structure comprised 2049 PSFs with a 128x128 radiance map resolution and an 8x8 tile resolution consuming 224MB memory. The scene has 169984 diffuse elements and 5619456 directional elements.

diffuse part and a specular part:

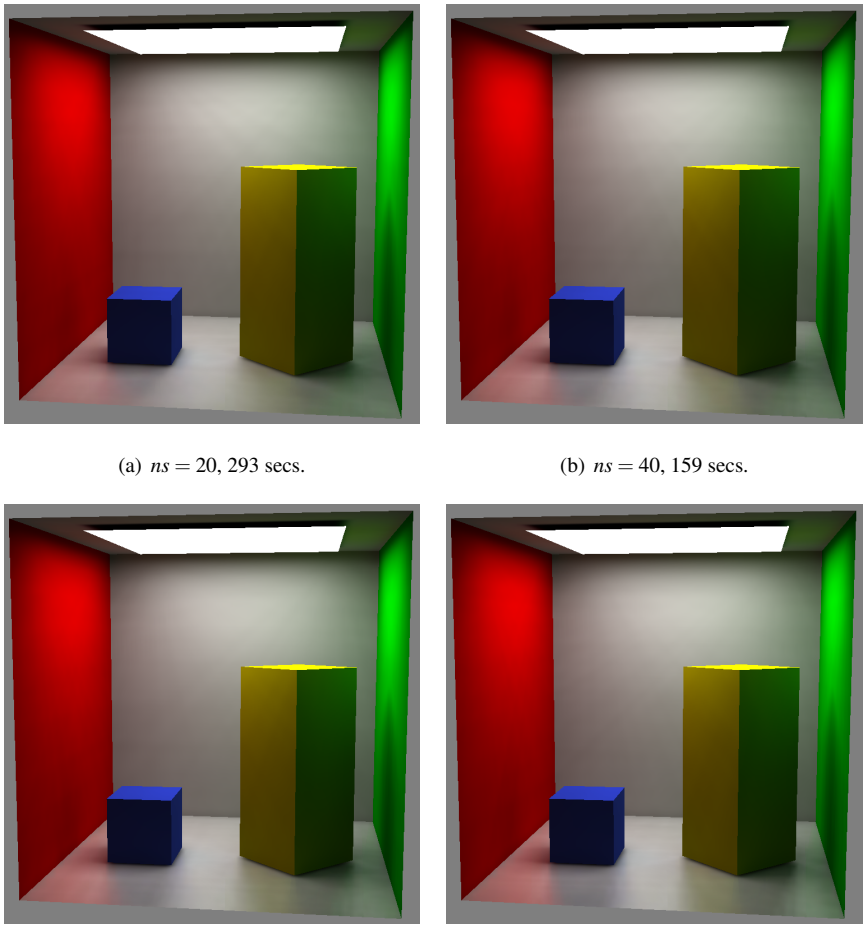
$$f_r(x,\Theta_i,\Theta_o) = k_d\frac{1}{\pi} + k_s\frac{n+2}{2\pi}\cos^{ns}\alpha \tag{6.1}$$

where:

- $\alpha =$ the angle between the perfect specular reflective direction and the outgoing direction.
- $k_d =$the diffuse reflectivity, i.e. the fraction of incoming energy that is reflected diffusely.
- $k_s =$the specular reflectivity, i.e. the fraction of the perpendicularly incoming energy that is reflected specularly.
- $ns =$the specular exponent. Higher values for ns result in sharper specular reflections.

In the VLF context the diffuse contribution modulated by k_d is accumulated in the irradiance maps and the non-diffuse contribution modulated by k_s is stored in the directional radiance maps. During propagation the reflection kernel is invoked when a non-diffuse receiver is encountered. At this point the radiance interface (RI) is filled with incoming radiance from the direction that is currently being propagated. This radiance must be scattered in directions over the hemisphere above the receiver face weighted by the BRDF assigned to the face. Given a face f and a PSF_i in direction ω_i every PSF_j in direction ω_j is visited and a reflection *from* the RI *to* the radiance map for face f in PSF_j is performed weighted by $k_s\cos^{ns}\alpha$, where α is computed using the normal of f and the directions ω_i and ω_j. If $\alpha < \varepsilon$ (a user defined small value) the reflection is ignored in order to avoid insignificant or negative cosine values.

Figure 6.10 shows a Cornell box with a glossy floor using a high directional resolution (2049 PSFs = 4098 directions). The specular exponent ns is set to values ranging from 20 to 160. With a value of $ns = 20$ the floor has a diffuse appearance with a reflective component that blurs quickly with increasing distance between the reflector and the object being reflected. With increasing ns the reflection becomes



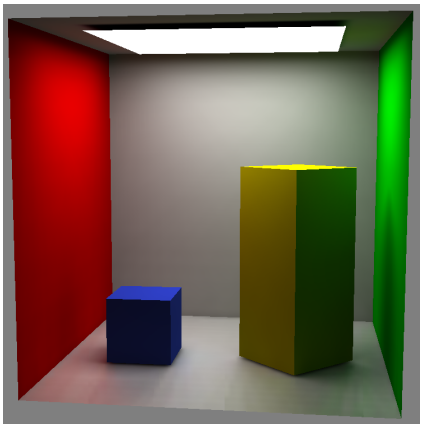
(a) $ns = 20$, 293 secs.

(b) $ns = 40$, 159 secs.

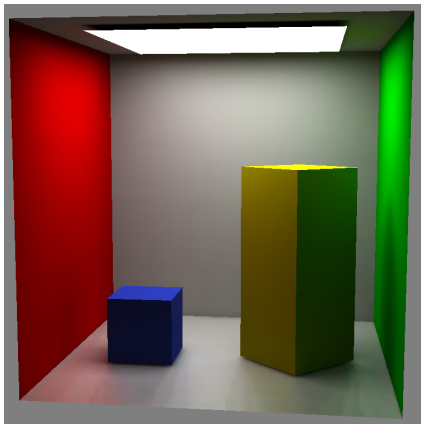
(c) $ns = 80$, 87 secs.

(d) $ns = 160$, 49 secs.

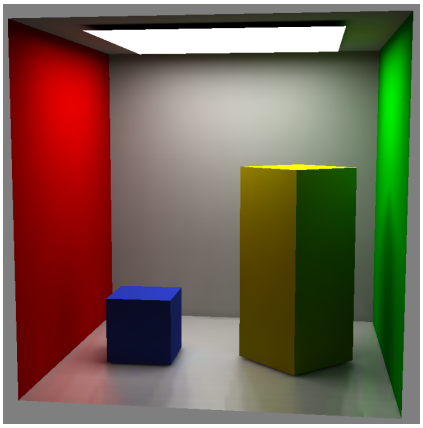
Figure 6.10: Cornell box with glossy material reflecting from 2049 directions. Images (a)-(d) uses a glossy material on the floor varying the specular exponent *ns* between 20-160. The diffuse reflectivity $k_d = 0.4$, specular reflectivity $k_s = 0.6$. The scene was propagated over 3 iterations using 2049 PSFs with a 128x128 radiance map resolution and an 8x8 tile resolution.



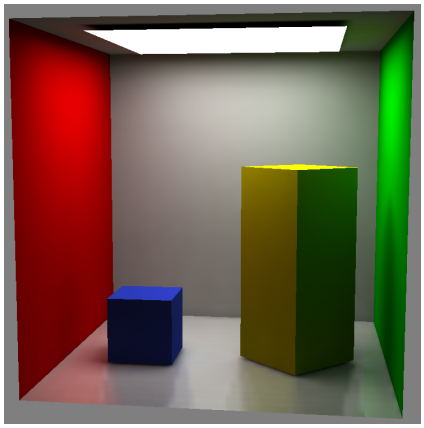
(a) $ns = 20$, 101 secs.



(b) $ns = 40$, 54 secs.

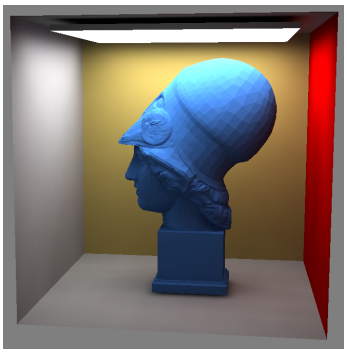


(c) $ns = 80$, 37 secs.

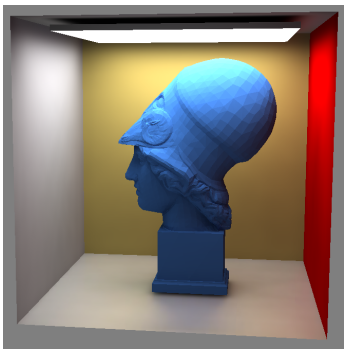


(d) $ns = 160$, 28 secs.

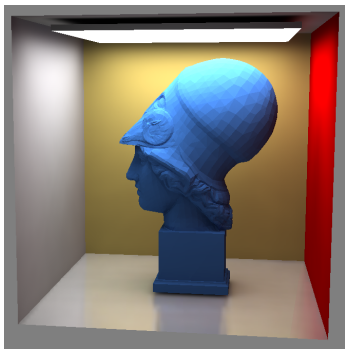
Figure 6.11: Cornell box with glossy material reflecting from 513 directions. Images (a)-(d) uses a glossy material on the floor varying the specular exponent ns between 20-160. The diffuse reflectivity $k_d = 0.4$, specular reflectivity $k_s = 0.6$. The scene was propagated over 3 iterations using 2049 PSFs with a 128x128 radiance map resolution and an 8x8 tile resolution.



(a) Diffuse, 419 secs.



(b) Glossy $ns = 20$, 708 secs.



(c) Specular/diffuse, 430 secs.

Figure 6.12: Atenea scene (9410 faces) with diffuse (a), glossy (b) and specular/diffuse (c) floor material. Image (b) uses a glossy material with specular exponent $ns = 20$ reflecting from 2049 directions. The diffuse reflectivity $k_d = 0.4$, specular reflectivity $k_s = 0.6$. The scene was propagated over 3 iterations using 2049 PSFs with a 128x128 radiance map resolution and an 8x8 tile resolution.

sharper. The propagation time varied between 49 seconds to 293 seconds. For comparison the propagation time using a mixed specular/diffuse floor was 15 seconds. Figure 6.11 shows a similar Cornell box with the same VLF resolution but performing reflections for only a quarter of the available directions. The propagation time of this experiment varied between 28-101 seconds. The images exhibit some banding due to the fixed nature of the directions. This could to some extent be converted to noise by using jittered random directions. The result would, however, be more than sufficient when used merely for transport in a renderer using final gathering.

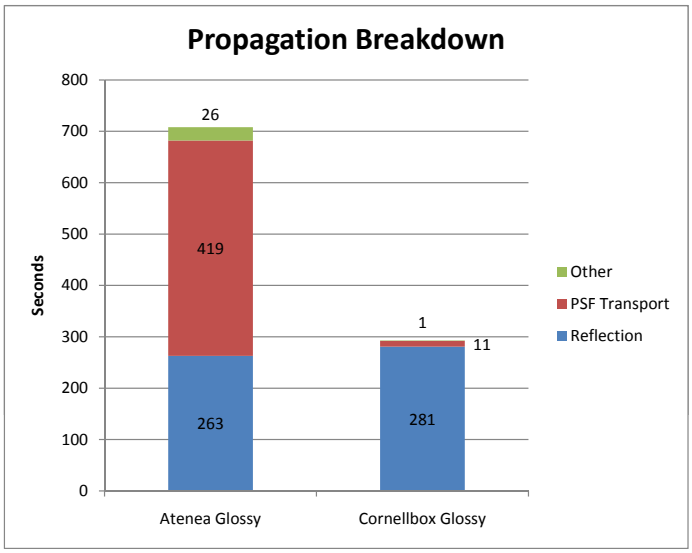


Figure 6.13: Breakdown of propagation time for the glossy Cornell box and the glossy Atenea scene ($ns = 20$); Figure 6.12(b) and Figure 6.10(a). The blue bar represents time spent on glossy reflection kernel, the red bar is time spent on transport between faces along PSFs, and the green bar is time spent on other propagation tasks.

