December 1997

An Image-Based Framework For Global Illumination In Animated Environments

Jeffry S. Nimeroff
University of Pennsylvania

University of Pennsylvania Institute for Research in Cognitive Science Technical Report No. IRCS-98-03.

An Image-Based Framework For Global Illumination In Animated Environments

Abstract
Interacting with environments exhibiting the subtle lighting effects found in the real world gives designers a better understanding of the scene's structure by providing rich visual cues. The major hurdle is that global illumination algorithms are too inefficient to quickly compute their solutions for reasonably sized environments. When motion is allowed within the environment, the problem becomes even more intractable. We address the problem of sampling and reconstructing an environment's time-varying radiance distribution, its spatio-temporal global illumination information, allowing the efficient generation of arbitrary views of the environment at arbitrary points in time. The radiance distribution formalizes incoming chromatic radiance at all points within a constrained view space, along all directions, at all times. Since these distributions cannot, in general, be calculated analytically, we introduce a framework for specifying and computing sample values from the distribution and progress through a series of sample-based approximations designed to allow easy and accurate reconstruction of images extracted from the distribution. The first approximation is based on storing time-sequences of images at strategic locations within the chosen view space. An image of the environment is constructed by first blending the images contained in the individual time-sequences to get the desired time and then using view interpolation to merge the proximate views. The results presented here demonstrate the feasibility and utility of the method but also show its major drawback. An inability to accurately model the temporal radiance variations using image sequences without resorting to a high sampling rate leads to the replacement of the image sequences by a sparse temporal image volume representation for storing randomly, or adaptively, placed radiance samples. Triangulation techniques are then used to reconstruct the radiance distribution at the desired time from a proximate set of stored spatio-temporal radiance samples. The results presented here show that temporal image volumes allow for more accurate and efficient temporal reconstruction requiring less sampling than the more traditional time-sequence approach.

Comments
University of Pennsylvania Institute for Research in Cognitive Science Technical Report No. IRCS-98-03.

AN IMAGE-BASED FRAMEWORK FOR GLOBAL ILLUMINATION IN ANIMATED ENVIRONMENTS

JEFFRY S. NIMEROFF

A DISSERTATION
in

COMPUTER AND INFORMATION SCIENCE

Presented to the Faculties of the University of Pennsylvania in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy
1997

Dr. Julie Dorsey
Supervisor of Dissertation

Dr. Mark Steedman
Graduate Group Chairperson

Acknowledgments

It takes the understanding of a lot of people to make a 5-plus year experience enjoyable. I have not only found a lot of understanding souls at the University of Pennsylvania and elsewhere, but am 100% sure that I made them re-think their positions! Being compulsively overstated, I'll try to be brief in describing what each of these souls has done for me.

Nothing could ever be accomplished in this world without a loving and supportive family. My life is no exception. My parents, Pat and Mike might have done some homework for me when I was younger, but let me solve my own problems, whatever the outcome, without passing judgment. They offered advice at the pace I required. The thing that I never had to worry about was how they felt about me. Their love is unconditional, as I've tested. I am a very lucky person. I only hope they can fathom how much I love them.

My sister Jami is a mirror of myself. Although we have chosen different careers, we are like peas in a pod. She pushes me, I push back. We both are better for the experience. We are both quick to anger, tenacious, and outspoken. That is why we get along so well. I love her very much.

My newest family member, my wife Jennifer, deserves the most credit here. Unlike my parents and my sister, she is a part of my life through a conscious choice she has made. I don't think I could easily tolerate someone as trying as myself, but she tolerates me and then some. I may question her sanity but nothing else. I love her with all my heart. She is my world.

I have had many friends in my adult life that have helped mold me into the person I have become. My friends from home, Al and Jerry, have always been there for me. Whether it is tripping to LA, Mexico, NYC, or AC, or just hanging out, they make sure I have fun when I need it most. My friends from undergrad, Dave, Eric, Frank, Issy, and Mark, are the exact opposite. Of course they are there for me, but they also make it a point to make sure I have fun when I can least afford to. I wouldn't have things any other way. My friends from my first graduate school experience, Greg, JJ, and Naseer showed me how diverse the world can be. We are, at the same time, very different and very similar. Although our paths have been different, they have been tailored to achieve the same type of success for each of us. They got me through the point in my life where dropping out of graduate school was a distinct possibility. My last group of graduate school friends, Barry, Christy, Doug, Ken, Harold, Mike, Nick, Paul, and Vangelis stoked my fire with stimulating conversation about computer graphics and life, and filled me full of libations and hot wings. I can't figure out if this is good or bad! A special acknowledgment has to go out to my officemates and good friends, the choads, Chris, and Charlie. Nothing would have been possible without Carolina's, "spaghetti," and a little bit of scotch.

On the technical side of things, I have been blessed to find a group of researchers who can sustain themselves when I spew my millions of incoherent ideas. They are so good, in fact, that they can even smile while I'm raving.

My friend and thesis advisor, Julie Dorsey, is a very diverse and talented individual. Handling computer science and architecture both technically and artfully is a difficult task that she performs very well. She has gotten me to focus my ideas so they can be expressed intelligibly. She has the knack for not squelching my unfocused ideas but instead helping me to prioritize them so they are not lost. This is a talent I wish I had myself. The best I can hope to do is to surround myself with more people like her. I hope I can find them.

My friend and external thesis advisor, Holly Rushmeier, is as technically proficient a researcher as I have come across. Along with fine-tuning the technical aspects of my work she was a great help in defining what the overall scope of my research needed to be. I still find it amazing that I found someone who was contemplating the same problem I was at exactly the same time (late 1994).

My friends and advisors, Dimitri Metaxas and Jean Gallier, were very helpful during those dry periods when my progress faltered. When I needed to get away from my work, they knew to leave me alone. When I needed a push, they were there to give it to me, even if it was over the edge of some intellectual cliff. For two of the most mathematically inclined people I know they also have great senses of humor.

Someone who deserves a special section all his own is my good friend, advisor, and "father figure," Norman Badler. Whether it was discussing my research, my wedding plans, or my stress level he always had the right advise. He didn't say the things that you wanted to hear. He said the things that you needed to hear, and defined those things that you ultimately needed to do. He greatest asset is his instinct. Although not as well versed in rendering as myself, his critique of my work was always accurate. He, flat out, knows how things should be presented. I owe him a great debt and intend to repay it in full.

I also need to acknowledge the help of some individuals and companies without whom my work would have severely suffered. Greg Ward at Lawrence Berkeley Labs answered more questions about graphics than I should have been allowed to ask. He is an encyclopedia of technical and non-technical graphics information; a true guru. Phil Thompson at MIT supplied me with great models, most that I couldn't have created in ten lifetimes. My work benefits immensely from his effort. Lightscape Technologies, Inc., MultiGen, Inc., Paradigm, Inc., and Silicon Graphics, Inc. supplied me with models, images, and equipment to keep my days and nights filled with work. Also, my deepest gratitude goes out to my co-researchers at the graphics lab who put up with me when I was stealing their machine cycles.

Last, but not least, since you can't work very well when you are hungry, my most heartfelt thanks goes out to Frita, Dimitri, and Bill for keeping me from going hungry even when I had no money.

Last but not least, I want to thank God for allowing me not to be born a *choad*, a *spaz* or a *sped*.

ABSTRACT

AN IMAGE-BASED FRAMEWORK FOR GLOBAL ILLUMINATION IN ANIMATED ENVIRONMENTS

JEFFRY S. NIMEROFF

DR. JULIE DORSEY

Interacting with environments exhibiting the subtle lighting effects found in the real world gives designers a better understanding of the scene's structure by providing rich visual cues. The major hurdle is that global illumination algorithms are too inefficient to quickly compute their solutions for reasonably sized environments. When motion is allowed within the environment, the problem becomes even more intractable. We address the problem of sampling and reconstructing an environment's time-varying radiance distribution, its spatio-temporal global illumination information, allowing the efficient generation of arbitrary views of the environment at arbitrary points in time. The radiance distribution formalizes incoming chromatic radiance at all points within a constrained view space, along all directions, at all times. Since these distributions cannot, in general, be calculated analytically, we introduce a framework for specifying and computing sample values from the distribution and progress through a series of sample-based approximations designed to allow easy and accurate reconstruction of images extracted from the distribution. The first approximation is based on storing time-sequences of images at strategic locations within the chosen view space. An image of the environment is constructed by first blending the images contained in the individual time-sequences to get the desired time and then using view interpolation to merge the proximate views. The results presented here demonstrate the feasibility and utility of the method but also show its major drawback. An inability to accurately model the temporal radiance variations using image sequences without resorting to a high sampling rate leads to the replacement of the image sequences by a sparse temporal image volume representation for storing randomly, or adaptively, placed radiance samples. Triangulation techniques are then used to reconstruct the radiance distribution at the desired time from a proximate set of stored spatio-temporal radiance samples. The results presented here show that temporal image volumes allow for more accurate and efficient temporal reconstruction requiring less sampling than the more traditional time-sequence approach.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

"A picture is worth a thousand words."

Fred R. Barnard

"At thirty frames per second, that's a lot of info..."

Jeffry Nimeroff

Observation is the key to understanding. Visual cues related to the interaction of light and the environment provide a wealth of information ranging from size, and shape to texture, roughness, and other material properties. Today, computer graphics research is concerned with generating this information via *global illumination* simulation algorithms. The results of these algorithms are accurate and visually impressive, albeit expensive to get. For all but the simplest scenes, an inordinate amount of time is needed to compute the subtle lighting effects you would expect to see if you were looking at a real environment. Designers want to be able to realistically model, illuminate and walk through complex environmental prototypes, but today's algorithms are not yet efficient enough to allow this.

How then does one expect to interact with large environments such as architectural structures, creating an efficient *walk-through system* (Figure 1.1)? The scenes we want to work with contain large numbers of geometrically complex static objects with complicated surface reflectance functions as well as large numbers of dynamic objects that add to the overall visual richness in the environment. Naively using today's rendering algorithms on these large datasets is futile. We instead need a new approach, one that makes judicious use of the available resources, pre-computing information and simplifying data wherever possible, exploiting any coherence that may exist in the environment.

The task of creating a unified approach for illuminating and interacting with realistic environments involves the synergy of solutions to three separate subproblems:

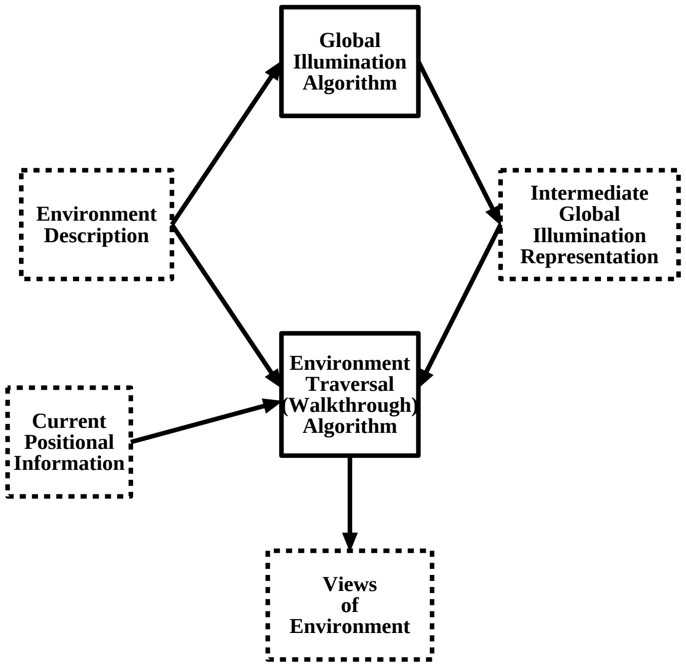

Global Illumination Algorithm
Environment Description
Intermediate Global Illumination Representation
Environment Traversal (Walkthrough) Algorithm
Current Positional Information
Views of Environment

Figure 1.1: A block diagram of a simple environment walk-through system.

1. How can we efficiently calculate a full global illumination solution for an arbitrary animated environment?

2. How can we efficiently store this time-varying global illumination and geometric information?

3. How can we efficiently use this stored information in a walk-through system that gives the user any view of the environment at any particular time?

When we can solve these individual subproblems and combine their solutions, we will have created the “ultimate visualization system.”

1.1 Perspective

Rendering research has taken two distinct paths towards the goal of creating a realistic color image of a synthetic environment. One path has concerned itself purely with the realism and physical correctness of the imagery produced while the other path has attempted to temper these accuracy issues with ideas for increased levels of interactivity. The two lines of research are now beginning to close in on each other.

1.1.1 Quest for realism

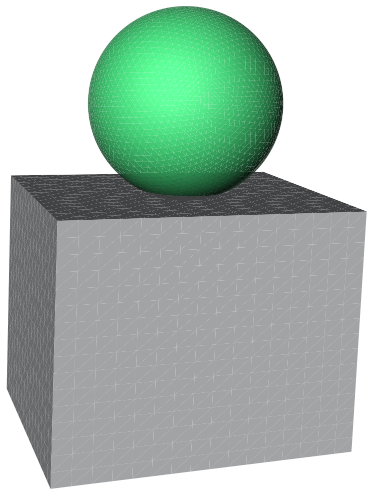


Figure 1.2: A simple environment rendered with the Phong smooth shading algorithm.

The incorporation of realistic illumination effects into computer environments can be traced back to the earliest hidden surface [328, 208, 55, 332] and polygon shading algorithms [44, 123, 240]. These algorithms resolved not only the problem of which objects could be seen from a particular viewpoint, but more importantly, what colors those objects appeared to be (Figure 1.2). The shading methods proposed were empirical, without a rigorous physical description, and *local*, without concern for the lighting effects generated by the interaction of an object with other objects in the scene, but did approximate the appearance of certain realistic materials like plastic and isotropic metal. While the interpolative methods of Gouraud [123] and Phong [240] became the staples of typical rendering/animation systems, their lack of any global illumination information hampered their use in systems where realism was paramount.

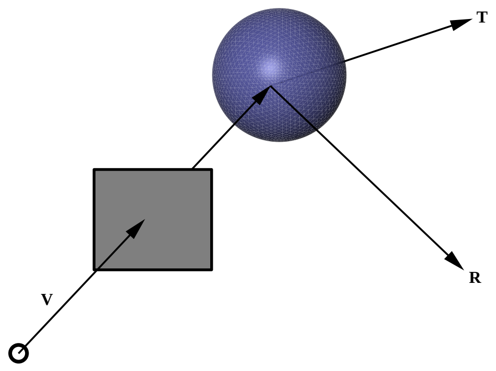


Figure 1.3: Geometry of a ray tracing system: viewing ray V, reflected ray R, transmitted ray T.

In the early eighties ideas from the ray casting literature [6] were adapted to implement the first non-local/*global* shading algorithm, *ray tracing* [335]. By following individual rays of light (Figure 1.3) from the viewpoint and using the laws of Hero for reflection and Snell for refraction, this recursive formulation yielded extraordinary results, accurately modeling shiny and transparent objects (Figure 1.4). Ray tracing is a view-dependent, brute-force method that requires many expensive calculations for each pixel and is inherently slow. Although coherence can be exploited under certain circumstances, e.g. which object was most recently seen near this location and what its color is, changes in the environment or the view often require a total re-calculation of the solution. This makes using ray tracing for computing global effects in a changing environment a burdensome task [111, 186]. Also, since it treats rays of light individually, many calculations are needed if the correct treatment of a large-scale, physically accurate environment is desired [74, 155, 232, 324].

The mid-eighties saw the move towards physically-based illumination algorithms with the introduction of the *radiosity* method [119, 221]. This method is based on the radiative heat transfer literature [280] and computes surface radiance (watt/meter^2/sr) via the solution of an integral

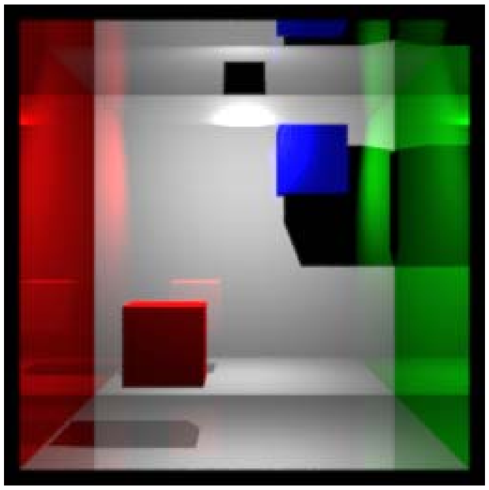


Figure 1.4: Simple environment rendered with the standard Whitted ray tracing algorithm.

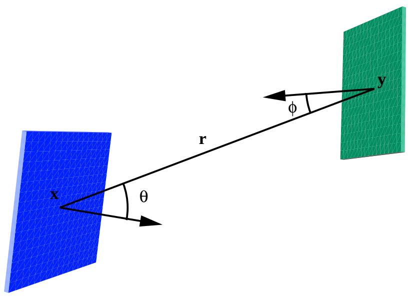


Figure 1.5: Pairwise surface geometry in a radiosity system.

equation which couples the surface points in the environment (Figure 1.5). A discrete linear system is created from the integral equation by both simplifying the type of reflection handled[1], and decomposing the environment into differential surfaces. The radiosity method is known for easily handling the subtle shading effects, such as soft shadowing and color bleeding, that pervade realistic environments (Figure 1.6). As with ray tracing, the radiosity solution must be re-calculated in some manner whenever objects in the environment move. It is not the case, however, that a radiosity solution has to be re-computed when the view configuration changes. The radiance values are associated with points on the surfaces in the environment and being independent of view direction, need not be re-calculated when the view changes. This is one advantage radiosity has over ray tracing.

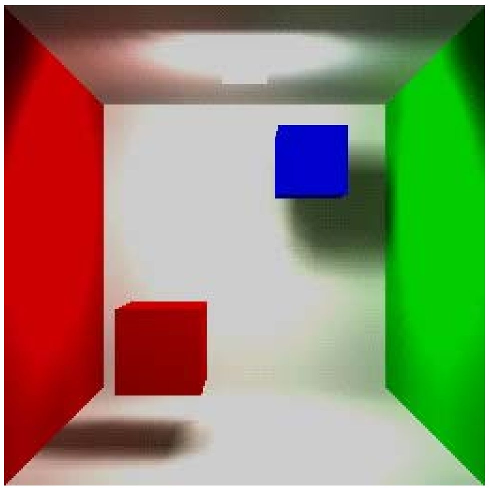


Figure 1.6: Simple environment rendered with a radiosity algorithm.

The *multi-pass methods* [284, 141, 59, 283, 255] of the nineties have attempted to remove the restrictions that are placed on the environment by the classical radiosity formulation. These methods allow both specular and diffuse surface reflectances by treating the *different modes of light transport* as separate calculations (Figure 1.7). The accuracy of these algorithms is unmatched but there is a penalty for this realism. They must do all the work involved in solving the indirect illumination problem, like a radiosity solver, as well as performing the work of the typical ray tracer when calculating direct and specular illumination effects. Since multi-pass algorithms capture

[1]Only view-independent diffuse reflection is modeled, radiance can be replaced with a simpler quantity radiosity (watt/meter^2)

view-dependent effects, their solutions must also be refined any time an object moves or the view changes.

Also included in this group of very expensive methods should be the *multi-point radiance transport algorithms* [11, 269, 62] which handle directionally varying illumination effects by coupling higher-order tuples of locations in the discrete linear system. Whereas a naive radiosity algorithm would run in $O(n^2)$ time, where n is the number of input patches, coupling pairs of points in space, multi-point transport algorithms intuitively run in $O(n^d)$ time where d is the number of points coupled in space.

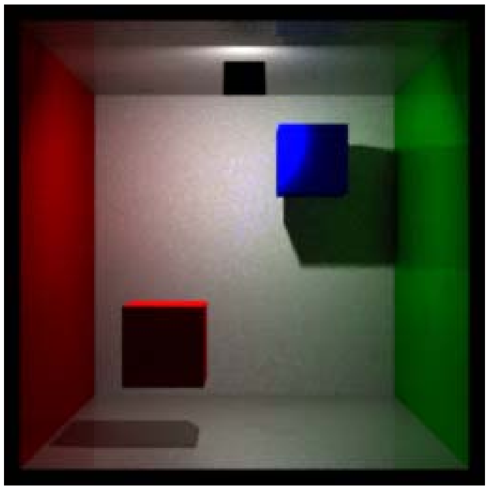


Figure 1.7: Simple environment rendered with a multi-pass rendering algorithm.

1.1.2 Quest for interactivity

With the advent of raster-based graphics technology, the earliest software rendering systems often combined the interaction and shading/rendering processes. Object motion or viewer movement would trigger the system to Gouraud/ Phong shade the scene using the new configuration of viewer, lights, and geometry. The speed of these algorithms, their scalability, soon dwindled as designers produced larger and more complicated environments to interact with.

One way this problem was alleviated, was the incorporation of the shading algorithms into reasonably priced hardware units [63, 5]. These systems provided the ability to rapidly perform the costly shading procedures, in some sense decoupling interaction and shading. Today, *interactive*

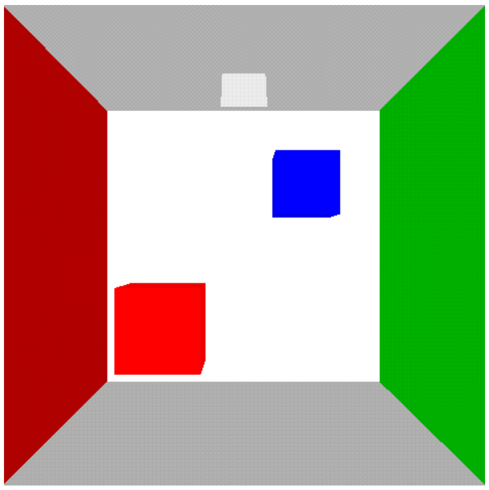


Figure 1.8: Simple environment rendered with a hardware shading algorithm.

hardware-based polygon rendering systems boast rates of up to ten million Gouraud shaded, textured polygons per second [4]. For smooth motion, a frame rate of 30 fps, these systems can render approximately 333,333 polygons per frame, far below the size of today's large environments. Relying on hardware speedups to achieve our interactivity goals is not the answer. There will always be larger and more complicated scenes to work with that “clog” the pipeline.

As mentioned in the previous section (Section 1.1.1), the lack of realism and control that led researchers towards physically accurate simulation methods is still a great limitation in hardware-based rendering (Figure 1.8). Physical units are still not incorporated directly into hardware algorithms that rely on a simple normalized trichromatic color space, RGB, in which to perform their computations. It is often the case that color space overflows affect the calculated solution yielding inaccurate results. Researchers are now focusing on how to more effectively represent realistic illumination effects using hardware rendering systems [84].

Although the hardware itself cannot handle units like radiosity, pre-computed radiosity solutions often form the basis for hardware-assisted walk-throughs in complex environments. The pre-computed radiosity values are converted into “reasonable” hardware color values and then used as vertex intensities by the hardware shading routines. The converted radiosity solution is used to replace the diffuse and/or ambient illumination components with the hardware shading algorithm approximating first-order specular highlights on-the-fly from the environment's geometric, lighting, and viewing configurations. Although effective results can be generated, the size of the environment is still directly coupled to the performance of the interaction mechanism.

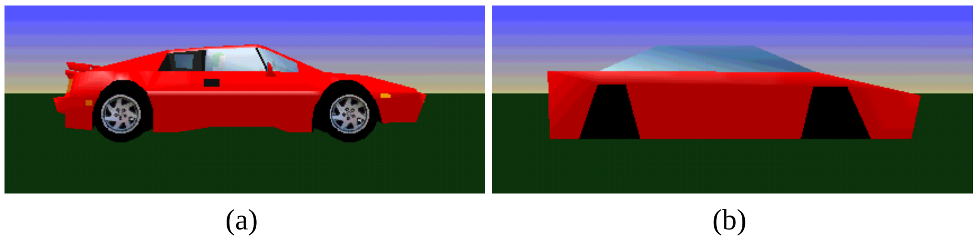


Figure 1.9: Level of detail processing: (a) very high detail, (b) low detail (MultiGen, Inc and Silicon Graphics, Inc).

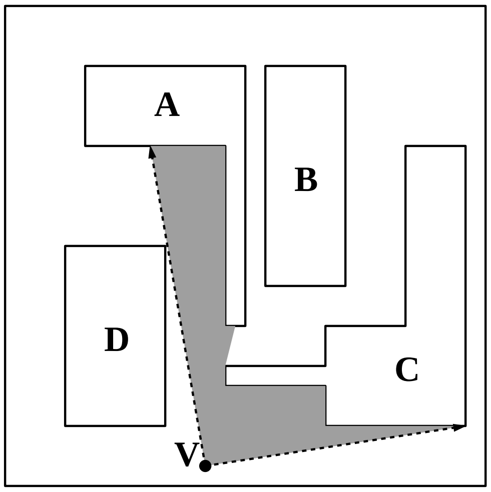


Figure 1.10: Visibility culling.

Recently, many researchers have proposed data manipulation techniques to alleviate the bottleneck that hardware systems have when dealing with extremely large scenes. To alleviate problems with visual fidelity and consistency researchers have proposed both static and dynamic scene modification algorithms in an attempt to keep the size of the environment below today's hardware processing threshold []. Techniques such as *texture mapping* [39] and *billboarding* [251] to replace scene geometry can be applied to the environment before rendering. *Level-of-detail processing* (Figure 1.9), which entails using different versions of geometry depending on the view configuration is performed during interaction with the environment. *Working set techniques*, which include *visibility culling* [304] to find potentially visible geometry (Figure 1.10), *spatial decomposition methods* [261], and *database pre-fetching* [105] also help in minimizing the amount of scene geometry that needs to be processed for each frame. All of these techniques are designed to allow a constant stream of frames to be presented to the user. A constant frame rate is one of the most important requirements for smooth interaction.

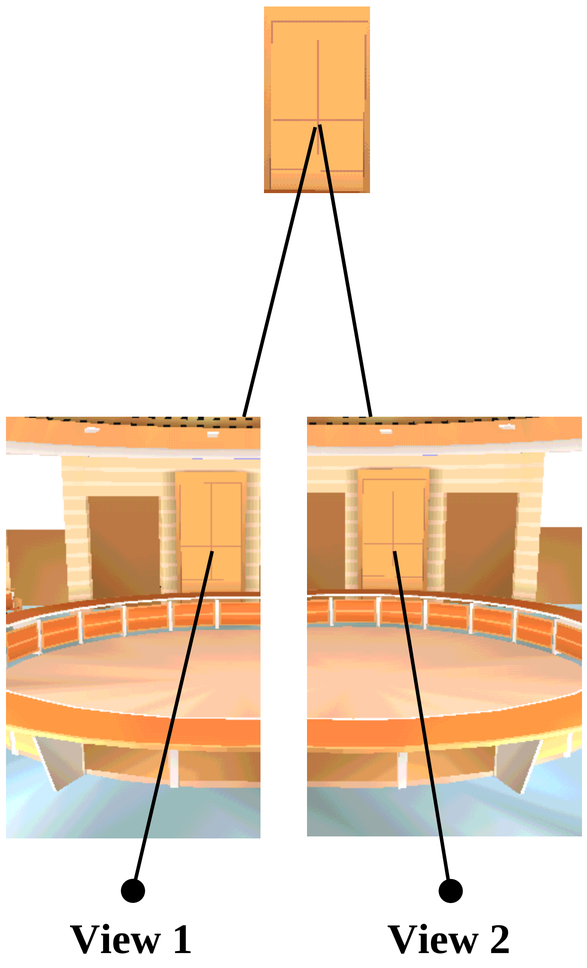


Figure 1.11: Image-based representation (Lightscape Technologies, Inc).

Researchers have also recently re-opened the avenues originally taken in flight and driving simulation design [34], and video-disc technology [179]. This work takes image mapping to its logical extreme in an attempt to decouple the complexity of the methods that generate the imagery from the methods designed to reconstruct views during a walk-through (Figure 1.11). These approaches compute global illumination into an intermediate data structure, usually a *basis*-like set of samples (views/images), that are, in turn, used to reconstruct the full *space* of desired views. Since the images are pre-computed, they can be created at any desired quality level, containing very accurate or approximated illumination, depending on the time afforded the simulation algorithm.

The consistent storage mechanism allows very complicated environments to be represented in a manner that is comparable in size to that needed for even the simplest of environments, creating a more constant relationship between environment complexity and the inherent level of interactivity. A difficulty with these methods is that sample-based structures like images are used in most of today's interactive systems as a modifier for scene geometry, a texture or bump map, not as a low-level primitive and are difficult to process efficiently. As the hardware is modified to include more operations on images, these problems will be alleviated making sample-based walk-through mechanisms prevalent.

Most recently work has been focused on *line-space/directional methods* for storing radiance information that travels around an environment [175, 121, 302]. With views of the environment being constructed as sets of radiance values flowing along slightly varying directions into the same view location, if an adequate coverage of line-space is achieved, a high level of interaction can be produced.

1.2 Thesis Claim

Today's global illumination algorithms keep getting more accurate and at the same time more expensive to execute. View-independent effects can be pre-calculated over an adequate spatial representation, but accurately modeling view-dependent effects requires that the illumination information either be pre-computed spatially *and* directionally, or be updated continually for different view positions. Faster graphics hardware and computers in general will ease some of the burden but there will always be larger and more complicated environments to offset the increased performance. Researching ways to overcome the inherent complexity in the simulation methods often yields incredible results, both in terms of accuracy and efficiency, but algorithmic breakthroughs cannot be relied on to occur consistently.

Instead, we propose to represent the problem of interacting with globally illuminated, complex, animated environments as one of sampling and reconstructing the time-varying radiance distribution [2] induced by the environment's geometric, lighting, and potential view configurations. This radiance distribution extends the analogy of the traditional computer graphics image, which can be viewed as the radiance coming from a small set of directions into a specific viewpoint at a specific time, to include the radiance seen at all points (x, y, z), along all incoming directions (θ, ϕ), at all times (t). The radiance distribution fully characterizes the time-varying visual information contained within an environment that has object and/or viewer motion. If we treat distribution as an arbitrarily complex function that we do not know much about, we are left with the task of trying to construct a set of samples that provides adequate information content so as to allow accurate reconstruction; the standard sampling/reconstruction problem.

Sample-based systems, with the pre-computed samples organized as images, have been used previously in flight simulators [34] and in computer graphics [60, 193, 58, 270]. While this research has demonstrated the potential of these types of systems to allow a user to tour a complex scene at interactive rates, the problem of efficiently rendering environments that can vary over time within the context of such systems has not yet been considered. Also, very little work has gone into verifying the often surmised premise that the radiance samples are best stored as sets of images as opposed to using a more general data structure.

Our method for approximating an environment's radiance distribution, originally presented in [217, 218] attempts to isolate spatio-temporal variances in the distribution without directly calculating illumination values. A set of scene analysis algorithms are defined that qualify and quantify changes in the radiance distribution as a function of changes in the environment configuration (e.g. object/viewer motion) over time. This classification is performed relative to a set of coherence metrics that are constructed to separate visibility, direct illumination, and indirect illumination effects. All the algorithms rely on a straightforward adaptive sampling/decomposition scheme and spatial and temporal discontinuities are treated separately. The outcome of this decomposition process is a set of radiance distribution samples, to be computed via a global illumination/rendering algorithm, organized as time-sequences of images at strategic locations within a chosen *view space* — an area of the environment in which free range of motion is desired.

How these images are produced is the focus of our multi-level spatio-temporal rendering method which uses a low resolution object space simulation/interpolation algorithm to track the variances in indirect illumination within the environment. This pre-process is used as input to a direct illumination algorithm which computes compute the complete images for the appropriate times. Once the time-sequences of images are complete, reconstruction of particular views can take place.

Reconstruction of a view of the environment from a particular location at a particular time is performed by first using image blending techniques to reconstruct the correct image from each time-sequence and then using view interpolation methods to merge neighboring views around the desired view position. The results presented show the feasibility and utility of the method but also demonstrate that an inordinately high temporal sampling rate is required to adequately reconstruct the direct illumination and visibility discontinuities generated by possible changes in the environment.

With this in mind we introduce our second radiance distribution approximation, originally presented in [215], which replaces the time-sequences of images at the strategic view space locations with a sparse volumetric representation for incoherent/randomized spatio-temporal radiance samples. Instead of sampling by decomposing time into a series of slices and computing a complete image of radiance samples for each slice, these temporal volumes store radiance samples that have been distributed randomly or adaptively over time and space — the image plane — and are stored in an unorganized manner as opposed to the typical $(row, column)$ image format. Reconstruction of an image of the environment at a particular time is achieved by using one of two possible methods. The first involves projecting the appropriate set of the radiance samples, chosen based on their proximity to the desired time, onto the desired temporal plane, triangulating the projected sample set, and then using the radiance values to blend intermediate radiances for the triangle interiors. The second involves building the $3D$ spatio-temporal radiance complex, (x, y, t) radiance samples connected into a nearest-neighbor mesh, and intersecting the complex with an axis-aligned (x, y) plane fixed at the desired time $(t^{'})$. Although the complex is more complicated to pre-compute, it allows more efficient reconstruction since the simplicization/triangulation does not have to be performed in the interaction phase. The results presented show that temporal image volumes allow for more accurate, believable, and efficient temporal/motion reconstruction requiring much less storage than the more traditional time-sequence approach. It should also be noted the complete image/temporal sequence is not needed as with a posteriori video compression techniques.

1.3 Thesis Overview and Contributions

This work [216] is different from previous efforts in global illumination and walk-through systems in that we want to pre-compute, into sample-based structures, as much information as possible about the spatio-temporal illumination variances in the environments we want to interact with. We show the feasibility of sample-based approaches to global illumination and walk-through of complex, animated environments by deriving a series of efficient radiance distribution approximations based on images, and sparse volumes. Our approximations lay the groundwork for future research into the use of data structures for pre-computed radiance values as an efficient and effective method for walking through accurately illuminated, complex, animated environments.

This thesis progresses chronologically through the previous work in global illumination, Chapter 2, and interaction/walk-through systems, Chapter 3.

Chapter 4 introduces our formalization for sample-based systems based on the full time-varying radiance distribution.

Chapter 5 introduces our new multi-stage rendering algorithm for efficiently computing global illumination within complex, animated environments.

Chapter 6 introduces our first radiance distribution approximation built around time-sequences of range-images, and discusses our adaptive coherence algorithms.

Chapter 7 discusses our second radiance distribution approximation that replaces the time-sequences of range-images with temporal image volumes.

We conclude the thesis in Chapter 8 by restating the goals of the work and how we accomplished them, and present ideas about the the future of this research.

The specific contributions of this thesis are:

1. A simple overview of radiometry, the modes of light transport, and simulation algorithms. This overview provides the information necessary to make informed choices when when constructing/choosing an efficient rendering method.