Scattering with a BRDF is an altogether *local* operation involving only the reflector and its directional radiance maps stored in the PSFs. The performance of propagation involving arbitrary BRDFs depend only on the number of PSFs and the nature of the BRDF. A BRDF with a very sharp lobe requires fewer reflections than a BRDF with a very wide lobe since the former has fewer directions contributing a significant amount and can be sampled more aggressively. The worst case is a perfectly diffuse BRDF but that is handled in the VLF by accumulating the irradiance and moving the BRDF evaluation $\frac{\rho}{\pi}$ outside the integral. Since the performance is tied only to the VLF resolution and BRDF and not to the number of faces the linear nature of the propagation stage is unaffected. The overhead of BRDF reflection will be a constant that diminishes when the geometric complexity is increased. Figure 6.12 shows a more complex scene with $\sim$9K faces propagated with the same resolution and parameters as those of Figure 6.10. Propagation times for pure diffuse, glossy and mixed specular/diffuse are shown. Figure 6.13

shows a breakdown of the propagation times for the Cornell box (Figure 6.10(a)) and the Atenea scene (Figure 6.12(b)) with a glossy floor with $ns = 20$. This clearly illustrates that the BRDF reflection is a constant that is independent of geometric complexity.

The prototype implementation samples the BRDF uniformly without any importance sampling. The number of reflections performed could, for many BRDFs, be reduced significantly by employing importance sampling techniques. Another observation is that low-frequency BRDFs are the most expensive to precompute but they can also be represented with fewer PSFs whereas high-frequency BRDFs require many PSFs but are cheaper to precompute. This duality could be exploited by a system that could adjust the number of PSFs used for storage and reflections based on the BRDFs used.

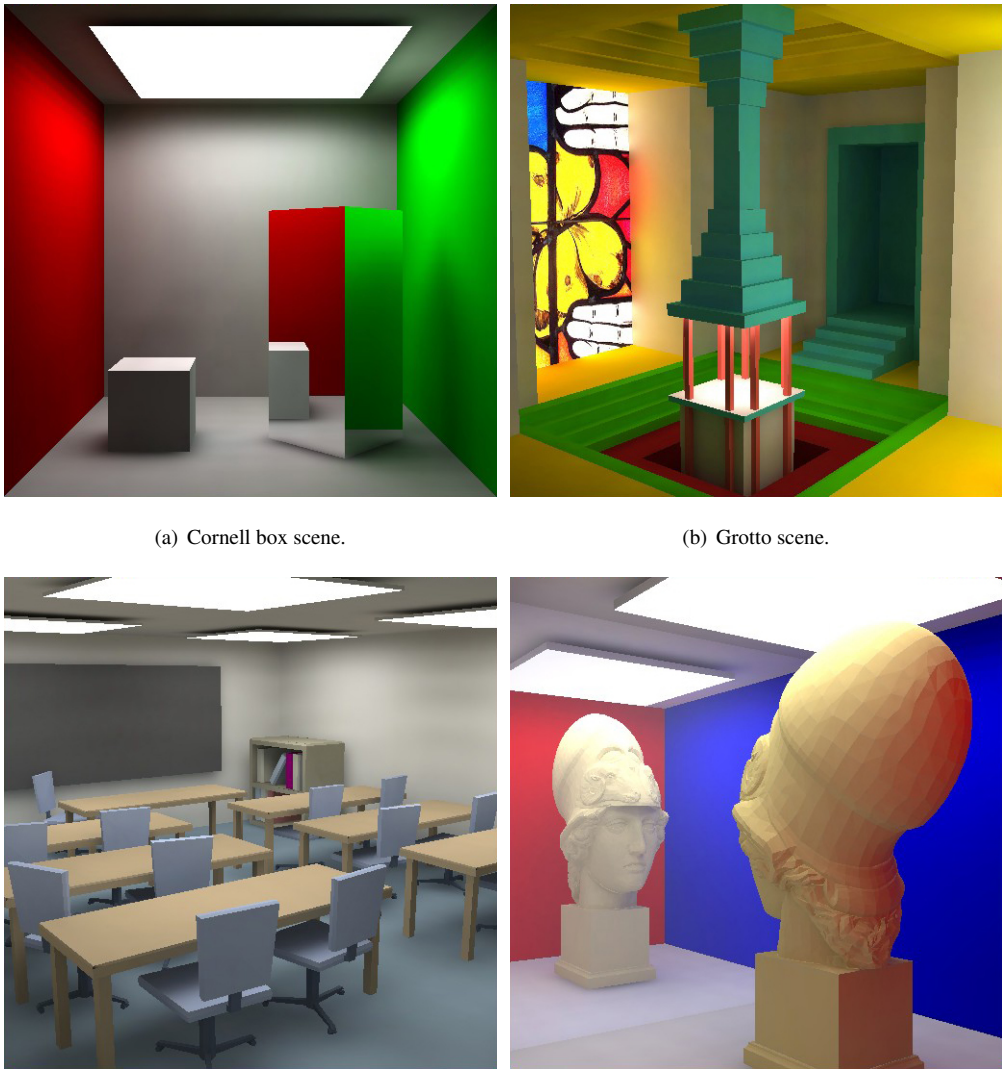
6.5 Comprehensive Results

Performance of the algorithm was tested on four scenes: two consist of only diffuse surfaces (Figure 6.14(b) and Figure 6.14(c)) and the remaining two of diffuse and specular surfaces (Figure 6.14(a) and Figure 6.14(d)). All figures use 512 bi-directional global directions, and Figure 6.14(a) uses PSFs constructed at a resolution of 64x64 cells (8x8 tiles, 8x8 cells per tile), whereas Figure 6.14(d) uses 256x256 cells (16x16 tiles, 16x16 cells per tile). The mirror wall in Figure 6.14(d) is partly diffuse and specular, containing a slightly bluish diffuse element. The ratio of directions to spatial resolution is a function of the distance between mutually visible polygons and can be computed automatically. We set these parameters manually.

Scene	Faces	Diff. elems.	Spec. elems.	Sort (sec)	Prop. (sec)	Total (sec)	Memory (prop.)	Memory (rend.)	Rend. (1024×768)
Cornell	19	35K	931K	0.031	3.9	4.1	23.2MB	2.9MB	122fps
Grotto	318	817K	0	0.549	16.7	17.2	19.6MB	2.5MB	124fps
Class.	3,024	906K	0	11.149	52.5	63.7	21.7MB	2.7MB	121fps
Atenea	9,410	2,435K	7,630K	18.656	133.1	153.2	241.5MB	30.2MB	121fps

Table 6.1: Performance of the VLF-GPU method. This table shows the propagation and rendering performance for a number of scenes. The number of polygons along with the finite element resolution are listed. The *Total* column lists the total propagation time including sorting and GPU initialisation. The memory used during propagation and rendering is listed. The memory for rendering is less since temporary radiance maps have been discarded and (ir)radiance atlases have been tonemapped and compressed to 24bit RGB.

Propagation timing results from the scenes are presented in Table 6.1. They are all based on three propagation iterations. Timings were obtained on an Intel Core 2 Quad (QX6700) 2.66GHz processor with a GeForce 8800 GTX with 768MB graphics memory and 4GB of host memory. Initial sorting was



(a) Cornell box scene.

(b) Grotto scene.

(c) Classroom scene.

(d) Atenea scene.

Figure 6.14: Our system can propagate light through scenes with arbitrary BRDFs and millions of ir/radiance elements in seconds 6.14(a) to minutes 6.14(d) and render them at frame-rates exceeding 120fps at 1024×768 screen resolution.

performed on the CPU in four threads using a 2-level BSP subdivision along the depth of the PSF and a single level across the width and height. For the Cornell box scene the sorting was faster on a single core without BSP subdivision due to the scene's low number of polygons.

A 64-bit floating-point RGBA format was used for storing ir/radiance. Memory consumption is given for the propagation stage as well as the rendering stage. After propagation all *current* and *next* ir/radiance maps were discarded accounting for two-thirds of the memory and the *total* ir/radiance maps were tonemapped and stored as 24bit RGB for display. For offline storage the maps can be further compressed using standard image compression methods. Storage during propagation can be roughly halved using 4 byte shared exponent format (similar to Greg Ward's RGBE format), which was introduced as a native pixel-format with the GeForce 8 series.

6.6 Immersive Virtual Reality Applications

In order to use a rendering technique in an immersive virtual reality (VR) device using stereo to display 3D content, the frame update rate needs to be stable and at least 30 frames per second in order not to cause discomfort in the participant [BH95]. Larger immersive VR devices work by consecutively rendering the view as seen from the left and right eye to the same screen. Only the appropriate view is presented to each eye by filtering out every other frame, using either shutter glasses or twin projectors coupled with polarised filters. Since the rendering is interleaved in this way it is not always practical to render each image on a different workstation[5], which means that the *effective* update rate of the rendering technique could be at least 60 frames per second in such scenarios. Besides rendering, a VR system also needs to handle other tasks related to the operating system, networking, synchronisation, tracking and scene graph management. The VLF-GPU can add global illumination to a rendering system with little overhead making this a practical technique for integrating into a complete VR system. This chapter discusses the integration of VLF-GPU into eXtremeVR (XVR) [CTB+05], which is a stand-alone integrated development environment for rapid development of complex VR applications.

6.6.1 Rendering the VLF in the CAVE™

When the VLF propagation step has converged, the GPU can render novel views from the data structure by interpolating between samples stored in the diffuse textures and non-diffuse view-dependent radiance tiles (see Section 5.4 for details). Diffuse surfaces are rendered directly using texturing with the diffuse textures available in the irradiance texture atlases. The GPU performs interpolation efficiently in this case.

Depending on the resolutions used for the VLF the specular surfaces may be too coarse for direct display, this is often true in the case of perfect mirrors. In order to improve the fidelity of specularly reflecting surfaces they could be rendered with ray-tracing by recursively following a view ray reflected in the specular face until it strikes a diffuse face where the visible radiance can be collected and displayed. This, however, can be expensive and could reduce the frame rate significantly. A similar idea, often used in real-time VR applications, is to use the stencil buffer to render a reflected view of the scene as seen

[5] For example when using passive stereo on commodity GPUs without hardware syncronisation.

through the specular face and then paste this onto the face with texturing [Kil02]. The latter technique is used in the current XVR integration for flat mirrors.

The use of global illumination in a VR scenario dramatically improves the visual appearance of the scene. Current VR research routinely use only local illumination rendering (see Figure 6.15(b) [6]). This causes flat looking images since indirect lighting is approximated with a single ambient term. Also, such images lack the soft shadows caused by area light sources and indirect lighting, which are important in anchoring objects in a scene [SUC95]. An example of this can be seen in Figure 6.15(a) where soft contact shadows are firmly anchoring the rocking chair, the table, and, the potted plant in their environment.

6.6.2 Dynamics Integration

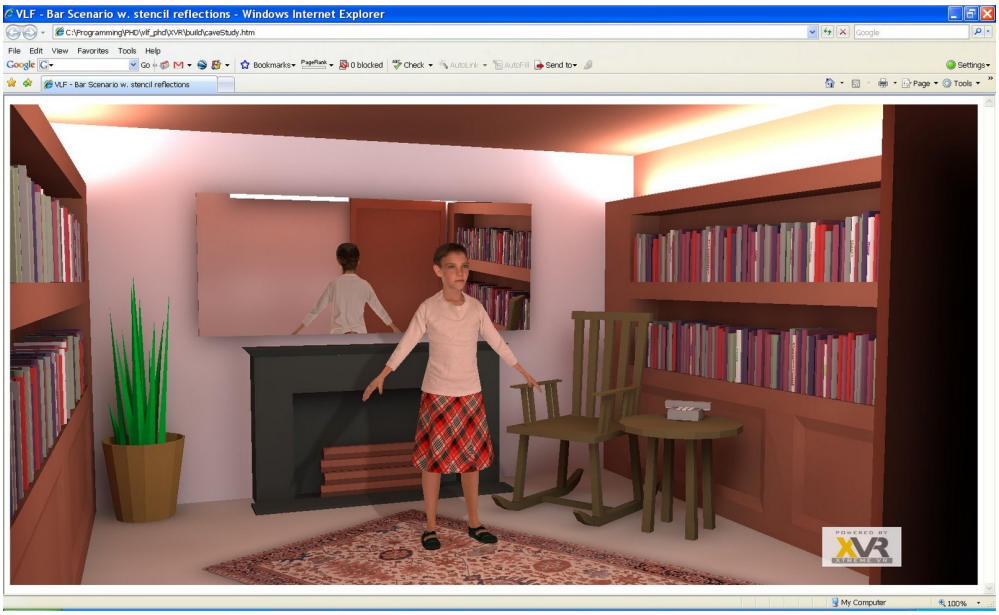
Integrating dynamic elements in a global illumination solution is a difficult task. The computation resources needed to solve the rendering equation numerically at real-time frame rates are currently not readily available. Popular approaches attempting this are ray tracing [WDB+06] and hierarchical finite element approaches [DSDD07]. At the time of writing these cannot deliver the frame-rates and resolution needed for virtual reality applications. Making some simplifying assumptions can, however, make the problem tractable. If we separate the scene geometry into dynamic and static elements we can pre-compute the global illumination for the static elements using the VLF and focus on the dynamic elements at run-time.

Dynamic elements undergoing only rigid body animation can be easily integrated using Pre-computed Radiance Transfer [SKS02] where the VLF can provide the input radiance. This approach does not apply to elements such as virtual characters (avatars) using skinned animation, which are a crucial element of many VR applications. The mesh of an avatar is typically made up of thousands of triangles, essentially undergoing unstructured motion, making it virtually impossible to accelerate through a pre-compute approach. However, breaking up the problem and attacking the modes of transport that contribute most to the image, real-time frame rates can be achieved with support for significant global illumination effects.

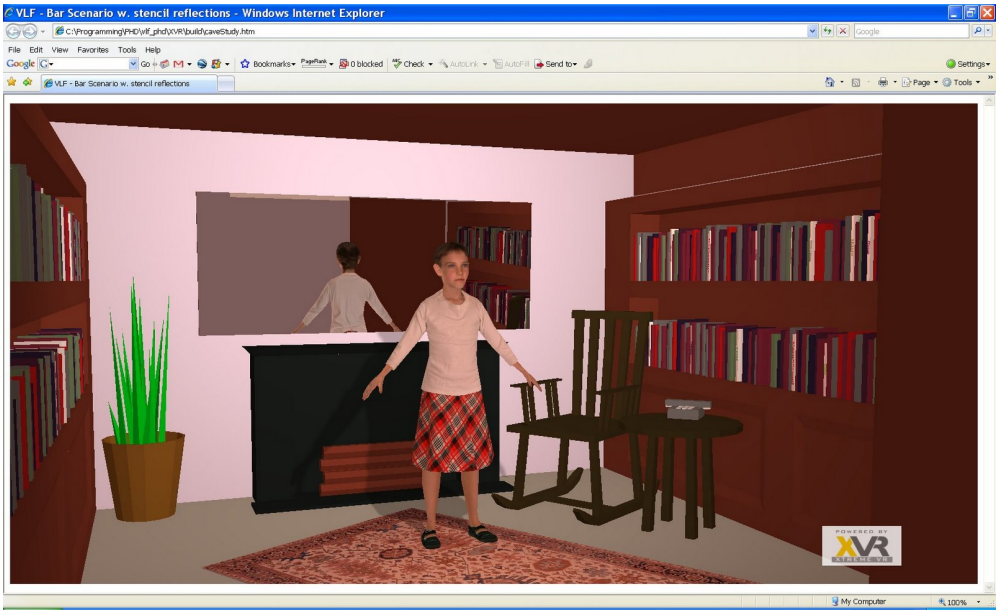
We can separate the problem into three main modes of transport contributing to the image and focus on solving these: 1) field radiance scattered off the avatar towards the eye, 2) soft shadows cast by the avatar and 3) specular reflections of the avatar. This does not solve for diffuse reflections of the avatar, and thus colour bleeding caused by the avatar will not be accounted for. However, the magnitude of illumination that has undergone multiple diffuse reflections is generally low and will add little to the image.

In order to solve 1) we need to be able to rapidly provide the irradiance at an arbitrary spatial position, a shader program can then use this to calculate the surface shading at points on the mesh. In order to provide irradiance calculations at real-time frame rates we introduce another pre-computed data structure derived from the VLF. The bounding volume of the scene is subdivided into a set of

[6] The ambient term supplied for the rendering was painstakingly estimated by adjusting the OpenGL lighting parameters and comparing the image to the VLF rendering.



(a) Rendered with VLF-GPU.



(b) Rendered with OpenGL.

Figure 6.15: Comparison of OpenGL and VLF-GPU. The lighting parameters of OpenGL were adjusted to best match the ambient lighting of the VLF-GPU counterpart. Note the realistic appearance due to global illumination effects, in contrast to standard OpenGL rendering, which is commonplace in VR research.

Figure 6.16: Lighting a character with a spherical harmonics light probe. For illustrative purposes the light probe is superimposed on the torso of the character.

voxels. A voxel stores irradiance retrieved from the VLF projected to a spherical harmonic (SH), and due to the properties of spherical harmonics they can be calculated at arbitrary positions by trilinear interpolation of the eight nearest voxels. See Figure 6.16 for an example of such a light probe and its application to a dynamic character. Such irradiance volume was suggested by Ramamoorti [RH01] and also Greger [GSHG98] albeit in a different form.

To solve 2) direct soft shadows cast by the avatar onto static geometry or other dynamic geometry are needed. Physically correct soft shadows are notoriously difficult to calculate. However, perceptually correct soft shadows can be rendered in real-time using the graphics processing unit (GPU). Percent Closer Soft Shadows [Ura05] samples a standard shadow map stochastically to provide approximate umbra and penumbra regions of a shadow due to an area light source. In order to combine this with the physically correct soft shadows (cast by static elements) already present in the scene, the visibility of shadow mapped light sources are also stored as texture maps in the VLF. This information is trivially available during the VLF pre-compute.

Finally, to solve 3) specular reflections of the avatar are required. Reflections (and caustics) are already appropriately accounted for in the VLF for the static parts of the scene, but this obviously does not include the dynamic elements. Also, as discussed earlier, depending on the resolution of the VLF and the BRDF of the surface they may benefit from reconstruction using the scene geometry. This can easily be achieved in real-time using a reflection rendering pass, rendering the visible scene onto a reflective surface [Kil02]. This can be extended to curved surfaces using traditional environment mapping techniques. In order to include the dynamic elements in the reflection they can merely be rendered into the reflection texture map along with the static scene. Occlusion is automatically handled by the depth buffer. This affords dynamic objects after a *single* reflective bounce, since after one reflection the VLF is used to render any subsequent non-diffuse reflections, in which the dynamic objects are *not* included. In order to include dynamic objects with more bounces a ray tracing approach would be more suitable. This would trace reflective rays until a diffuse surface (or dynamic object) was struck or a specified maximum

number of bounces was reached at which point the VLF would be looked up.

The effect of these techniques used in conjunction is quite striking. The dynamic elements merge well with the surrounding scene featuring impinging colour bleeding, caustics and casting soft shadows as well as being visible in reflective surfaces (see Figure 6.15(a)). Also, the soft shadows provide a subtle but important anchoring effect as discussed earlier. Often when using standard local illumination without shadows the dynamic elements can be hard to place relative to one another and objects on a surface may appear to be floating. This is not the case using these techniques. Reflections, shadows and colour bleeding are all cues that help place the dynamic elements in their environment.

6.6.3 The XVR framework

We chose XVR as our implementation framework since in addition to real-time graphics rendering, it includes the ability to handle many collateral aspects of VR programming such as sound, haptics and interaction. To allow for CAVE™applications development, XVR has a dedicated module called Network Renderer [MVT+07]. This module allows for cluster-based rendering of XVR applications. By using a cluster of workstations, the rendering load is distributed among several machines. In particular, we let each PC of the cluster manage the rendering of a different screen of the CAVE™system.

The Network Renderer is totally transparent to the original XVR application: each of the OpenGL calls performed by the master application is intercepted by the module that catches all the information about the calls, and stores them into an internal memory buffer. Each time that it is necessary, the Network Driver sends the content of the buffer to a set of remote executables, named graphic slaves, which run on the machines composing the cluster. Each of the slaves then executes the OpenGL calls received from the master. The VLF-GPU rendering technique integrates naively in this system, since the VLF rendering can be expressed entirely by OpenGL calls and execution of GLSL shader programs (see Section 5.4), which are supported by the network renderer. This is in contrast to many other approaches that require additional information to be computed on the rendering nodes, which cannot be expressed as OpenGL calls. An example of this is ray tracing based approaches. In fact VLF-GPU integrates easily in either a rasterisation or ray tracing based system.

The immersive capabilities provided by a CAVE™system include three main factors: stereo graphics rendering, head tracking, and the fact that the participant is surrounded by the visual display. All these factors must be considered in order to make the visualisation system work properly and in a consistent way. In particular it is necessary to calculate the proper perspective projection matrix for each of the screens, according to the position and the orientation of participant's viewpoint and head direction. Performing all these operations is a task independent of the specific application running, and it can be exactly defined given the specifications of the particular CAVE™in use. For these reasons the XVR Network Renderer relieves the application programmer from being concerned with these issues. The programmer only needs to render monoscopic images, and XVR takes care of properly rendering in stereo and onto the CAVE™screens. In our CAVE™setup four rendering clients drive the front, left, right and floor projections respectively.

6.7 Summary

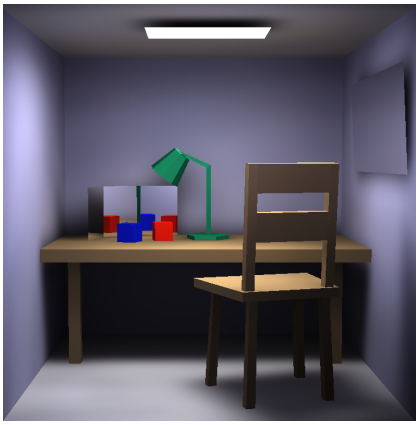
If we, in a *hypothetical* scenario, were to consider each sample transported in the propagation step as requiring *one* ray intersection computation for visibility, we would in effect achieve ∼33M ray intersections per second on average for the four scenes in Table 6.1 (between ∼14MRays/sec for the Cornell box scene and ∼74MRays/sec for the Grotto scene). This is partly due to the fact that we exploit coherence by "tracing" bundles of parallel rays and partly because the incremental approach eliminates the need for traversing a data structure to resolve visibility. To our knowledge, the fastest CPU ray tracing method to date is *AEPSA* [FCM09] reporting 21MRays/sec for a scene with 10k faces, for shaded non-textured *coherent* primary rays. Our memory bandwidth is much higher than that of AEPSA since each ray intersection requires HDR texture lookups of radiance values; this yields a memory bandwidth of >3.56GB/sec for the Grotto scene.

Comparably, GPU ray tracing implementations have not proven effective [FS05, CHH02, PBMH02], reporting less than ∼1MRay/sec (extrapolated to current hardware). However, with the recent development of more general purpose GPUs, ray tracing on that platform is being revisited. Nvidia has recently developed the ray tracing API OptiX. The performance of OptiX was analysed in [LE10]. They report between 17.3Mrays/sec and 53Mrays/sec running on an Nvidia Quadro FX 5800 GPU. This GPU has 2.7x the processing power of the Nvidia GeForce 8800 GPU (933.12GFLOPs vs. 346GFLOPs), 2x higher fill rate and 1.6x higher memory bandwidth.