2. A chronological overview of past geometric and non/quasi-geometric walk-through methods. This overview provides a set of criteria, based on quality and speed of presentation, that can be used when evaluating the different methods for constructing walk-through mechanisms.

3. A framework for defining synthetic environments in terms of a single time-varying radiance distribution.

4. A multi-stage rendering method that separately computes indirect and direct illumination at different quality levels. Indirect irradiance is shared via ambient files that are constructed using wavelet radiosity, jittered resampling, and a mesh-based irradiance gradient adaptation.

5. A sampling and reconstruction methodology for deriving a series of approximations for the radiance distribution of an environment.

6. An approximation of the radiance distribution based on time-sequences of range-images stored at important locations within a user defined view space.

7. A set of coherence measures for finding spatio-temporal illumination and visibility variances that do not require illumination information to be computed. These measures isolate the causes of illumination variances as opposed to the variances themselves.

8. A sparse spatio-temporal sampling and reconstruction method, based on the Delaunay complex in either $2D$ or $3D$, that allows for the replacement of time-sequences of images with sparse temporal image volumes.

Chapter 2

Global Illumination

“Let there be light.”

God – Genesis i. 3.

Whether computer generated or not, an environment’s visual complexity is related to the interaction of its structure and the light filling its space. The problem of the propagation of light, which is essential to the accurate depiction of spatial relations, is “solved” in real-time in nature. Unfortunately, when dealing with computers, we are not so lucky. Simulating the propagation of light around an environment is an expensive process. An understanding of global illumination theory and its simulation algorithms is necessary if we are to understand why walk-through systems for our realistic environments are not yet commonplace. It also helps us judge the tradeoffs when choosing a rendering method and environment representation scheme.

2.1 Radiometry

Radiometry [70, 285, 116] is concerned with the measurement of radiant energy and its transfer[1]. An understanding of radiometry theory allows for the construction of an energy balance equation for all the surface points in an environment. In an effort to insure accuracy while retaining a reasonable level of efficiency, global illumination implementations approximate the solution by using simplifications of this general equation. In order to motivate this discussion we first start with a few definitions.

[1]Photometry provides an equivalent set of perceptual quantities.

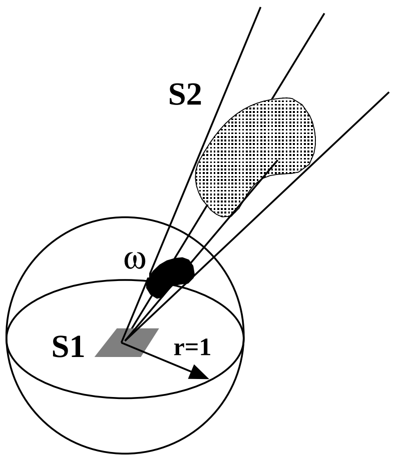


Figure 2.1: Solid angle ω of a surface $S2$ as seen from $S1$.

Radiant Energy. Radiant energy Q, measured in joules, is a fundamental quantity in radiative transfer.

Radiant Flux (Power). Radiant flux (power) Φ, measured in watts, is radiant energy per unit time,

$$d\Phi = \frac{dQ}{dt} \quad .$$

Solid Angle. Solid angle (Figure 2.1) is the $3D$ analog of the $2D$ central angle and is measured in *steradians* (sr). The solid angle ω of a surface $S2$, as seen from another surface $S1$, is the area on the unit sphere surrounding $S1$ covered by the projection of $S2$. Since the area of a unit sphere is 4π, the solid angle of the hemisphere Ω above $S1$ is 2π sr.

The solid angle of a very small (differential) surface, with area dA, as seen by a reference surface is often approximated by the area of the small surface projected onto a plane perpendicular to the line between the two surfaces divided by the distance between the two surfaces,

$$d\omega = \frac{dA\cos\theta}{r^2} \quad . \tag{2.1}$$

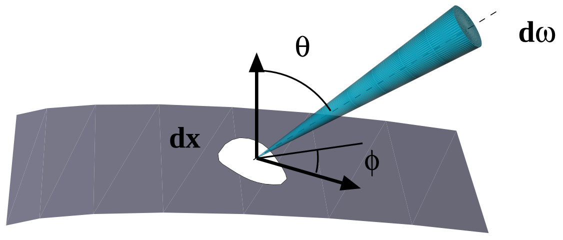


Figure 2.2: Definition of radiance.

Radiance. Radiance $L(x, \theta, \phi)$ (Figure 2.2) at a point traveling in the direction of a ray is radiant power per unit projected area[2] per unit solid angle in the direction of the ray,

$$d^2\Phi = L(x, \theta, \phi) \underbrace{dx \cos\theta}_{\text{projected area}} d\omega \ .$$

Radiant Flux Density (Radiosity/Irradiance). Radiant flux density at a point, either leaving from (radiosity, $B(x)$) or impinging on a surface (irradiance, $E(x)$), is the total power per unit area on the surface,

$$B(x) = \frac{d\Phi}{dx} \ .$$

Defining total power by integrating radiance over the sphere of possible infinitesimal solid angles,

$$\begin{aligned} d\Phi &= \int_\Omega d^2\Phi \\ &= dx \int_\Omega L(x, \theta, \phi) \cos\theta d\omega \ , \end{aligned}$$

yields

$$B(x) = \int_\Omega L(x, \theta, \phi) \cos\theta d\omega \ .$$

Bidirectional Reflectance. The bidirectional reflectance distribution function (BRDF, Figure 2.3) describes the directional distribution of reflected light and is defined as the ratio of radiance in the outgoing direction to the radiant flux density (irradiance) in the incoming direction,

$$\rho_{bd}(x, \theta_o, \phi_o, \theta_i, \phi_i) = \frac{L(x, \theta_o, \phi_o)}{L(x, \theta_i, \phi_i) \cos\theta_i d\omega} \ . \tag{2.2}$$

[2]This measurement is taken perpendicular to the ray, thus inducing the $\cos\theta$ term

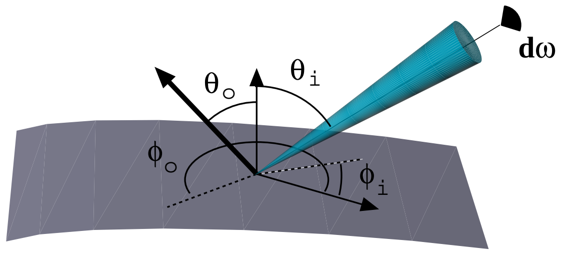


Figure 2.3: Definition of BRDF.

Note that the bidirectional reflectance ρ_{bd} has units of $1/\text{sr}$ which can be unbounded. The BRDF is often replaced with a more intuitive term, *reflectance* ρ, which is the fraction of incident radiant flux density that is reflected by the surface. Reflectance, or more formally *directional-hemispherical reflectance*, is a dimensionless quantity between 0 and 1.

Energy Balance Equation. Energy equilibrium is achieved by coupling the incident energy at a surface with the reflected energy at the surface boundary. The energy balance integral equation,

$$L(x,\theta_o,\phi_o) = L_e(x,\theta_o,\phi_o) + \int_\Omega \rho_{bd}(x,\theta_o,\phi_o,\theta,\phi) L_i(x,\theta,\phi)\cos\theta d\omega \ , \tag{2.3}$$

specifies the outgoing energy as the sum of an emitted component ($L_e(x,\theta_o,\phi_o)$) and a reflected component ($\int_\Omega \rho_{bd}(x,\theta_o,\phi_o,\theta,\phi)L_i(x,\theta,\phi)\cos\theta d\omega$).

There is an intuitive explanation and formal solution to this integral equation [285]. Start by defining a reflection operator (R) that represents the effect that a single reflection has on a given radiance distribution:

$$(RL)(x,\theta_o,\phi_o) = \int_\Omega \rho_{bd}(x,\theta_o,\phi_o,\theta,\phi) L_i(x,\theta,\phi)\cos\theta d\omega \ ,$$

which reduces Equation 2.3 to

$$L = L_e + RL \ ,$$

which can then be inverted,

$$L = [I - R]^{-1} L_e \tag{2.4}$$

$$= \sum_{n=0}^{\infty} (R)^n L_e \ , \tag{2.5}$$

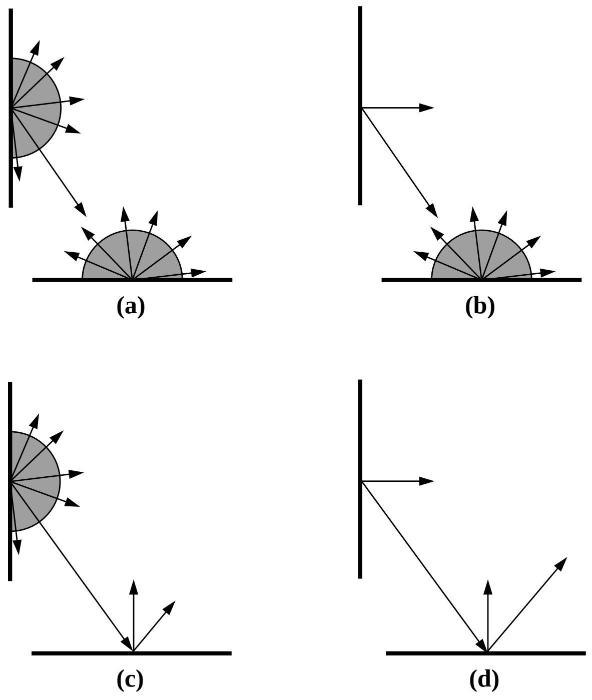


Figure 2.4: Modes of light transport: (a) diffuse-diffuse, (b) specular-diffuse, (c) diffuse-specular, (d) specular-specular.

yielding a Neumann series solution. Each application of the reflection operator adds the information from another bounce of light to the solution. The infinite series converges to the exact solution.

Even with the existence of a formal solution, integral equations like this are difficult to solve analytically and instead are approximated using a simplified form and numerical simulation methods. The simplifications usually involve a choice as to which visual effects are important and need to be calculated accurately and which can be approximated with low-order methods.

2.2 Modes of Light Transport

The energy balance equation just presented intrinsically handles diffuse/matte and specular/shiny surfaces, as well as reflection and refraction, in a unified manner by considering subsets of incident directions. The different transport modes [317, 141] such as *diffuse* versus *specular* (Figure 2.4) and *direct* versus *indirect* (Figure 2.5), generate a widely varying set of effects, measured both on an object's surface or in the final generated image, view-independent versus view-dependent.

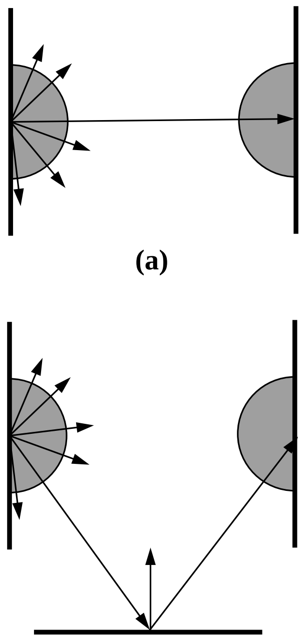


Figure 2.5: Modes of light transport: (a) direct, (b) indirect.

Depending on how important the desired effect is, methods specifically designed to accurately calculate it are often used. This leads to a widely divergent set of tools to choose from when deciding how to simulate the illumination that would be present in a computer environment. *View-dependent* algorithms like traditional ray tracing [335] which incorporate information isolating those directions that lead back to the viewer are good for handling specular highlights, multi-bounce reflections, and refractions. *View-independent* algorithms such as traditional radiosity [119, 221] are good for handling non-directional effects such as diffuse interreflections. Algorithms can also be classified by how they choose to initially sample and compute illumination within the environment. *Image space* algorithms sample the image plane in an attempt to compute only that illumination that affects the final image. *Object space* algorithms compute illumination over all the points on all the surfaces in the environment. *Hybrid object/image space* algorithms solve for the different modes of light transport using a combination of the two techniques, noting that each is good for handling different effects.

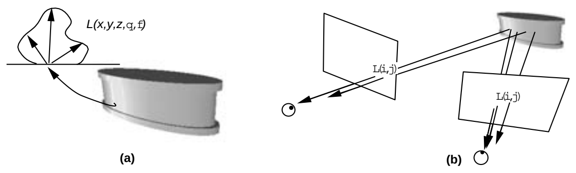


Figure 2.6: Simulation methods: (a) object space methods, radiance L is computed for each point and in each direction for each object, (b) image space and hybrid methods, radiance L is computed only for the objects which are visible in the image, and only for the points and directions which are visible.

2.3 Simulation Methods

As just mentioned, we can classify global illumination simulation algorithms based on the space in which they choose to initially sample the environment: image space, object space, and hybrid object/image space methods. Each has their pros and cons exhibiting some form of accuracy/efficiency tradeoff based on a simplification of Equation 2.3.

2.3.1 Image space

Image space methods (classical [335] and distribution ray tracing [74], Monte Carlo path tracing [155, 308, 309]), and photon tracing [231] compute the radiance $L(i,j)$ which will be seen through a pixel at location (i,j) in the image (Figure 2.6b). These methods have the advantage that if an object in the environment does not appear in a particular image, its radiance does not have to be computed. This is an advantage for environments which have many more objects than the image representing it has pixels. Image space methods have the disadvantage that they do not exploit spatial coherence – each pixel is computed independently.

Classical ray tracing

Classical ray tracing [335] (Figure 2.7) was the first algorithm to incorporate object-object illumination effects by reversing the formulation of the original computer graphics ray casting problem [6] and solving it in a recursive manner. By solving the ray casting problem not only in the camera coordinate system but also in coordinate systems transformed/oriented by using the laws for perfect specular reflection and refraction (Figure 2.8), Whitted could use the rays as a convenient bookkeeping mechanism for handling the local diffuse and specular contributions made

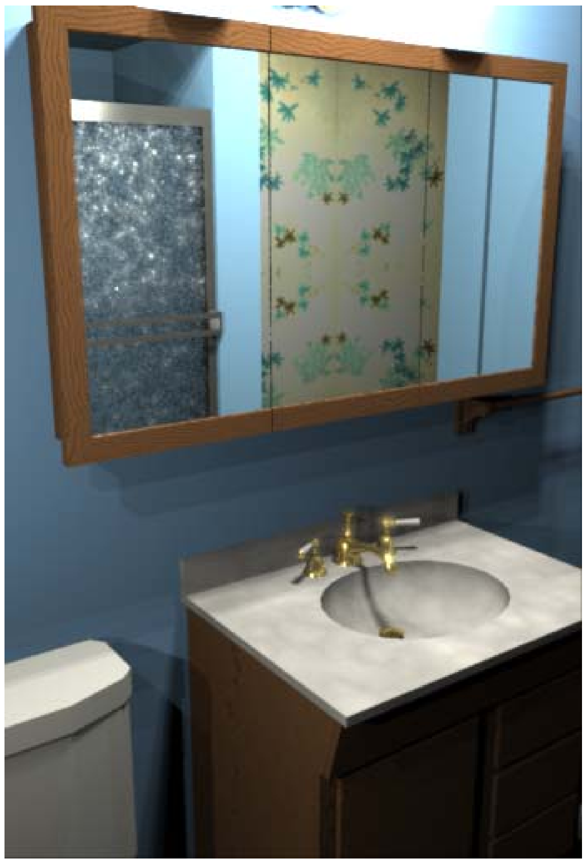


Figure 2.7: Greg Ward's bathroom displaying sharp shadows, and reflections (Whitted-style ray tracing).

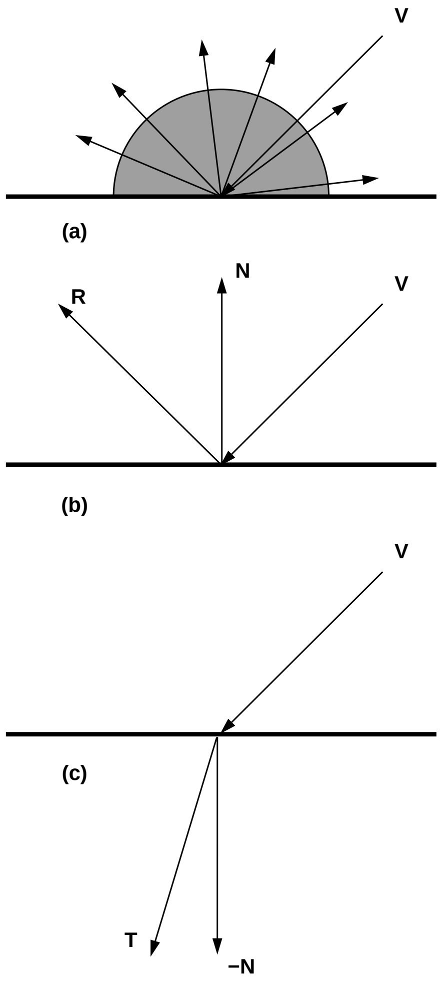


Figure 2.8: Sampling directions in classical ray tracing: (a) sampling light source directions over hemisphere, (b) perfect specular reflection (Hero's law), (c) perfect specular transmission (Snell's law).

by point light sources and the global specular contributions made between objects in the environment. He essentially solved the energy balance equation (Equation 2.3) by replacing the domain of integration with a weighted summation over all light source directions (n) and perfect specular reflection and transmission directions.

$$\begin{aligned} L(x,\theta_o,\phi_o) &= \left[\sum_{j=1}^{n} \rho_{bd}(x,\theta_o,\phi_o,\theta_j,\phi_j) L_i(x,\theta_j,\phi_j)\right] + \\ &\quad \rho_{bd}(x,\theta_o,\phi_o,\theta_R,\phi_R) L_i(x,\theta_R,\phi_R) + \\ &\quad \rho_{bd}(x,\theta_o,\phi_o,\theta_T,\phi_T) L_i(x,\theta_T,\phi_T) \ , \end{aligned} \tag{2.6}$$

where the R and T directions are the perfect specular reflection and transmission/refraction directions based on (θ_o, ϕ_o) and the normal $\mathbf{N}$ to the visible object at the point x. Whitted's work, although not physically based, was the first quantum leap in computer-generated imagery. His pictures included visual effects that up until that point, had been missing in computer graphics. This was not an incremental improvement but a new way of thinking about the problem of rendering.

Distribution ray tracing

Distribution ray tracing [74] statistically estimates the integral of Equation 2.3 using a prohibitively large number of rays (Figure 2.9).

$$\begin{aligned} L(x,\theta_o,\phi_o) &= \left[\sum_{j=1}^{n} \sum_{k=1}^{s} \rho_{bd}(x,\theta_o,\phi_o,\theta_{j_k},\phi_{j_k}) L_i(x,\theta_{j_k},\phi_{j_k})\right] + \\ &\quad \left[\sum_{k=1}^{s} \rho_{bd}(x,\theta_o,\phi_o,\theta_{R_k},\phi_{R_k}) L_i(x,\theta_{R_k},\phi_{R_k})\right] + \\ &\quad \left[\sum_{k=1}^{s} \rho_{bd}(x,\theta_o,\phi_o,\theta_{T_k},\phi_{T_k}) L_i(x,\theta_{T_k},\phi_{T_k})\right] \ , \end{aligned} \tag{2.7}$$

where the subscript j_k denotes the kth subsample of the jth light source, and R_k and T_k denote the kth subsamples around the perfect reflection and transmission directions. The radiance of a point is calculated by averaging the radiances from sampled directions on the incoming sphere in a recursive manner. Different sampling methods are used when sampling indirect illumination, using the BRDF to guide the sampling, and direct illumination, using the light source locations to guide the sampling. With distribution ray tracing we are not limited to sampling spatial dimensions. The original formulation includes a methodology for distributing rays over the lens area of the

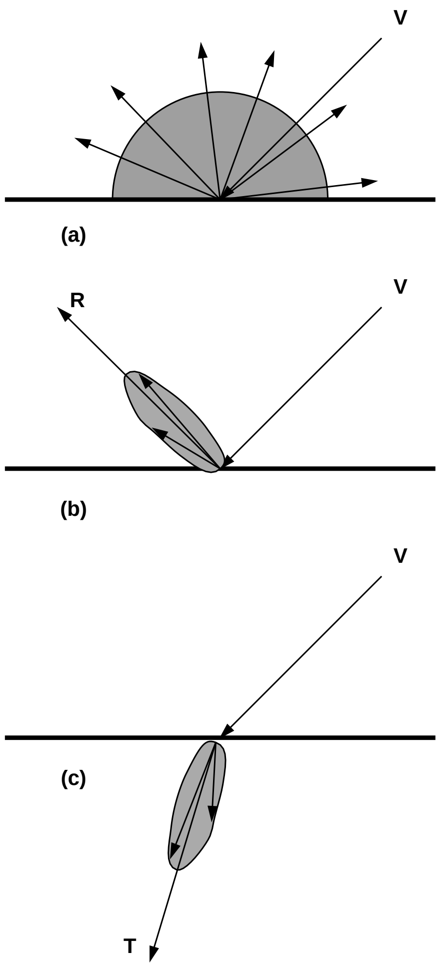


Figure 2.9: Sampling functions in distribution ray tracing: (a) sampling indirect illumination, (b) sampling specular reflection, (c) sampling specular transmission.

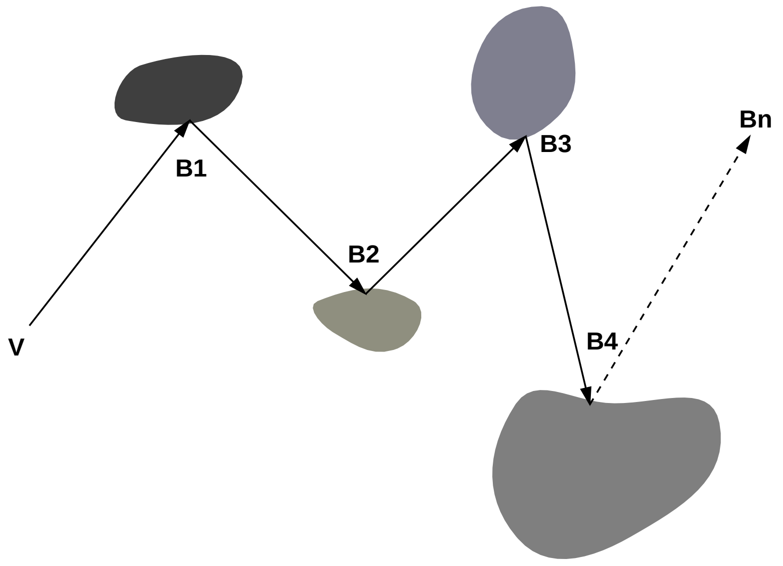


Figure 2.10: Gathering radiance along a path.

synthetic camera, the color spectrum being sampled, and the time that the shutter on the camera is open. With these extensions, effects such as depth-of-field, color anti-aliasing, and motion blur can be handled.

Monte Carlo path tracing

Monte Carlo path tracing [155] uses a truncated Neumann series (Equation 2.5) expansion to solve the global illumination problem. This expansion takes place in the form of histories that are kept for individual particles interacting with the environment until they are absorbed (Figure 2.10). At each intersection, a new direction for each particle is chosen as opposed to spawning new rays like the other ray casting approaches. Monte Carlo estimation techniques are used to solve the resulting integrals sending more particles into the environment until a certain statistical confidence in the solution is achieved. Importance-based and stratified sampling can be used to reduce the variance in the solution as pure Monte Carlo techniques produce very noisy solutions. Recently, advances in Monte Carlo estimation techniques [308, 309] have made this method more accurate and efficient by working in a bidirectional manner. The crux of this new approach is being able to combine these estimators in an optimal way so as to not bias the results.

Photon tracing

Photon tracing [231] is the approach most like the original ray casting approach [6]. Photons are generated at the light sources and sent into the environment where they interact with surfaces. Each photon that hits a surface has a chance of being absorbed or reflected, depending on the reflectance of the surface, with each photon depositing some of its energy onto the surface at the appropriate location. This process continues until the particle is absorbed. Usually some form of surface discretization is performed a priori and the energy deposited on the surface by the photons is stored at the vertices of the discretized mesh. Statistical measures are used to drive the initial emittance of photons and their reflection patterns off surfaces. Once enough energy has been deposited on the discretized surfaces, traditional ray casting, or hardware shading can be used to construct the final images.

The performance of image space algorithms is directly related to the number of samples required on the image plane. Even if we assume one sample per pixel, each sample ray must be intersected with the environment to find the visible surface. For the visible surface, illumination calculations must be performed. The process is not interactive. Most images take between several minutes, for small environments, and many hours, for very complicated environments. If advanced anti-aliasing techniques are added to remove error in the final image, the problem becomes even worse.

2.3.2 Object space

Object space methods, such as the radiosity method, compute radiance distributions $L(x, \theta, \phi)$ for objects (Figure 2.6a) without reference to the images in which the object will appear. The object space approach has the advantage that no interreflection calculation has to be performed as images are finally computed. It has the disadvantage that many radiances have to be computed which never appear in any images.

Classical radiosity

The classical radiosity method [119, 221] (Figure 2.11) makes certain assumptions about the environment in order to simplify Equation 2.3. The assumptions are:

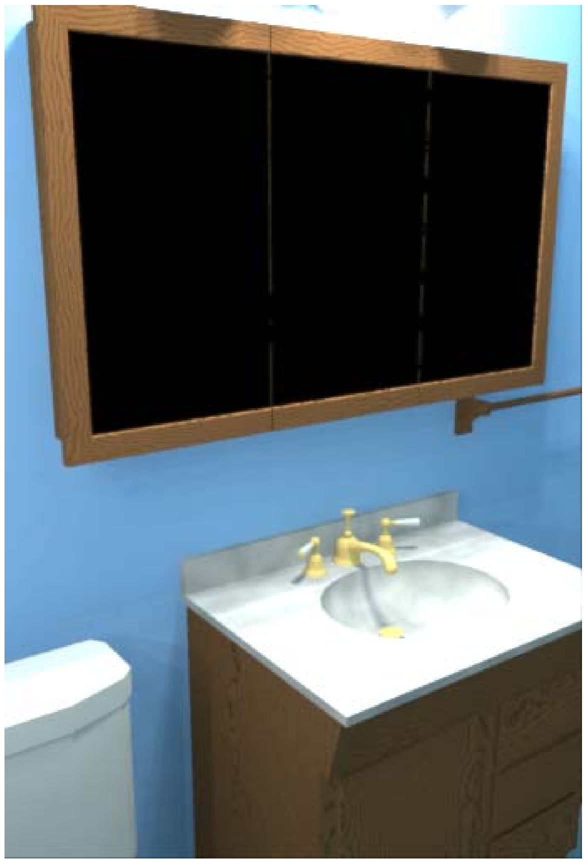


Figure 2.11: Greg Ward's bathroom displaying diffuse interreflection (Goral-style radiosity).

1. All surfaces are diced into small planar patches.

2. The patches are ideal diffuse reflectors/emitters.

These assumptions simplify the energy balance equation yielding the classical radiosity equation,

$$B(x) = B_e(x) + \rho_d \int_{y \in S} B(y) \frac{\cos\theta \cos\theta'}{\pi r^2} V(x,y)dy \quad , \tag{2.8}$$

where $V(x, y)$ is a binary visibility function that tests if x and y are visible to each other (occlusion limits coupling).

Instead of solving this integral equation directly, radiosity algorithms impose one more restriction that allows for the discretizing/simplifying of the equation into a linear system that can be solved with an iterative method. Restricting each patch to have a constant radiosity, for the purpose of reflection only, transforms Equation 2.8 into

$$B_i = B_{e_i} + \rho_i \sum_{j=1}^{N} B_j \frac{1}{A_i} \int_{x \in P_i} \int_{y \in P_j} \frac{\cos\theta \cos\theta'}{\pi r^2} V(x,y)dydx \quad .$$

This equation has a more concise form,

$$B_i = B_{e_i} + \rho_i \sum_{j=1}^{N} F_{ij} B_j \quad , \tag{2.9}$$

where

$$F_{ij} = \frac{1}{A_i} \int_{x \in P_i} \int_{y \in P_j} \frac{\cos\theta \cos\theta'}{\pi r^2} V(x,y)dydx \quad , \tag{2.10}$$

is called the *form-factor* and relates the proportion of total power that is transferred from patch i to patch j.

If we enumerate over i in Equation 2.9 we are left with a system of equations that can be represented in matrix form,

$$\begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{pmatrix} = \begin{pmatrix} B_{e_1} \\ B_{e_2} \\ \vdots \\ B_{e_n} \end{pmatrix} + \begin{pmatrix} \rho_1 F_{11} & \rho_1 F_{12} & \cdots & \rho_1 F_{1n} \\ \rho_2 F_{21} & \rho_2 F_{22} & \cdots & \rho_2 F_{2n} \\ & \vdots & \vdots & \\ \rho_n F_{n1} & \rho_n F_{n2} & \cdots & \rho_n F_{nn} \end{pmatrix} \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{pmatrix} \quad ,$$

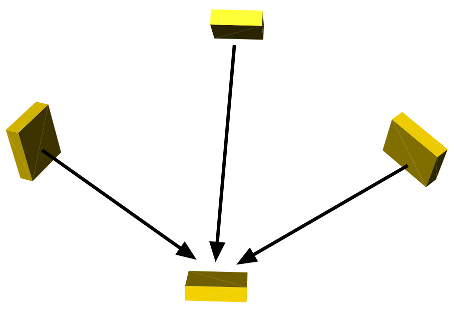


Figure 2.12: Gathering energy in a radiosity formulation.

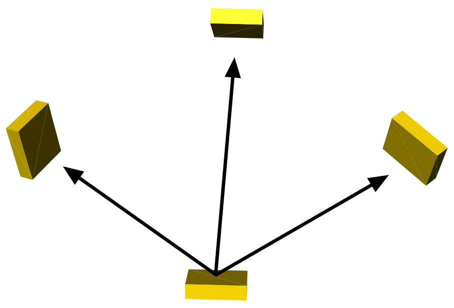


Figure 2.13: Shooting energy in a progressive refinement formulation.

or even more concisely as

$$\begin{pmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \cdots & -\rho_1 F_{1n} \\ -\rho_2 F_{21} & 1-\rho_2 F_{22} & \cdots & -\rho_2 F_{1n} \\ \vdots & \vdots & & \\ -\rho_n F_{n1} & -\rho_n F_{n2} & \cdots & 1-\rho_n F_{nn} \end{pmatrix} \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{pmatrix} = \begin{pmatrix} B_{e_1} \\ B_{e_2} \\ \vdots \\ B_{e_n} \end{pmatrix} ,$$

$$MB = B_e \ . \tag{2.11}$$

Jacobi or Gauss-Seidel relaxation are iterative techniques, $O(N^3)$ time-complexity, that are used to solve this type of linear system. They solve for the radiosity of each patch i in turn and are analogous to a “gathering” of the energy from all other patches in the environment (Figure 2.12).

Progressive refinement

The progressive refinement approach of Cohen et al. [67] modifies the order in which the patch radiosities are calculated based on the observation that many energy transfers in an environment are insignificant. They propose the idea of “shooting” energy into the environment from the patch

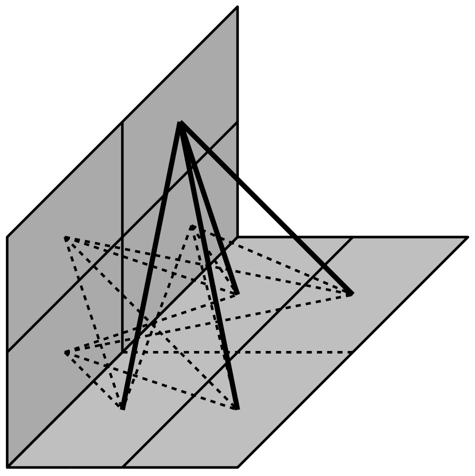


Figure 2.14: Linking patches in a hierarchical radiosity formulation.

that, at that point in the simulation, has the most energy to send out (Figure 2.13). Progressive radiosity has been shown to converge and be equivalent to the iterative technique called Southwell relaxation [120].

There are a few problems with the progressive refinement approach that should also be mentioned. Although in practice very few shootings are necessary to distribute most of the energy, for convergence, $O(N^3)$ operations are still needed. It is also the case that the choice for the current patch is based on a global notion of energy, often leaving a significant number of locally important transfers to fall by the wayside.

Hierarchical radiosity

The first attempt at addressing the asymptotic complexity of the radiosity formulation was made by Hanrahan et al. [138, 139] who equate it to the n-body problem in physics. The similarity between the two problems comes from the $1/distance^2$ falloff in the interactions between objects, and the ability to treat clusters of objects as a single "mega-object" when they are beyond a certain distance threshold. The solution techniques are asymptotically more efficient while maintaining guaranteed error bounds. The algorithm constructs a hierarchical decomposition of the individual surfaces in the scene (Figure 2.14). It then traverses the hierarchy deciding if pairs of patches should be linked using a decision procedure, an *oracle*, to estimate the form-factors between the patches.

This process proceeds recursively until the linking of all the surfaces at an appropriate level of

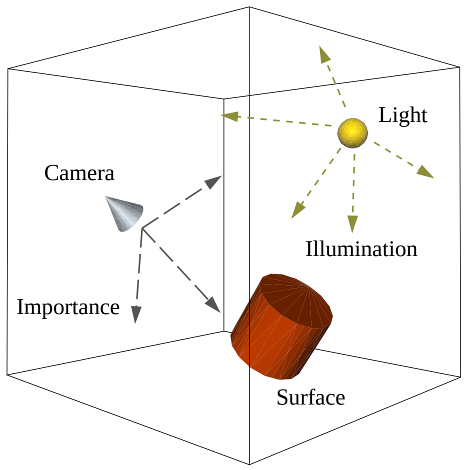


Figure 2.15: Illumination and importance traveling through a scene.

interaction is created. An area error criteria is kept to make sure that the recursion is not "infinite." The key to the efficiency of this algorithm is the efficiency of the oracle which must be trivial to calculate compared to calculating the true form-factors between the patches. The authors use a counting argument to show an asymptotic complexity of $O(N)$ interactions per iteration instead of the traditional $O(N^2)$.

Importance-driven radiosity

An interesting application of duality extends hierarchical radiosity to *importance-driven* radiosity [289]. Importance is a quantity that can propagate through the environment, like radiosity, and loosely measures the effect that a surface has to the set of views of the environment (i.e. a camera position or path, Figure 2.15). Those surfaces that are visible from the most views have a high initial importance with the amount of importance decreasing for those surfaces that are seen less often. Instead of solving the single system of equations for radiosity propagation, a dual system with importance and radiosity propagation is solved simultaneously. Importance-driven radiosity can be thought of as *view-dependent* radiosity and yields significant quality increase over traditional methods for the same cost.