Our visibility performance is comparable (or better) to that of AEPSA and OptiX. We stress, however, that the method here does not easily apply to general ray tracing, the numbers are given in order to put the visibility performance of the incremental approach in context.

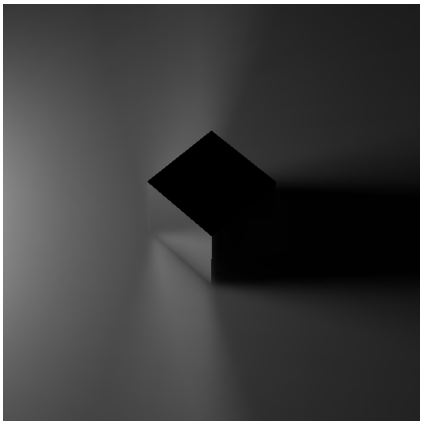
In Figure 6.17(a) - Figure 6.17(d) the VLF-GPU method is compared to the original VLF method presented in [SMKY04]. Figure 6.17(a) and Figure 6.17(b) shows a small office scene composed of 141 faces. Both methods use 2049 PSFs with a 128x128 radiance map resolution and an 8x8 tile resolution. The VLF-GPU method has comparable image quality with less structured aliasing on the floor and propagates 2125x faster. The rendering speeds quoted in [SMKY04] for a 256x256 image were 1.1fps when rendering directly from the radiance maps and 20fps when reconstructing the reflective surfaces with ray tracing (shown). The VLF-GPU method renders a similar view at 83fps directly from the radiance maps and at 1946fps using stencil reflections at 1024x768 screen resolution. The hardware used in [SMKY04] was an 1.7Ghz Xeon CPU for propagation and two 2.8Ghz Xeon CPUs for rendering. The hardware used to propagate the scene and render the images for the VLF-GPU method was an Nvidia GeForce 8800GTS GPU with 640MB video memory. The CPU was a 3.3Ghz Intel Q6600 Quad Core CPU of which only a single core was used.

Figure 6.17(c) and Figure 6.17(d) shows the caustic test scene also shown in Figure 6.9. Again both methods use 2049 PSFs with a 128x128 radiance map resolution and an 8x8 tile resolution. The VLF-GPU method additionally uses 8192 jittered directions. The image quality is comparable although it proved difficult to set exactly the same viewpoint and tone mapping parameters. The VLF-GPU method propagates the scene 2414x faster. Rendering times for the original VLF methods was 27fps for ray

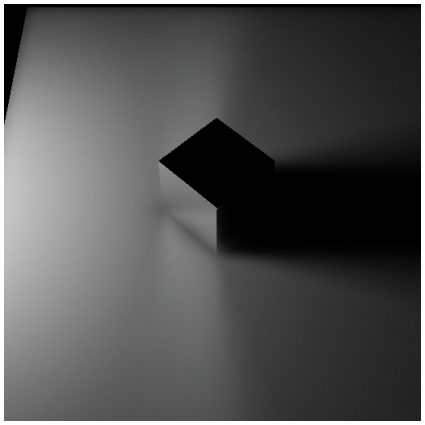


(a) Propagated with VLF-GPU, 61 secs.

(b) Propagated with VLF [SMKY04], 36 hrs.

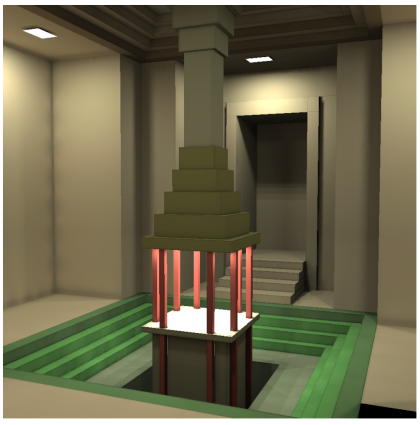


(c) Propagated with VLF-GPU, 50 secs.



(d) Propagated with VLF [SMKY04], 33.53 hrs.

(e) Propagated with VLF-GPU, 17.2 secs.



(f) Grotto-Lo scene [CHL04], 64.78 secs.

Figure 6.17: Images (a) - (d) compares the VLF-GPU and the original VLF method described in [SMKY04]. Images (e) and (f) compares the VLF-GPU method to the GPU radiosity method described in [CHL04].

tracing also with a 256x256 screen resolution. The VLF-GPU method renders such view at 89fps directly from the radiance maps and 3316fps using stencil reflections with a 1024x768 screen resolution.

Figure 6.17(e) is a slightly modified recreation of the Grotto scene used in [CHL04]. The method presented by Coombe et. al. is based on progressive radiosity on graphics hardware. Since it is a pre-compute method that can render global illumination at high sustainable frame rates on high resolution displays it is a suitable method for comparison. The propagation time quoted in [CHL04] are 64.78 seconds for a version using 260K diffuse elements (shown in Figure 6.17(f)) and 86.81 seconds for a version with 1M elements i.e. 11.7K elements per second. For comparison the scene used with VLF-GPU had 817K diffuse elements i.e. 47.5K elements per second[7]. The hardware used in [CHL04] was an Nvidia GeForce FX 5900 GPU with a P4 1.8Ghz host system. Since the Nvidia GeForce 8800GTS GPU has 3.2x higher fill rate than the Nvidia GeForce FX 5900 GPU and a 2.3x higher memory bandwidth our performance is comparable to that of [CHL04] (extrapolating to current hardware). But, the VLF-GPU method has fewer artifacts and supports non-diffuse BRDFs. The current implementation supports specular, glossy, diffuse and mixed specular/diffuse, thus accounting for $L(S|G|D)^*E$ paths, whereas [CHL04] is diffuse only.

Overall, propagation time shows >3 orders of magnitude improvement (i.e. $>$2000x) over earlier VLF propagation results [SMKY04] for simple scenes with $<$200 faces. More complex scenes are intractable with the previous approach due to its $O(N^2)$ complexity. This significant improvement is due equally to the improved propagation scheme, incremental transfer and GPU implementation.

Rendering can easily be done in real-time at high resolutions. The frame-rates given in Table 6.1 are for 1024×768 frames. Frame-rates are stable regardless of the viewpoint due to the fact that the multiple rendering passes are performed for *all* pixels in the frame. Clearly, this could be optimised by only performing the view-dependent rendering on non-diffuse pixels. This would, however, make the frame-rate variable with a worst case performance of that given in Table 6.1 when the viewpoint is such that only non-diffuse pixels are viewed.

[7]The parameters used for the VLF-GPU method are quoted in Section 6.5.

Chapter 7

Conclusion

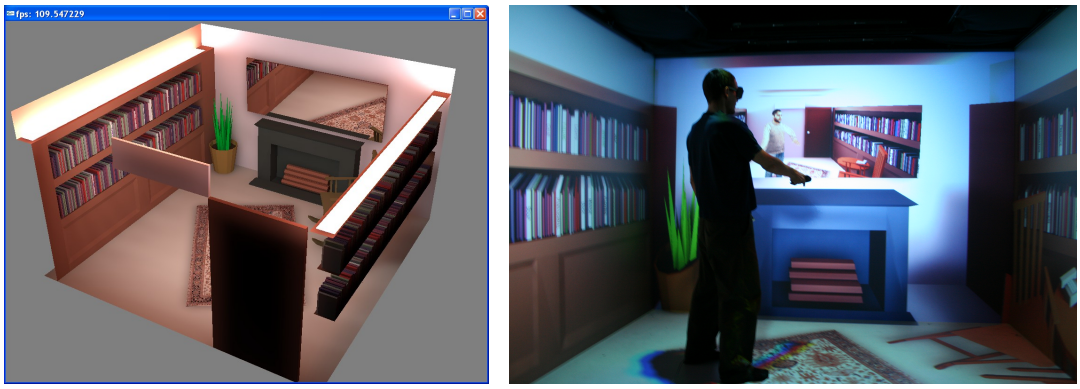
A novel approach to the problem of global illumination on the GPU has been introduced in this thesis. The VLF-GPU method can produce 5D light fields for virtual scenes in linear time in the number of faces and render them, requiring only a small constant time overhead on top of a standard renderer. The goal has been to achieve real-time walkthrough for scenes with full global illumination. The scope of the problem was mainly to allow the use of global illumination in immersive VR scenarios that hitherto has typically been restricted to local illumination rendering techniques.

7.1 Contributions

An incremental approach for propagating light through the scene has been introduced. This has $O(N)$ complexity in the number of input faces. This allows simple to moderate scenes in terms of geometric complexity with high illumination complexity - millions of elements - to be solved in seconds to minutes on the GPU [MKS07]. This is an improvement over earlier results [SMKY04] of nearly three orders of magnitude and exhibits linear rather than quadratic scaling. Propagation time is decoupled from the number and cumulative area of emitters. This distinguishes this approach from many recent methods that does not support area emitters and, often, will only work with *few* point emitters. The approach includes direct lighting so there is no need to compute direct lighting separately at run-time.

A novel GPU approach for rendering from Virtual Light Fields has been introduced [MKS07]. This can render scenes of arbitrary illumination complexity in real-time; $>$ 160 frames per second. It adds a small constant to the time taken to rasterise the scene with a traditional local illumination approach. Thus, the approach can guarantee a stable frame rate which is critical for some applications [BH95].

Furthermore, the VLF-GPU technique has been integrated into a fully fledged VR system. Dynamic soft shadows and relighting for dynamic elements are supported and work well with the technique, ensuring that dynamic elements blend well with the improved rendering. The system was used for a presence experiment aimed at determining the effect of global illumination on the sense of presence. The scenario featured global illumination with area lights, specular reflections, dynamic globally illuminated avatars and dynamic soft shadows. The XVR/VLF-GPU system was able to render the environment at sustained frame rates at 30 frames per second in stereo (effectively 60fps). This frame rate was not a limitation of the VLF-GPU rendering system but rather of the VR system as a whole, which requires strict



(a) Scene used for experiment. (b) Subject in Cave^{TM}

Figure 7.1: Scene used in the CAVE^{TM}presence experiment [YMKS10].

frame synchronisation to avoid tearing artifacts to appear where the CAVE walls meet. See Figure 7.1 for an overview of the scene used in the experiment. Preliminary results of this experiment are reported in [YMKS10].

One of the main drawbacks is that the approach does not work well with very small or point emitters. In order to do so a pre-pass must be included that lights up surfaces directly visible to the lights; this has been discussed in [SK98, SKSMT00]. Another issue is the rather large memory requirement needed to store directional radiance for non-diffuse surfaces. Compression could solve this issue given that there is very high coherence between the outgoing radiance of adjacent PSFs. Also, although it was shown that the Vis-Sort algorithm is linear time, the constant is rather high due to the *stop-and-wait* GPU implementation. This could be solved by implementing the optimised approach in [GLM04].

7.2 Directions of Future Work

Future work - in addition to addressing the drawbacks discussed above - includes implementing application support for a wider range of BRDFs, which is inherently supported by the method. Also, inspired by the work in [RGK^{+}08] experimenting with using approximate visibility for jittered PSFs would be an interesting avenue of research. One could for example re-cycle visibility from nearby PSFs without re-sorting them. In addition, lowering the precision used for the propagation of later bounces might be possible since the illumination of bounced light is likely to be lower frequency. In the limit, the light transport could be performed for indirect lighting only and the direct lighting could be computed at run-time, potentially making it possible to lower the VLF precision enough to allow for some interactivity. This would, however, remove support for area lights and make the run-time complexity dependent on the number of lights in the scene, which is an undesirable property. Given that the PSFs contain similar data, compression may work very well and could allow larger scenes. Currently, there is a two-level hierarchy for storing (ir)rradiance; the directional radiance maps and the (directionless) irradiance maps. An option could be to introduce a directional hierarchy and represent radiance at an appropriate level in the hierarchy based on its BRDF. The current directional subdivision already exposes such a hierarchy.