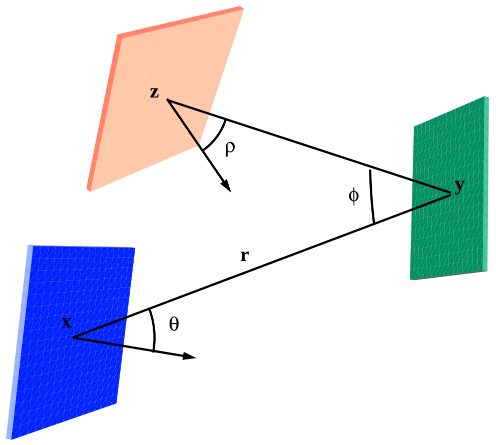


Figure 2.16: Three-point transport for the calculation of radiance.

Wavelet radiosity and radiance methods

Recent advances in object space methods for global illumination are yielding outstanding results. Wavelet bases [122, 266] have been applied to the hierarchical formulation as a means of compactly representing the radiosity function across a surface at the different levels of detail required for the individual interactions. Since wavelets encode smooth information at the base of the hierarchy and encode detail information in the subsequent levels, they mesh nicely with the hierarchical radiosity criteria.

The radiosity restriction on diffuse emission/reflection has also recently been removed through the introduction of *three-point transport* [11, 12] techniques which couple triples of surfaces instead of pairs. The energy transferred from surface point $\mathbf{x}$ to $\mathbf{z}$ goes through the intermediate surface point $\mathbf{y}$ (Figure 2.16).

The performance of object space methods is directly proportional to the number of interactions that must be computed. Even with hierarchical methods which keep the number of interactions down, computing a full radiance solution for a large environment can take hours or days. This is no where near the performance needed to both compute the illumination and interact with an environment at the same time.

2.3.3 Hybrid object/image space

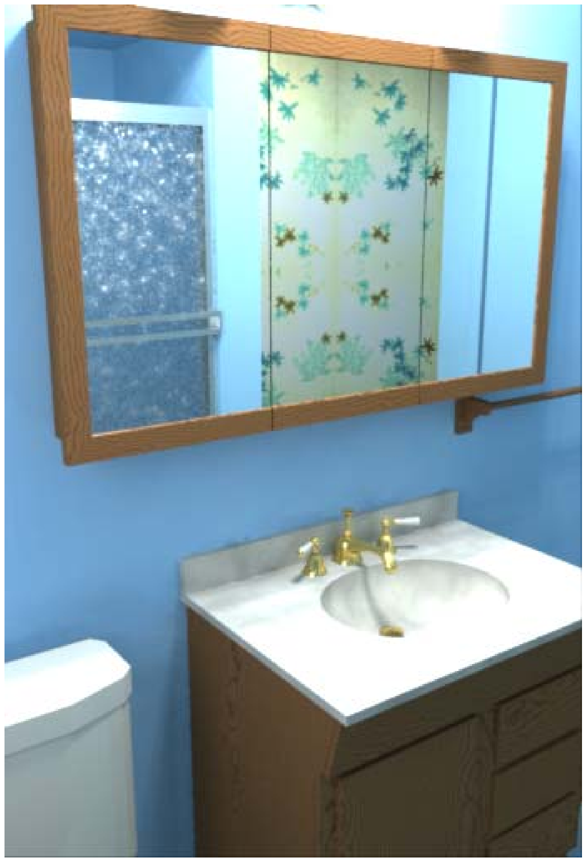


Figure 2.17: Greg Ward's bathroom displaying soft shadows, interreflections, and anisotropic reflection (multi-pass global illumination).

Hybrid Object/Image Space methods for global illumination combine the advantages of object and image space approaches. Detailed radiance is only computed for objects which appear in the image. Spatial coherence is exploited by calculating multiple reflections in object space.

Specifically, in hybrid methods, visibility and direct illumination calculations, for which the level of detail is limited by the pixel resolution, are computed in image space. Indirect illumination is computed in object space. In general, illumination is a continuous field, with sharp discontinuities occurring only when point light sources are instantaneously obscured [7]. Since indirect illumination is the result of multiple reflections, giving rise to many extended "secondary" light sources, indirect illumination is generally a smoother function than direct illumination. As a result, indirect illumination can be sampled relatively sparsely in space, and intermediate values found by interpolation.

Deterministic and stochastic ray tracing

The *Radiance* system [324] is a hybrid global illumination simulation method built using deterministic and stochastic ray tracing. *Radiance* uses a functional interface to allow general reflectances and patterns for objects and has withstood rigorous testing of its solutions. *Radiance* converges quickly to a solution by isolating those sources of variance in the solution and sampling smartly. It solves the indirect illumination problem only as needed and "caches" the information for interpolation at inbetween locations.

Multi-pass methods

Multi-pass progressive refinement methods [141, 59, 255] (Figure 2.17) calculate only those energy transfers that will have a significant effect on the final image. Integrating diffuse and non-diffuse transfers into a single algorithm appears difficult but if we are willing to traverse the environment following the propagation of light both from the light sources as with radiosity and from the eye as with ray tracing we can "meet in the middle" for a solution. Multi-pass methods usually perform a view-independent radiosity calculation and then perform a ray tracing post-process effectively simulating a BRDF for each surface that is a linear combination of a ideal diffuse component plus a specular component. Unfortunately, BRDF's for real surfaces are not adequately described as an ideal diffuse component plus a specular component. Specular reflections should be integrated into the object space energy transfer simulation instead of added as a post-process [284].

Multi-pass extensions of the radiosity method have been developed to account for the full range of BRDF's which occur in real life. In a method developed by Sillion et al. [283] a directional radiance function $L(\theta, \phi)$ is computed for sample points (x, y, z) on surfaces in the environment, rather than simply a radiance value L. Such a method however substantially increases the precomputation time and storage requirements for a complex scene.

The performance of hybrid methods is lower than either of the methods discussed earlier. This performance decrease is directly related to the fact that most hybrid methods perform all the work of both the image and object space methods, computing both indirect and direct illumination.

2.3.4 Summary

An environment's visual richness is directly related to the interaction of its structure and the illumination conditions. In computer graphics, we must simulate this process using global illumination algorithms designed to compute approximate solutions to the energy balance equation (Equation 2.3). Creating useful global illumination simulations requires a knowledge of radiometry and, more specifically, a knowledge of which visual effects need to be present in the images we will be generating. Image space algorithms sample the image plane in attempt to compute only that illumination that effects the final image and are good at capturing specular effects. Object space algorithms compute illumination over all the points on all the surfaces in the environment and are useful for calculating interreflections. Hybrid object/image space algorithms often use multiple passes combining the best of both algorithms to capture the most visual effects. It should be noted again that in general global illumination algorithms are not quick. The methods that have the highest accuracy do not run at interactive rates. Any walk-through system, in fact, that desires interaction with globally illuminated environments is going to require a separation of the illumination computation from the walk-through mechanism. This is a challenging task.

Chapter 3

Walk-Through Systems

“Anywhere is walking distance, if you’ve got the time.”

Steven Wright

While the simulation algorithms of the previous chapter are concerned with the physical accuracy and visual realism of the solution, for our particular application we have to be equally concerned with the ability to walk-through, or interact with, the environments we are creating and illuminating. The effects are not worth simulating if you cannot freely observe them.

The quality of a walk-through system is based on being able to quickly and accurately compute the content of a series of images that are going to be presented to the user. The imagery produced must meet certain resolution and field-of-view requirements and must be computed quickly enough as to allow a consistent rate of presentation. There should be no noticeable gaps or lags in the set of frames as the walk-through is taking place. As mentioned in the previous chapter, this is a difficult if not impossible goal to attain. Using the highest quality images that can be produced by today’s global illumination algorithms is not feasible since they take too long to compute. There are, however, ways to mitigate this problem. We can lower our goals slightly and work towards creating *effective* walk-through systems; those systems that meet the minimum criteria for quality and speed while still presenting the effects the user *needs* to see.

There are two basic approaches to constructing walk-through systems, based on the type of low-level drawing primitive that is used. Geometric methods work with versions of the environment's geometry, often using polygons as the drawing primitive, and either pre-computing illumination as with radiosity walk-throughs, or computing illumination on the fly using advanced hardware rendering. Quasi/non-geometric methods replace the geometric drawing primitive with sample-based structures, like images, that contain pre-computed illumination/radiance information. Pre-computed animations, animation networks [179], and view interpolation [60, 58, 193, 270] are examples of sample/image-based walk-through mechanisms. Recent advances such as the lumigraph [121] and light field rendering [175] are examples of abstract radiance structures useful as the basis for advanced walk-through systems.

3.1 Geometric Approaches

Geometric walk-through systems are constructed leaving the environment intact and using a simple geometric structure like the polygon as the low-level drawing primitive. These systems are often based on either hardware rendering algorithms which compute object illumination on-the-fly, or based on radiosity pre-processing with the hardware shading algorithm being used for fast display of the illumination information. Geometric walk-through systems usually offer the highest levels of interactivity, but lack the photo-realism desired in most walk-through applications. The biggest bottleneck with these types of systems is that their performance is tied directly to the complexity of the environment being used.

3.1.1 Hardware rendering

The first approach, which has been growing in popularity since the mid-eighties is to rely on graphics hardware to perform all the illumination and display functions. Almost all computer workstation manufacturers today offer graphics hardware [63] that can represent and rapidly shade environments polygonal objects illuminated using local, empirical lighting models.

With hardware rendering (Figure 3.1), the illumination of a surface is calculated on-the-fly but depends solely on its own characteristics and that of non-physical light sources such as points. No interaction of the surfaces takes place which simplifies the energy balance equation (Equation 2.3)

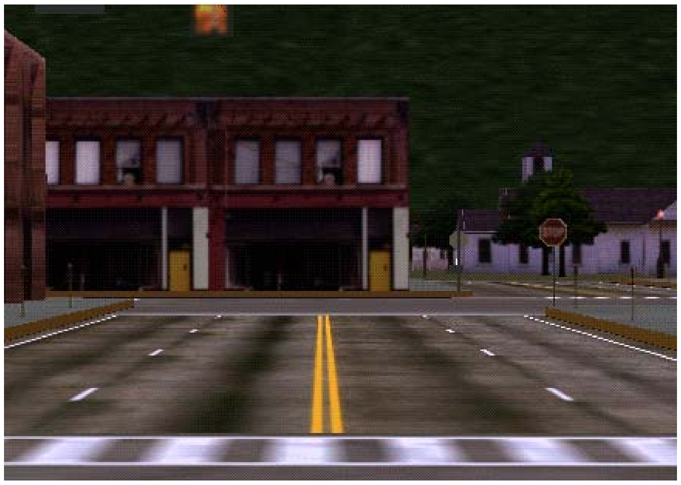


Figure 3.1: A hardware rendered environment (Paradigm, Inc and Silicon Graphics, Inc).

into a summation of the incident energy over all the light sources. Material properties are limited to setting of diffuse and/or specular reflectivity with a constant ambient term included to approximate all the global illumination effects. Lacking in these solutions are the color bleeding and soft shadowing of radiosity solutions as well as object-to-object mirror effects of ray traced images, although the newest high-end machines include image mapping and shadowing algorithms to portray more subtle shading effects, and give the illusion of additional geometric detail.

While hundreds of thousands, or even millions of polygons can be rendered per second on today's fastest graphics hardware, for complex scenes this is not enough. Realistic environments can have many millions of objects before decomposition into polygons. These environments are too complicated to be used "as is" for hardware shaded walk-throughs.

In general, hardware rendering effects can be useful for giving a sense of traversing a space for some applications. However, they are far from realistic because of the non-physical lighting models available and the limitations on numbers of polygons that can be used to model the environment.

3.1.2 Radiosity walk-throughs

As discussed in the previous chapter, radiosity techniques explicitly model the diffuse interreflection of light (Equation 2.8) in a scene to compute the radiance distribution L leaving each object [119], [221].

Since radiosity correctly models the physical propagation of light and takes into account the

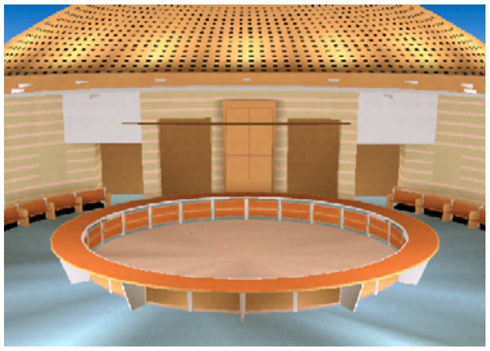


Figure 3.2: A radiosity rendered environment (Lightscape Technologies, Inc).

subtle interreflection effects that are responsible for the atmosphere of a scene, it is useful for helping to understand the three-dimensional nature of the objects represented (Figure 3.2). The main problem is that although classical radiosity correctly models diffuse interreflection, with L being independent of the direction of view (θ, ϕ) from the surface normal, the world is not made up of strictly diffuse surfaces. Diffuse scenes do not efficiently communicate spatial relations, since the human visual system is very good at detecting and interpreting highlights which result from the directional variation of L.

This problem with the radiosity method is actually one of its more salient features when placed in the correct context. For environments where specular characteristics are minimized, the view-independent character of the results is very useful. A radiosity solution is a set of radiance values, possibly a spectrum of values corresponding to different wavelengths, attached to the points or surface elements in an environment, can be converted to RGB values and be used in place of the hardware lighting values. Given such a pre-computed solution, graphics hardware can be used to render images from any viewpoint. Radiosity approaches, therefore, exhibit all the pros and cons of hardware-based environment traversal with increased accuracy of the illumination effects. This increased accuracy does not come without a cost. The radiosity solution itself is very sensitive to changes in the environment, as are all global illumination solutions.

If the changes in the environment are known a priori, as in animated environments, a special data-structure, the *back-buffer* [22], can be used to keep track of changes in form-factor geometry. This allows for an incremental calculation of radiosity over the time of the animation and yields a significant improvement over generating a new radiosity solution for each time step of the animation.

For environments in which no a priori information is known, incremental techniques [57, 108] which use an “energy correction” or redistribution term to correct the radiosity solution without starting from scratch are often employed. More recently, advanced data-structures, the *shadow form-factor list*, for allowing the quick computation of changes in the environment [202], and dynamic scene analysis in hierarchical radiosity [101] have also been presented.

These approaches, although relying on different amounts of information, allow for a quicker re-calculation of the radiosity solution than actually restarting the process. All aid in the use of radiosity for interacting with realistic environments.

3.1.3 Reducing object complexity

If we are willing to live with the accuracy afforded us by the combination of a radiosity pre-process for diffuse interreflection with hardware approximated first-order specular effects, the biggest bottleneck left to overcome with geometric walk-through systems is the amount of geometry that can be passed into the graphics pipeline for processing. Complex geometry is needed to add visual richness but is very costly to process. Recent research has gone towards replacing “expensive” forms of complexity/geometry with simpler forms of complexity such as textures and billboards. The goal is to reduce the environment size in ways that will not be very apparent to the user.

Textures

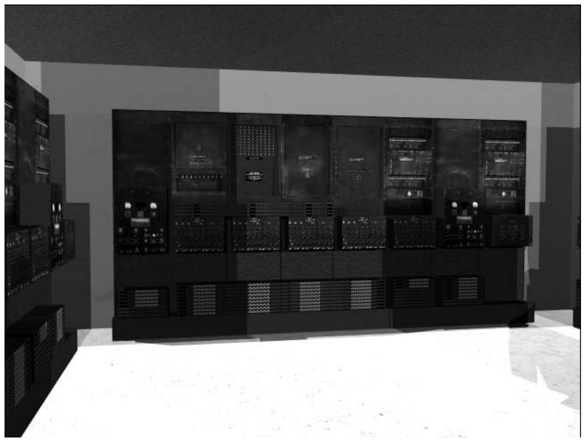


Figure 3.3: Textures replacing geometric detail (Dill Pixels, Inc).

Textures are used to replace geometry that would otherwise be very costly to build (Figure 3.3).

They can be used to add complexity down to the pixel level, something that would be very difficult to model. Landscapes and backgrounds that are sufficiently far away relative to the viewer's position in the environment are often represented as textures as they would require millions of surfaces to physically model. Geometry that is visible at non-grazing angles, so that the structure is inferred as opposed to explicit, can be easily represented using textures or bump maps whereas objects that will be seen from grazing angles are often modeled explicitly. Texturing has progressed beyond the standard decal mapping, shrink-wrapping, and now incorporates alpha-channel transparency useful in making asymmetric objects.

Billboards

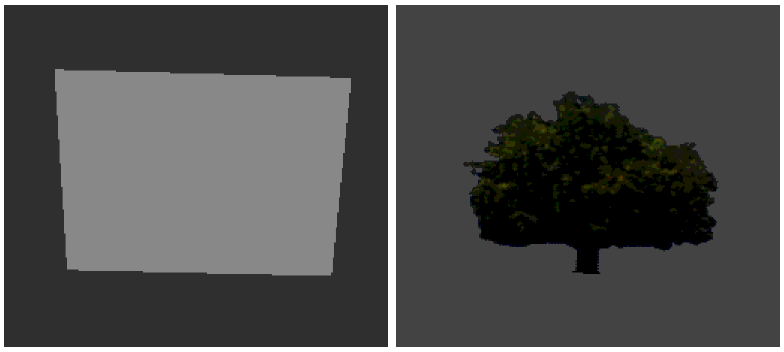

(a) (b)

Figure 3.4: Billboard replacing geometric detail (Silicon Graphics, Inc).

Billboards [251] are a specific combination of geometry and textures useful for radially or spherically symmetric objects. Objects, like trees, can be represented as a plane with a tree texture (Figure 3.4). The billboarding feature keeps the polygon oriented towards the viewer at all times. The effect can be disconcerting for billboards that are very close to the viewer but the effect is dramatic at a distance and allows for objects that would normally contain hundreds to hundreds of thousands of polygons to be composed of a single polygon and a single texture.

Multi-level modeling

Multi-level modeling [251] is a method for providing perceptually effective detail to the viewer during a walk-through. Different versions of objects in the environment, varying by resolution, are stored and the appropriate object is chosen based on how it will appear in the current view

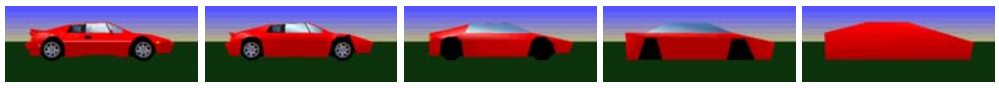

(a) (b) (c) (d) (e)

Figure 3.5: Multiple levels of detail: (a) - (e) very high to very low detail (MultiGen, Inc and Silicon Graphics, Inc).

(Figure 3.5). Low resolution objects are used when the object projects onto a small set of pixels in the current view. Higher resolution versions of objects are used when the object projects to a larger area of the current view. The major hurdle with these methods tends to be the transition between the different versions of objects. Care has to be taken so as not to cause any noticeable shifts between consecutive frames of the walk-through. To this extent, designers often choose between weighted blending or morphing to alleviate any noticeable artifacts.

3.1.4 Environment data management

The ideas in the previous section were designed to reduce the complexity of individual objects in an attempt to lower the overall size of the environment. Researchers have also tried to incorporate “working set” principles in an attempt to limit the amount of the environment that needs to be processed for the current frame.

Spatial organization

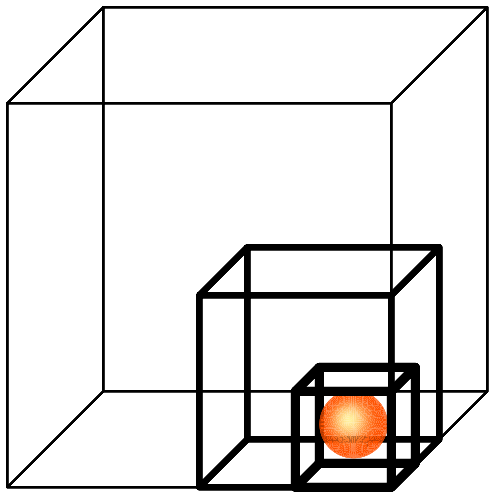
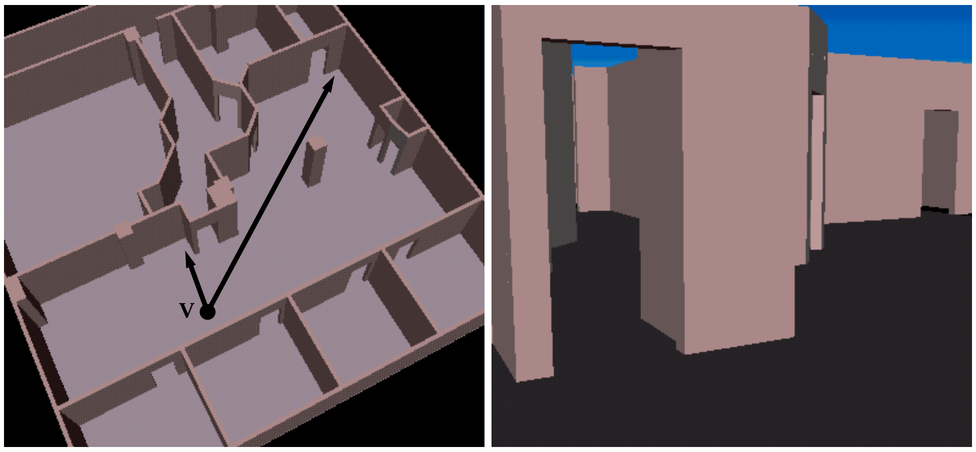


Figure 3.6: Spatial decomposition.

The use of bounding volumes as a method of calculating potentially visible objects has been used for many years in the rendering community and exists as a basic feature of many walk-through architectures such as IRIS Performer [251]. In this approach, the view frustum is intersected with the hierarchical bounding volumes (Figure 3.6) in the environment to find the geometry that should be processed for the correct view. When the environment consists of clustered groups of objects, this technique works very well.

Visibility culling



(a) (b)

Figure 3.7: Visibility culling: (a) overhead, (b) viewer

Visibility culling [3, 304, 251] is an automatic approach that pre-processes an environment into sets of surfaces visible from a current room through the *portals* from that room (Figure 3.7). Each room in an environment is tagged with information about those surfaces that are potentially visible from the current location. Only those surfaces that could be visible are dealt with during processing. The overhead of these methods for large environments can be enormous.

Efficient primitives

Although having an efficient data representation applies to individual objects in the environment, in today’s walk-through systems it often means using a simple primitive object to represent the entire environment. Such is the case on most hardware rendering platforms where the pipeline is optimized to process triangles and triangular meshes (Figure 3.8).

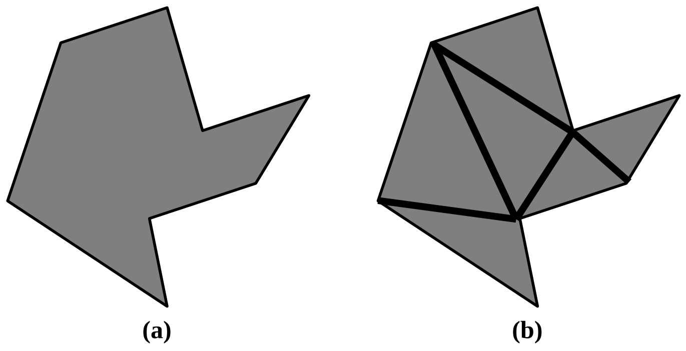


Figure 3.8: Efficient primitives, (a) concave polygon, (b) hardware-optimized triangular mesh.

3.2 Quasi/Non-Geometric Approaches

Quasi- or non-geometric approaches to walk-through systems store illumination information into sample-based structures, often images, which are then combined to reconstruct different views of the environment. Since these structures store radiance information, we are able to capture any visual effects such as diffuse interreflection, and higher-order specular and glossy reflections, that can be generated by a global illumination algorithm, provided we are willing to wait for its completion. The use of this type of storage mechanism is very efficient, the visibility and illumination problems are already solved. Only incoming radiance values seen along particular directions are stored, removing the relationship between environment complexity and walk-through performance. The amount of space needed to store a complex environment is then comparable to the amount of space needed to store the simplest environment.

3.2.1 Pre-recorded animation

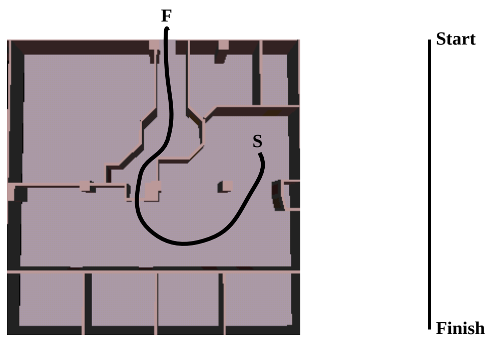


Figure 3.9: An animation path.

To date, the most realistic but restrictive walk-through method is the pre-recorded animation. Animations show snapshots of an environment over time allowing for both viewer and object motion (Figure 3.9).

Since animations are sets of images, each of which is computed with the environment configuration for the appropriate time, any type of global illumination algorithm can be used. Since the entire walk-through is pre-computed, the frames can be of the highest quality. Most animations are created using the current state-of-the-art multi-pass algorithms which accurately model both diffuse and specular reflection. Creating an animated sequence with these algorithms is very time consuming but all of the time is consumed by the rendering process.

Although it is easy to say that animations do not allow any interaction with the environment, one can just as easily say that you have the ability to walk along one particular path through the environment as it is changing over time. This appears very limiting but sometimes a single path, such as walking down a corridor is all that is needed for a particular effect.

3.2.2 Animation networks

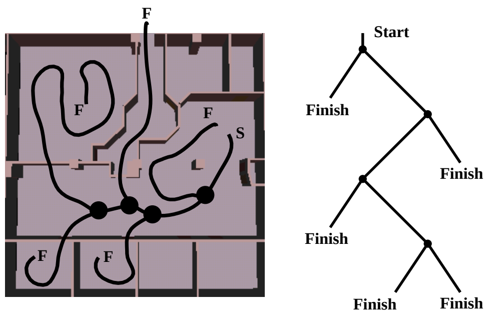


Figure 3.10: A network of animation paths.

It is often the case that differing amounts of freedom of movement can be added by recording a network or tree of paths for the viewer to tour. This is similar to many video-disc applications. Animation networks constructed for interactive applications, such as walk-throughs and video games, have been around since the early eighties [179]. Pre-recorded animations are constructed for sets of paths through the environment and then an interactive application is used to control the traversal of these paths (Figure 3.10). The viewer is limited to the set of paths that have been constructed but if care is taken, a thorough depiction of an environment can be created. It is possible to highlight all the major features within an environment and since each path is fully computed, the highest quality illumination effects can be exhibited.

3.2.3 Image-based walk-throughs

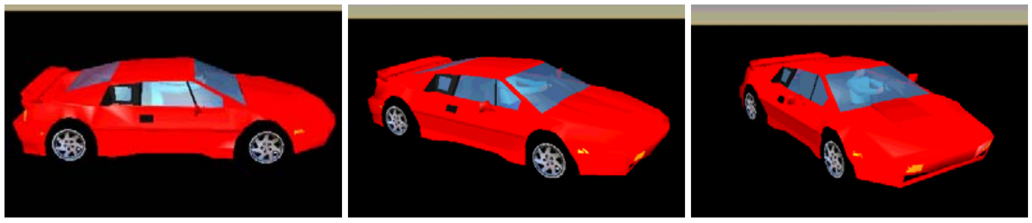


Figure 3.11: Image-based view interpolation, (a) view one, (b) intermediate view constructed from the two views, (c) view two (MultiGen, Inc and Silicon Graphics, Inc).

Image-based interpolation has been employed in flight simulators [34], and has been applied to more general graphics applications by a wide array of graphics and vision researchers [60, 58, 193, 270]. In this approach, often called view interpolation, geometry in the environment is replaced by images placed at strategic points (Figure 3.11). The images need to be spaced closely enough as to adequately represent the particular view of the geometry they are replacing but not so dense as to require an inordinate amount of pre-processing time[1]. Simple spatial relationships are used to decide, based on the current view parameters, which structures contain the correct visual information about the environment. The structures are then combined to yield an image of the environment from the particular view.

3.2.4 Higher-dimensional data structures

Recently, a small number of research groups have focused on using higher-dimensional structures to store radiance information within static environments. The lumigraph [121], and light field rendering [175] work is applicable to sets of views that circle the convex hull of an object, or clusters of objects, within the environment, or to outwards views within a region of free space in the environment. The $4D$ storage structure (Figure 3.12) used indexes radiance traveling along rays through pairs of slabs that surround the object for internal views or surround the viewer for

[1]The logical extreme of animation networks would store a different pre-computed image for a very dense set of spatial locations in the environment allowing total freedom of movement through the switching of images.

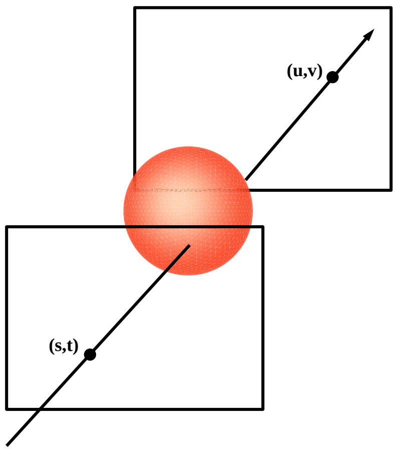


Figure 3.12: Higher-dimensional data structures: 4D slab.

external views. New views are constructed by mapping the desired new view configuration into the bounding slabs and isolating the endpoints of those rays whose radiance values contribute to the current view. This research is very promising. Research on radiance interpolants [302] attempts to compute conservative bounds on the radiance distribution leaving objects in an environment by enclosing each object or cluster within a local set of slabs. The outgoing radiance distribution is represented as rays through the boundaries of the slabs with subdivision being performed if sources of aliasing such as gaps and occlusions are found within the slab. The slabs can then replace the objects themselves during a walk-through phase with illumination queries accessing the radiance slabs for the appropriate objects. No work as of yet has been done on using these higher-dimensional structures for storing a time-varying radiance distribution as needed when working with animated environments.

3.3 Summary

Geometric walk-through systems offer above average levels of interactivity provided the object count is kept low. There are many different approaches to doing this, some concentrating on handling individual pieces of geometry and others concentrating on minimizing the overall size of the environment used in producing the current frame. With these techniques it is possible to manage the size of the geometric database so that it can be processed efficiently. The main problem with using geometry-based walk-through techniques is that we have to trade off certain aspects of realism in order to achieve reasonable interactivity. Diffuse interreflection can be handled in a radiosity pre-process but accurate specular highlights are not handled. Low-order specular highlights can be approximated using hardware shading but this is often not enough accuracy in scenes where mirror highlights are prevalent. These pros and cons are the opposite of those associated with quasi/non-geometric walk-through systems.

Current quasi/non-geometric walk-through systems offer below average to average levels of interactivity. Either the performance of the walk-through mechanism is good but the conditions where it applies are limited, as with the higher-dimensional radiance structures, or the conditions where the method applies are general but the performance is limited, as with the view interpolation algorithms used to merge proximate images which are more expensive than hardware rendering algorithms. All the quasi/non-geometric methods offer much greater flexibility in the visual effects that can be presented since the global illumination method is encapsulated in the environment pre-process.

Of the major methods for constructing walk-through systems, sample-based systems are better when radiometric accuracy is desired, but for interacting with animated environments, they can be very costly both in the storage required and the time needed to create the images that are to be presented. For environments where interactivity is paramount, the geometry-based systems with algorithms designed to monitor the size of the environment are still the best bet. For our application where we desire the utmost in radiometric accuracy, we want to work with the newer sample-based methods.

Chapter 4

A Framework for Sample-Based Systems: Modeling the Time-Varying Radiance Distributions Within an Environment

"The will of the people is the only legitimate foundation of any government, and to protect its free expression should be our first object."

Thomas Jefferson

The prototypical global illumination and walk-through methods just presented appear to solve vastly different problems. This is not really the case. They can all be thought of as approaches to calculating and presenting the time-varying illumination information within an environment and as such can be formalized using the same framework. This framework is most easily expressed as an analogy to human vision and is a useful starting point for formulating solutions to our more specific problem of efficiently computing walk-throughs of complex, animated environments.

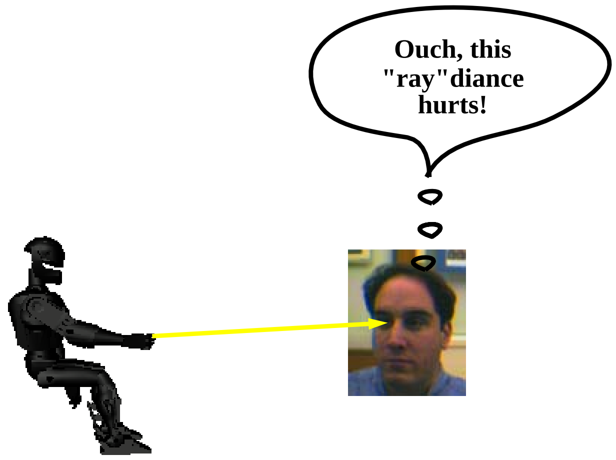


Figure 4.1: Viewing radiance along a ray.

4.1 Radiance Distributions - An Analogy

The distribution of light, radiance, inside an environment that changes over time can be presented as an analogy to human vision (the plenoptic function [2]). For any position where a viewer might be standing, a spot on the retina is “energized” by the light that travels along a ray that ultimately enters the eye and strikes the spot (Figure 4.1). Each ray carries radiance, or luminance, information. If we allow the retina to be exposed to a set of rays coming from different directions, an image is formed. If we extend this metaphor over time, a movie is projected onto the retina. To be able to allow free range of movement and orientation within an environment, movies encompassing the desired time interval, for every viewpoint and incoming ray direction over the sphere encompassing that view must be stored.

4.1.1 Formalization

By parameterizing each potentially varying quantity (Figure 4.2), we can pose the problem of capturing the radiance distribution of an environment using the language of sampling and reconstruction. The viewer position can change spatially, (x, y, z) . The view direction/orientation can vary spherically, (θ, ϕ), using a fixed viewing coordinate system. Time, (t), can vary. With all the parameters fixed the formalization yields the radiance seen at (x', y', z') along direction

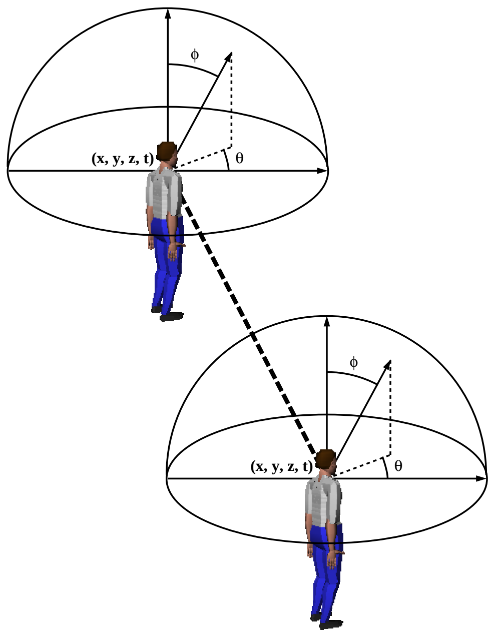


Figure 4.2: Parameterizing the time-varying radiance distribution, $L(x, y, z, \theta, \phi, t)$.