In summary a novel physically based global illumination approach for the GPU has been presented.

It can pre-compute general $L\{S|G|D\}^*E$ solutions in $O(N)$ time and render from this in $O(1)$ time on top of rasterisation or ray casting from the eye. This research has made real-time global illumination possible in VR applications. The approach has been integrated into a complete VR system and has been used to conduct VR research in the CAVE™with globally lit virtual environments. This is believed to be the first successful complete VR system rendering fully globally lit environments in the CAVE™at real-time frame rates. It will allow significantly more realistic environments to be used with immersive VR and will thus improve current applications and open up new avenues of research in VR.

Appendix A

Symbols

A.1 Geometric Symbols

Symbol	Description
A	combined set of surface points
x	surface point
n_x	normal at surface point x
dA_x	differential surface area at point x
$\Theta, -\Theta, \Psi, -\Psi$	direction vectors
Ω_x	hemisphere above x $\{\forall\Theta : \Theta \in \Omega_x \iff \Theta \cdot n_x \geq 0\}$
$d\omega_\Theta$	differential solid angle around direction Θ
θ, φ	angles
r_{xy}	distance between surface points x and y
$\overline{xy}$	direction from x to y
$V(x,y)$	visibility function; $V(x,y) = \begin{cases} 1, & \text{if } x \text{ and } y \text{ are mutually visible;} \\ 0, & \text{otherwise} \end{cases}$
$r(x,\Theta)$	ray casting function; $r(x,\Theta)$ is the nearest visible point to x in direction Θ

A.2 Radiometric Symbols

Symbol	Description
Q	radiant energy [(J)oule]
Φ	radiant flux [(W)att]
E	irradiance $[W \cdot m^{-2}]$
M, B	radiant exitance, radiosity $[W \cdot m^{-2}]$
I	radiant intensity $[W \cdot sr^{-1}]$
L	radiance $[W \cdot m^{-2} \cdot sr^{-1}]$
$L_e(x, \Theta)$	self emitted radiance leaving x in direction Θ
$L_e(x, y)$	self emitted radiance leaving x in direction $\overline{xy}$
$L_e^i(x, \Theta)$	incident emitted radiance arriving at x from direction Θ
$L_e^i(x, y)$	incident emitted radiance arriving at x from direction $\overline{yx}$
$L_o(x, \Theta)$	exitant radiance leaving x in direction Θ
$L_o(x, y)$	exitant radiance leaving x in direction $\overline{xy}$
$L_i(x, \Theta)$	incident radiance arriving at x from direction Θ
$L_i(x, y)$	incident radiance arriving at x from direction $\overline{yx}$
W	potential
$W_e(x, \Theta)$	self emitted potential leaving x in direction Θ
$W_e(x, y)$	self emitted potential leaving x in direction $\overline{xy}$
$W_e^i(x, \Theta)$	incident emitted potential arriving at x from direction Θ
$W_e^i(x, y)$	incident emitted potential arriving at x from direction $\overline{yx}$
$W_o(x, \Theta)$	exitant potential leaving x in direction Θ
$W_o(x, y)$	exitant potential leaving x in direction $\overline{xy}$
$W_i(x, \Theta)$	incident potential arriving at x from direction Θ
$W_i(x, y)$	incident potential arriving at x from direction $\overline{yx}$
$f_r(x, \Theta_i, \Theta_r)$	bidirectional reflectance distribution function (BRDF) at surface point x with incident direction Θ_i and reflected direction Θ_r
ρ	hemispherical reflectance

A.3 VLF Notation

Symbol	Description
VLF	virtual light field
PSF_ω	parallel subfield in direction ω
L_T	total or accumulated radiance
L_U	unshot radiance to be distributed in the next iteration
$L(\omega,s,t,P)$	outgoing radiance map for tile (s,t) in direction ω for surface P
$L(\omega,s,t,u,v,P)$	outgoing radiance of cell (u,v) in tile (s,t) in direction ω for surface P
$V(\omega,s,t,u,v,P)$	binary visibility of cell (u,v) in tile (s,t) in direction ω for surface P, $V(\omega,s,t,u,v,P)=1$ if P is rasterised to this cell in PSF_ω
$X(\omega,s,t,u,v,P)$	exchange buffer of cell (u,v) in tile (s,t) in direction ω for surface P, $X(\omega,s,t,u,v,P)$ is the index of the closest visible surface along the ray in direction ω originating at (s,t,u,v)
D_T	total or accumulated irradiance
D_U	unshot irradiance
D_R	reflected irradiance to be distributed in the next iteration
$D(P)$	irradiance map for surface P
$D(u,v,P)$	irradiance of cell (u,v) for surface P
$L^D(\omega,s,t,u,v,P)$	irradiance for ray (ω,s,t,u,v) from surface P

A.4 Global Illumination Feature Table

In Table A.4 a selection of global illumination algorithms described in Chapter 2 are listed by feature. The categories are as follows:

- *Reference*: Paper reference.
- *Keyword*: Keyword to identify paper.
- *Dynamic viewpoint*: Whether dynamically moving viewpoints are supported.
- *Dynamic lights*: Whether dynamically moving lights are supported.
- *Dynamic geometry*: Whether dynamically moving geometry is supported.
- *Dynamic materials*: Whether dynamically changing materials are supported.
- *Specular transport*: Whether perfect specular transport is supported.
- *Glossy transport*: Whether glossy transport is supported.
- *Diffuse transport*: Whether diffuse transport is supported.
- *Bounces*: Number of indirect bounces supported.
- *Area lights*: Whether the algorithm supports area lights.
- *Direct lighting*: Specifies whether the algorithm includes direct lighting computations. Some algorithms require that direct lighting is compute separately using graphics hardware.
- *Illumination Frame rate*: Time taken to render and illuminate a frame. This is given in either a number denoting the number of frames per second or specified as (s)econds, (m)inutes, (h)ours or (d)ays.
- *Pre-processing*: Time taken to precompute data necessary structures. This is given as (s)econds, (m)inutes, (h)ours, (d)ays or (-) if no pre-processing is needed.
- *Memory usage*: Memory needed for precomputation or rendering, specified as megabytes (mb) or gigabytes (gb).
- *Notes*: Any miscellaneous information.

Reference	Keyword	Dynamic viewpoint	Dynamic lights	Dynamic geometry	Dynamic materials	Specular transport	Glossy transport	Diffuse transport	Bounces	Area lights	Direct lighting	Illumination Frame rate	Pre-processing	Memory usage	Notes
Thesis	VLF-GPU.	✓	×	×	×	✓	✓	✓	Any	✓	✓	100+	s-m	mb-gb	Dynamic object relighting supported.
[SMKY04]	VLF.	✓	×	×	×	✓	×	✓	Any	✓	✓	20	h-d	mb-gb	256x256 screen resolution.
[CHL04]	GPU radiosity.	✓	×	×	×	×	×	✓	Any	✓	✓	100+	s-m	-	Requires tesselation.
[SB97]	Interactive glossy.	✓	×	×	×	✓	✓	✓	Any	×	×	s	n/a	n/a	Precomputes photon map.
[ML09]	ISPM.	✓	✓	✓	✓	✓	✓	✓	Any	×	×	26	-	-	Requires eight CPU cores. Linear in # lights.
[HOJ08]	Progressive PM.	×	×	×	×	✓	×	✓	Any	✓	✓	h-d	-	-	Final glossy reflection possible.
[Kel97]	Instant radiosity.	✓	✓	✓	✓	×	✓	✓	Any	✓	✓	s	-	-	Uses shadow maps for visibility.
[LSK+07]	Incr. instant rad.	✓	✓	✓	✓	×	×	✓	1	×	×	65	-	-	Uses PCF soft shadows for direct lighing.
[RGK+08]	ISM.	✓	✓	✓	✓	×	✓	✓	Any	✓	✓	17	-	-	5 fps for more than one bounce.
[RGKS08]	CSSM.	✓	✓	✓	✓	×	×	✓	Any	×	×	10	-	-	Final glossy reflection possible.
[DS05]	RSM.	✓	✓	✓	✓	×	×	✓	1	×	×	28	-	-	No visibility for bounce.
[DS06]	RSM splatting.	✓	✓	✓	✓	×	✓	✓	1	×	×	85	-	-	No visibility for bounce.
[NW09]	Multi-res RSM.	✓	✓	✓	✓	×	✓	✓	1	×	×	68	-	-	No visibility for bounce.
[WFA+05]	LightCuts.	✓	✓	✓	✓	×	✓	✓	Any	✓	✓	m	-	-	Final glossy reflection possible.
[CPWAP08]	Non-linear cuts.	✓	✓	×	✓	×	×	✓	1	✓	✓	4	m-h	mb	Final glossy reflection possible.
[HPB07]	Matrix sampling.	✓	✓	✓	✓	×	✓	✓	Any	✓	✓	s	s	-	Temporal flickering.
[DSDD07]	Anti-radiance.	✓	✓	✓	✓	×	✓	✓	Any	✓	×	15	-	mb	7 fps for reasonably sized scene.
[DKTS07]	Implicit visibility.	✓	✓	✓	✓	×	✓	✓	Any	✓	✓	8	-	-	At most two bounces were shown in results.
[WKB+02]	Fast ray tracing.	✓	✓	✓	✓	✓	✓	✓	Any	✓	✓	<5	-	-	Requires clusters. Low quality during dynamics.
[WDB+06]	Fast ray tracing.	✓	✓	✓	✓	✓	✓	✓	Any	✓	✓	<10	-	-	Requires clusters. Low quality during dynamics.
[KGPB05]	Radiance caching.	✓	✓	✓	✓	×	✓	✓	1	✓	✓	s-m	-	mb	Speed depends on cache completeness.
[WS99]	Holodeck cache.	✓	×	×	×	✓	✓	✓	Any	✓	✓	s	h	mb-gb	Dynamic object relighting supported.
[SS00]	Tapestry cache.	✓	×	×	×	✓	✓	✓	Any	✓	✓	<7	s-m	n/a	Poor quality due to sparse sampling.
[BWG03]	Edg/pnt images.	✓	✓	✓	×	×	✓	✓	Any	✓	✓	<6	m	n/a	One bounce glossy shown.
[TPWG02]	Shading cache.	✓	✓	✓	×	×	✓	✓	Any	✓	✓	2	s	n/a	Requires clusters. Low quality during dynamics.

Table A.1: Global Illumination Feature Table

A.5 Heckbert Light Transport Notation

Symbol	Description
L	light
E	eye
D	diffuse reflection event
S	specular reflection event
G	glossy reflection event
$(x)^+$	one or more x events
$(x)^*$	zero or more x events
$(x)^?$	zero or one x event
$(x\|y)$	either x or y event

Bibliography

[AK90] James Arvo and David Kirk. Particle transport and image synthesis. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, volume 24, pages 53–66, 1990.

[App67] Arthur Appel. The notion of quantitative invisibility and the machine rendering of solids. In *Proceedings of the 1967 22nd national conference*, pages 387–393, New York, NY, USA, 1967. ACM Press.

[App68] Arthur Appel. Some techniques for shading machine renderings of solids. *AFIPS 1968 Spring Joint Computer Conference*, 32:37–45, 1968.

[Arv86] James Arvo. Backward ray tracing. *Developments in Ray Tracing, ACM SIGGRAPH course notes*, 12:259–263, August 1986.

[Arv93] James Arvo. Transfer equations in global illumination. In *ACM SIGGRAPH 1993 Course Notes - Global Illumination*, volume 42, 1993.

[Arv95] James Arvo. The role of functional analysis in global illumination. In *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, pages 115–126, 1995.

[Azu99] Daniel Azuma. Interactive rendering of surface light fields. Technical report, University of Washington, University of Washington, May 1999.

[BF89] Chris Buchalew and Donald Fussell. Illumination networks: fast realistic rendering with general reflectance functions. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 89–98, New York, NY, USA, 1989. ACM Press.