(θ', ϕ') at time (t') as in the scenario mentioned in the previous section. If we have the radiance value $L(x, y, z, \theta, \phi, t)$ for every configuration of the input parameters, we have a complete representation of the environment's radiance distribution This representation fully characterizes the time-varying illumination in the environment capturing the illumination seen from all possible view positions, along all directions, at all times. Traditional imagery can then be extracted from the radiance distribution by slicing the $6D$ distribution with a $2D$ plane, varying directions (θ, ϕ) for a fixed viewpoint (x', y', z') at a fixed time (t'), and then cropping the unnecessary directional information.

The problem of processing and displaying a complex, globally illuminated, animated environment involves computing the desired/necessary portion of its radiance distribution and providing an efficient mechanism for extracting and presenting images. It is infeasible, if not impossible to fully calculate the distribution, or to describe it analytically so approximation methods are required. These techniques have been designed to either efficiently or accurately construct portions of the radiance distribution. Typical walk-through mechanisms quickly compute that portion of the radiance distribution that is required for the current view either from pre-computed illumination values or using on-the-fly shading algorithms. Typical global illumination methods often compute only those radiance values necessary for a particular view at a particular time, thus requiring a new solution for each new configuration of the environment, a new portion of the full radiance distribution. We want to do better. Our approximations will be based on effectively sampling the radiance distribution looking for coherence that may exist and efficiently constructing images from a more “complete” set of radiance samples.

4.1.2 Sampling - structure and coherence

Although the structure of individual radiance distributions is not easily visualized given the time-varying geometric, view, and lighting configurations of the environment, we are not dealing with completely unknown functions when we attempt to sample and reconstruct different distributions. There is structure within an environment’s visual information that can be tracked provided we can describe what effects we want to look for. This structure, slowly varying illumination information between discontinuities, is useful in helping us construct the most complete set of samples we can while still maintaining a reasonably sparse sampling rate. Attempting to find this type of coherence forms the basis for most adaptive subdivision algorithms.

The key to deciding on the importance of different visual effects, rating their importance in the presentation of the environment to the user, is tied to human physiology, how the human visual system samples the environment, and perception, how the human visual system interprets those samples. Since global illumination simulation algorithms and animation techniques are designed to compute images of the environment just like those projected onto the retina, the metaphor is directly relevant.

4.1.3 The human visual system

To get a feel for how we might want to sample a typical radiance distribution, we can look at how the human visual system deploys its sampling mechanisms (Figure 4.3). The human visual system extracts a limited number of samples from four dimensions of the radiance distribution at each instance of time. Each eye, spatial dimension, takes a snapshot of a dense set of directions, directional dimension, with foveal directions being sampled more densely than the periphery. The temporal dimension is sampled continuously, with neurons producing continuous output, and differs from the other dimensions in that any filtering must be causal/asymmetric.

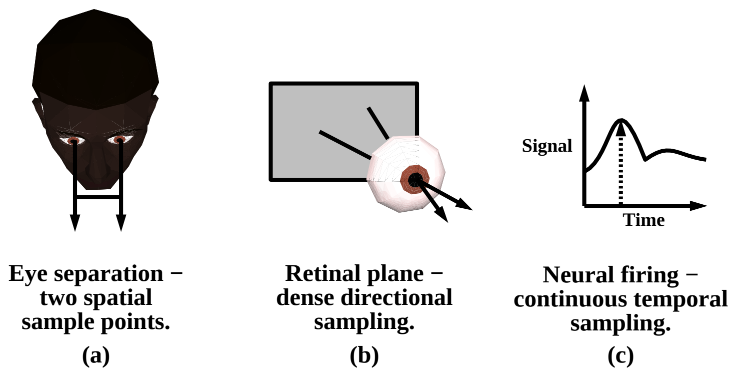

Signal
Time
Eye separation − two spatial sample points.
(a)
Retinal plane − dense directional sampling.
(b)
Neural firing − continuous temporal sampling.
(c)

Figure 4.3: Sampling patterns in the human visual system: (a) spatial, (b) directional, (c) temporal.

This distribution of samples in the human visual system maps very naturally onto the sampling patterns used in computer graphics and vision when generating imagery. First, the two eye/view methodology is used to capture and reconstruct depth information, facilitating view reconstruction, in a manner that closely matches the epipolar formulation in computer vision [295, 193]. Second, individual images traced on the retina require a large number of discrete samples that are densely packed. This sampling pattern matches the pixel/image methaphor used to generate typical computer images. In fact, a large amount of work has been done on mimicking the retinal sampling distribution in a computer graphics context [73, 197]. Third, the most interesting sampling similarity exists in the temporal dimension where the continuous neural signal pattern maps naturally to the computer graphics idea of sampling densely in time. In an attempt to generate the highest quality images and animations, intuition and experience evolved into a sampling scheme that matches the way human vision performs its similar task.

4.2 Constructing Radiance Distribution Samples

The overview just presented is abstract and high-level. It assumes that you have the radiance distribution, it can be manipulated, and easily sampled. This is not the case with synthetic environments. The radiance distribution must first be constructed from the geometric, view, and lighting configurations with synthesis taking place before analysis. Each radiance sample to be evaluated requires a global illumination calculation and, therefore, is difficult to compute quickly (Figure 4.4). This leaves us in no position to compute a dense sampling of the distribution and inhibits our ability

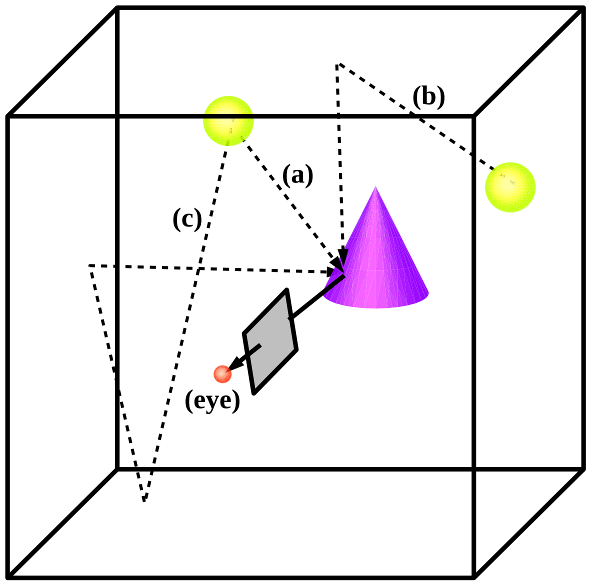


Figure 4.4: Constructing the radiance (global illumination) seen along a direction: (a) direct illumination, (b) two-bounce illumination, (c) three-bounce illumination.

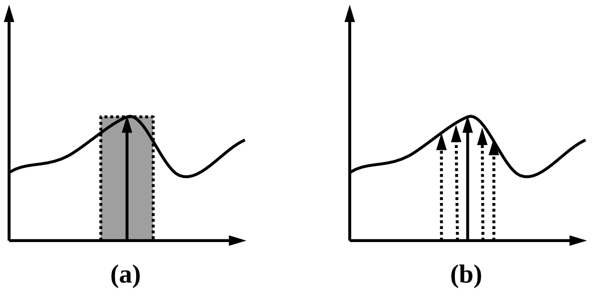


Figure 4.5: Approximating $1D$ local averages: (a) box filtering (continuous), (b) jittered sampling (discrete).

to perform accurate analysis. We need a set of techniques that minimize the amount of sampling required, and store more information about the structure of the radiance distribution with each sample. These requirements are met by traditional computer graphics techniques: adaptive sampling, jittered supersampling, and derivative approximation using averaging and differencing (Figure 4.5 and Figure 4.6), that guide the placement of samples within a distribution. The derivative and local average information that can be computed with these techniques augments the radiance information that is stored with each sample and is very useful when it comes time to formulate a reconstruction scheme.

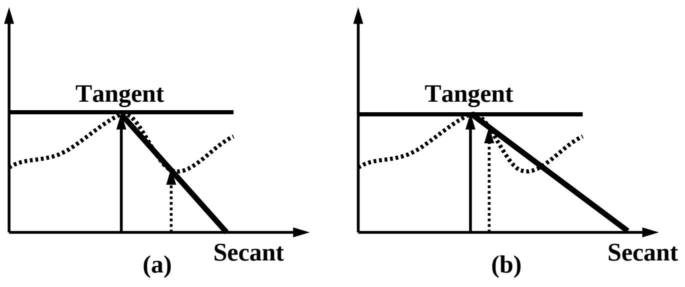


Figure 4.6: Approximating $1D$ derivatives using the secant line (difference) between two points: (a) poor approximation, (b) better approximation.

4.2.1 Separable illumination modes

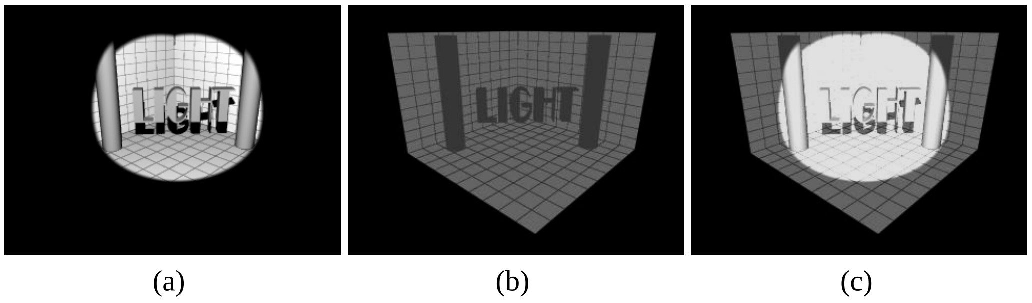


Figure 4.7: Separable illumination modes: (a) direct, (b) indirect, (c) direct and indirect.

In addition to trying to place the sample locations as appropriately, yet as sparingly as possible, we would like to be able to minimize the typical cost of the global illumination simulation (Chapter 2) for computing the values of these samples. This involves analyzing the rendering process and deciding which effects are most important for our type of presentations and which algorithms can garner us those effects. The combination of both object space and image space techniques is simplified when we divide the construction of radiance distribution samples into the separate computation of direct and indirect illumination (Figure 4.7).

4.2.2 Separable dimensions

Decomposing the global illumination computation into indirect and direct components using typical simulation algorithms exposes a separability in the different dimensions of the radiance distribution. This separability is exhibited both in the analogy used to formulate the distribution's definition and in the way the human visual system differentiates its sampling mechanisms. Using

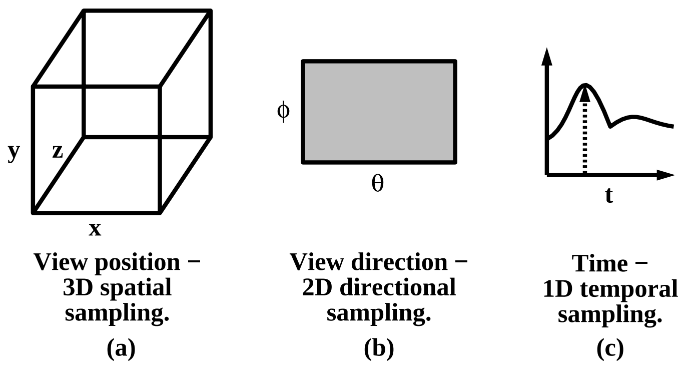

y
z
x
View position − 3D spatial sampling.
(a)
ϕ
θ
View direction − 2D directional sampling.
(b)
t
Time − 1D temporal sampling.
(c)

Figure 4.8: Separable dimensions: (a) spatial (b) directional, (c) temporal.

these standard methods to construct a representation for a radiance distribution entails rendering a set of images, directional variances, in the form of animations, temporal variances represented as time-sequences, for a dense set of important view locations, spatial variances, placed within the environment (Figure 4.8).

The problem with using this separable representation is that although the analogy intuitively makes sense and fits in well with the technology we want to work with, a strict set of boundaries are created that we might not be able to easily cross. As nice as it is to work with standard algorithms, each concentrating their efforts in the most natural way, the task of combining these techniques in an efficient way so as to allow us to reconstruct the complete radiance distribution might be more difficult than taking a more general approach early on. This is the single most important decision when it comes to producing a system that will efficiently process complex, globally illuminated, animated environments.

4.3 Organizing Samples - Storage Mechanisms

It is important to remember that our goal is not only to be able to efficiently compute the illumination seen in a particular image of the environment but to be able to represent the temporal variances in the illumination in a manner that allows the easy creation of any number of images of the environment. The two tasks are similar but the former is well suited to the direct environment-image transformation that rendering methods are known for while the latter is best suited to an environment-intermediate radiance structure-image transformation. There is nothing that precludes this intermediate structure from being image/sample-based, and in fact it will be for our purposes,

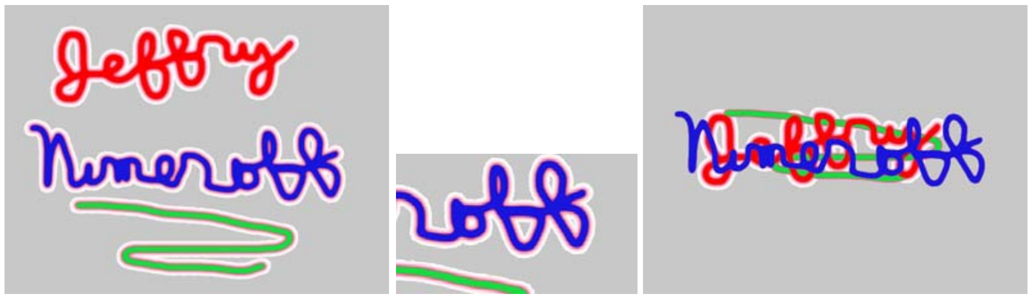

(a) (b) (c)

Figure 4.9: Images: (a) total, (b) partial, (c) multi-layer.

but it should be noted that we do not have to organize our radiance samples in the forms of images.

4.3.1 Images - aggregate of radiance samples

The typical image-based, row-column ordered, storage mechanism (Figure 4.9) is a useful and convenient way to store radiance information but can be prone to oversampling in areas with minimal variance. The problem is not so much with individual images as with the coherence that exists between images in a sequence designed to capture temporal illumination variances. If the background of an environment does not vary much, why must it be densely sampled in each image? It turns out that the images used to store the radiance information do not, in fact, have to have the same shape. This allows for multi-layered approaches to image-based storage [320] where the background can be rendered into a single layer with varying information being stored as small image slivers, partial images on individual layers. It can also be the case that the ability to reconstruct from multiple images can be assisted through the use of extra pseudo-geometric information included with the stored radiance values. Typically computer images can be augmented to include depth and orientation information about the objects seen through the individual pixels. This information represents the $3D$ structure for all the visible surfaces in this particular image, a view-dependent geometric representation.

4.3.2 Advanced data structures - individual radiance samples

Recently, abstract higher-dimensional structures such as the lumigraph [121], light fields [175], and other "slab-style" radiance structures [302] have become prominent in the processing of static environments (Figure 4.10) free of atmospheric effects. The idea behind these structures is to

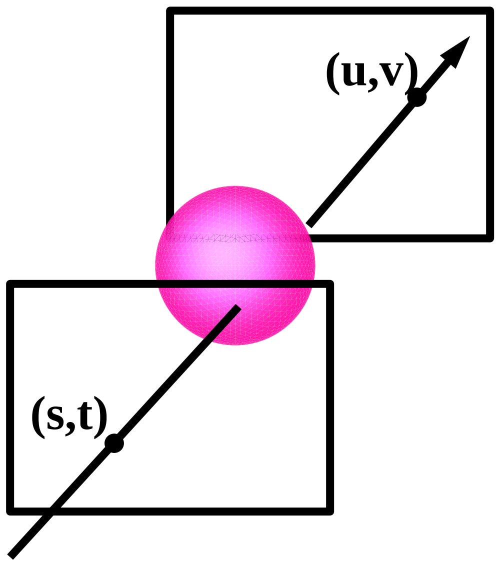


Figure 4.10: Advanced data structures: $4D$ slab.

store directional radiance information for a set of view positions within a single structure instead of directional radiance information for a single view (image). This is usually done by aligning pairs of planes (slabs) that can be used as a $4D$ indexing scheme *(line-space)* for rays that pass through the space bounded by the planes. A pair of indices (s, t) is used for the first plane and a pair of indices (u, v) is used for the back plane and represent a $3D$ oriented line segment. A view position outside of the space bounded by the planes that looks through the space accesses the data structure through the set of viewing rays ($3D$ oriented line segments) that pierce the slab. Slab-based structures have only been implemented in $4D$ relying on the radiance invariance along the particular direction of interest. This is very limiting when the environments we want to work with contain the "nooks and crannies" and occlusions that exist in complex, realistic environments. It is possible to extend the slab structure to work in $5D$ allowing complete freedom of movement. This relies on exploiting that fact that radiance invariance exists along a ray between any two points provided there are no occlusions between them (not just between the viewer and the first object seen).

A more well researched alternative, a high-dimensional data structure that has recently been used in computer graphics is the Delaunay complex [15, 340]. In N dimensions, the Delaunay complex (Figure 4.11) decomposes the space into a set of non-overlapping simplices (N dimensional triangles) that obey the empty "circumcircle" property. A simplex obeys this property when

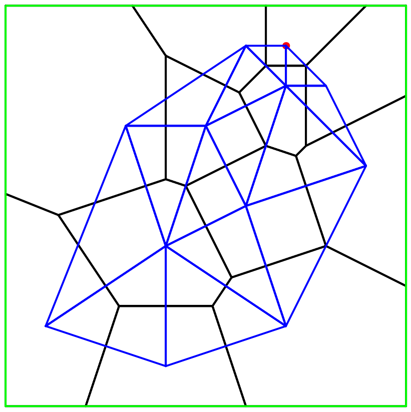


Figure 4.11: Advanced data structures: Delaunay triangulation and its Voronoi dual (Jean Gallier, vordelau1).

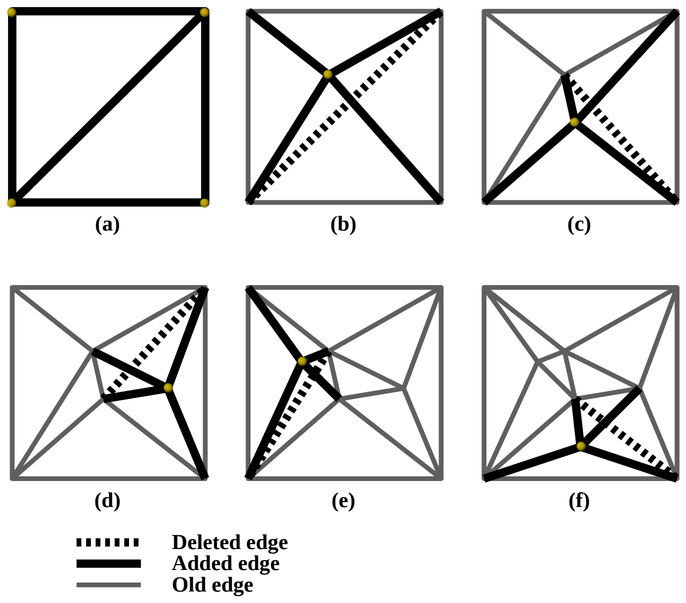


Figure 4.12: Incremental construction of the $2D$ Delaunay complex with 9 points being inserted.

a sphere of dimension N (circle, sphere, hyper-sphere, etc.) contains the vertices of the simplex, and no vertices from other simplices are contained within the sphere. The Delaunay complex constructs a set of ND connected components that share $(N-1)D$ faces. The complex is also useful as an incremental method (Figure 4.12) for decomposing a space through non-uniform adaptive sampling. As random ND points are generated, they can be added to the Delaunay complex. To do this, the simplex that surrounds the new point is found, the simplex is subdivided, and the circumcircle property is tested (with offending facets removed). Each simplex can then be tested for coherence using a metric defined over the simplex vertices. Any simplex whose vertex metric shows that the vertex data values differ by a wide margin is a candidate for subdivision. A $6D$ Delaunay complex might be a very appropriate data structure for storing the distribution as it can handle temporal as well as spatial and directional variances in one unified structure.

4.4 Reconstruction - Combining Radiance Distribution Samples

As just demonstrated, we can store radiance distribution samples in many different structures, some more traditional (images) than others (high-dimensional). Storage is only half the battle. We not only want to be able to efficiently store the radiance distribution information but to reconstruct an approximation of the continuous distribution from the samples as well. To do this, indexing the radiance data should be quick and easy, as should manipulating the data to form an image.

4.4.1 Image operations

When working with images either containing radiance data by itself, or augmented with geometric information, there is still a limited set of operations that can be performed to aid in the reconstruction of the radiance distribution.

In the spatial domain, re-projection methods (Figure 4.13) that work on complete images or in a pixel-by-pixel manner can be used to warp or morph one image into another [26], or a series of source images into a destination image. These mappings can be linear as well as non-linear. Projective transformations can be used to map from the image space of source images into the image space of a destination image to perform mosaicing [34, 297, 58] as in view reconstruction [60, 193, 270].

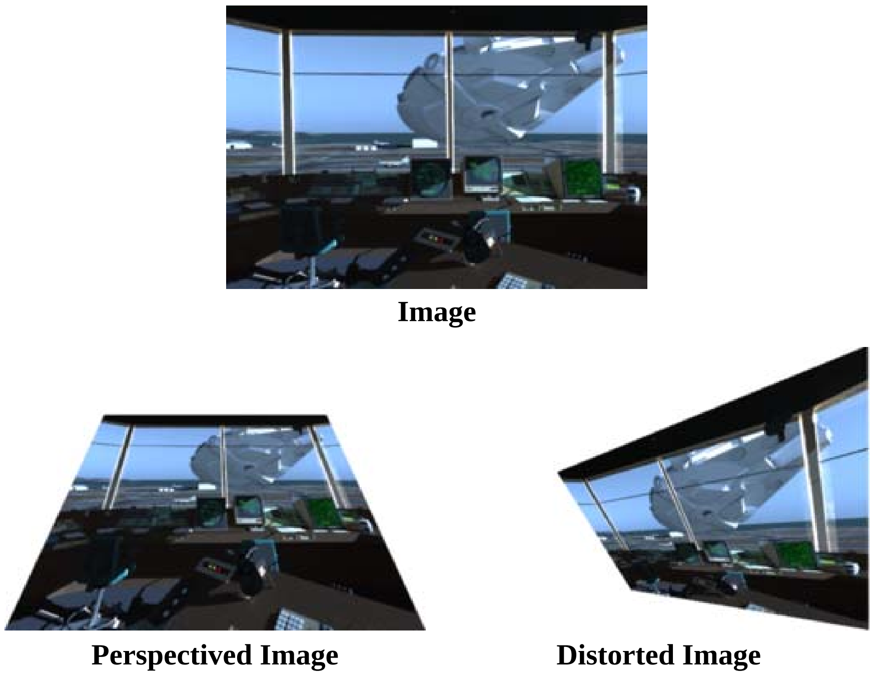


Figure 4.13: Image re-projection: spatial reconstruction via warping.

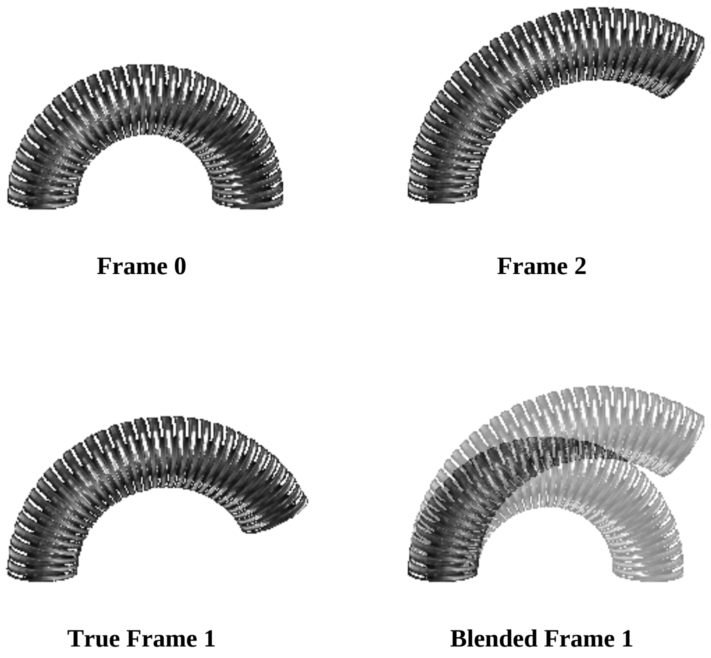


Figure 4.14: Image blending: temporal reconstruction via pixel operations.

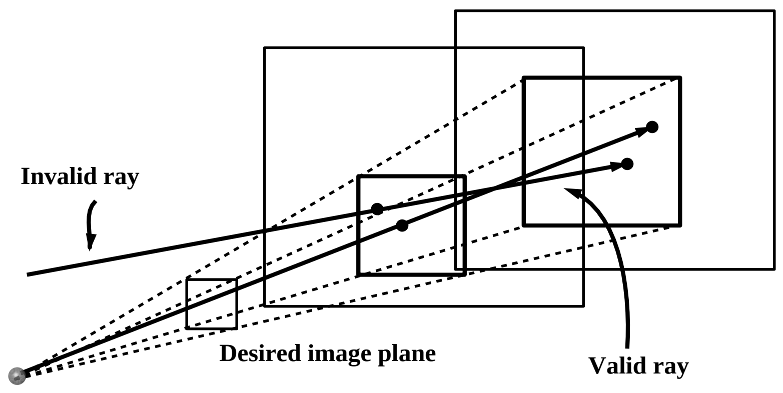


Figure 4.15: Slab-based reconstruction indexes direction sets between (s,t) and (u,v) planes.

In the temporal domain, where object motion is prevalent, blending techniques (Figure 4.14) are not generally appropriate, except in the case of a very dense set of images, as the projection information needed to incorporate motion reconstruction can be different for every pixel in the image. For a moving view, all that is needed, a priori, are the object space to image space transformations for the source and destination images. In the temporal case the object space to image space transformations are necessary, but are augmented by the time-varying object space to object space transformations for the individual objects visible through each pixel. Image-based techniques from computer vision [295] incorporate this information by using time-sequences to compute either the apparent image velocity field (optical flow) or actual image velocity field (visual flow) under the assumptions that the objects in the environment are mostly rigid and the image radiances/luminances are mostly constant (within a small ϵ). The blending of pixels values for an image becomes the process of "sliding" the pixels along piecewise-linear planar image space approximants of their object space paths. For a moderately dense set of images, this works adequately. Neither of the techniques, straightforward image blending or pixel-wise motion, work well for a sparse set of images.

4.4.2 Advanced data structures

Although image-based storage mechanisms have been used for many years, the newer advanced data structures have been designed to allow for sparser sampling, and efficient storage while maintaining accurate reconstruction capabilities.

As mentioned in Section 4.3.2, slab-based structures like lumigraphs, and light fields can be

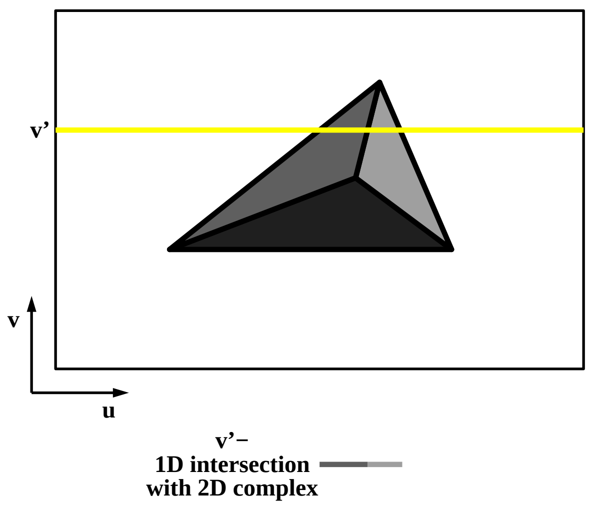


Figure 4.16: Delaunay complex reconstruction: $2D$ simplicial slicing.

used to store the radiance seen along a set of directions through two parallel planes that bound an environment (Figure 4.15). A novel use of projective texturing allows fast access of the directions needed to compose the desired view. Both the front and back planes are parameterized using the standard (u, v) texture parameterization. Each of these planes is projected onto the desired view configuration. After the projection each frame-buffer location holds the (u, v) coordinate for the particular plane. These values can be used to access the $4D$ structure directly. $4D$ radiance interpolants can also be constructed for individual objects and accessed in a ray tracing context. Although using a software only approach is slower, using a slab-like structure on a per object basis allows for the processing of environments with multiple objects, and better freedom of movement. One important issue that is addressed with these high-dimensional structures is that of aliasing. Each slab in the pair can be filtered individually (bilinear), or as a unit (trilinear). Wavelets can also be used to represent the slabs in an efficient manner and aid in filtering.

In order to reconstruct an image from the Delaunay complex (Figure 4.16) we need to be able to isolate those dimensions that constitute the image. This entails fixing the spatial $(x^{'}, y^{'}, z^{'})$ and temporal $(t^{'})$ parameters and allowing limited range on the directional (θ, ϕ) parameters. The easiest way to do this is to walk the simplex list for the complex (using the parametric edge equations) to intersect each simplex against the plane representing all the desired view directions. Once all the simplices have been intersected against the plane, the resulting $2D$ point set can be triangulated and the rendering hardware can be used to prune the undesired directional information. The radiance for each $2D$ point is interpolated from the vertex radiances using the same parametric value used to compute/interpolate the intersection point.

4.5 Summary

In this chapter we presented the radiance distribution, an adaptation of the plenoptic function, that is well suited to serve as the formalization for global illumination and walk-through mechanisms. The formalization is most easily understood in terms of an analogy to human vision. For any position where a viewer might be standing, a spot on the retina is "energized" by the light that travels along the ray that ultimately enters the eye and strikes the spot. Each ray carries radiance (luminance) information. If we allow the retina to be exposed to a set of rays coming from different directions, an image is formed. If we extend this metaphor over time, a movie is projected onto the retina. To be able to allow free range of movement within an environment, movies (encompassing the full time interval desired) for every viewpoint and incoming ray direction must be stored. The $6D$ parameterization of this analogy is the radiance distribution $L(x, y, z, \theta, \phi, t)$.

There are many structures that can be used to store radiance data. Each of the structures, whether they are traditional computer graphics structures like images, new high-dimensional (directional) structures like lumigraphs or light fields, or even abstract N-dimensional structures like the Delaunay complex, have pros and cons in terms of how much of the radiance distribution they can span, how efficiently they can be accessed, and how useful they are in allowing the reconstruction of arbitrary regions of the radiance distribution.

Chapter 5

Computing the Radiance Distribution for an Environment - Multi-Stage Rendering

"Does it have anything to do with shmalts (rendered fat)?"

Anonymous member of the Nimeroff family

In this chapter we present a modified rendering method designed to compute global illumination within complex environments. The method we propose is multi-stage and separates the processing of the different illumination modes by the importance of their visual effects to the overall presentation of the environment. Our method is designed to be flexible allowing accuracy to be traded for efficiency and vice-versa.

An object space radiosity algorithm is used to compute view-independent indirect illumination. Since indirect illumination contains the interreflection effects that add to the "feel" of the environment, it cannot be eschewed. These soft effects can, however, be overwhelmed by direct lighting so they should be able to be calculated to their own degree of accuracy, treated separately from direct illumination. The rendering method needs to be flexible. With this in mind we compute a sparse indirect illumination solution using a *geometrically simplified* version of the environment. The simplified environment and the sparse sampling density elicits an indirect solution that is in some sense "filtered" but the increase in performance is often worth the error that is introduced.

To compute an image of the environment using a particular view configuration, the view-independent indirect illumination information is converted into an *ambient irradiance field* in which the true environment is embedded. An image space direct illumination algorithm, using the current view, is then executed with the indirect illumination value at each visible surface location being used to augment the direct illumination value that is calculated. The direct illumination algorithm uses a variant of deterministic and non-deterministic/Monte Carlo ray tracing to perform its task. Each illumination value is computed as the combination of an indirect component and a direct component, each computed to their own desired accuracy, with the computational savings being fine-tuned so as to outweigh any losses in accuracy.

5.1 Motivation

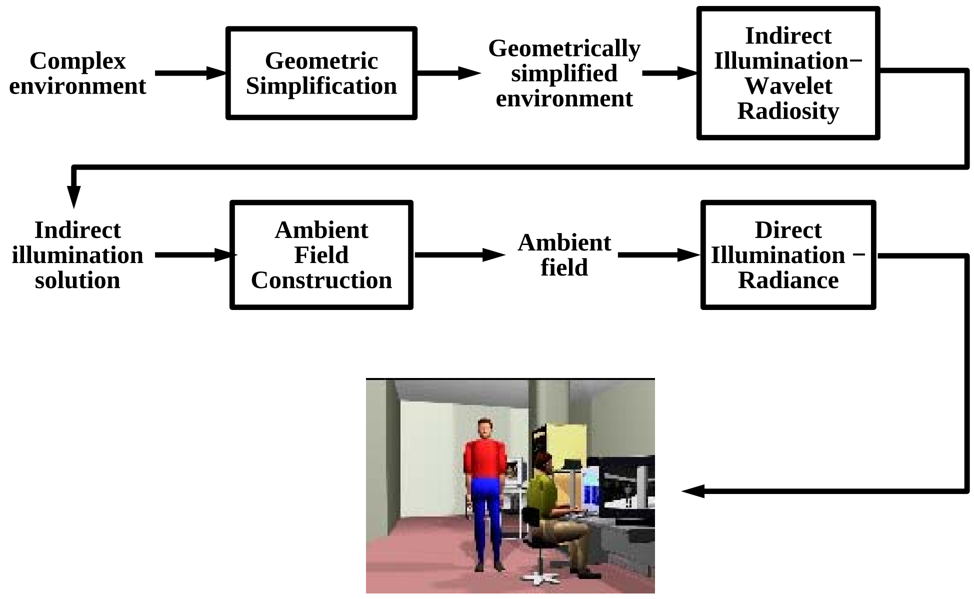

Complex environment
Geometric Simplification
Geometrically simplified environment
Indirect Illumination– Wavelet Radiosity
Indirect illumination solution
Ambient Field Construction
Ambient field
Direct Illumination – Radiance

Figure 5.1: A block diagram of our multi-stage rendering method.