[BH95] Woodrow Barfield and Claudia Hendrix. The effect of update rate on the sense of presence within virtual environments. *Virtual Reality*, 1(1):3–15, June 1995.

[Bli82] James F. Blinn. Light reflection functions for simulation of clouds and dusty surfaces. In *SIGGRAPH '82: Proceedings of the 9th annual conference on Computer graphics and interactive techniques*, pages 21–29, New York, NY, USA, 1982. ACM Press.

[Bly06] David Blythe. The direct3d 10 system. *ACM Trans. Graph.*, 25(3):724–734, 2006.

[BNN+98] Philippe Bekaert, László Neumann, Attila Neumann, Mateu Sbert, and Yves D. Willems. Hierarchical monte carlo radiosity. In *Rendering Techniques '98 (Proceedings of the 9th Eurographics Rendering Workshop)*, June 1998.

[Bri07] Robert Bridson. Fast poisson disk sampling in arbitrary dimensions. In *SIGGRAPH '07: ACM SIGGRAPH 2007 sketches*, page 22, New York, NY, USA, 2007. ACM.

[Bun05] Michael Bunnell. *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*, chapter Dynamic Ambient Occlusion and Indirect Lighting, pages 223–234. Addison Wesley, 2005.

[BWG03] Kavita Bala, Bruce Walter, and Donald P. Greenberg. Combining edges and points for interactive high-quality rendering. *ACM Transactions on Graphics*, 22(3):631–640, 2003.

[Cam01] Emilio Camahort. *4D Light-Field Modeling and Rendering*. PhD thesis, The University of Texas at Austin, May 2001.

[Cat78] Edwin Catmull. A hidden-surface algorithm with anti-aliasing. In *SIGGRAPH '78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, pages 6–11, 1978.

[CCWG88] Michael F. Cohen, Shenchang Eric Chen, John R. Wallace, and Donald P. Greenberg. A progressive refinement approach to fast radiosity image generation. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 75–84, New York, NY, USA, 1988. ACM Press.

[CF99] Emilio Camahort and Donald Fussell. A geometric study of light field representations. Technical Report TR9935, Department of Computer Sciences, The University of Texas at Austin, 1999.

[CG85] Michael F. Cohen and Donald P. Greenberg. The hemi-cube: A radiosity solution for complex environments. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, volume 19, pages 31–40, New York, NY, USA, August 1985. ACM Press.

[CGIB86] Michael Cohen, Donald P. Greenberg, Dave S. Immel, and Philip J. Brock. An efficient radiosity approach for realistic image synthesis. *IEEE Computer Graphics and Applications*, 6:3, 1986.

[Che90] Shenchang Eric Chen. Incremental radiosity: An extension of progressive radiosity to an interactive image synthesis system. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, volume 24, pages 135–144, New York, NY, USA, August 1990. ACM Press.

[CHH02] Nathan A. Carr, Jesse D. Hall, and John C. Hart. The ray engine. In *HWWS '02: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 37–46, 2002.

[CHH03] Nathan A. Carr, Jesse D. Hall, and John C. Hart. GPU algorithms for radiosity and subsurface scattering. In *HWWS '03: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 51–59, 2003.

[CHL04] Greg Coombe, Mark J. Harris, and Anselmo Lastra. Radiosity on graphics hardware. In *GI '04: Proceedings of the 2004 conference on Graphics interface*, pages 161–168, 2004.

[CKM+99] Joao Comba, James T. Klosowski, Nelson Max, Joseph S.B. Mitchell, Cludio T. Silva, and Peter L. Williams. Fast polyhedral cell sorting for interactive rendering of unstructured grids. *Computer Graphics Forum*, 18(3):369–376, September 1999.

[CLF98] Emilio Camahort, Apostolos Lerios, and Donald Fussell. Uniformly sampled light fields. In *Rendering Techniques '98 (Proceedings of Eurographics Rendering Workshop '98)*, pages 117–130, 1998.

[CMS98] Francesc Castro, Roel Martínez, and Mateu Sbert. Quasi-monte carlo and extended first-shot improvement to the multi-path method. In Laszlo Szirmay-Kalos, editor, *Proceedings of the Spring Conference on Computer Graphics '98*, pages 91–102, Budimerce, Slovakia, 1998. Comenius University.

[CNSD+92] Carolina Cruz-Neira, Daniel J. Sandin, Thomas A. DeFanti, Robert V. Kenyon, and John C. Hart. The cave: audio visual experience automatic virtual environment. *Commun. ACM*, 35(6):64–72, 1992.

[Coo86] Robert L. Cook. Stochastic sampling in computer graphics. *ACM Transactions on Graphics*, 5(1):51–72, January 1986.

[CPC84] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, volume 18, pages 137–145, New York, NY, USA, July 1984. ACM Press.

[CPWAP08] Ewen Cheslack-Postava, Rui Wang, Oskar Akerlund, and Fabio Pellacini. Fast, realistic lighting and material design using nonlinear cut approximation. *ACM Trans. Graph.*, 27:128:1–128:10, December 2008.

[CRMT91] Shenchang Eric Chen, Holly E. Rushmeier, Gavin Miller, and Douglass Turner. A progressive multi-pass method for global illumination. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, volume 25, pages 165–174, New York, NY, USA, July 1991. ACM Press.

[CS98] Francesc Castro and Mateu Sbert. Application of quasi-monte carlo sampling to the multi-path method for radiosity. In *Proceedings of the Third International Conference on Monte Carlo and quasi Monte Carlo methods in scientific computing*, pages 163–176, Berlin, 1998. Springer.

[CTB+05] Marcello Carrozzino, Franco Tecchia, Sandro Bacinelli, Carlo Cappelletti, and Massimo Bergamasco. Lowering the development time of multimodal interactive application: the real-life experience of the xvr project. In *ACE '05: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 270–273, 2005.

[dBOS92] Mark de Berg, Mark Overmars, and Otfried Schwarzkopf. Computing and verifying depth orders. In *SCG '92: Proceedings of the eighth annual symposium on Computational geometry*, pages 138–145, New York, NY, USA, 1992. ACM Press.

[DKTS07] Zhao Dong, Jan Kautz, Christian Theobalt, and Hans-Peter Seidel. Interactive global illumination using implicit visibility. In *PG '07: Proceedings of the 15th Pacific Conference on Computer Graphics and Applications*, pages 77–86, Washington, DC, USA, 2007. IEEE Computer Society.

[DLW93] Philip Dutré, Eric P.F. Lafortune, and Yves D. Willems. Monte carlo light tracing with direct computation of pixel intensities. In *Proceedings of Compugraphics '93*, pages 128–137, 1993.

[Dob00] William H. Dobelle. Artificial vision for the blind by connecting a television camera to the visual cortex. *ASAIO journal (American Society for Artificial Internal Organs : 1992)*, 46(1):3–9, Jan–Feb 2000.

[DS05] Carsten Dachsbacher and Marc Stamminger. Reflective shadow maps. In *I3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 203–231, New York, NY, USA, 2005. ACM.

[DS06] Carsten Dachsbacher and Marc Stamminger. Splatting indirect illumination. In *I3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 93–100, New York, NY, USA, 2006. ACM.

[DSDD07] Carsten Dachsbacher, Marc Stamminger, George Drettakis, and Frédo Durand. Implicit visibility and antiradiance for interactive global illumination. *ACM Trans. Graph.*, 26(3):61, 2007.

[Dut96] Philip Dutré. *Mathematical Frameworks and Monte Carlo Algorithms for Global Illumination in Computer Graphics*. PhD thesis, Katholieke Universiteit Leuven, September 1996.

[Dut03] Philip Dutré. *Global Illumination Compendium*. Katholieke Universiteit Leuven, 2003.

[FCM09] C. Fowler, S. Collins, and M. Manzke. Accelerated entry point search algorithm for real time ray tracing. In *ACM Spring Conference on Computer Graphics*, 2009.

[FDFH90] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley, 2 edition, 1990.

[Fek90] György Fekete. Rendering and managing spherical data with sphere quadtrees. In *VIS '90: Proceedings of the 1st conference on Visualization '90*, pages 176–186, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press.

[FS05] Tim Foley and Jeremy Sugerman. KD-tree acceleration structures for a GPU raytracer. In *HWWS '05: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 15–22, 2005.

[GDW00] Xavier Granier, George Drettakis, and Bruce Walter. Fast global illumination including specular effects. In B. Peroche and H. Rushmeier, editors, *Rendering Techniques 2000 (Proceedings of the Eleventh Eurographics Workshop on Rendering)*, pages 47 – 59. Eurographics, Springer Wien, 2000.

[GGSC96] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 43–54, 1996.

[GHLM05] Naga K. Govindaraju, Michael Henson, Ming Lin, and Dinesh Manocha. Interactive visibility ordering and transparency computations among geometric primitives in complex environments. In *Proceedings of ACM Symposium on Interactive 3D Graphics and Games 2005*, 2005.

[GLM04] Naga K. Govindaraju, Ming Lin, and Dinesh Manocha. Vis-sort: Fast visibility ordering of 3-d geometric primitives. Technical report, University of North Carolina at Chapel Hill, 2004.

[GMN94] Jay S. Gondek, Gary W. Meyer, and Jonathan G. Newman. Wavelength dependent reflectance functions. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 213–220, New York, NY, USA, 1994. ACM Press.

[GSHG98] Gene Greger, Peter Shirley, Philip M. Hubbard, and Donald P. Greenberg. The irradiance volume. *IEEE Computer Graphics & Applications*, 18(2):32–43, March 1998.

[GTGB84] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modeling the interaction of light between diffuse surfaces. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 213–222, New York, NY, USA, 1984. ACM Press.

[GWH01] Michael Garland, Andrew Willmott, and Paul S. Heckbert. Hierarchical face clustering on polygonal surfaces. In *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 49–58, New York, NY, USA, 2001. ACM.

[Hav01] Vlastimil Havran. *Heuristic Ray Shooting Algorithms*. PhD thesis, Czech Technical University in Prague, Prague, April 2001.

[Hec90] Paul S. Heckbert. Adaptive radiosity textures for bidirectional ray tracing. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 145–154, New York, NY, USA, 1990. ACM Press.

[HG83] Roy Hall and Donald P. Greenberg. A testbed for realistic image synthesis. *IEEE Computer Graphics & Applications*, 3(8):10–20, November 1983.

[HM99] John F. Hughes and Tomas Möller. Building an orthonormal basis from a unit vector. *Journal of Graphics Tools*, 4(4):33–35, 1999.

[HOJ08] Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. Progressive photon mapping. *ACM Trans. Graph.*, 27:130:1–130:8, December 2008.

[HPB07] Miloš Hašan, Fabio Pellacini, and Kavita Bala. Matrix row-column sampling for the many-light problem. *ACM Trans. Graph.*, 26, July 2007.

[HSA91] Pat Hanrahan, David Salzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 197–206, New York, NY, USA, 1991. ACM Press.

[HW91] Paul S. Heckbert and James M. Winget. Finite element methods for global illumination. Technical report, University of California, Computer Science Division, July 1991.

[ICG86] David S. Immel, Michael F. Cohen, and Donald P. Greenberg. A radiosity method for non-diffuse environments. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 133–142, New York, NY, USA, 1986. ACM Press.

[IPL97] I. Ihm, S. Park, and R. K. Lee. Rendering of spherical light fields. In *Proceedings of the 5th Pacific Conference on Computer Graphics and Applications*, pages 59–68, Seoul, South Korea, October 1997. IEEE Computer Society.

[JB07] C. R. Johns and D. A. Brokenshire. Introduction to the cell broadband engine architecture. *IBM Journal of Research and Development*, 51(5):503–519, 2007.

[JC95] Henrik Wann Jensen and Niels Jørgen Christensen. Photon maps in bidirectional monte carlo ray tracing of complex objects. *Computers & Graphics*, 19(2):215–224, 1995.