Radiance is a fundamental quantity that exists and can be measured in the real world. We are not so lucky in computer graphics. Our environments are not described in terms of the time-varying flow of radiance around the environment, even though this is a complete environmental characterization. We instead have to use simulation algorithms to produce the radiance values we want to work with from the geometry, view, and lighting configurations we have (Chapter 2). These algorithms are both expensive to use and do not yet encompass all the visual phenomena we see in the real world. Anyone who has used one of today's newest multi-pass methods for computing global illumination knows that there is still a tradeoff between accuracy and speed. So, although we like the increased realism provided by these new methods, using them to compute the frames for walk-throughs in temporally varying environments is infeasible. The question then is how do we "give back" imperceptible amounts of accuracy to "get back" reasonable increases in efficiency and levels of interactivity. Our multi-stage rendering method (Figure 5.1) achieves increases in efficiency by:

1. Using different accuracy levels when computing indirect and direct illumination.

2. Using a geometrically simplified environment when computing indirect illumination.

3. Using an efficient hierarchical algorithm to compute indirect illumination.

4. Using an efficient deterministic and non-deterministic ray tracing algorithm to compute direct illumination.

5.2 Varying Accuracy Levels - Indirect and Direct Illumination

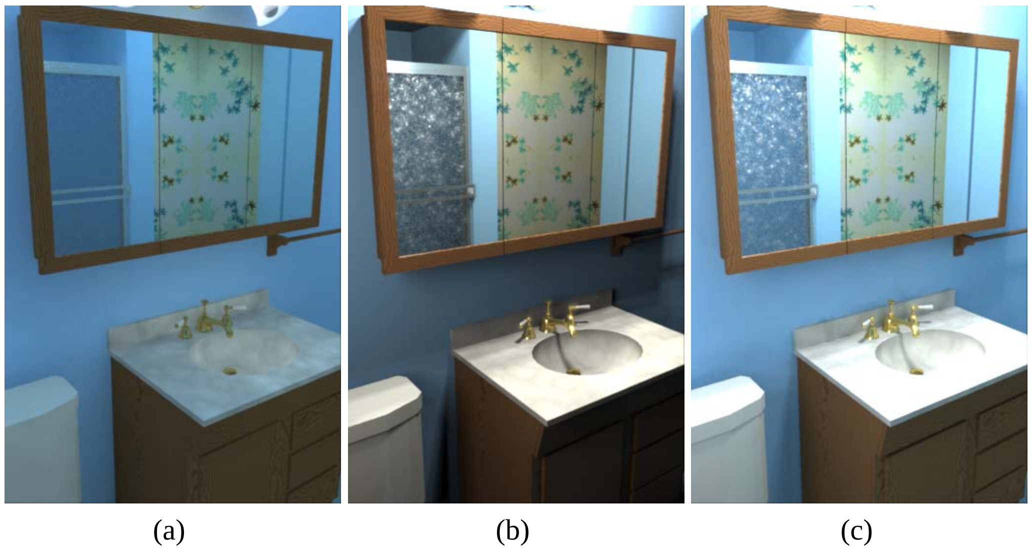


Figure 5.2: Global illumination solutions: (a) indirect, (b) direct, (c) direct and indirect.

As mentioned earlier, indirect illumination (Figure 5.2a) is a much smoother function than direct illumination [7] and as such can be decoupled from the method used to compute these direct effects (Figure 5.2b). Using different algorithms, each designed to accurately and efficiently handle a certain set of the desired visual effects, provides a flexible method for evaluating total illumination (Figure 5.2c). This is useful for architectural scenes which are often composed of many large diffuse objects and a few specular objects. Large-scale indoor omni-directional lighting configurations create subtle interreflections that should be modeled accurately whereas specular highlights can be computed less accurately to maintain efficiency. If we process typical computer environments that contain a small number of directional lights and many semi-shiny/shiny objects, specular reflection can be accurately handled and diffuse interreflection can be processed in a more cursory manner. Whenever possible, we want to exploit any coherence that may exist in the illumination in the environment.

5.3 Indirect Illumination

To efficiently compute indirect illumination and capture its variances we designed a variant of traditional radiosity methods [119, 221, 67]. The method is hierarchical [138, 139] and efficiently computes indirect illumination using a geometrically simplified version of the environment. Many traditional rendering methods apply the global illumination algorithm for each desired configuration of the environment without exploiting any redundancy/coherence that might exist. We will be exploiting coherence in the global solution whenever possible, re-using computed information where we can.

5.3.1 Coherence

Coherence measures are useful in finding the similarity between a set of solutions, finding those regions of the solution space where we have already computed "enough" information. Those areas of the solution space that fall in regions that are "coherent" are considered to be reconstructible from proximate elements in the solution space. Coherence measures form the basis for most computer graphics adaptive subdivision algorithms. In our particular implementation, spatial coherence is exploited by calculating the indirect illumination only at a sparse set of sample points on the surfaces in the geometrically simplified version of the environment and then using simple interpolation methods to fill in the missing information.

5.3.2 Geometric simplification

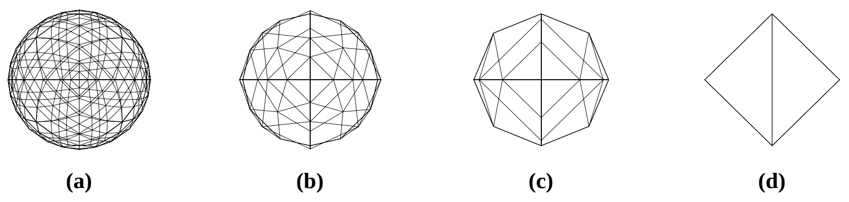


Figure 5.3: Geometric simplification: (a) tessellated sphere, (b) - (d) replacement by different geometric simplifications.

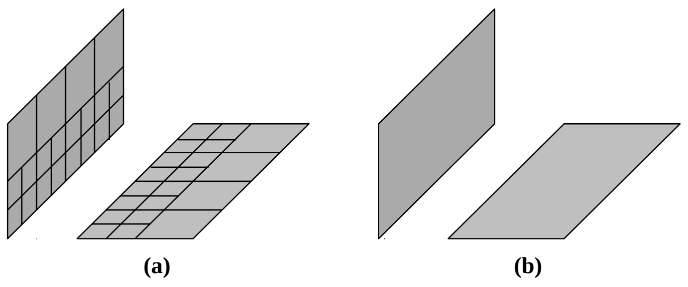


Figure 5.4: Mesh simplification and tessellation criteria: (a) accurate patch subdivision, (b) subdivision limitation due to mesh simplification.

Geometric simplification algorithms are used to minimize the amount of geometry that has to be processed when computing global illumination solutions. Very complicated geometry or large clusters of geometry are replaced by simpler pieces (Figure 5.3) that are in some sense radiometrically equivalent (same average reflectance, etc.).

A number of methods can be used to simplify the complex environment to a radiometrically similar but much smaller number of surfaces for use in the indirect illumination simulation. We have tried two methods. The first uses a simple spatio-temporal visibility criterion. For each of some number of trials, a random time is chosen. For the true environment configured for the appropriate time, random view configurations within the environment are chosen. For each view, a large number of rays are shot through a wide viewing angle. A record is kept of how many times each object is hit in this process. Based on the number of hits, an object may be ignored — a very small number of hits, simplified — somewhat larger number of hits, or restricted in subdivision — large number of hits, in the indirect illumination solution. The second method uses an area/volume limit criteria and prunes surfaces comparing their size to the average size of the environment. Also incorporated in this method is a tessellation criteria that allows us to simplify geometric primitives that pass the area/volume limit criteria into a polygonal mesh (Figure 5.4). Both of these methods provide a simple mechanism for reducing the complexity in the environment to a reasonable size so that it can be efficiently processed.

5.3.3 Wavelet radiosity - hierarchical indirect illumination

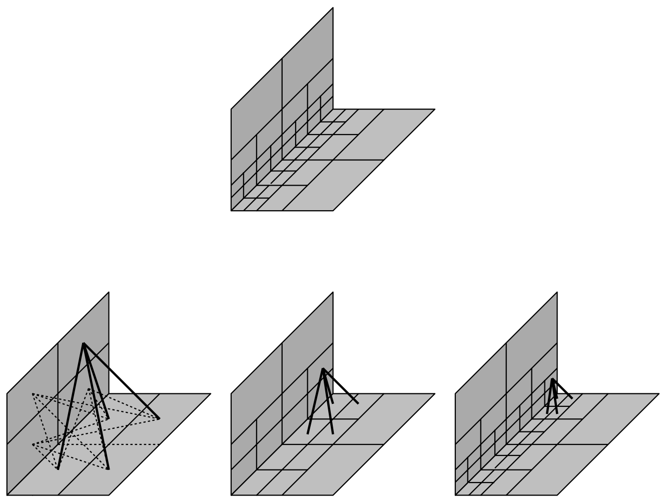


Figure 5.5: Hierarchical decomposition.

Once the environment is simplified, the indirect illumination simulation can take place. Our indirect illumination algorithm is based on the hierarchical radiosity work (Figure 5.5) of Hanrahan et al. [139] and the wavelet radiosity work of Gortler et al. [122] (Chapter 2) but has been modified to separate out the indirect component of the global solution. The wavelet solver separately keeps track of the illumination that has arrived from emitters and from intermediate surfaces and can return the complete solution, the direct component, or the indirect component. We use the indirect component asking the algorithm to return a sparse set of indirect irradiance values on the surfaces of objects in the environment. Since the direct illumination algorithm uses ray tracing and also computes radiance on the surface boundaries of objects this is not a limitation. We do, however, want to use a dense set of surface radiance values to be able to generate accurate imagery. This requires a mechanism for spatial interpolation of the indirect illumination values across the interiors of objects/patches in the environment.

5.3.4 Ambient field reconstruction

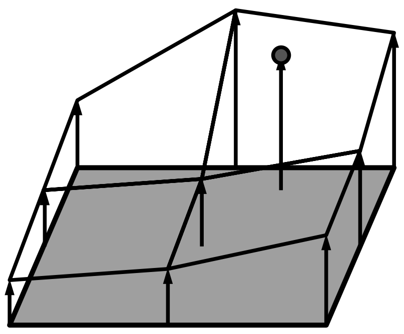


Figure 5.6: Spatial indirect illumination reconstruction.

Once we have an indirect illumination solution for our environment we want to be able to reconstruct an ambient field, dense indirect irradiance solution, for the environment to be used as input to the direct illumination algorithm.

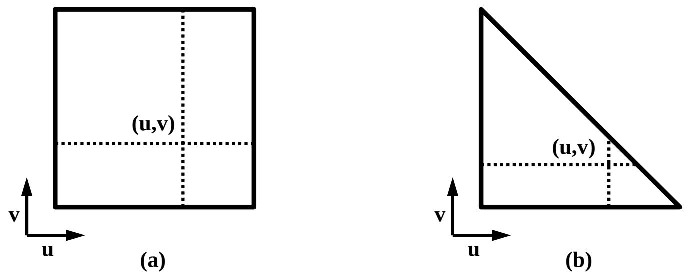


Figure 5.7: Reconstruction domains: (a) rectangular, (b) triangular.

Ambient field construction (Figure 5.6) is performed using nearest-neighbor techniques. Nearest-neighbor interpolation across the patches in object space is used to blend vertex radiances into the interior of the regions they bound. Bilinear reconstruction can be employed to interpolate within quadrilateral regions, and barycentric interpolation can be used to interpolate within triangular regions.

A bilinear interpolant [337] can be used to map the unit square in two-space onto an arbitrarily oriented quadrilateral in three-space (Figure 5.7a),

$$[x, y, z, Radiance] = [uv, u, v, 1] \begin{bmatrix} x_3 & y_3 & z_3 & Radiance_3 \\ x_2 & y_2 & z_2 & Radiance_2 \\ x_1 & y_1 & z_1 & Radiance_1 \\ x_0 & y_0 & z_0 & Radiance_0 \end{bmatrix} . \tag{5.1}$$

For a dense random sampling of $(0.0 \leq u, v \leq 1.0)$ we can compute the position of the sample in space (Equation 5.1) as well as its radiance value.

A barycentric interpolant [79] can be used to map a reference triangle in two-space onto an arbitrarily oriented triangle in three-space (Figure 5.7b),

$$[x, y, z, Radiance] = [s, t, 1 - s - t] \begin{bmatrix} x_2 & y_2 & z_2 & Radiance_2 \\ x_1 & y_1 & z_1 & Radiance_1 \\ x_0 & y_0 & z_0 & Radiance_0 \end{bmatrix} . \tag{5.2}$$

As with the bilinear interpolant, for a dense random sampling of $(0.0 \leq s, t \leq 1.0)$ we can compute the position of the sample in space (Equation 5.2) as well as its radiance value.

In an effort to be more useful to the direct illumination algorithm, the spatial information returned in the ambient field can be augmented by translational and rotational illumination gradient information. Following the reasoning in [325], a mesh-based radiosity algorithm can compute a

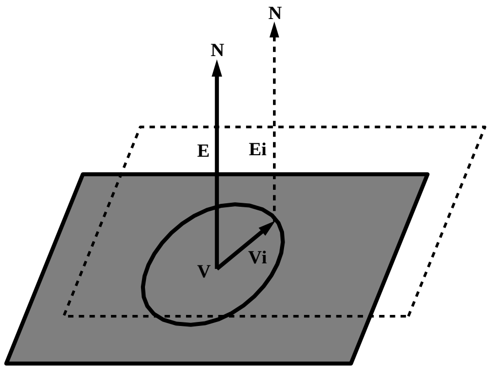


Figure 5.8: Translational gradient sampling.

simplified translational gradient (Figure 5.8) for each patch vertex, $\mathbf{V} = (x, y, z)$ with normal $\mathbf{N}$, by:

1. Computing indirect irradiance, E, at the patch vertex.

2. Choosing a set of normalized translation directions, $\mathbf{V_i} = (x_i, y_i, z_i)$, for the patch vertex (in the tangent plane containing the patch).

3. Moving the patch vertex along each of these directions, $\mathbf{V} + \mathbf{V_i} = (x + x_i, y + y_i, z + z_i)$.

4. Computing the irradiance, E_i, at each of these locations (using normal $\mathbf{N}$).

5. Computing the irradiance differences, $\Delta E_i = |E - E_i|$.

6. Computing the translational gradient vector by taking a weighted summation:

$$\vec{\nabla}_{trans} V = \sum_i (\Delta E_i) V_i \quad .$$

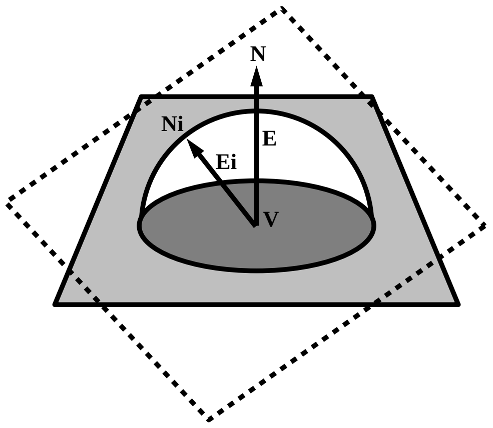


Figure 5.9: Rotational gradient sampling.

A simplification of the rotational gradient (Figure 5.9) can be computed at each patch vertex $\mathbf{V} = (x, y, z)$ with normal $\mathbf{N}$, by:

1. Computing indirect irradiance, E, at the patch vertex.

2. Choosing a set of normalized orientation directions directions, $\mathbf{N_i} = (n_i, n_i, n_i)$, for the patch at the vertex.

3. Computing the irradiance, E_i, for each of these new patch orientations (using $\mathbf{N_i}$).

4. Computing the irradiance differences, $\Delta E_i = |E - E_i|$.

5. Computing the rotation gradient vector by taking a weighted summation:

$$\vec{\nabla}_{rot} V = \sum_i (\Delta E_i) N_i \quad .$$

5.4 Direct Illumination - Radiance

Once the ambient field for the environment has been constructed it is passed into the direct illumination algorithm, *Radiance* [324], which computes the total illumination in the environment reflected back to the viewer.

The *Radiance* lighting simulation package uses a combined deterministic/Monte Carlo ray tracing approach to achieve accurate and controllable in a reasonable amount of time. Rays of light are traced backwards from the viewpoint to the sources/emitters. The computation of total illumination is divided into three components which are calculated separately: the direct illumination component, the specular indirect component, and the diffuse indirect component. In our implementation, we replace the computation of diffuse indirect illumination with our pre-computed ambient field.

The direct component is made up of illumination arriving at a surface directly from light sources or via one or more perfectly specular transfers, reflections or transmissions, from other surfaces. The list of emitting surfaces is sorted based on potential contribution to the environment and used to minimize the number of rays required for visibility testing. Monte Carlo sampling is coupled with the adaptive subdivision of large sources to provide a mechanism to capture accurate penumbra. Specular transfers from large planar surfaces are handled efficiently through the use of *virtual* light sources, which guide the direct calculation to the original sources.

The specular indirect component consists of light arriving at a surface from other surfaces and being reflected off or transmitted through in a directional manner. Perfectly specular transfers are handled by simply redirecting the ray in the appropriate reflected or transmitted direction. Rough specular transfers are modeled with Monte Carlo sampling of the reflected and/or transmitted direction.

We replace the computation of the diffuse indirect component, that light which arrives from an intermediate surface in a non-directional manner, with our pre-computed ambient field. Computing non-directional indirect illumination takes a prohibitively large amount of rays to perform accurately and is the most difficult task for a ray tracing algorithm. The pre-computation of this indirect illumination gives us a big savings in overall rendering time, especially when an environment requires many different renderings.

Radiance is a very complex program containing many possibly conflicting parameter settings, and can be difficult to use. We augmented *Radiance* with a utility that converts from a simple

Radiance Parameter Lookup Table					
Parameter	Description	Minimum	Low	Medium	High
-ps	pixel sampling rate	16	8	4	1
-pt	sampling threshold	1	.15	.05	0
-pj	anti-aliasing jitter	0	.6	.9	1
-dj	source jitter	0	0	.7	1
-ds	source substructuring	0	.5	.15	.02
-dt	direct thresholding	1	.5	.05	0
-dc	direct certainty	0	.25	.5	1
-dr	direct relays	0	1	3	6
-dp	direct pretest density	32	64	512	0
-sj	specular jitter	0	.3	.7	1
-st	specular threshold	1	.85	.15	0
-lr	limit reflection	0	4	8	16
-lw	limit weight	.05	.01	.002	0

Table 5.1: *Radiance* parameter lookup table (Greg Ward).

quality parameter $(0.0 \leq q \leq 1.0)$ to a set of "empirical" settings that have been shown to work well in the past. Our mapping is a piece-wise interpolant, either linear or cubic, based on the data contained in (Table 5.1).

5.5 Results

In this section we evaluate the time and quality tradeoffs involved in using our multi-stage illumination method when compared to a set of reference images. Since we are concerned with constructing images that are as perceptually accurate as those generated by traditional multi-pass simulation algorithms, we use image space comparisons to determine the quality of different solutions. Solutions computed with the *Radiance* system computing both indirect and direct illumination are used as reference images. These settings have been determined empirically over the lifetime of the program and often lead to the best time/quality ratio in the solutions that are generated. Although it is possible to compute full Monte Carlo reference images, these images are prohibitively expensive to generate.

5.5.1 Metric

Now that we have chosen the form of our reference solution, and the space in which we will compare our solutions, a suitable metric used to compute the error must be chosen. We use relative luminance RMS error, luminance RMS/reference solution average luminance, as a metric throughout the rest of this thesis. More sophisticated metrics are being researched [258] but they often rely on psycho-visual research [40] which has yet to be fully formalized and verified. *Radiance* generates real-valued images with pixel samples having the units of luminance [323]. This image format provides a greater dynamic range for interpreting the error than the traditional RGB format.

5.5.2 Geometric simplification

Geometric simplification can yield a significant reduction in the computation of indirect illumination with minimal loss in image fidelity. The problem is knowing how much simplification can be performed before a noticeable error has been introduced.

We used a model of a studio environment that contains a large number of surfaces and a set of varied surface BRDF's to test the effectiveness of our geometric simplification module. The model includes approximately 70,000 surfaces and has many windows that admit sunlight. A very challenging view has been selected in which indirect illumination from the sunlight reflected off the vertical surfaces is very important.

We simplified the studio model using object pruning and decomposition. A user-defined quality parameter was used to calculate the minimum area/volume requirement for primitives and the tessellation detail level specifier that is used to decompose those primitives that meet the area/volume requirement. For this analysis we simplified the studio environment using low, medium, and high quality settings. We then ran the wavelet radiosity (WR) solver on each of the environments to generate the indirect illumination solutions. These solutions were converted to ambient fields and then used to generate the final solution images in *Radiance* . We compared these three images to an image generated by *Radiance* , using both typical and high quality settings, using the relative luminance RMS metric.

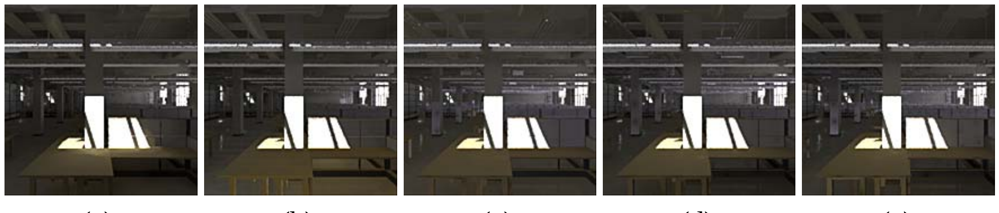


Figure 5.10: Comparison of studio images generated using geometrically simplified indirect illumination solutions: (a) Radiance HQ, (b) Radiance typical, (c) Wavelet radiosity low, (d) Wavelet radiosity medium, (e) Wavelet radiosity high.

Studio				
Scene Decomposition				
Quality	Initial Surfaces	Pruned Surfaces	Final Surfaces	Time (secs)
Low - .3	68718	54710	14008	32.33
Medium - .6	68718	37120	31598	38.41
High - .9	68718	2616	66102	34.95

Analysis		
Image	Computation Time (hours)	LRMS Error/Rad. HQ Avg. Lum.
Radiance HQ	42.201	0.000/5.468 = 0.000
Radiance Typical	23.719	1.967/5.468 = 0.360
WR Low Quality	18.240	2.081/5.468 = 0.381
WR Medium Quality	26.867	1.858/5.468 = 0.340
WR High Quality	35.960	1.853/5.468 = 0.339

Table 5.2: Multi-stage rendering statistics for the studio.

5.6 Analysis

The results (Table 5.2 and Figure 5.10) of this analysis are very promising. The low and medium quality indirect solutions are comparable to the *Radiance* typical solution and take a similar amount of time to compute. The advantage of our approach is that once the view-independent indirect solutions are computed, they need not be re-computed as long as the environment is unchanged. This is very important when sequences of frames are generated, as the time for computing solution sets from scratch in *Radiance* grows approximately geometrically, indirect and direct illumination calculated on a per frame basis. In our framework the indirect component is computed once and the direct illumination is added in on a per frame basis, yielding a significant overall savings. Another advantage of our method is that since we compute the indirect component in object space, with the resulting values being attached to surfaces, we have the ability to investigate spatio-temporal algorithms that would otherwise be intractable.

5.7 Summary

In this chapter we presented a new rendering algorithm designed to compute global illumination within complex environments. The method we proposed is multi-stage and decouples the processing of the different illumination modes by the importance of the visual effects they compute to the overall presentation of the environment. Our method is designed to be flexible allowing accuracy to be traded for efficiency and vice-versa.

An object space radiosity algorithm is used to compute view-independent indirect illumination. Since indirect illumination contains the interreflection effects that add to the "feel" of the environment, it cannot be eschewed. These soft effects can, however, be overwhelmed by direct lighting so they should be able to be calculated to their own level of accuracy, treated separately from direct illumination. The rendering method needs to flexible. With this in mind we compute a sparse indirect illumination solution using a geometrically simplified version of the environment. The simplified environment and the sparse sampling density elicits an indirect solution that is in some sense "filtered" but the increase in performance is often worth the error that is introduced.

To compute an image of the environment at a particular time using a particular view configuration, the view-independent indirect illumination information is converted into an ambient field in which the true environment, placed at the appropriate time, is embedded. An image space direct illumination algorithm, using the current view configuration, is then executed with the indirect illumination value at each intersection location being used to augment the direct illumination value that is calculated. The direct illumination algorithm uses a variant of deterministic and non-deterministic/Monte Carlo ray tracing to perform its task. Each illumination value is computed as the combination of a different quality indirect component and direct components with the computational savings being fine-tuned so as to outweigh imperceptible losses in accuracy.

Chapter 6

Approximating the Time-Varying Radiance Distribution Using Range-Image Sequences

“Whoever controls the media – the images – controls the culture.”

Allen Ginsberg

In this chapter we present a method for representing the radiance variances in an environment that treats the spatial, directional, and temporal degrees of freedom separately. It is based on work presented at the Sixth Eurographics Workshop on Rendering in Dublin, Ireland [217] and in the IEEE Transactions of Visualization and Computer Graphics [218]. The radiance distribution within an environment is represented as time sequences of range-images, images and depth maps, stored at important positions in the environment. The important locations and times are selected using a set of adaptive coherence measurements.

We introduce an object space decomposition algorithm that harnesses the power of our multi-stage rendering algorithm (Chapter 5) to efficiently track indirect illumination coherence and compute the necessary indirect illumination solutions. Augmenting this algorithm is a temporal indirect illumination interpolation method that allows us to represent time-varying indirect illumination as a set of spatio-temporal object space interpolants.

We then introduce image space algorithms that quickly evaluate spatial, directional, and temporal direct illumination and visibility coherence without computing any illumination information. These coherence algorithms are designed to produce a “recipe” that tells us which points of view, positions and directions, in the environment are important, and which times need to be accounted for in order to adequately represent the radiance distribution using image-sequences. Once the recipe has been constructed, the range-image sequences are produced by interpolating from the pre-computed indirect illumination solutions, and refining shadows, highlights, and specular lighting effects using the direct illumination phase of our rendering method.

After the range-image sequences have been produced, we use simple and efficient algorithms for reconstructing arbitrary views (image-based temporal and view interpolation) of the environment at any desired times. We finish the chapter with an implementation, and present results and an analysis that demonstrate the feasibility and effectiveness of this type of radiance distribution approximation in complex, animated environments.

6.1 Motivation

The radiance distribution (Chapter 4) is an abstraction that encapsulates all the information needed to visualize a particular environment. No geometric information is necessary once we have a time-varying representation of the flow of radiance within the desired space. Since the distribution is very complex the problem, again, is deciding which portions of the distribution need to be accurately represented and which can be “approximated.” We also have to decide how to produce the necessary radiance values. Our rendering method from last chapter was designed to efficiently produce radiance values, computing complete images of the globally illuminated environment at a desired time using the desired view configuration, but does not provide a mechanism for deciding, for an arbitrary environment, which view configurations and times are “important.” Choosing these important areas of the radiance distribution is a difficult problem, one that is glossed over in many image-based applications [60, 58, 193, 270].

Relying, like our rendering method does, on traditional algorithms to produce radiance values elicits a separation of the spatial, directional, and temporal portions of the radiance distribution.

This separation is analogous to the metaphor used in Chapter 4 which defines the radiance distribution. We can represent an environment by the radiance seen by a set of hypothetical viewers/cameras located at every position in the environment, looking in every possible direction, over all times. Parameterizing each of dimension yields the radiance distribution function. Since it is impossible to process all the information required to match our analogy, we instead relax it. Instead of requiring an infinite set of hypothetical viewers to be watching all directions at all times, we can imagine a fixed set of spherical (360° field-of-view) video cameras recording what goes on in the environment. We also need to modify our cameras so that they are sensitive to motion, turning on just in time to capture object motion, but do not "waste tape" when nothing is going on. Capturing the radiance distribution for an environment now becomes the problem of placing "smart" video cameras within the environment and letting time (the animation) progress normally.

In computer graphics, video sequences are represented as sets of images, so our analogy codifies an image-based approach to approximating the radiance distribution. It turns out that if we extend our representation from simple radiance images to radiance images with depth (range-images), the approach is not just feasible but efficient as well. While previous research has demonstrated the potential of range-image systems to allow a user to tour a complex, static environment [34, 60, 58, 193, 270] at interactive rates, the problem of processing time-varying environments in the same context has not yet been considered extensively. We build upon previous work by considering how the radiance distribution for complex, animated environments (i.e. environments in which objects as well as the user can move) can be represented as time sequences of range-images. We explore how to select a set of base views (video camera positions) for the range-images as well as the time steps required to capture the effects of object motion (temporal lighting variances). Further, we consider how time-varying global illumination can be efficiently computed to generate each of the range-images utilizing the power of our multi-stage rendering method. Previous global illumination methods have successfully exploited spatial coherence by separating the calculation of direct and indirect illumination [326, 59]. We build on this idea and exploit temporal coherence as well by separating the calculation of temporal variations in direct and indirect illumination.

6.2 Why Use Images? Why Not!

As mentioned in Section 3.2.3, image-based systems have the advantage of very efficiently representing geometrically complex environments. This advantage over polygon-based systems is compounded when we wish to represent a photometrically accurate version of the environment. This advantage can be seen by referring again to Figure 2.6.

As shown in Figure 2.6a, for a photometrically accurate radiosity solution, we must compute a representation of the directional radiance distribution $L(\theta, \phi)$ for all points (x, y, z) on every object. As shown in Figure 2.6b, in an image-based system we need only to compute and store the radiance for the small number of points and directions for each object visible in each image.

The advantage of the image-based approach over radiosity-style approaches for photometrically accurate scenes extends even further when allowable error in the solution is considered. For any global illumination solution, the computation time can be drastically reduced by allowing some known error level in the results, rather than attempting to compute results to machine precision [139]. *But what is an allowable error level?* The allowable error depends on viewer perception, not on the characteristics of the object. In a radiosity solution, a high degree of accuracy is required, because the view of the object is unknown. The “worst case” must be assumed. Perceptual error metrics are inherently image based, since the accuracy required for a particular solution depends on the radiance distribution in the visual field [40, 258]. In an image-based system then, much better estimates can be made of allowable error in the radiance solution, and the solution can be computed much more efficiently. In this initial radiance distribution approximation, the global illumination solution will be computed in the form of sets of images (range-images), each of which can be interpolated to produce a frame at each time step. The user will then be able to move freely within the space spanned by sets of range-images.

6.3 Exploiting Temporal Coherence in Global Illumination

As discussed earlier (Section 3.1.2), several researchers have proposed object space solutions that efficiently exploit the temporal coherence in global illumination variations as objects move. In these approaches, the location and radiance distribution is stored for each object. As the observer moves through the scene (Figure 6.1a) objects are placed where they should be at the appropriate time and projected on to the view with the newly computed radiance value. The advantage of this approach is that the global illumination for each time step can be incrementally computed by exploiting object space coherence. That is, a completely new global illumination solution is not needed for every frame. The disadvantages are the pre-computation time and storage space required as the number of objects becomes very large.

In a temporally varying range-image based system (Figure 6.1b), we move through time by interpolating between images in a time series for each base view. In this case, we are producing frames that represent a full global illumination solution, taking into account both diffuse and non-diffuse illumination. Radiances do not need to be stored for every object for every time.

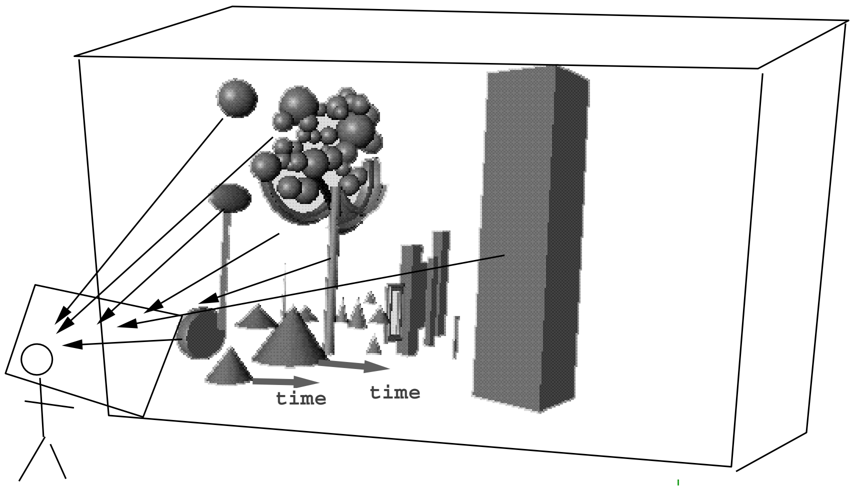
6.3.1 Coherence

Illumination and visibility variances come about from both object and viewer motion. We want to be able to track when these variances occur without having to compute illumination information (using the expensive global illumination algorithm). This requires understanding the fundamental causes of illumination variances within an environment so that we might find other mechanisms for tracking them. A good analogy is the shadow volume algorithm [338] which finds the shadow regions from geometric relationships without requiring the rendering algorithm to be executed. The algorithm might not be as accurate and elegant as using ray casting to compute shadow boundaries, but can be executed more efficiently since no lighting information is needed. We want to use the same insight for all forms of radiance variances.

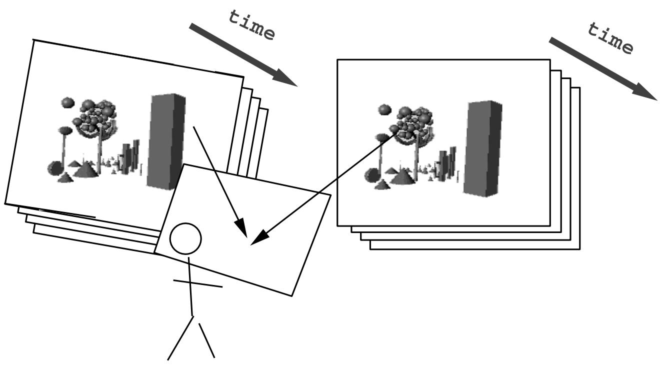
6.3.2 Direct illumination and visibility.

Direct illumination variances (Figure 6.2) occur when objects or light sources within the environment move over time. These variances can either be discontinuous (moving shadows, specular highlights), or continuous (increases/decreases in diffuse lighting). Since the human visual system is good at tracking moving edges [117], we are most concerned with finding discontinuities (shadow and highlight boundaries).

Visibility variances (Figure 6.3) are discontinuities that occur relative to the view configuration over time. Objects appear and disappear from the field-of-view, or occlusion amongst objects changes. These types of variances are very disconcerting which is why high temporal sampling



(a)



(b)

Figure 6.1: The effects of object motion are stored differently in object space and image space systems. In an object space approach, (a), motion is displayed by reprojecting objects in their new positions as time increases and either recomputing or looking up the radiances for those objects at that time. In an image space approach (b), motion is displayed by interpolating the radiances in a time series of base range-images, and morphing the range-images to produce the appropriate view.

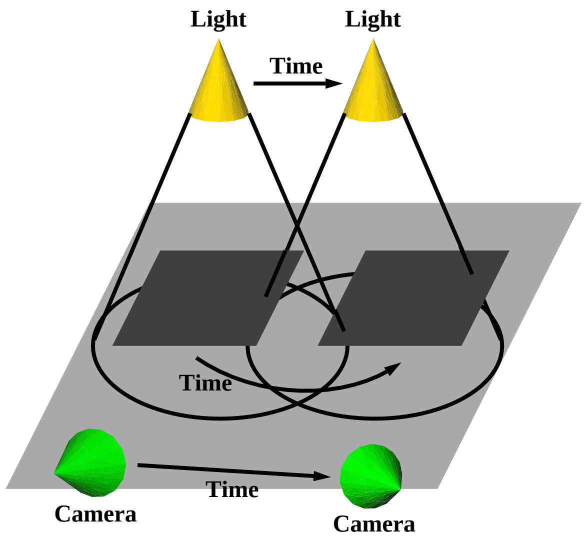


Figure 6.2: Direct illumination variances such as changing overall illumination, moving shadow boundaries, and moving specular highlights can be caused by light source motion, object motion, and camera motion.

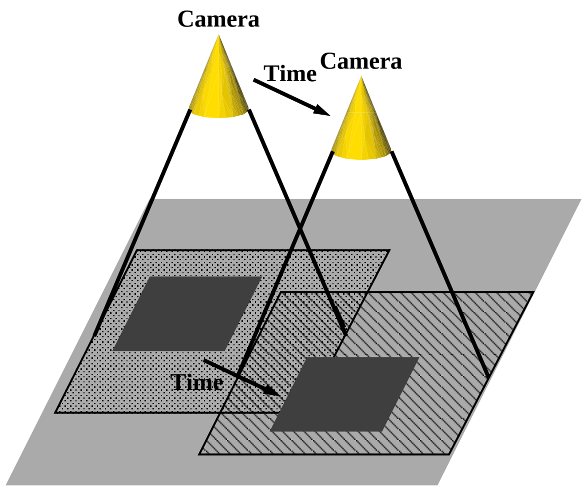


Figure 6.3: Visibility variances (occlusion changes) can be caused by, object motion (including light sources), and camera motion.

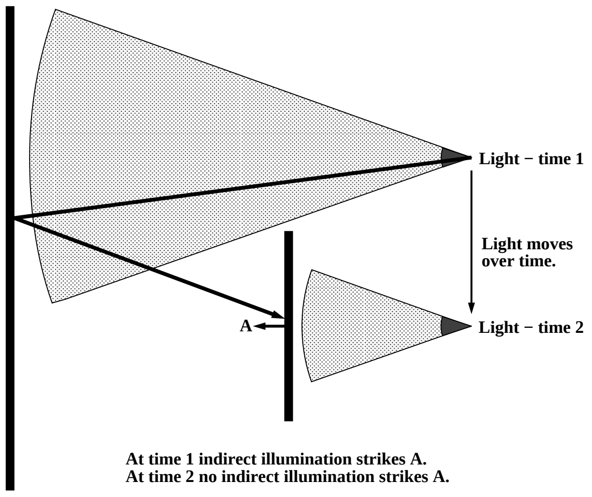

Light − time 1
Light moves over time.
A
Light − time 2
At time 1 indirect illumination strikes A.
At time 2 no indirect illumination strikes A.

Figure 6.4: Indirect illumination variances can be caused by object motion: the point A is indirectly illuminated at the early time, but not at the late time.

rates (frame rates) are used when capturing motion effects for traditional animations. Since we desire quality for our walkthroughs that rivals frame sequences computed with a static view configuration, visibility variances have even a higher importance.

Relatively standard techniques for ray tracing animations can be used to exploit the coherence of direct visibility and shadowing in the base range-images. Rather than computing images for every $1/30$ sec, the time steps for these base images can be determined by detecting the amount of geometry or shadowing change over longer lengths of time. Even with this reduced number of base-time images, how can we avoid a pixel by pixel recalculation of the global illumination for each time step? As in the temporal radiosity methods we seek to exploit the temporal coherence in the full global solution to reduce the calculations.

6.3.3 Indirect illumination.

Like direct illumination variances, indirect illumination variances (Figure 6.4) can also be discontinuous. It turns out that these discontinuities occur in the higher derivatives of the illumination function [143, 7] and are less objectionable to the average viewer. Since our rendering method treats indirect illumination as being completely diffuse, using a radiosity algorithm, we can track the variances in object space. Tracking the indirect illumination discontinuities in object space

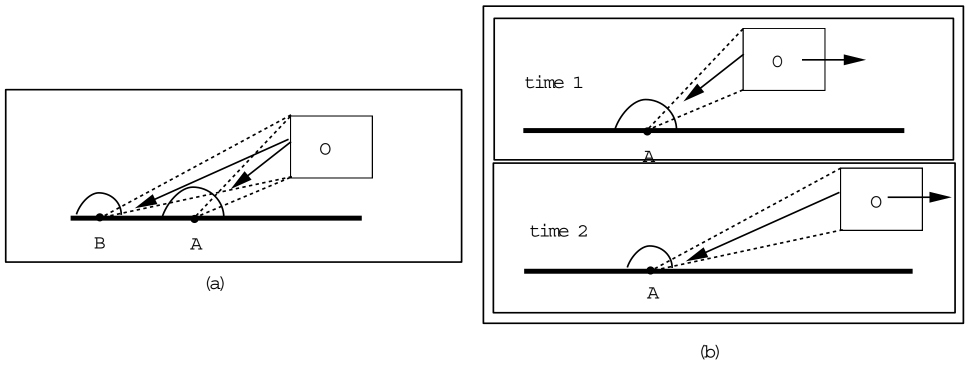


Figure 6.5: Figure (a) illustrates that the indirect illumination of point A by object O is only slightly different than the indirect illumination of point B by object O. Figure (b) illustrates that the difference in indirect illumination of point A caused by the movement of object O from time 1 to time 2 is the same as the difference in indirect illumination between A and B in the static case.

possesses a dual method that allows us to interpolate between the indirect solutions once the discontinuities are found, constructing a piecewise-linear illumination interpolant.

As noted in Section 2.3.3 and Chapter 5, hybrid approaches exploit spatial coherence by computing indirect illumination effects using relatively sparse spacing in object space and interpolating between the samples. We can use the same sparse sampling of object space to exploit temporal coherence. Figure 6.5 illustrates the relationship between sampling indirect illumination in time and in space. Consider Figure 6.5a. For a static environment, we can sample indirect illumination at points A and B and interpolate between them, because the effect of object O on diffuse or near-diffuse reflection varies continuously between A and B. The amount of light from O that is reflected from A is slightly higher than the amount reflected from B because O subtends a larger solid angle from A, and because the angle of O to the surface normal is slightly lower at A.

If object O is moving (Figure 6.5b), indirect illumination at point A at times “time 1” and “time 2” varies in the same manner that the indirect illumination varied with position between A and B in the static case. At “time 2” the light A reflects from O is a bit less because the solid angle subtended by O is smaller, and the angle of O to the surface normal has increased.

Because the changes in indirect illumination resulting from object motion are equivalent to the changes in indirect illumination as a function of distance in the static case, we can sparsely sample indirect illumination in time as well as in space.

Our approach then is to compute indirect illumination for sparsely separated points for very large time steps. Interpolation between these solutions will then be used to compute the indirect illumination for the time series of range-images at each base view point.

6.4 The Whole System

The overall framework of our approach is shown in Figure 6.6. The indirect illumination is sampled sparsely in time and space in object space (top row). The indirect illumination solution is then interpolated and used as the basis to produce the full global illumination for each of the base images (middle row). Finally, for any path in the chosen region of the environment space, images are generated for each frame by interpolating the base images.

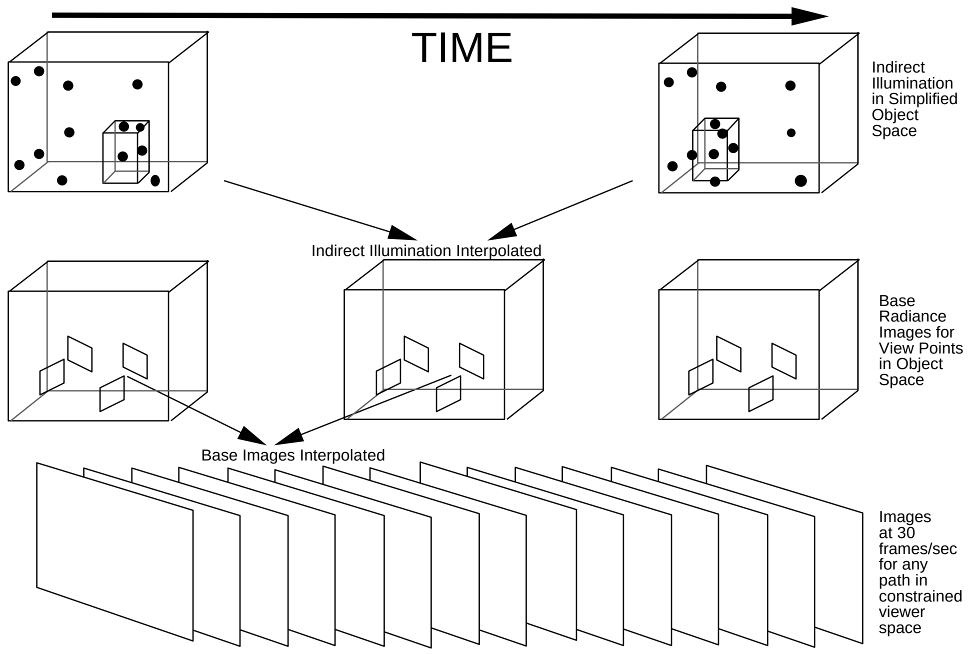

TIME
Indirect Illumination in Simplified Object Space
Indirect Illumination Interpolated
Base Radiance Images for View Points in Object Space
Base Images Interpolated
Images at 30 frames/sec for any path in constrained viewer space

Figure 6.6: Indirect illumination is sampled sparsely in time and space in a simplified, animated object space. Radiance images are computed more frequently in time for selected base views, using interpolated values from the indirect illumination solution. One image per frame is computed for any path in the chosen area of the environment by interpolating base images.

To build the system diagrammed in Figure 6.6 we need the following:

1. Rules for selecting the base view (camera) positions so that the entire space a user may wish to tour is spanned (decomposing space).

2. Rules for selecting the times steps at which base views are computed so that motions are

adequately represented/captured (decomposing time - view dependent direct illumination coherence).

3. Rules for selecting the points and times (where and when to sample) for which indirect illumination is to be computed (decomposing time - view independent indirect illumination coherence).

Once these views and times are chosen, the range-image sequences are created using our multi-stage rendering method. Temporal and spatial reconstruction techniques are then applied to quickly produce the desired views of the environment and the desired times.

The structure of our initial implementation is shown in Figure 6.7. As input, we have the environment description, the specification of objects and the view space. Locations for the base views are found by adaptive subdivision. The initial time discretization is estimated by accounting for direct lighting variances. The time discretization for each view is then further refined. In parallel to the establishment of base views, the sparsely sampled (geometrically simplified) indirect illumination solutions are calculated. The indirect illumination solutions are then interpolated and used to compute the base image-sequences.

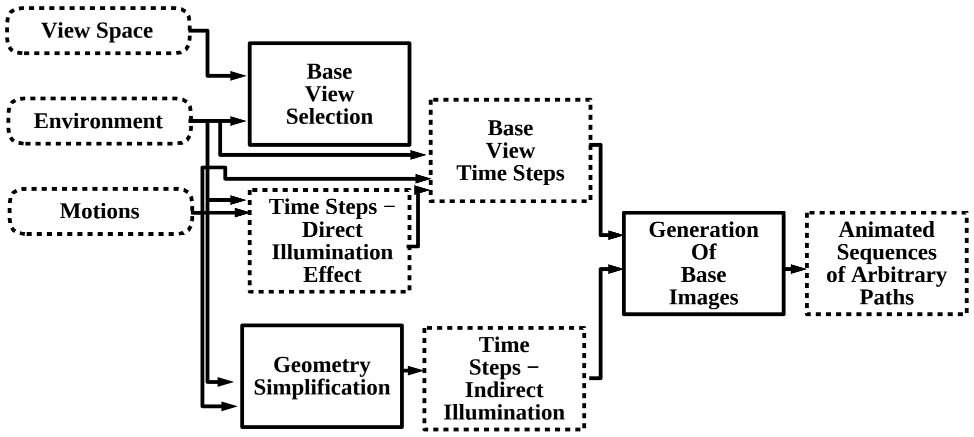

View Space
Environment
Motions
Base View Selection
Base View Time Steps
Time Steps − Direct Illumination Effect
Geometry Simplification
Time Steps − Indirect Illumination
Generation Of Base Images
Animated Sequences of Arbitrary Paths

Figure 6.7: Overview of initial implementation.

6.5 Decomposing Space - Motion Independence

Taking a top-down view of the problem, we first want to decide where to place our cameras to be able to best capture spatial variability in the environment. These changes exhibit themselves

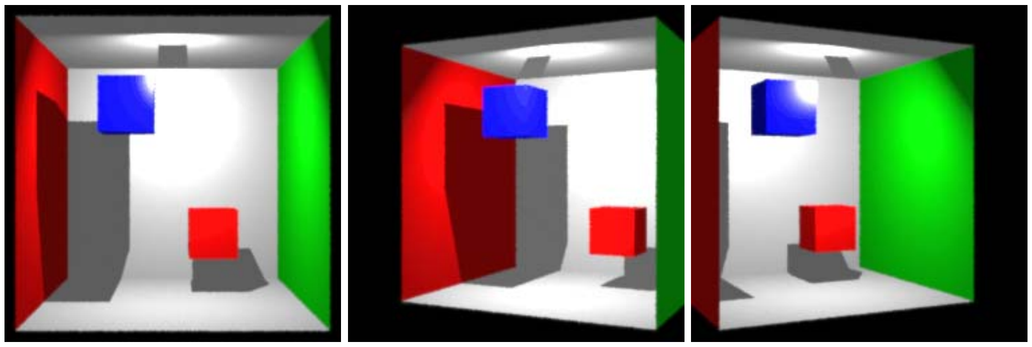

(a) (b) (c)

Figure 6.8: Three views of the same environment showing different visual information.

as visibility and illumination differences as we walk through the environment. A path through the scene that exhibits minimal differences in visibility and illumination is considered to be adequately represented by a combination of the information contained at the endpoints of the path.

Since it is infeasible to test the visibility coherence of arbitrary paths through the chosen environment, an infinite number of possibilities, we propose isolating the coherent regions of the environment using spatial subdivision driven by a coherence metric. The coherence metric is used to test visual similarity in the information seen from different spatial locations and should be computable without the need to execute the expensive global illumination simulation (Figure 6.8). In some sense we want a coherence metric that can warn us about changes in the illumination and visibility without requiring us to compute any illumination. We bound the spatial subdivision algorithm by introducing the idea of a *view space*. A view space is a region of the environment where the user would like unlimited freedom of movement. The boundaries of the view space are used as the limiting values for our subdivision algorithm.

Our spatial subdivision algorithm takes as input the view space of interest and a particular coherence metric. The boundary of the view is tested using the coherence metric. If the endpoints see similar information, are coherent, the region bounded by the endpoints is considered coherent. Subdivision is performed on those regions that do not pass the coherence test. The endpoints that are left once the coherence criteria has been met by all the subregions being tested, form a network of spatial information that will be used to reconstruct the spatial information for intermediate points in the view space.

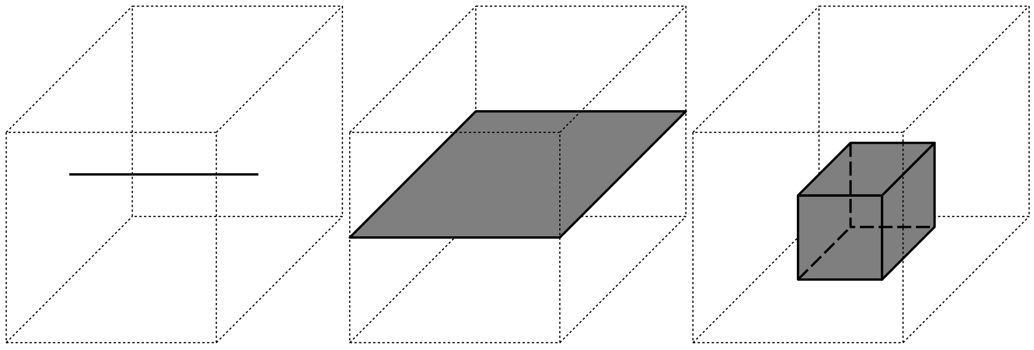


Figure 6.9: View space geometries: (a) one dimensional, (b) two dimensional, (c) three dimensional.

6.5.1 View space definition

The view space S is a set of view locations that is some subset of the full environment. The view space may be $1D$, $2D$, or $3D$ (Figure 6.9), allowing the user to move along a line, within a plane or through a volume respectively.

Ideally, to allow the user to have a free range of movement within S, the view for the entire sphere of directions should be stored, (Figure 6.10, left). To simplify certain types of walkthroughs, hemispherical fish-eye views (Figure 6.10 right) can be used. This restricts the set of paths within the view space to those that are encompassed by the directions and fields of view available within the hemispherical base views. When sampling the sphere of directions, a pattern that samples view directions nearly uniformly (i.e. avoiding concentrations of samples at the "poles") is used. Two hemispherical cameras with opposite view directions can be used to test the sphere of all possible viewing directions.

6.5.2 Base view selection - adaptive subdivision and visibility coherence

The base view locations in S are determined by adaptive subdivision. The subdivision level is controlled by a pre-selected quality parameter q, which ranges from zero to one. A value of q equal to zero will result in no subdivision returning the boundary elements of the space as the base locations, a value of q equal to one will result in a continuous, infinitely dense, population of view locations.

The subdivision begins by defining synthetic cameras at the corners of the view space. As

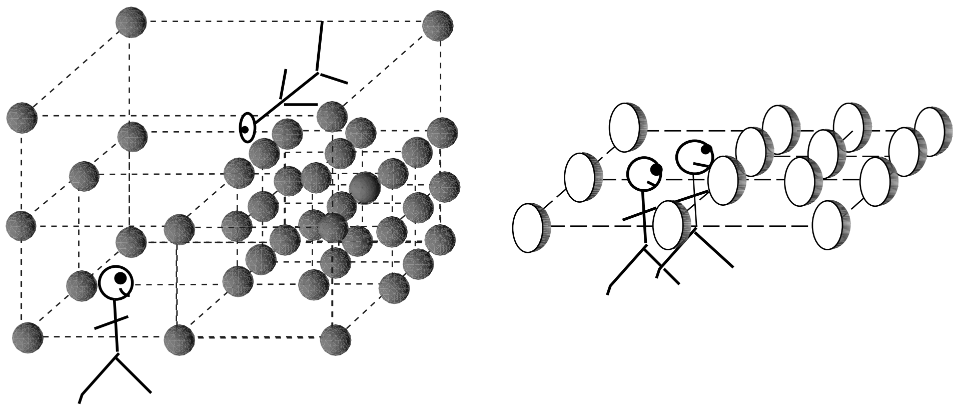


Figure 6.10: A viewer has maximum freedom in a $3D$ view space with full views stored (left). Hemispherical views can also be used. The positions of these base views can be computed using an adaptive subdivision algorithm.

opposed to using the global illumination algorithm, a P by Q resolution *id-image* (Figure 6.11) is formed for each camera. In an id-image the pixel values are unique numerical identifiers for the objects visible at the pixel. Depending on the geometry of the objects, the id-image could be formed by a variety of techniques (i.e. scan-line, ray tracing, etc.). In our implementation we use the SGI display hardware to form the image, by assigning a unique 32-bit color to each polygon in place of its physical color. The number of pixels N that have the same id for all of the cameras is counted. The "fitness" f of this set of locations is computed as N/PQ. If f is less than the quality parameter q, the space is subdivided, and the test is applied recursively. There are a set of drawbacks associated with using the graphics hardware to implement this approach. First, the resolution and field-of-view of our cameras are limited in a hardware based approach. Second, sub-pixel accuracy is not possible as the graphics hardware would "average" the colors (object identifiers) seen through the same pixel, leading to erroneous results. Finally, there can always be a degenerate configuration of objects in the environment that an adaptive subdivision scheme cannot handle well, causing oversampling where it might not be necessary. Different coherence metrics might help to alleviate this problem, but it is unknown to the author at this point what those metrics might be.

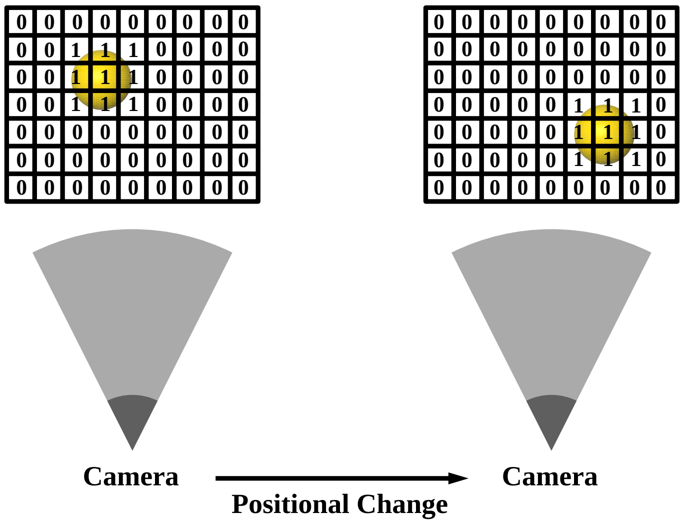


Figure 6.11: Base view selection: id-images are computed after cameras have been placed at the boundary of the view space.

6.6 Decomposing Time - Temporal Illumination Coherence

Capturing the temporal information inherent in the environment can also be performed using a coherence-based approach. We track the temporal changes in the environment by independently monitoring the changes in indirect illumination, and direct illumination and visibility at the chosen base view locations. Indirect illumination, the light that impinges on surfaces after reflecting off intermediate surfaces, has been shown to vary less rapidly than direct illumination [7]. We exploit this by decoupling indirect and direct illumination and sampling indirect illumination more sparsely. The idea of sampling indirect illumination more sparsely than direct illumination leads to what we have termed the *temporally decomposed illumination model*. This model affords us the opportunity to investigate reconstructing indirect illumination in object space prior to adding in the direct component as is typically done in today's multi-pass methods.

6.6.1 Indirect illumination coherence

If we were to monitor the surface radiance values over time and compare corresponding values we would gain insight into the temporal coherence of the indirect distribution of illumination. A low variability signifies a scenario where the indirect illumination may be adequately represented with a small number of radiance values stored with each patch vertex.

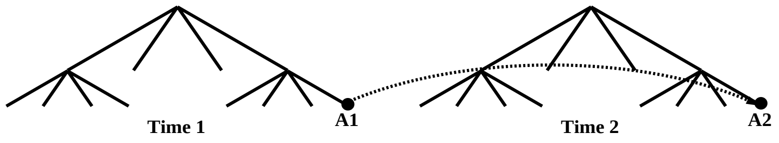


Figure 6.12: Indirect illumination coherence: indirect illumination solutions for times 1 and 2 are compared by walking the hierarchies and comparing the irradiance stored at matching hierarchy locations.

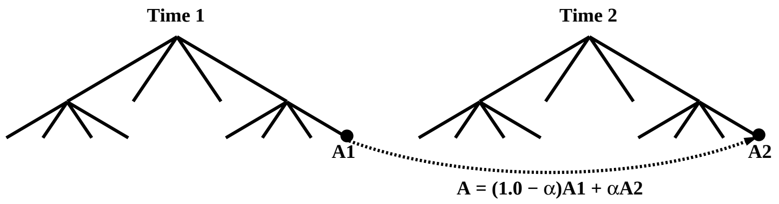


Figure 6.13: Indirect illumination interpolation: indirect illumination solutions for times 1 and 2 are interpolated by walking the hierarchies and interpolating between the irradiance stored at matching hierarchy locations.

We test the temporal coherence of the indirect illumination in the environment by specifying a time interval and a coherence metric. For the endpoints of the time interval, the objects in the environment are placed in the appropriate locations and the indirect illumination is calculated. The distributions for the endpoints are then compared by walking each hierarchy and comparing the irradiance gathered at the particular surface locations for the two different times. This tree-walking mechanism (Figure 6.12) is used to decide if the time interval is coherent. Interval subdivision and a recursive execution of the coherence test on the two subintervals continues until all the subintervals pass the coherence test. Once all the subintervals are coherent, the indirect illumination solutions generated form a set of piecewise-linear object space interpolants that can be used to generate an indirect distribution of light for the environment at any point in time by first interpolating the positions of the surface points and then interpolating the radiance values for those surface points (Figure 6.13). Notice that this process is only slightly more expensive than the typical calculation of indirect illumination in that we add our coherence test. Of course, when you realize that for any views of the environment that are generated we need an indirect illumination component we can see that the cost of our coherence test will more than be offset by the fact that we only have to generate a small subset of the indirect distributions that would normally be required, usually on a per-frame basis, while being able to quickly reconstruct the rest.

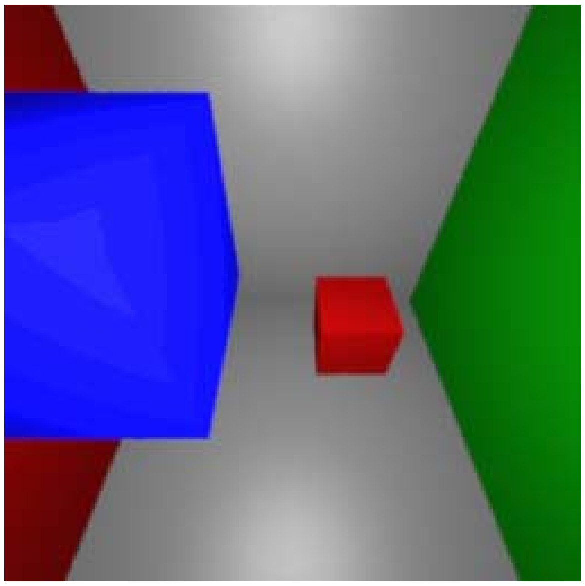


Figure 6.14: Visual information as seen from a light source.

6.6.2 View independent direct illumination coherence

Direct illumination varies more rapidly than indirect illumination and effects the environment through the addition of directional highlights and shadows. We would like to test the variability of these effects and rely on a simple duality argument to formulate our testing mechanism. The direct illumination that falls on a surface falls on those points that are directly visible from the light sources (Figure 6.14). Those surface points within the light volume but not the first points seen along any ray from the light source are in shadow. Those points within the light volume that are visible may be subject to highlights.

To check the general variability in direct illumination, we place a camera at the position of each light source and project all objects in the scene onto the (hemispherical) camera using a fisheye projection. Initially, this is performed for the beginning and ending times of the animation, and a comparison of the resulting id-images (Figure 6.15) is used to compute a value of f for the two points in time. The time interval is subdivided recursively until the value of f for all pairs of successive times exceeds q.

Our coherence measure must consider that we may have a large number of light sources and that each will contribute to the direct illumination variability in the scene. We use the desired time interval for testing as well as our coherence metric, making sure to take into account all the

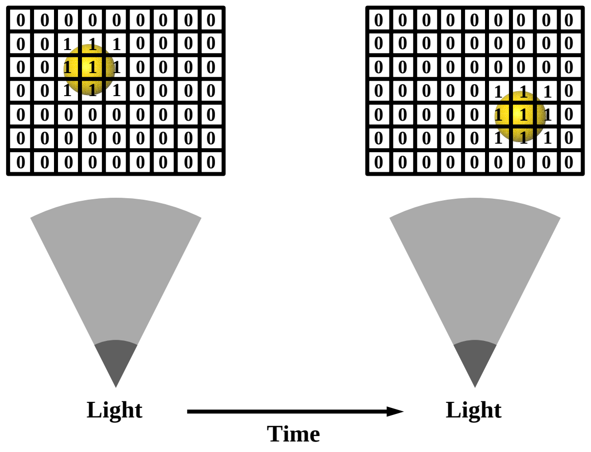


Figure 6.15: View independent direct illumination coherence: id-images are computed after objects and light sources are positioned at the appropriate times.

light sources. For each light source we look at the surfaces that are visible at the endpoints of the chosen time interval and see how similar this visibility information is. When the subintervals are all deemed to be coherent we are left with a time sequence that denotes where the direct illumination information in the environment should be computed and where the information can be adequately represented by combining the information computed for the closest endpoints in the time sequence.

The approach implicitly assumes point light sources. These points may be the centers of mass of clusters of small light sources, or may be sample points chosen on area light sources. Note that no illumination calculations are being performed with this process, we are simply estimating the time frequency for which highlights and shadows will need to be updated.

6.6.3 View dependent temporal refinement

Now that a particular temporal decomposition has been defined for the environment in terms of the variability of the direct illumination effects, we would like to see if it adequately captures the temporal variance seen by each of the locations generated in our spatial decomposition. For each of the base views, we refine the time sequence further by using visibility tests for each of the base view locations. Each base view could have its own time sequence – i.e. there may be relatively little motion in some views relative to others. To do the base view time refinement, the procedure is the same as just described, but the list of base view locations (Figure 6.16) is used in place of the light source locations. Once our decomposition of both time and space is complete, we need to

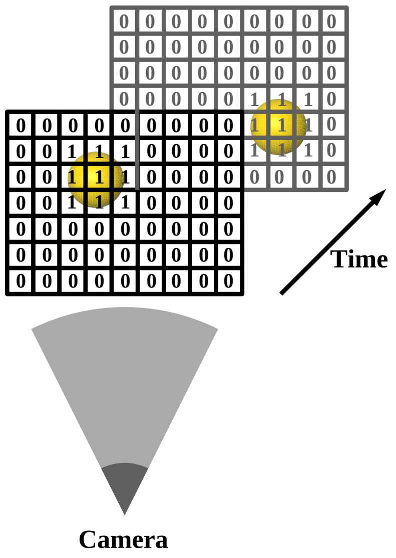


Figure 6.16: View dependent temporal refinement: id images are computed after objects, light sources, and camera are positioned at the appropriate times.

generate our spatio-temporal basis using an appropriately chosen multi-pass rendering algorithm.

6.7 Reconstruction

Once all the base images are computed for the selected locations and times, frame sequences for arbitrary paths through the view space can be generated by reconstructing from the images closest in time and space. For each base view location temporal reconstruction is performed first, isolating the correct time in the frame sequence. Once a single frame for each base location has been reconstructed, spatial reconstruction from proximate views around the desired view location is performed. This process can be repeated for any sequence of desired frames.

6.7.1 Temporal

Temporal reconstruction (Figure 6.17) is performed by finding the N frames in each temporal sequence that are closest to the desired time. A smooth spline, using the times for each of the images as the knot values, is constructed and applied pixel-by-pixel to reconstruct the individual pixel values in the complete image for each base view location:

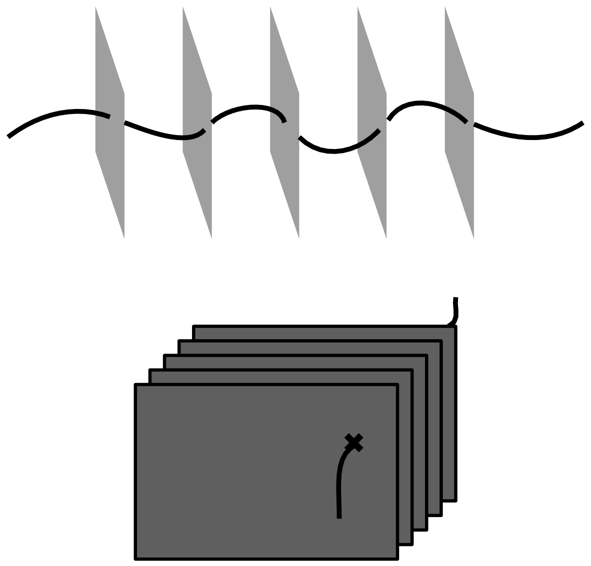


Figure 6.17: Temporal reconstruction using splines.

6.7.2 Spatial

The spatial reconstruction algorithm (Figure 6.18), a reprojection-based view interpolator, takes the proximate set of source images and reprojects each of the pixel locations onto the desired view configuration. The reprojection operation is a simple projection matrix composed of the inverse of the world-to-image projection for the source view and the world-to-image projection for the desired view:

put formula here!!!

Due to the possibly differing source image density, it cannot be guaranteed that the reprojection operation covers all the pixel locations in the destination image. Overlap in the destination image can occur which requires either a source ordering algorithm or a source averaging algorithm to handle pixels that map to the same destination locations. Holes in the destination image may occur due to insufficient density in the source images. Either foreground or background fill methods can be used to alleviate these problems. Foreground fill methods (a priori) project the boundary of the pixels instead of the pixel center so as to cover the area in the destination image with the warped source pixel area. Background fill methods (a posteriori) walk the destination using standard flood fill algorithms to fill in missing information. Both overlap and holes can occur in a single destination image which requires both types of error reduction algorithms to be used.

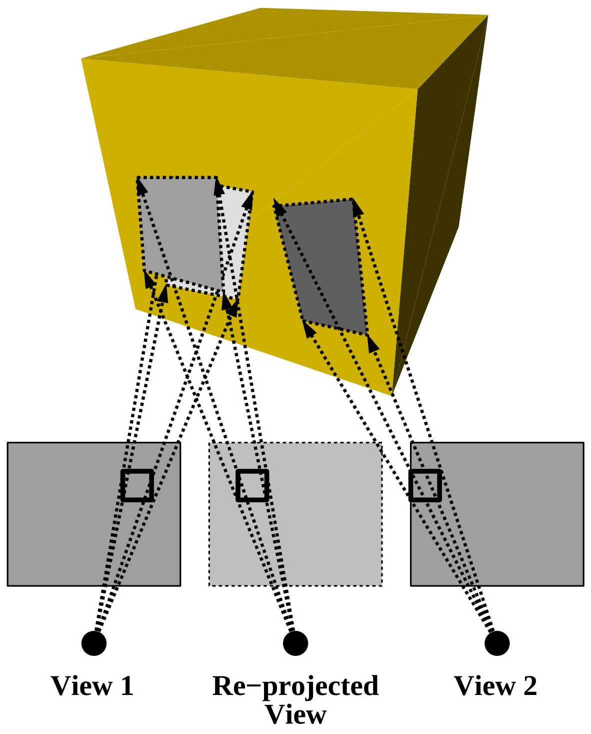


Figure 6.18: Spatial reconstruction using pixel reprojection.

6.8 Implementation Details

Each of the methods discussed above were written in the C++ programming language. SGI frame-buffer hardware was used to facilitate the image space coherence algorithms for direct illumination and visibility. Our indirect illumination coherence and object space spline-based interpolation methods were built on top of a variation of the Princeton radiosity system [138, 139, 266] (our multi-stage rendering method from Chapter 5). Our temporal and spatial reconstruction algorithms are based on the *pcomb* and *pinterp* programs included in the *Radiance* distribution [324].