[JDO08] David Luebke Simon Green John E. Stone James C. Phillips John D. Owens, Mike Houston. Gpu computing. *Proceedings of the IEEE*, 96(5):879–899, May 2008.

[Jen95] Henrik Wann Jensen. Importance driven path tracing using the photon map. In *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, pages 326–335, New York, NY, 1995. Springer-Verlag.

[Jen96] Henrik Wann Jensen. Global illumination using photon maps. In *Rendering Techniques '96 (Proceedings of the 7th Eurographics Workshop on Rendering)*, pages 21–30, London, UK, 1996. Springer-Verlag.

[JMLH01] Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy, and Pat Hanrahan. A practical model for subsurface light transport. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 511–518, New York, NY, USA, August 2001. ACM Press.

[Kaj86] James T. Kajiya. The rendering equation. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 143–150, New York, NY, USA, 1986. ACM Press.

[KBLD08] Jakub Kurzak, Alfredo Buttari, Piotr Luszczek, and Jack Dongarra. The playstation 3 for high-performance scientific computing. *Computing in Science and Engg.*, 10(3):84–87, 2008.

[KBW06] Jens Krüger, Kai Bürger, and Rüdiger Westermann. Interactive screen-space accurate photon tracing on GPUs. In *Rendering Techniques (Eurographics Symposium on Rendering - EGSR)*, pages 319–329, June 2006.

[Kel96] Alexander Keller. Quasi-Monte Carlo Radiosity. In *Rendering Techniques '96 (Proceedings of the 7th Eurographics Workshop on Rendering)*, pages 101–110, New York, NY, 1996. Springer-Verlag/Wien.

[Kel97] Alexander Keller. Instant radiosity. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 49–56, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[KGPB05] Jaroslav Krivanek, Pascal Gautron, Sumanta Pattanaik, and Kadi Bouatouch. Radiance caching for efficient global illumination computation. *IEEE Transactions on Visualization and Computer Graphics*, 11(5):550–561, 2005.

[KH84] James T. Kajiya and Brian P Von Herzen. Ray tracing volume densities. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 165–174, New York, NY, USA, 1984. ACM Press.

[Kil02] Mark J. Kilgard. Improving shadows and reflections via the stencil buffer. White paper, Nvidia Corporation, 2002.

[Lar98] Greg Ward Larson. The holodeck: A parallel ray-caching rendering system. In *Proceedings of Eurographics Workshop on Parallel Graphics and Visualization*, 1998.

[LB83] You-Dong Liang and Brian A. Barsky. An analysis and algorithm for polygon clipping. *Commun. ACM*, 26(11):868–877, 1983.

[LE10] H. Ludvigsen and A. C. Elster. Real-time ray tracing using nvidia optix. In *Eurographics 2010 short papers*, 2010.

[LH96] Marc Levoy and Pat Hanrahan. Light field rendering. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42, 1996.

[LR98] Dani Lischinski and Ari Rappoport. Image-based rendering for non-diffuse synthetic scenes. In *Rendering Techniques '98 (Proceedings of the 9th Eurographics Rendering Workshop)*, pages 301–314, Vienna, Austria, June 1998.

[LSK+07] Samuli Laine, Hannu Saransaari, Janne Kontkanen, Jaakko Lehtinen, and Timo Aila. Incremental instant radiosity for real-time indirect illumination. In *Proceedings of Eurographics Symposium on Rendering 2007*, pages 277–286. Eurographics Association, 2007.

[LW93] Eric P. Lafortune and Yves D. Willems. Bi-directional path tracing. In *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93)*, pages 145–153, Alvor, Portugal, 1993.

[LW94a] E.P. Lafortune and Y.D. Willems. A theoretical framework for physically based rendering. *Computer Graphics Forum, Special Issue on Rendering*, 13(2):97–107, June 1994.

[LW94b] Eric P.F. Lafortune and Yves D. Willems. Using the modified phong brdf for physically based rendering. Technical report, Katholieke Universiteit Leuven, Department of Computer Science, K.U.Leuven, 1994.

[Max54] James Clerk Maxwell. *A Treatise on Electricity and Magnetism*, volume 1. Dover Publications, 3 edition, 1954.

[Max86] Nelson L. Max. Atmospheric illumination and shadows. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 117–124, New York, NY, USA, 1986. ACM Press.

[MB95] Leonard McMillan and Gary Bishop. Plenoptic modeling: an image-based rendering system. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 39–46, New York, NY, USA, 1995. ACM Press.

[MGAK03] William R. Mark, R.Steven Glanville, Kurt Akeley, and Mark J. Kilgard. Cg: A system for programming graphics hardware in a c-like language. *ACM Transactions on Graphics*, 22(3):896–907, 2003.

[MKS07] Jesper Mortensen, Pankaj Khanna, and Mel Slater. Light field propagation and rendering on the gpu. In *AFRIGRAPH '07: Proceedings of the 5th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, pages 15–23, New York, NY, USA, 2007. ACM.

[MKYS07] Jesper Mortensen, Pankaj Khanna, Insu Yu, and Mel Slater. Real-time global illumination in the cave. In *VRST '07: Proceedings of the 2007 ACM symposium on Virtual reality software and technology*, pages 145–148, New York, NY, USA, 2007. ACM.

[ML09] Morgan McGuire and David Luebke. Hardware-accelerated global illumination by image space photon mapping. In *HPG '09: Proceedings of the Conference on High Performance Graphics 2009*, pages 77–89, New York, NY, USA, 2009. ACM.

[MRP98] Gavin Miller, Steven M. Rubin, and Dulce Ponceleon. Lazy decompression of surface light fields for precomputed global illumination. In G. Drettakis and N. Max, editors, *Rendering Techniques '98 (Proceedings of the 9th Eurographics Rendering Workshop)*, pages 281–292, July 1998.

[MVT+07] Giuseppe Marino, Davide Vercelli, Franco Tecchia, Paolo Simone Gasparello, and Massimo Bergamasco. Description and performance analysis of a distributed rendering architecture for virtual environments. In *ICAT 2007: Proceedings of the 17th Annual ICAT Conference on Artificial Reality and Telexistence*, 2007.

[MYK+08] Jesper Mortensen, Insu Yu, Pankaj Khanna, Franco Tecchia, Bernhard Spanlang, Giuseppe Marino, and Mel Slater. Real-time global illumination for vr applications. *IEEE Computer Graphics and Applications*, 28(6):56–64, 2008.

[Neu95] László Neumann. Monte carlo radiosity. *Computing*, 55(2):23–42, March 1995.

[NMN87] Tomoyuki Nishita, Yasuhiro Miyawaki, and Eihachiro Nakamae. A shading model for atmospheric scattering considering luminous intensity distribution of light sources. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 303–310, New York, NY, USA, 1987. ACM Press.

[NN85] Tomoyuki Nishita and Eihachiro Nakamae. Continuous tone representation of three-dimensional objects taking account of shadows and interreflection. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, volume 19, pages 23–30, New York, NY, USA, July 1985. ACM Press.

[NNS72] M.E. Newell, R.G. Newell, and T.L. Sancha. A solution to the hidden surface problem. In *ACM'72: Proceedings of the ACM annual conference*, pages 443–450, 1972.

[Nov05] Justin Novosad. *GPU Gems 2*, chapter 27 - Advanced High-Quality Filtering, pages 417–435. Addison Wesley, 2005.

[NRH+77] F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, and T. Limperis. Geometric considerations and nomenclature for reflectance. Monograph 161, National Bureau of Standards (US), October 1977.

[NW09] Greg Nichols and Chris Wyman. Multiresolution splatting for indirect illumination. In *I3D '09: Proceedings of the 2009 symposium on Interactive 3D graphics and games*, pages 83–90, New York, NY, USA, 2009. ACM.

[OR02] J J O'Connor and E F Robertson. Light through the ages: Ancient greece to maxwell, August 2002.

[Pat93a] Sumanta N. Pattanaik. *Computational Methods for Global Illumination and Visualisation of Complex 3D Environments*. PhD thesis, NCST Birla Institute of Technology and Science, Pilami, India, 1993.

[Pat93b] Sumanta N. Pattanaik. The mathematical framework of adjoint equations for illumination computation. In *ICCG '93: Proceedings of the IFIP TC5/WG5.2/WG5.10 CSI International Conference on Computer Graphics*, pages 123–138. North-Holland, 1993.

[PBMH02] Timothy J. Purcell, Ian Buck, William R. Mark, and Pat Hanrahan. Ray tracing on programmable graphics hardware. *ACM Transactions on Graphics*, 21(3):703–712, July 2002.

[PCI02] PCI-SIG. *PCI Express Base Specification, Revision 1.0*, July 2002. www.pcisig.org.

[Pel95] M. Pellegrini. Monte carlo approximation of form factors with error bounded a priori. In *SCG '95: Proceedings of the eleventh annual symposium on Computational geometry*, pages 287–296, New York, NY, USA, 1995. ACM Press.

[PH04] Matt Pharr and Greg Humphreys. *Physically Based Rendering : From Theory to Implementation*. Morgan Kaufmann, 2004.

[Pho75] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, 1975.

[PM92] S. N. Pattanaik and S. P. Mudur. Computation of global illumination by monte carlo simulation of the particle model of light. In *Rendering Techniques 92 (Proceedings of the Third Eurographics Workshop on Rendering)*, 1992.

[PMS+99] Steven Parker, William Martin, Peter-Pike J. Sloan, Peter Shirley, Brian Smits, and Charles Hansen. Interactive ray tracing. In *Proceedings of the ACM Symposium on Interactive 3D Graphics 1999*, pages 119–126, April 1999.

[RGK+08] T. Ritschel, T. Grosch, M. H. Kim, H.-P. Seidel, C. Dachsbacher, and J. Kautz. Imperfect shadow maps for efficient computation of indirect illumination. *ACM Trans. Graph.*, 27(5):1–8, 2008.

[RGKM07] Tobias Ritschel, T. Grotsch, Jan Kautz, and S. Muller. Interactive illumination with coherent shadow maps. In *Proc. Eurographics Symposium on Rendering*, 2007.

[RGKS08] Tobias Ritschel, Thorsten Grosch, Jan Kautz, and Hans-Peter Seidel. Interactive global illumination based on coherent surface shadow maps. In *GI '08: Proceedings of graphics interface 2008*, pages 185–192, Toronto, Ont., Canada, Canada, 2008. Canadian Information Processing Society.

[RH01] Ravi Ramamoorthi and Pat Hanrahan. An efficient representation for irradiance environment maps. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 497–500, 2001.

[Rom10] Vitkovskiy Roman. Comparison of nvidia graphics processing units. Wikipedia, April 2010.

[Rot82] Scott D. Roth. Ray Casting for Modeling Solids. *Computer Graphics and Image Processing*, 18(2):109–144, February 1982.

[RSH00] Erik Reinhard, Brian Smits, and Chuck Hansen. Dynamic acceleration structures for interactive ray tracing. In *Rendering Techniques 2000 (Proceedings of the Eleventh Eurographics Workshop on Rendering)*, pages 299–306, June 2000.

[RSH05] Alexander Reshetov, Alexei Soupikov, and Jim Hurley. Multi-level ray tracing algorithm. *ACM Transactions on Graphics*, 24(3):1176–1185, 2005.

[RT90] Holly E. Rushmeier and Kenneth E. Torrance. Extending the radiosity method to include specularly reflecting and translucent materials. *ACM Trans. Graph.*, 9(1):1–27, 1990.

[SAG94] Brian Smits, James Arvo, and Donald Greenberg. A clustering algorithm for radiosity in complex environments. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 435–442, New York, NY, USA, 1994. ACM Press.

[SAS92] Brian E. Smits, James R. Arvo, and David H. Salesin. An importance-driven radiosity algorithm. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 273–282, New York, NY, USA, 1992. ACM Press.