6.9 Results and Analysis

In this section we critically evaluate the time and quality tradeoffs involved in using the framework. We take the approach of evaluating individual phases of the framework for a small number of environments to get some initial insights into the tradeoffs involved. Specifically, we consider the following components: base view selection, indirect illumination interpolation, and temporal decomposition and reconstruction. We conclude with the application of the complete framework to a complex environment that includes object motion. We use the error metric and comparison techniques described in the results section of the previous chapter (Section 5.5).

6.9.1 Base view selection

In order to test the error incurred by performing quality-driven view space decomposition (Figure 6.20) and view reconstruction, we return to the studio environment used in the last chapter to test geometric simplification. The large number of surfaces challenges the coherence testing procedure, and the view-dependent illumination effects test the reconstruction procedure.

In order to test the effectiveness of the view space decomposition and interpolation modules we have chosen a two-dimensional view space within the studio environment in which we will move freely. Figure 6.19 provides a diagram of the view space. Figures 6.20-6.23 show sample base views and different quality reconstructions using four fill algorithms.

For simplicity in analyzing the effect of quality on a single image, we considered a simplified one-dimensional subset of the view space. The spatial decomposition module was executed with three different quality levels; it returned three pairs of base locations (Figure 6.24, corresponding to the respective quality levels) to be used for the different quality reconstructions. For each pair of base locations, 180° range-images were generated in *Radiance* for use as the base images. We then chose a viewpoint and field-of-view to reconstruct that fell in the middle of the pairs of base locations. Using the three pairs of base images, the view interpolation module, *pinterp* , was used to reconstruct the different quality images of the environment from the chosen view. *Radiance* was used to generate a reference image for the chosen view and field-of-view. The analysis was completed by calculating the relative LRMS error between the various images.

The raw statistics and images (Table 6.1 and Figure 6.25) for the pixel-based view reconstruction are especially interesting. One source of error is due to the fact that *pinterp* produces slightly different results depending on the ordering of the input files and a tolerance factor. We ordered the files by linear distance from the interpolation point under the assumption that the images from closer base viewpoints would be more valid than those from greater distances. Since the method uses reprojection of pixels from the base images as its means of reconstruction, a high initial resolution is required to remove any quantization effects and resampling errors. This effect is shown

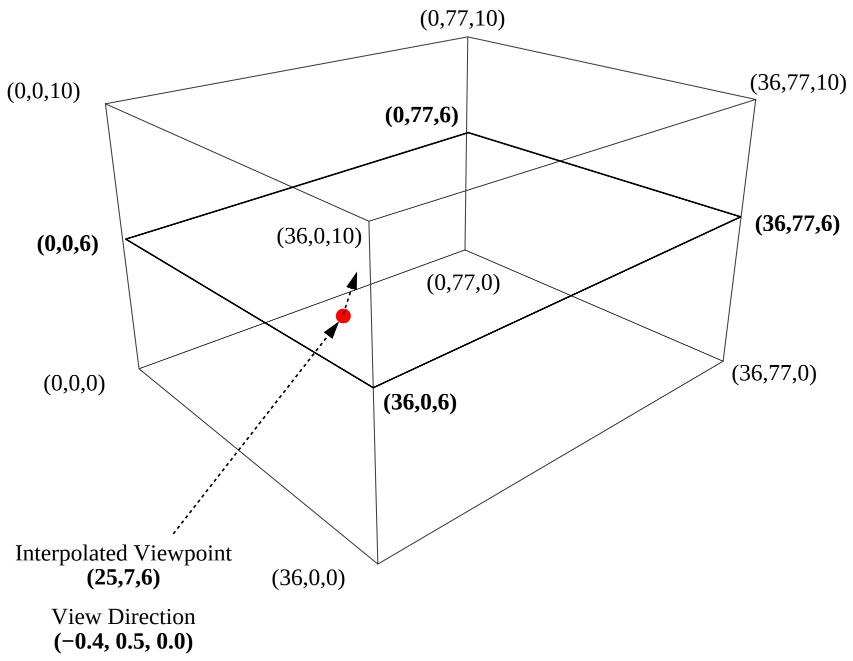


Figure 6.19: Studio dimensions - 2D view space.

View Reconstruction - Studio		
Scene Decomposition		
Quality	Number of Views	Computation Time (secs)
Low Quality - .3	2	10.77
Medium Quality - .6	3	34.12
High Quality - .9	5	116.05
Base View Synthesis		
Image		Computation Time (mins)
180° camera		728.7
Base 1 - Low Quality (.3)		621.8
Base 2 - Low Quality (.3)		570.0
Base 1 - Medium Quality (.6)		725.6
Base 2 - Medium Quality (.6)		647.2
Base 1 - High Quality (.9)		707.2
Base 2 - High Quality (.9)		592.3
Synthesis		
Image	Computation Time (mins)	LRMS Error/Rad. HQ Lum. Avg.
Radiance HQ	45.7	0.000
Low Quality (.3)	0.048	8.75/13.72 = 0.638
Medium Quality (.6)	0.061	8.57/13.72 = 0.625
High Quality (.9)	0.046	1.61/13.72 = 0.117
Resampled Radiance	0.035	1.64/13.72 = 0.120

Table 6.1: View reconstruction statistics for the studio environment.

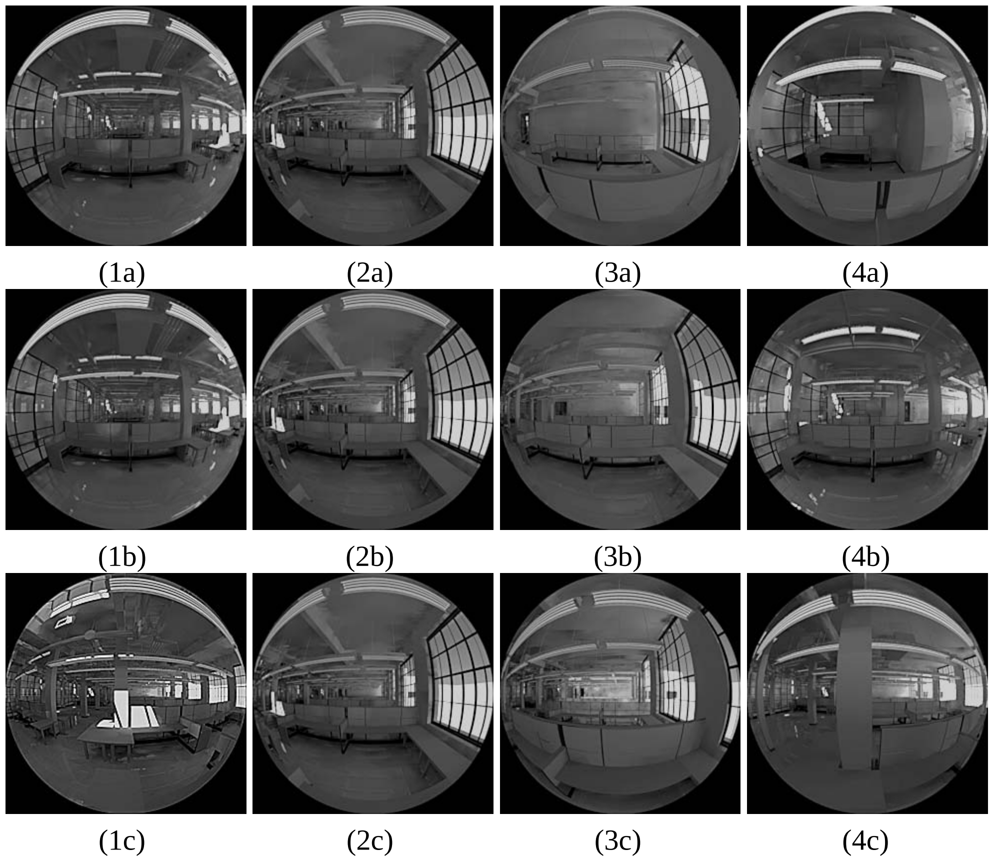


Figure 6.20: Four base view locations in the studio view space: (a) low quality spatial decomposition, (b) medium quality spatial decomposition, (c) high quality spatial decomposition.

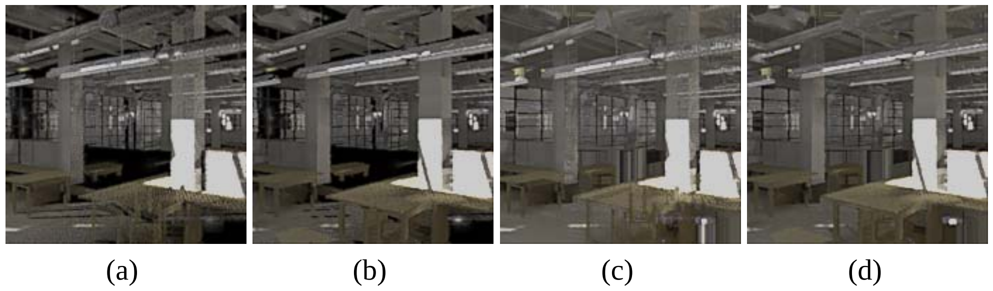


Figure 6.21: Four pixel-based view reconstruction fill algorithms (low quality spatial decomposition): (a) No fill, (b) Foreground fill, (c) Background fill, (d) Foreground and background fill.

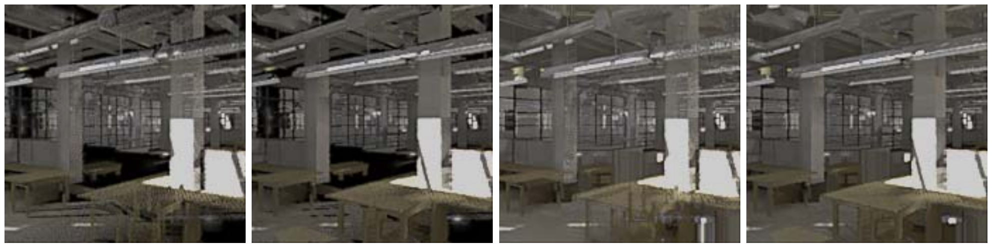

(a) (b) (c) (d)

Figure 6.22: Four pixel-based view reconstruction fill algorithms (medium quality spatial decomposition): (a) No fill, (b) Foreground fill, (c) Background fill, (d) Foreground and background fill.

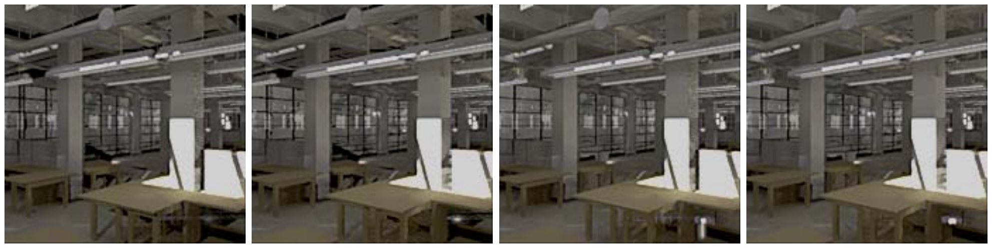

(a) (b) (c) (d)

Figure 6.23: Four pixel-based view reconstruction fill algorithms (high quality spatial decomposition): (a) No fill, (b) Foreground fill, (c) Background fill, (d) Foreground and background fill.

in the row labeled “Resampled Radiance” which specifically measures the error involved in downsampling from a spherical image at the chosen viewpoint to the desired field-of-view image at the same point. This error is very large and suggests that a more sophisticated approach is needed; future work will address this shortcoming. It is important to recognize the existence of this error when viewing the statistics.

In addition, to ensure good results, the setting for q for this step has to be high ($\geq .9$), as the results are very sensitive to having the correct objects visible. Although q was high, we were able to get reasonable results by interpolating views that are seven meters apart.

The reconstruction process itself works well for views that flank the chosen view. However, the error increases dramatically as we start to move our base images away from the chosen viewpoint. Mutually occluded information in the base images that must be “filled” in the reconstructed image can also cause severe discrepancies in the compared images and a large RMS error. Fortunately, the areas that are most problematic are somewhat isolated. Perhaps rather than having a spatially denser set of complete base images, a few partial images for areas of the view that change rapidly

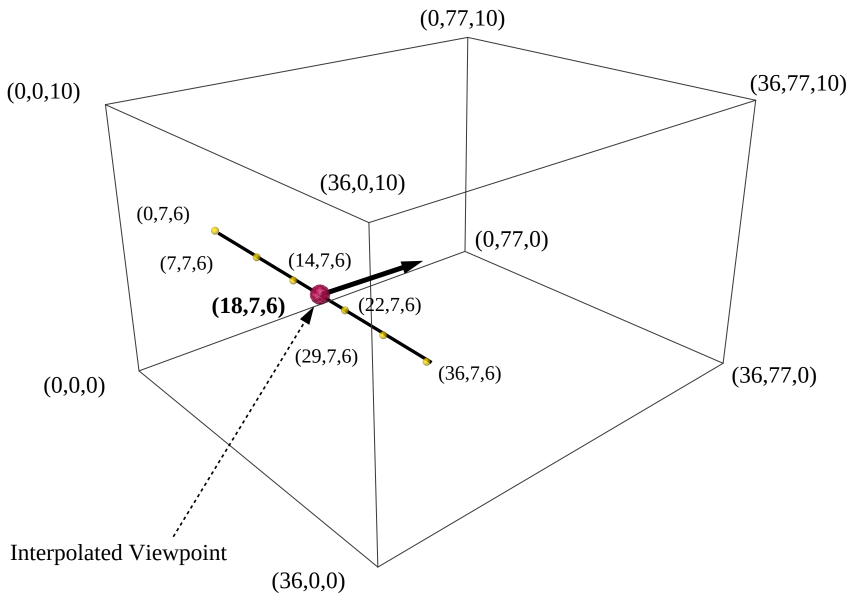


Figure 6.24: Studio dimensions - 1D view subspace.

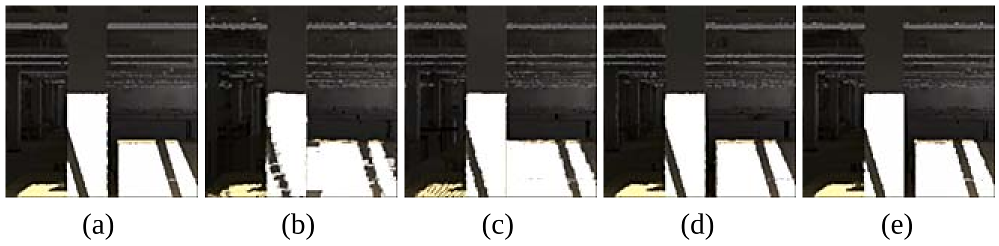


Figure 6.25: Comparison of studio images generated via view reconstruction: (a) Radiance HQ, (b) Low quality reconstruction, (c) Medium quality reconstruction, (d) High quality reconstruction, (e) Resampled radiance.

with location could be used. These partial samples could benefit other image-based approaches as well, including our own motion interpolation method.

6.9.2 Indirect illumination interpolation

To examine our approach to modeling the temporal variations in indirect illumination, we elected to work with the Cornell box environment. We chose this simple model as a means to help isolate the interpolation error from the geometric simplification error presented in the previous section. We modified this model to include time-based motion paths for the enclosed boxes.

We decomposed the environment into temporal segments in which the indirect illumination on the surfaces did not vary beyond the user-specified quality. For the environment and the chosen time interval, we placed the objects where they should be at the ends of the time interval and then generated an indirect solution for both times using the wavelet radiosity solver. We then computed the average of the object space pointwise illumination RMS over the irradiance average and compared it to the quality parameter. This computation weights the magnitude of the irradiance RMS by the average irradiance of the two solutions to give a sense of the relative magnitude of the RMS. If the particular time interval is deemed to be "incoherent," the interval is subdivided and the same procedure is applied recursively to the two subintervals. The decomposition procedure terminates with a set of times representing the endpoints of coherent indirect illumination intervals from which the intermediate indirect solutions can be computed via interpolation. Interpolation is performed by placing the objects where they should be at the appropriate time and linearly interpolating pointwise indirect irradiance values for the surfaces.

We decomposed the box environment using three different quality parameters, which yielded three time lists. For each element in the time lists, an indirect illumination solution was stored. We then chose an intermediate time and generated four indirect illumination solutions. Three were linearly interpolated from the nearest indirect solutions in the different quality time lists. One was generated by the wavelet radiosity solver for the chosen time. We then converted these indirect solutions to *Radiance* ambient files and generated multi-pass images. These four images were compared to a *Radiance* HQ image using the relative LRMS described earlier. The results (Table 6.2, and Figure 6.26) show that indirect illumination interpolation performs well and yields significant increases in performance over the typical process of generating both the indirect illumination and

Indirect Illumination Interpolation - Cornell Box			
Image	Phase	Comp. Time - 30 Frame Seq. (mins)	LRMS Error/Rad. HQ Avg. Lum. (per frame)
Radiance HQ	Rendering	1179.00	0.000
Radiance Typical	Rendering	190.80	0.020/0.109 = 0.183
WR Low Quality	Scene Decomposition	0.04	
	Indirect Generation	0.73	
	Indirect Interpolation	0.94	
	Rendering	88.20	
	Total	89.91	0.016/0.109 = 0.147
WR Medium Quality	Scene Decomposition	0.14	
	Indirect Generation	1.03	
	Indirect Interpolation	0.94	
	Rendering	93.60	
	Total	95.71	0.015/0.109 = 0.138
WR High Quality	Scene Decomposition	0.31	
	Indirect Generation	1.69	
	Indirect Interpolation	0.94	
	Rendering	111.60	
	Total	114.54	0.013/0.109 = 0.119

Table 6.2: Temporal indirect illumination interpolation statistics for the Cornell Box environment.

the direct illumination on a per frame basis.

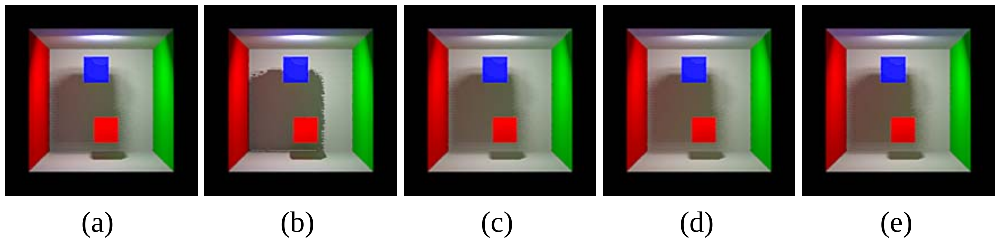


Figure 6.26: Comparison of Cornell box images generated with temporally interpolated indirect components: (a) Radiance HQ, (b) Radiance typical, (c) Wavelet radiosity low, (d) Wavelet radiosity medium, (e) Wavelet radiosity high.

6.9.3 Temporal decomposition and reconstruction

Reconstructing temporal radiance variances (object motion) from images is a difficult problem. Pixel-based interpolation methods are inadequate for the reconstruction of object motion unless a large set of images is generated. To keep our initial implementation as simple as possible, we used pixel-wise linear interpolation as a temporal reconstruction algorithm. This, of course, means that our temporal decomposition approach oversamples in an attempt to remove undesirable visual artifacts.

In order to test this phase of the framework, we chose to return to the Cornell box model. The simplicity of this environment allows us to easily isolate the error introduced by motion interpolation from other errors that can be introduced by the framework.

We analyzed temporal decomposition and reconstruction from the point of view of a single,

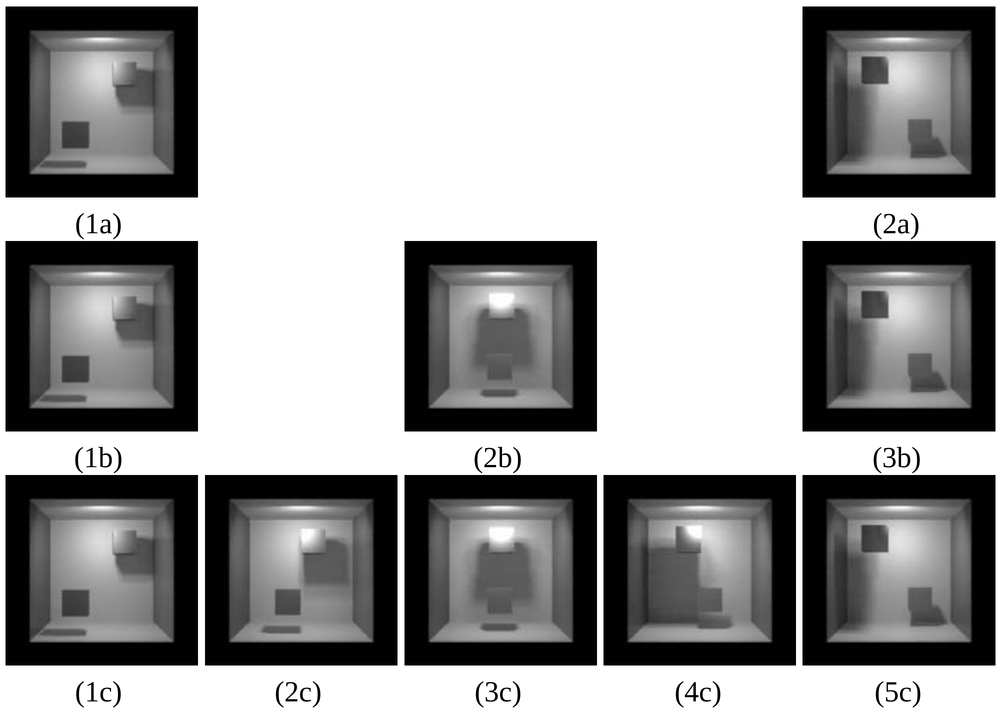


Figure 6.27: Temporal decompositions: (a) low quality decomposition, (b) medium quality decomposition, (c) high quality decomposition.

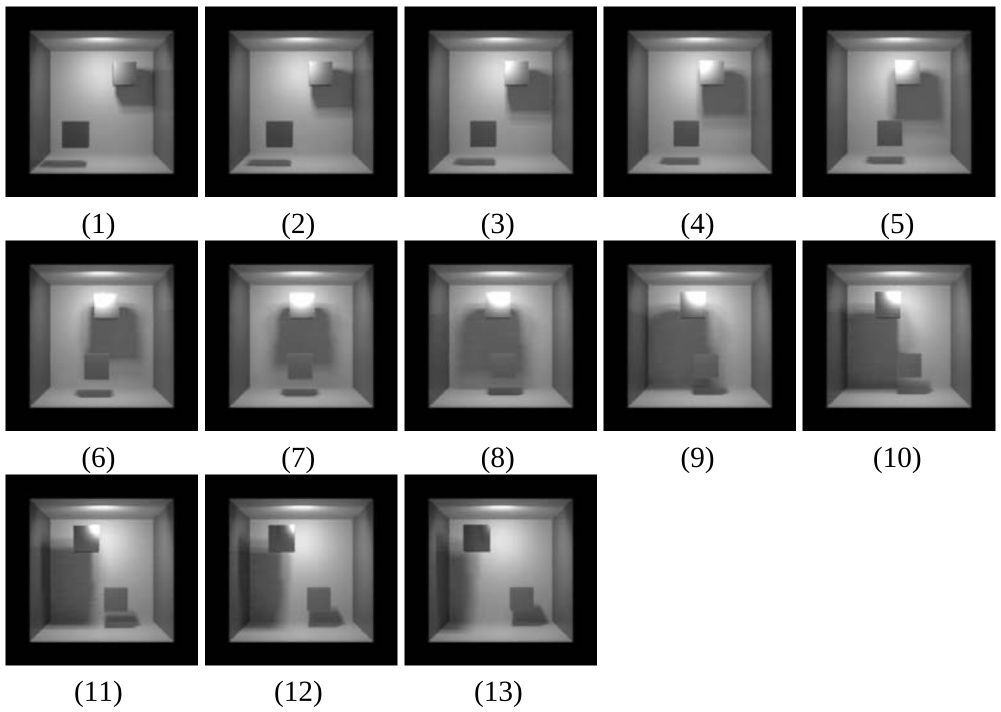


Figure 6.28: Fully rendered sequence of frames containing object motion.

Temporal Decomposition and Reconstruction - Cornell Box			
Decomposition			
Quality	Number of Solutions	Generated Time List	Computation Time (secs)
Low Quality - .3	2	0, 48	0.76
Medium Quality - .6	3	0, 24, 48	1.93
High Quality - .9	5	0, 12, 24, 36, 48	6.64
Synthesis			
Time (secs)			
Fully Rendered	Low Quality - .3	Medium Quality - .6	High Quality - .9
15204	2711	3837	6030
Analysis			
Frame 10 - LRMS/Rendered Frame Avg. Lum.			
Fully Rendered	Low Quality - .3	Medium Quality - .6	High Quality - .9
0.000	0.118/0.139 = .849	0.104/0.139 = .748	0.080/0.139 = .576

Table 6.3: Temporal decomposition and reconstruction statistics for the Cornell box environment.

fixed view. For the chosen view, we decomposed time into coherent segments using our visible object coherence and direct illumination coherence algorithms. At the ends of the chosen time interval, if the percentage of pixels that “sees” the same object is greater than the user-specified quality parameter, the interval is deemed coherent with respect to object visibility. A similar test is performed from the point of view of the light sources to test direct illumination coherence. If the interval passes both tests, it is coherent. Subdivision occurs for any interval that fails either test until a list of coherent time intervals is created.

We decomposed the box environment using three different quality levels which yielded three time lists. For each of the time lists, we generated the set of temporal base images (Figure 6.27). We then compared a fully rendered set of twelve *Radiance* images (Figure 6.28) against the different quality base image sets; intermediate frames were filled in by linear reconstruction. This simple test allowed us to isolate the error introduced by motion reconstruction alone.

The results (Table 6.3, Figure 6.29 and Figure 6.30) from this phase indicate that we can achieve reasonable accuracy by using a large number of temporal samples. However the error is of a type that is especially objectionable. Pixel-flow algorithms would provide a better mechanism for this type of reconstruction but are more complicated and require more information than is typically stored in a range-image. The pixel flow must be calculated and stored with the pixel values.

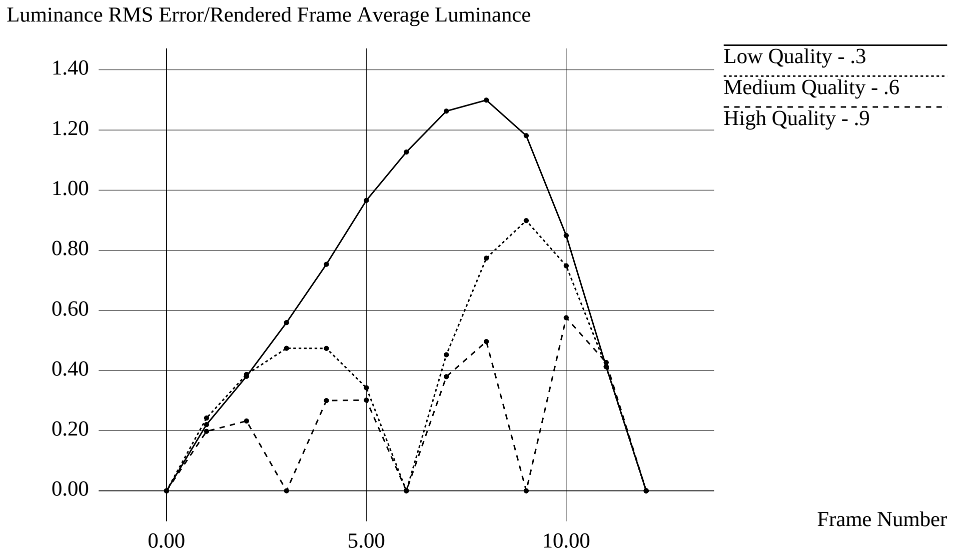


Figure 6.29: Error graph for motion interpolation within the Cornell box environment.

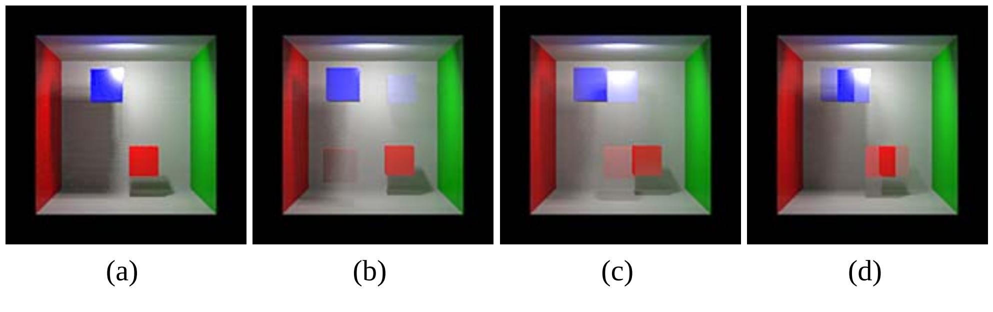


Figure 6.30: Comparison of frame 10 in the motion sequence: (a) Radiance HQ, (b) Low quality reconstruction, (c) Medium quality reconstruction, (d) High quality reconstruction.

6.9.4 Putting it all together – the soda shop

In this section, we examine the performance of the framework as a whole by taking an example from the specification stage through to the walk-through stage. This analysis involves integrating all of the phases of the framework, including indirect and direct illumination, and both domains, spatial and temporal. We use a soda shop environment of moderate complexity for this analysis. The scene contains complex surface BRDF's and textures and small objects that move.

We began by performing a decomposition for the indirect illumination. This entailed simplifying the environment and then decomposing it as a function of time. We performed this decomposition for three different quality levels: low, medium, and high; the same quality parameters were used through the different phases of the framework. After treating indirect illumination, direct illumination and visibility were computed yielding the following sample points in space and time:

- *Low quality*: two spatial locations with two times for each location.
- *Medium quality*: three spatial locations with three times for each location.
- *High quality*: five spatial locations with five times for each location.
- *Very high quality*: fifteen spatial locations with thirty times for each location.

For each of the spatial locations and each of the times, a 360° base image was generated for use in the reconstruction phase.

After the bases were constructed, a path through the environment was chosen, including the associated camera parameters. Frames were then rendered using *Radiance* with its high quality settings. We also reconstructed the path using our low, medium, and high quality bases. The individual frames were then compared using the relative LRMS error metric.

The results (Table 6.4, Figure 6.31, and Figure 6.32) for the walk-through are encouraging. The high quality reconstructions are comparable to the high quality *Radiance* renderings. In addition, we can successfully control the quality of the sequence and trade time for numerical accuracy with the parameters we have defined. For our results, we used the chosen quality level through all the steps in the framework. It is clear, however, that quality (as we have currently defined it) does not have the same impact on each step in the process. The framework needs to be refined to take a

Walkthrough Reconstruction - Soda Shop	
Scene Decomposition/Base Image Construction	
Quality	Time (mins)
Low Quality - .3	1213.53
Medium Quality - .6	2894.22
High Quality - .9	7746.76
Very High Quality (Converged Solution) - .99	20508.93
Walkthrough Construction - 30 Frames	
Quality	Time (mins)
Fully Rendered	4113.54
Low Quality - .3	90
Medium Quality - .6	210
High Quality - .9	480
Very High Quality (Converged Solution) - .99	1890

Table 6.4: Execution times for individual phases of the framework on the soda shop environment.

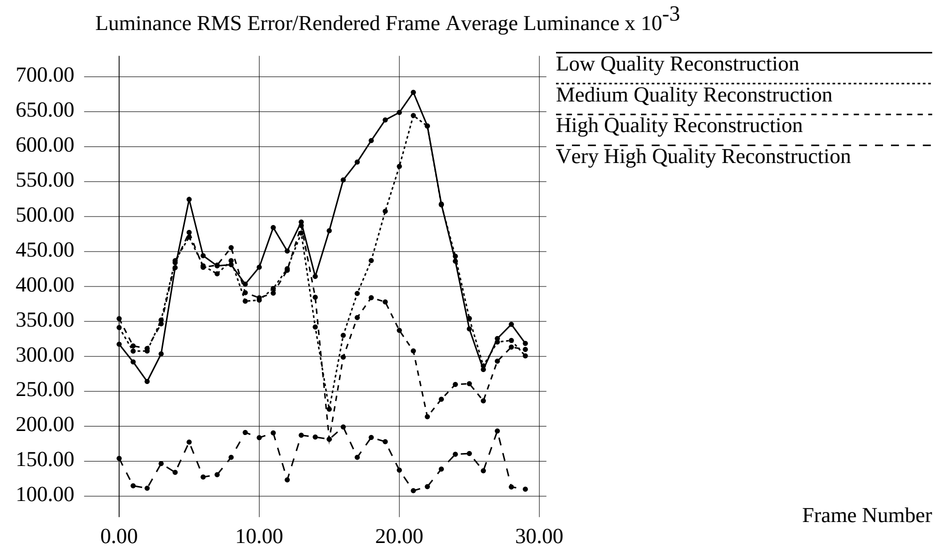
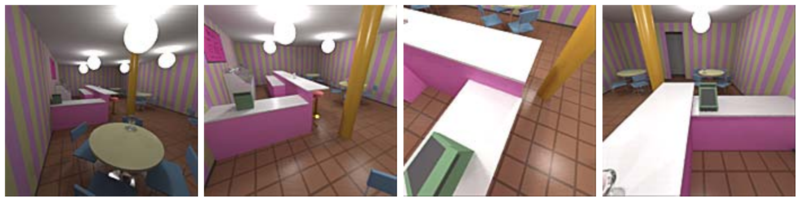
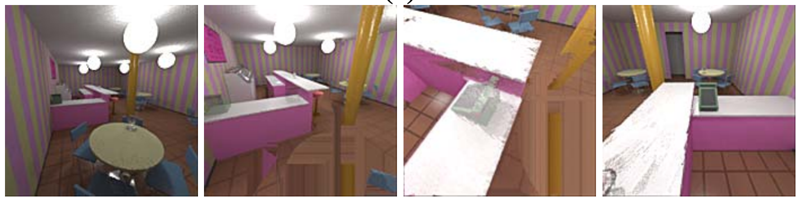


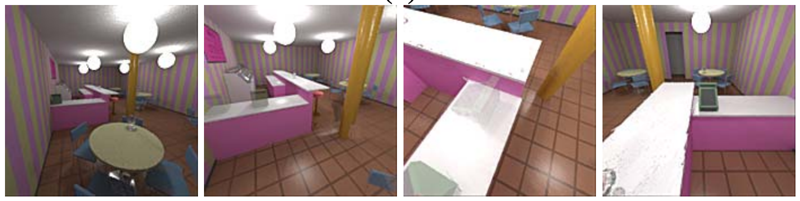
Figure 6.31: Error graph for walk-through reconstruction within the soda shop environment.