[SAWG91] François X. Sillion, James R. Arvo, Stephen H. Westin, and Donald P. Greenberg. A global illumination solution for general reflectance distributions. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 187–196, New York, NY, USA, 1991. ACM Press.

[SB97] Wolfgang Stürzlinger and Rui Bastos. Interactive rendering of globally illuminated glossy scenes. In *Proceedings of the Eurographics Workshop on Rendering Techniques '97*, pages 93–102, London, UK, 1997. Springer-Verlag.

[Sbe93] Mateu Sbert. An integral geometry based method for fast form-factor computation. *Computer Graphics Forum*, 12(3):409–420, 1993.

[SBM94] Clifford M. Stein, Barry G. Becker, and Nelson L. Max. Sorting and hardware assisted rendering for volume visualization. In *VVS '94: Proceedings of the 1994 symposium on Volume visualization*, pages 83–89, October 1994.

[SGwHS98] Jonathan Shade, Steven Gortler, Li wei He, and Richard Szeliski. Layered depth images. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 231–242, 1998.

[Shi90a] Peter S. Shirley. *Physically Based Lighting Calculations for Computer Graphics*. PhD thesis, University of Illinois at Urbana-Champaign, December 1990.

[Shi90b] Peter S. Shirley. A ray tracing method for illumination calculation in diffuse-specular scenes. In *Proceedings on Graphics interface '90*, pages 205–212, Toronto, Ont., Canada, Canada, 1990. Canadian Information Processing Society.

[Shi91] Peter Shirley. Time Complexity of Monte Carlo Radiosity. In Werner Purgathofer, editor, *Eurographics '91*, pages 459–465, Amsterdam, North-Holland, 1991. Elsevier Science Publishers.

[SHSS00] Marc Stamminger, Joerg Haber, Hartmut Schirmacher, and Hans-Peter Seidel. Walkthroughs with corrective texturing. In B. Peroche and H. Rushmeier, editors, *Rendering Techniques 2000 (Proceedings of the Eleventh Eurographics Workshop on Rendering)*, pages 377–388, New York, NY, 2000. Springer Wien.

[SJ00] Brian Smits and Henrik Wann Jensen. Global illumination test scenes. Technical report, University of Utah, Computer Science Department, University of Utah, 2000.

[SK98] László Szirmay-Kalos. Global ray-bundle tracing. Technical Report TR-186-2-98-18, Institute of Computer Graphics and Algorithms, Vienna University of Technology, 1998.

[SKP98] László Szirmay-Kalos and Werner Purgathofer. Global ray-bundle tracing with hardware acceleration. In *Rendering Techniques '98 (Proceedings of the 9th Eurographics Rendering Workshop)*, pages 247–258, June 1998.

[SKS02] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 527–536, New York, NY, USA, 2002. ACM.

[SKSMT00] László Szirmay-Kalos, Mateu Sbert, Roel Martinez, and Robert F. Tobler. Incoming first-shot for non-diffuse global illumination. In *Spring Conference on Computer Graphics*, Budmerice, Hungary, 2000.

[SKTNB97] László Szirmay-Kalos, Foris Tibor, Laszlo Neumann, and Csebfalvi Balazs. An analysis of quasi-monte carlo integration applied to the transillumination radiosity method. *Computer Graphics Forum*, 16(3):271–281, 1997.

[Sla02] Mel Slater. Constant time queries on uniformly distributed points on a hemisphere. *Journal of Graphics Tools*, 7(1):33–44, 2002.

[SMKY04] Mel Slater, Jesper Mortensen, Pankaj Khanna, and Insu Yu. A virtual light field approach to global illumination. In *Proceedings of Computer Graphics International (CGI 2004)*, pages 102–109. IEEE Computer Society Press, June 16-19 2004.

[SP89] François X. Sillion and Claude Puech. A general two-pass method integrating specular and diffuse reflection. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, volume 23, pages 335–344, New York, NY, USA, July 1989. ACM Press.

[SP07] Kelvin Sung and Michael Panitz. Developing applications on the xbox 360 console. *J. Comput. Small Coll.*, 23(2):71–72, 2007.

[SPL88] Min-Zhi Shao, Qun-Sheng Peng, and You-Dong Liang. A new radiosity approach by procedural refinements for realistic image synthesis. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, volume 22, pages 93–101, August 1988.

[SPNP96] Mateu Sbert, Xavier Pueyo, László Neumann, and Werner Purgathofer. Global multipath monte carlo algorithms for radiosity. *The Visual Computer*, 12(2):47–61, February 1996.

[SPP95] Mateu Sbert, Fredric Pérez, and Xavier Pueyo. Global monte-carlo: A progressive solution. In *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, pages 231–239, 1995.

[SS95] Peter Schröder and Wim Sweldens. Spherical wavelets: efficiently representing functions on the sphere. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 161–172, New York, NY, USA, 1995. ACM Press.

[SS00] Maryann Simmons and Carlo H. Séquin. Tapestry: A dynamic mesh-based display representation for interactive rendering. In *Rendering Techniques 2000 (Proceedings of the Eleventh Eurographics Workshop on Rendering)*, pages 329–340, London, UK, 2000. Springer-Verlag.

[SSS74] Ivan E. Sutherland, Robert F. Sproull, and Robert A. Schumacker. A characterization of ten hidden-surface algorithms. *ACM Computing Surveys*, 6(1):1–55, March 1974.

[SUC95] Mel Slater, Martin Usoh, and Yiorgos Chrysanthou. The influence of dynamic shadows on presence in immersive virtual environments. In *Computer Science, editor, Virtual Environments 95*, pages 8–21. Springer, 1995.

[SWH+95] Peter Shirley, Bretton Wade, Philip Hubbard, David Zareski, Bruce Walter, and Donald P. Greenberg. Global illumination via density estimation. In *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, June 1995.

[TPWG02] Parag Tole, Fabio Pellacini, Bruce Walter, and Donald P. Greenberg. Interactive global illumination in dynamic scenes. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 537–546, New York, NY, USA, 2002. ACM Press.

[Ura05] Yury Uralsky. *GPU Gems 2*, chapter 17 - Efficient Soft-Edged Shadows Using Pixel Shader Branching, pages 269–282. Addison Wesley, 2005.

[Vea97] Eric Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, 1997.

[VG94] Eric Veach and Leonidas J. Guibas. Bidirectional estimators for light transport. In *Proceedings of the 5th Eurographics Workshop on Rendering*, pages 147–162, 1994.

[VG95] Eric Veach and Leonidas J. Guibas. Optimally combining sampling techniques for monte carlo rendering. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 419–428, New York, NY, USA, August 1995. ACM Press.

[VG97] Eric Veach and Leonidas J. Guibas. Metropolis light transport. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 65–76, New York, NY, USA, August 1997. ACM Press/Addison-Wesley Publishing Co.

[WAA+00] Daniel Wood, Daniel Azuma, Wyvern Aldinger, Brian Curless, Tom Duchamp, David Salesin, and Werner Stuetzle. Surface light fields for 3d photography. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 287–296, New York, NY, USA, August 2000. ACM Press/Addison-Wesley Publishing Co.

[WAL+97] Bruce Walter, Gün Alppay, Eric Lafortune, Sebastian Fernandez, and Donald P. Greenberg. Fitting virtual lights for non-diffuse walkthroughs. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 45–48, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[Wal04] Ingo Wald. *Realtime Ray Tracing and Interactive Global Illumination*. PhD thesis, Saarland University, 2004.

[War92] Gregory J. Ward. Measuring and modeling anisotropic reflection. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, volume 26, pages 265–272, New York, NY, USA, August 1992. ACM Press.

[War94] Gregory J. Ward. The radiance lighting simulation and rendering system. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 459–472, New York, NY, USA, July 1994. ACM Press.

[War08] Greg Ward. The hopeful future of high dynamic range imaging: invited paper. In *SIGGRAPH '08: ACM SIGGRAPH 2008 classes*, pages 1–3, New York, NY, USA, 2008. ACM.

[WBS03] Ingo Wald, Carsten Benthin, and Philipp Slusallek. Distributed interactive ray tracing of dynamic scenes. In *Proceedings of the IEEE Symposium on Parallel and Large-Data Visualization and Graphics (PVG)*, pages 77–86, 2003.

[WCG87] John R. Wallace, Michael F. Cohen, and Donald P. Greenberg. A two-pass solution to the rendering equation: A synthesis of ray tracing and radiosity methods. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 311–320, New York, NY, USA, 1987. ACM Press.

[WDB+06] Ingo Wald, Andreas Dietrich, Carsten Benthin, Alexander Efremov, Tim Dahmen, Johannes Günther, Vlastimil Havran, Hans-Peter Seidel, and Philipp Slusallek. A ray tracing based framework for high-quality virtual reality in industrial design applications. In *Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing*, pages 177–185, September 2006.

[WDG02] Bruce Walter, George Drettakis, and Donald P. Greenberg. Enhancing and optimizing the render cache. In *Rendering Techniques 2002 (Proceedings of the 13th Eurographics workshop on Rendering)*, pages 37–42, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

[WDP99] Bruce Walter, George Drettakis, and Steven Parker. Interactive rendering using the render cache. In *Rendering techniques '99 (Proceedings of the 10th Eurographics Workshop on Rendering)*, volume 10, pages 235–246, Jun 1999.

[WEH89] J. R. Wallace, K. A. Elmquist, and E. A. Haines. A ray tracing algorithm for progressive radiosity. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, volume 23, pages 315–324, New York, NY, USA, 1989. ACM Press.

[WFA+05] Bruce Walter, Sebastian Fernandez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P. Greenberg. Lightcuts: a scalable approach to illumination. *ACM Trans. Graph.*, 24(3):1098–1107, 2005.

[WH92] Gregory J. Ward and Paul S. Heckbert. Irradiance gradients. In *Rendering Techniques 92 (Proceedings of the Third Eurographics Workshop on Rendering)*, pages 85–98, Bristol, UK, May 1992.

[Whi80] Turner Whitted. An improved illumination model for shaded display. *Communications of the ACM*, 23(6):343–349, 1980.

[WHSG97] Bruce Walter, Philip M. Hubbard, Peter Shirley, and Donald P. Greenberg. Global illumination using local linear density estimation. *ACM Transactions on Graphics*, 16(3):217–259, 1997.

[Wil83] Lance Williams. Pyramidal parametrics. *SIGGRAPH Comput. Graph.*, 17(3):1–11, 1983.

[Wil92] Peter L. Williams. Visibility-ordering meshed polyhedra. *ACM Transactions on Graphics*, 11(2):103–126, 1992.

[WKB+02] Ingo Wald, Thomas Kollig, Carsten Benthin, Alexander Keller, and Philipp Slusallek. Interactive global illumination using fast ray tracing. In *Rendering Techniques 2002 (Proceedings of the 13th Eurographics workshop on Rendering)*, 2002.

[WMS98] Peter L. Williams, Nelson L. Max, and Clifford M. Stein. A high accuracy volume renderer for unstructured data. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):37–54, March 1998.

[WRC88] Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A ray tracing solution for diffuse interreflection. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, volume 22, pages 85–92, New York, NY, USA, August 1988. ACM Press.

[WS99] Gregory Ward and Maryann Simmons. The holodeck ray cache: an interactive rendering system for global illumination in nondiffuse environments. *ACM Transactions on Graphics*, 18(4):361–368, 1999.

[WS03] Michael Wand and Wolfgang Straßer. Real-time caustics. In *Computer Graphics Forum*, volume 22(3), pages 611–620, 2003.

[WWZ+09] Rui Wang, Rui Wang, Kun Zhou, Minghao Pan, and Hujun Bao. An efficient gpu-based approach for interactive global illumination. In *ACM SIGGRAPH 2009 papers*, SIGGRAPH '09, pages 91:1–91:8, New York, NY, USA, 2009. ACM.

[YMKS10] Insu Yu, Jesper Mortensen, Pankaj Khanna, and Mel Slater. A note on the influence of realistic rendering on presence in immersive virtual environments. University College London, June 2010.