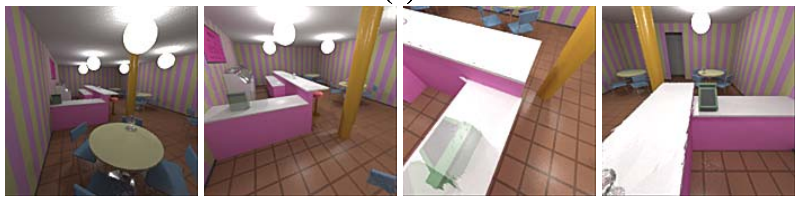
(a)



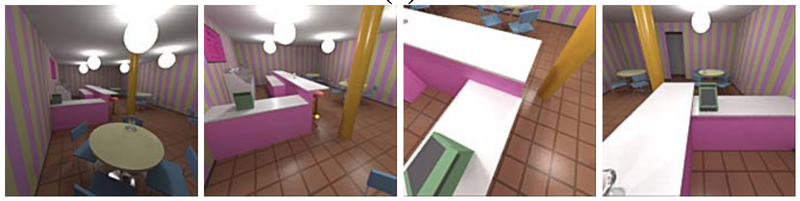
(b)



(c)



(d)



(e)

Figure 6.32: Four frame comparison in the soda shop environment: (a) Fully rendered, (b) Low quality reconstruction, (c) Medium quality reconstruction, (d) High quality reconstruction, (e) Very high quality reconstruction.

user-specified quality and interpret it relative to each step in the process; this will be a subject of future work.

6.10 Summary

In this chapter we presented a method for representing the radiance variances in an environment based on range-image sequences at a sparse set of important locations within the chosen view space of the environment. Our approach exploits coherence by computing direct and indirect illumination separately in both space and time. We introduced a mechanism for temporally interpolating indirect illumination to allow us to generate a coherent time-varying indirect distribution from a set of sparse spatial and temporal samples. We demonstrated that the range-image approach allows the rapid generation of the views along arbitrary paths within the view space. This radiance distribution approximation represents a major step toward the ultimate goal of allowing users to interact with accurately rendered, animated, geometrically complex environments, as it allows for a user to tour a view space rendered with full global illumination effects in which objects move. Within the context of our approximation, there are several areas that require future research. The most important deficiency that we will address in the next chapter is that of efficiently sampling and reconstructing temporal illumination variances (object motion) from sample-based structures (images).

Chapter 7

Capturing Directional and Temporal Radiance Variations Using Sparse Image Volumes

"I put instant coffee in a microwave oven and almost went back in time." *Steven Wright*

This chapter presents a method on efficiently capturing directional and temporal radiance variations for fixed spatial locations in an environment, approximating the $3D$ directional and temporal portions of an environment's radiance distribution. It is based on work presented at the Seventh Eurographics Workshop on Rendering in Porto, Portugal [215]. In our implementation from last chapter, directional and temporal radiance variances are represented by time-sequences of range-images, with spatial variations captured by using sets of time-sequences. Our new approach attempts to obviate the high directional and temporal sampling rate normally required to adequately capture the radiance variances caused by an object's motion over time — a very dense time-sequence of images is required. It is our contention that a balance can be found between the sampling rate and density required to extract the directional and temporal information in the radiance distribution of an animated environment, and the ability to reconstruct believable and accurate sequences of images from the samples. The typical frame-based process focuses its sampling efforts on each image in the sequence often neglecting the redundancy between subsequent frames.

The method presented here is designed to replace the image sequences in our original implementation and uses sparse directional and temporal environment sampling, a resolution-independent, unordered temporal file format, and a Delaunay complex image reconstruction method. Temporal reconstruction usually achieved by rendering a complete time-sequence of images of the environment at a high frame rate, flipbook style frame-based animation, can be performed using many fewer samples — often on the order of those required to generate a single, complete, high quality frame — taken from the sparse radiance data stored in bounded regions of our temporal file. The environment is rendered using a ray tracing algorithm modified to randomly sample over direction, the image plane (x, y), and time (t), yielding (x, y, t) samples that are stored in our image volumes. Reconstructing an image of the scene at a desired time can be performed in one of two manners. In the first reconstruction method, we orthogonally project a proximate subset of the (x, y, t) samples onto the desired temporal plane $(t^{'})$ with the appropriate weighting, construct the $2D$ Delaunay complex of the projected points, and Gouraud/Phong shade the resulting triangles using the vertex radiances as the interpolated parameter. In the second reconstruction method, we build the $3D$ Delaunay complex from the (x, y, t) samples and intersect it with an axis-aligned (x, y) plane fixed at the desired time $t^{'}$. Both first and higher order visual effects, illumination and visibility, are handled as this information is pre-computed into the individual samples. Silhouette edges and other discontinuities are more difficult to track but can be addressed with a combination of triangle filtering and image post-processing.

7.1 Motivation

As mentioned in the previous chapters, sample-based rendering is a rapidly growing area of research in the computer graphics and computer vision communities. Although the spatial domain is an area of ongoing research (view interpolation [60, 58, 193, 270]), efficient and accurate temporal reconstruction methods, those that can reproduce the motion of objects over time via image operations, are still scarce if not non-existent. Both the vision and graphics communities tend to look at the problem of temporal reconstruction in terms of image sequences, either decomposing the sequences after they have been created to track temporal/motion information (visual/optical flow in computer vision [295]), or attempting to generate smooth temporal/motion information by creating densely packed sequences (computer-generated animation in computer graphics). The problem can be solved in a more general manner by providing an way to sample the directional and temporal information in the radiance distribution of an environment as efficiently as possible, and providing an accurate and efficient reconstruction method, requiring an image sequence until one is desired during a walkthrough. The goal then, is to create image sequences that are as believable as those generated a single frame at a time, while being much more efficient to compute. Although our method is designed to construct an image at a particular time under fixed viewing conditions and is made to augment our framework, it meshes well with the current work in spatial reconstruction, helping to overcome those algorithms' limitation to static scenes. Directional and temporal reconstruction using our approach can be performed yielding images of the environment at the appropriate time, suitable for use as the input to a spatial reconstruction algorithm.

To augment our framework from the previous chapter we need to describe our methodology for:

1. Sparsely sampling the directional and temporal domains in animated environments.
2. Storing the radiance samples in an unordered temporal volume (file).
3. Reconstructing image sequences from the temporal volume.

Our approach is based on randomized and adaptive ray tracing and is designed to minimize the number of samples required to adequately capture the directional and temporal effects radiance variances by object motion. We want to strike a balance between the sampling rate and density required to extract the directional and temporal radiance information, and the ability to reconstruct believable and accurate sequences of images from the samples.

7.2 Directional and Temporal Sampling

Although the analogy is often made between the image creation/rendering process and a camera receiving illumination reflected off surfaces in the environment, we are not restricted to using a physically realizable model. Nothing precludes us from changing or even extending the information our "virtual camera" can detect (i.e. depth, orientation, etc.). More specifically, we are not limited to the snapshot notion of a camera, taking a large number of samples, a complete image, at a prescribed time. We instead want to build up a small set of samples, via a directional and temporal sampling process, which contains enough information about the variances in an environment's radiance distribution, so that accurate reconstruct is possible.

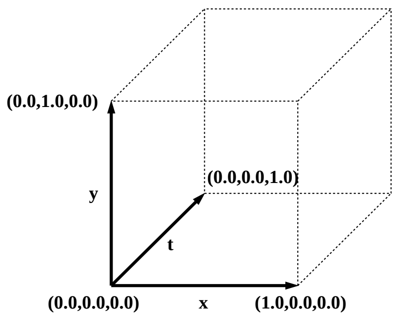


Figure 7.1: Sample space: projective $3D$ space-time.

A camera, physical or virtual, by its design takes a large number of samples for each instant of time. This precludes the use of any redundant information that may exist across multiple instances (i.e. small objects moving against a static background). In the worst case there may be no redundancy, but if there is, typical camera mechanisms do not even attempt to harness it. We want to enhance the design of a virtual camera so that we can choose which view directions at which times are important. The radiance along any particular view direction might not change over time. Why should this information be captured in every image? Instead of fixing the time t and then designing a sampling pattern for the image plane (x, y) to construct an image that is part of the desired sequence, we look at the total number of samples that would be required for the full image sequence $(x \times y \times t)$ and design the sampling pattern for (x, y, t) (Figure 7.1). The sampling pattern should be sparse around any chosen time $t + t_\epsilon$ but allow for accurate and efficient reconstruction of an image of the environment at the desired time.

7.2.1 Random sampling

In order to simplify our particular implementation, we initially constructed our sample points by randomly choosing positions on the image plane, (x, y), like a standard ray tracing algorithm, and also over the temporal extent of the motion, t.

Within our sample space, each sample point (x, y, t) satisfies the conditions that $(0.0 \leq x, y \leq 1.0)$ are uniform random numbers that represent coordinates over the normalized image plane

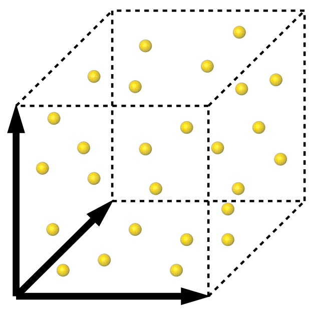


Figure 7.2: Random sampling.

and $(0.0 \leq t \leq 1.0)$ is a uniform random number that represents a temporal value within the normalized boundaries of the motion (Figure 7.2). Jittered supersampling is provided allowing a particular (x, y, t) sample to index the radiance information averaged over a collection of rays within $(x \pm x_\epsilon, y \pm y_\epsilon, t \pm t_\epsilon)$. It should be noted that although this sampling scheme is not optimal, it is easily implemented and therefore useful in an implementation.

We use a normalized space because the number of samples is only loosely coupled to the desired final image resolution. Also, any dependence that our final image sequence might have on a specific frame rate/temporal resolution has been removed. Instead, a representation has been chosen that is sparse in the directional and temporal region around any chosen time $(t' \pm t'_\epsilon)$.

7.2.2 Uniform adaptive sampling

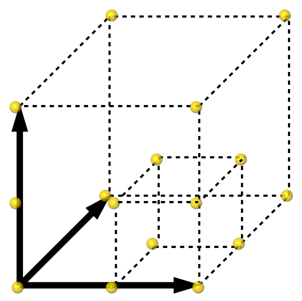


Figure 7.3: Adaptive sampling.

An alternative to a strict random sampling scheme that is better for isolating discontinuities is adaptive sampling (Figure 7.3). In two dimensions, image space adaptive subdivision compares the sample values at the vertices of the current image plane quadrilateral. Subdivision is performed if these boundary samples have distinctly different values, a radiance discontinuity. This method can be extended to encompass the temporal domain, a third dimension, as well with the comparison utilizing the vertex values of the current directional and temporal $3D$ object. We found that a combination of adaptive and random sampling yields better results than either technique used alone.

7.2.3 Non-uniform adaptive sampling - Delaunay tetrahedral subdivision

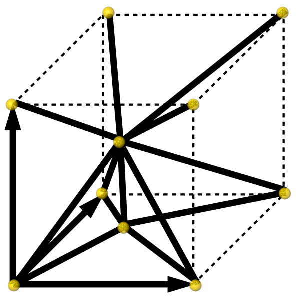


Figure 7.4: Delaunay tetrahedral sampling.

An alternative adaptive sampling scheme with many of the desirable properties of random sampling is Delaunay tetrahedral decomposition (Figure 7.4). Once sample points on the boundary of the directional and temporal region have been evaluated and added to the base decomposition, incremental subdivision can be performed by constructing a new random point to be added into the decomposition, evaluating it, and adding it to the decomposition using an incremental Delaunay tetrahedrization algorithm [15]. Testing the decomposition to see if it needs to be subdivided requires walking the list of tetrahedral regions, and comparing the boundary vertex values. If the values are different, subdivision takes place and the process repeats itself.

7.3 Computing Sample Radiance

For each sample location that is generated, a radiance/illumination value has to be computed. This radiance value represents the energy seen along the ray that enters the viewpoint through the location (x, y) on the image plane at the time t. Standard ray tracing techniques can be used but are inefficient for use in animated environments.

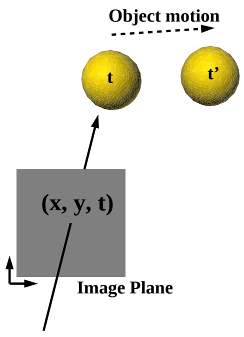


Figure 7.5: A moving sphere and its positions at time t and $\acute{t}$. The sample (x, y, t) traces a ray through the image plane position (x, y) into the environment with objects placed at time t.

7.3.1 Modified space-time ray tracing

We instead compute the radiance seen along the particular ray, associated with a particular sample location, using a variant of the space-time ray tracing algorithm (Figure 7.5) described by Glassner [111]. In order to efficiently trace rays in an environment where objects are moving, we use the construct efficient bounding volumes in $3D$ using a combination of uniform space subdivision and bounding volume hierarchies and then extend the technique to $4D$ space-time. This structure creates and efficient mechanism for tracing rays in an animated environments. The $4D$ bounding volumes enclose $3D$ objects for a particular duration in time, the 4th dimension, are fixed once the animation timeline has been constructed, and offer a good performance increase when testing many rays against the time-varying environment configuration. Instead of calculating complete frames at each chosen time like Glassner does, we never generate a complete image for any particular time. Instead, using our randomized temporal sampling pattern, we deposit a small number of samples within any time interval surrounding any particular time ($\acute{t} \pm t'_\epsilon$). This sparse, randomized sampling minimizes the redundancy found in the static portions of traditional computer-generated imagery by not oversampling those regions.

7.3.2 Temporal image volumes

The sampling schemes implemented do not use the typical indexing method, $(rows, columns)$, usually associated with computer-generated imagery.

Unordered storage mechanism

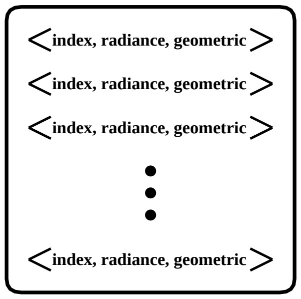


Figure 7.6: Unordered file format.

Since the sample locations are generated randomly, as well as adaptively, and therefore do not lie on a regular lattice that can be exploited when storing the data (Figure 7.6), the image structure must account for this by storing the indexing information with the sample (x, y, t).

Extended sample information

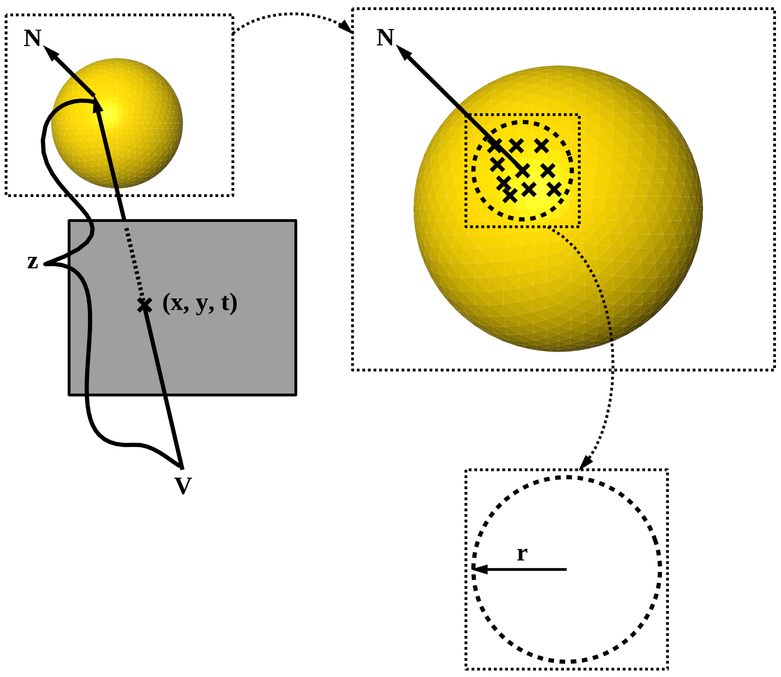


Figure 7.7: Extended sample information.

As with traditional ray tracing algorithms, the radiance/color is returned for the particular sample. Along with this information, pseudo-geometric information is returned to aid in formulating reconstruction algorithms, both current and future. The depth of the object seen along the ray at the particular time is returned along with the normal to the object at the intersection point. This provides depth and orientation information that is useful for spatial as well as direction and temporal reconstruction. A final piece of information that is returned is the size of the valid reconstruction region for this sample. This value is estimated from the largest directional and temporal sphere centered at the sample location encompassing all the jittered supersamples. These values (Figure 7.7) are stored into the temporal volume, represented using a simple ASCII file.

7.4 Reconstruction

In order to formulate our reconstruction mechanism, we borrow from the pixel-independent ray tracing approach [340]. Instead of focusing on the problem of sampling the image plane alone, performing adaptive image sampling relative to a Delaunay simplicization of the samples already taken, we use the research to motivate simplicial-based image reconstruction.

After rendering a scene into the temporal volume, the reconstruction of individual frames is performed. We investigated two methods for performing this task. One works in $2D$, triangulating a set of projected radiance values. The other works in $3D$ slicing an image from the tetrahedral radiance complex.

7.4.1 $2D$ reconstruction

The first reconstruction method involves isolating those samples within a certain tolerance around the chosen time, $t^{'} \pm t_{\epsilon}$, representing that information on the $2D$ image plane as a set of sample values, and filling in the information between the samples that is missing.

Temporal indexing and filtering

The samples that will be used to reconstruct the image, those with temporal indices in $\acute{t} \pm t_{\epsilon}$ (Figure 7.8), are weighted and projected onto the $\acute{t}$ plane. An orthographic projection is used, with the radiance of the sample being attenuated by a user-defined filter whose domain is $0.0 \leq |t^{'} - t| \leq 1.0$. This leaves a set of colored image plane, (x, y), samples with which to perform the reconstruction. Due to the randomness of the sampling scheme, small regions in time contain

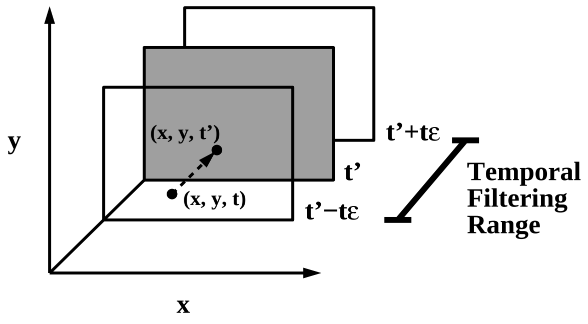


Figure 7.8: The temporal filtering range. All samples between $t' - t_\epsilon$ and $t' + t_\epsilon$ are included in the reconstructed solution. The spatio-temporal sample (x, y, t) is projected onto (x, y, t') with the radiance weighted by FILTER($|t' - t|$).

a small number of samples. This sample sparsity leaves large gaps in the (x, y) plane that need to be filled.

Delaunay triangulation

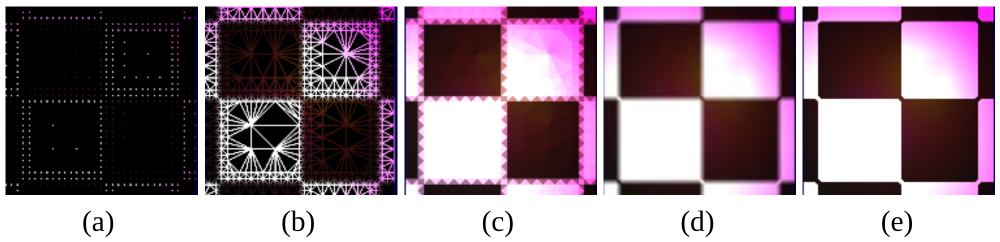


Figure 7.9: Directional and temporal reconstruction via Delaunay triangulation: (a) projected points after temporal filtering, (b) edges of the triangles, (c) flat-shaded triangles, (d) smooth-shaded triangles, (e) edge-detected triangles.

Projection provides a sparse, disconnected set of (x, y) samples on the chosen temporal plane. Without connectivity information, ordered methods of reconstruction such as nearest neighbor blending or other interpolation methods cannot be directly applied. To create topological and geometric information from this unorganized data we construct the incremental Delaunay triangulation [127, 227, 184] of the sample set. The triangles generated here can be drawn quickly as a set of vertices, edges, or faces, tiling the image plane (Figure 7.9). Since the sample values contain the illumination information, no new shading calculations need to be performed.

7.4.2 $3D$ reconstruction

The second requires building a $3D$ directional and temporal Delaunay complex. This complex is a piecewise approximation to the true directional and temporal radiance variances within the particular region. Generating an image from the complex requires slicing the $3D$ construct with a $2D$ plane that is positioned at the appropriate time.

Complex slicing

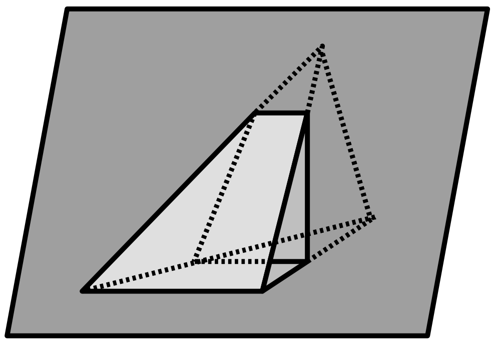


Figure 7.10: Slicing the $3D$ Delaunay complex with a plane: intersecting the edges of each simplex with the cutting plane.

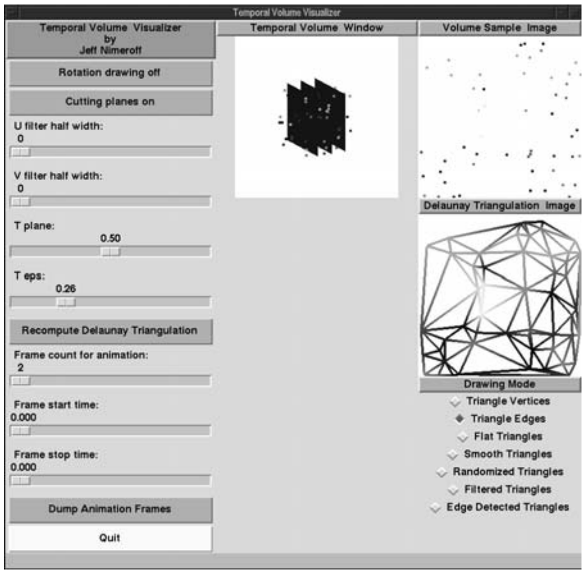
An alternative to the sample projection method that requires a $2D$ triangulation to be executed for each desired image, requires pre-computing the $3D$ tetrahedrization of all the (x, y, t) values (the Delaunay complex). Creating an image from the complex is equivalent to slicing the complex with an (x, y) axis-aligned plane for the desired time t (Figure 7.10). It can be shown that for each simplex (tetrahedron in $3D$) in the complex (connected set of simplices) its intersection with a plane is a quadrilateral [79]. The simplest way to find this quadrilateral is to intersect each edge of the simplex against the plane using a parametric edge representation. Since at most four of the edges of the simplex will intersect the plane, we can construct two triangles with the best aspect ratio and render them directly with the hardware rendering algorithm.

Vertex interpolation

Using the parametric form of the edge equation when computing the planar intersection is useful because the same interpolation value that is used to position the intermediate vertex coordinates can be used to combine the vertex radiances used to shade the triangle interior. Hardware shading can be used to fill in the interior of the triangles once all the simplices have been sliced with the plane. An incremental version of the slicing approach, using working set principles, can be created by keeping a list of the current edges that intersect the current temporal plane. Differential forms of the parametric edge equations can be used to update current edges provided they still span the temporal plane. Edges that are connected to current edges need to be tested to see if they should be included in the current working set.

7.5 Implementation Details

Our multidimensional sampling methods were implemented as C++ modules designed to interface with our modified radiance computation algorithm which was implemented in C [158] and C++ programming languages as a modification of the Rayshade ray tracing algorithm written by Craig Kolb [169].



U filter half width: 0
V filter half width: 0
T plane: 0.50
T eps: 0.26
Frame count for animation: 2
Frame start time: 0.000
Frame stop time: 0.000

Figure 7.11: An interactive reconstruction module. The user can change various parameters like the current time, t_ϵ, and manipulate the view of the temporal file. The user can choose a method for reconstructing from the Delaunay mesh as well as dump an entire animation sequence.

Our interactive temporal file visualization and reconstruction module (Figure 7.11) was written in a combination of C, Tcl/Tk [228], and OpenGL [205].

7.6 Results

This section presents our results, focusing on examples that capture both the temporal effects of object motion (translation and rotation) and higher order visual effects (shadows, reflections, and refractions). It should be noted that our primary focus is not on reconstructing individual high quality frames but instead on reconstructing sequences of frames from a small number of samples that believably present the visual effects that occur due to temporal changes in the radiance field of the environment. The results in this context are very promising.

Each image sequence presented was produced using an individual temporal volume containing $\approx 200,000$ samples (produced without jittering). $200,000+$ radiance samples are not usually adequate for producing even a single high-quality frame, but remember that the goal was the efficient reconstruction of believable temporal effects in globally illuminated environments.

7.6.1 Rotation, reflections, and refractions

Figure 7.12 shows 5 frames of a 25-frame sequence of a textured plane rotating under a mirrored ball and a transparent cone. Since rotation is particularly difficult to reconstruct using image manipulations, this is an appropriate first example. The fact that the objects in the scene exhibit directional lighting effects only adds to the difficulty of the problem. It can be seen that our method handles the example very well. The object motion is smooth, except edge discontinuities, and specular reflections and refractions are clearly visible and move smoothly through the frames. The rendering time was 13 minutes 24.93 seconds to send $200,000+$ samples into a temporal volume. Reconstruction of the individual frames, including file I/O, was approximately 30 seconds per frame on an Indigo2 XZ.

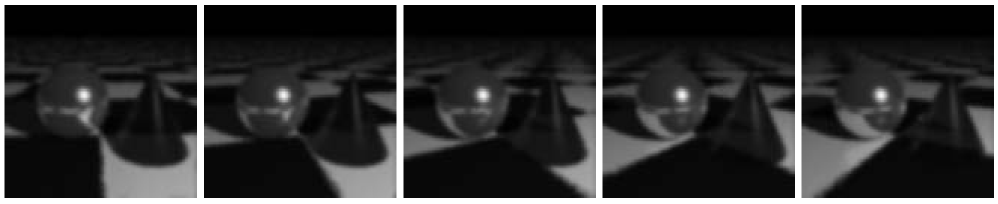

(1) (2) (3) (4) (5)

Figure 7.12: 5 frames of a 25-frame sequence reconstructed from a 200,000 sample temporal volume. This example shows a textured plane rotating under a mirrored sphere and a refractive cone.

7.6.2 Translation, reflections, and refractions

Figure 7.13 shows shows 5 frames of a 25-frame sequence of the same scene with the plane translating under the ball and the cone. First order object motion is handled well as are higher order specular effects, which are visible in the mirrored ball and through the cone. There is even an accurately reconstructed multiple bounce highlight on the floor in front of the mirrored ball. This example shows that edge shimmering is a consistent problem that needs to be remedied. The rendering time was 13 minutes 4.87 seconds to send $200,000+$ samples into a temporal volume. Reconstruction of the individual frames, including file I/O, was approximately 30 seconds per frame on an Indigo2 XZ.

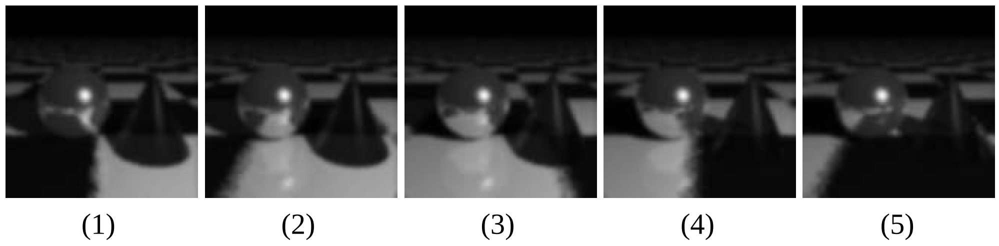


Figure 7.13: 5 frames of a 25-frame sequence reconstructed from a 200,000 sample temporal volume. This example shows a textured plane translating under a mirrored sphere and a refractive cone.

7.6.3 Rotation, translations, reflections, and refractions

Figure 7.14 shows shows 5 frames of a 25-frame sequence of the textured plane rotating and translating under a mirrored ball and a transparent cone. All the effects that were captured in the previous examples were combined here to show the generality of the method. The rendering time was 11 minutes 5.57 seconds to send $200,000+$ samples into a temporal volume. Reconstruction of the individual frames, including file I/O, was approximately 30 seconds per frame.

7.6.4 Visibility and shadows

Figure 7.15 shows 5 frames of a 25-frame sequence of a transparent triangle moving to the right in front of an opaque triangle moving to the left. This example exhibits multiple object motion, changes in visibility and transparent and opaque shadowing. Although edge shimmering is a problem, no extra information is needed when more temporal variances are introduced. The rendering

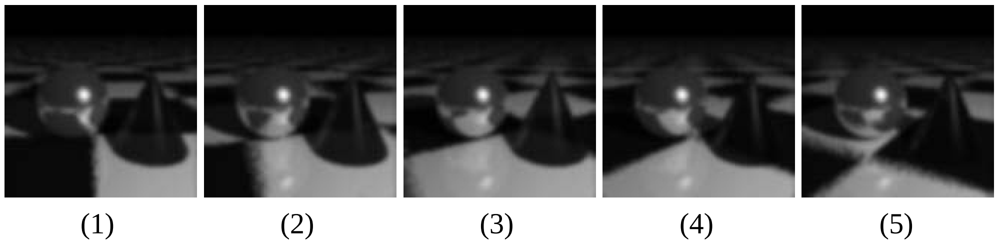


Figure 7.14: 5 frames of a 25-frame sequence reconstructed from a 200,000 sample temporal volume. This example shows a textured plane rotating and translating under a mirrored sphere and a refractive cone.

time was 10 minutes 23.71 seconds to send $200,000+$ samples into a temporal volume. Reconstruction of the individual frames, including file I/O, was approximately 30 seconds per frame on an Indigo2 XZ.

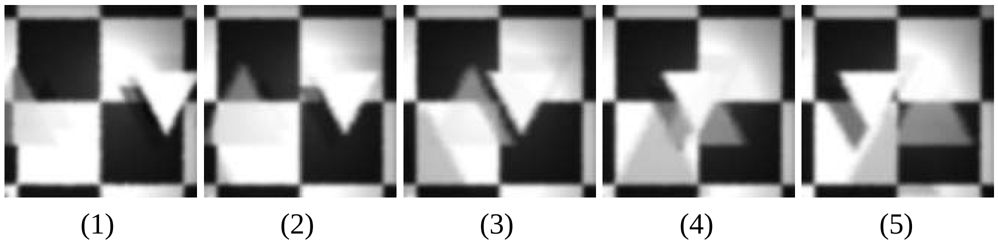


Figure 7.15: 5 frames of a 25-frame sequence reconstructed from a 200,000 sample temporal volume. This example shows two triangles passing by each other with the transparent one on top.

7.6.5 Mirrored Reflections - Apparent Motion

Figure 7.16 shows 5 frames of a 22-frame sequence of a mirror standing perpendicular to a checkered ground plane. The motion in this scene is completely from higher order reflections as the moving object is actually located behind the camera. This is a very convincing example of the power of our method. The $200,000+$ directional and temporal rays we cast into the scene required a total evaluation time of 12 minutes 2.96 seconds. Reconstruction time was approximately 1 second per frame on our RealityEngine using the accumulation buffer and 10x oversampling.

7.6.6 Refraction

Figure 7.17 shows 5 frames of a 25-frame sequence of a transparent cylinder spinning over a checkered ground plane illuminated by three lights. The refractive effects are handled accurately and the

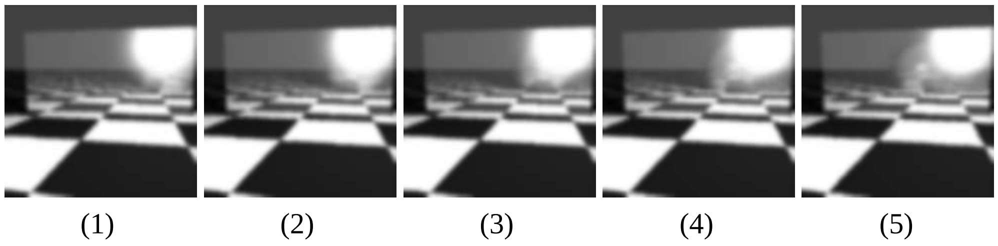


Figure 7.16: 5 frames of a 22-frame sequence reconstructed from a 200,000 sample temporal volume.

edge aliasing is mollified by accumulation buffering. The $200,000+$ directional and temporal rays we cast into the scene required a total evaluation time of 9 minutes 37.52 seconds. Reconstruction time was approximately 1 second per frame on our RealityEngine using the accumulation buffer and 10x oversampling.

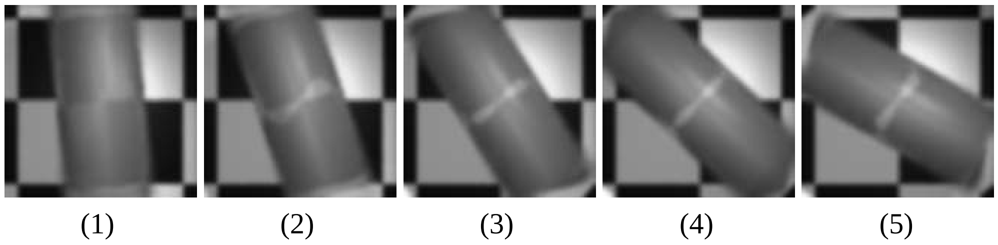


Figure 7.17: 5 frames of a 25-frame sequence reconstructed from a 200,000 sample temporal volume.

7.7 Analysis

The results shown in the previous section are promising. With less sampling than would be done to produce a single high-quality frame, the method constructs a temporal volume that allows for the believable reconstruction of the temporal variances in the radiance field of an environment. Although edges produce a shimmering effect in the frame sequences produced, the ability to reconstruct higher order visual effects such as reflections and refractions shows that this method provides a foundation for rendering methods capable of handling animated environments as easily as handling static environments. The reconstruction methods are efficient and their simplicity opens up many avenues of future research. In order to achieve even more sparsity in the sampling pattern while still maintaining the ability to perform believable reconstruction, research into different methods for achieving point set connectivity is very relevant. The theory of alpha shapes [94], weighted Delaunay triangulations designed to give the "shape" of a point set as opposed to just topological and geometric information, can be used to help refine this reconstruction using the color discrepancy amongst neighboring vertices as weights. Sample velocity information or connectivity constraints amongst small sets of samples might yield more accurate and stable reconstruction algorithms and lead to even sparser sample sets; these are worth investigation. Data quantization techniques will help garner the most compression in the temporal file format. Even with these open areas for future research, the method presented here achieves its goal of performing accurate/believable sample-based motion reconstruction for globally illuminated, animated environments.

7.8 Summary

This chapter presented an approximation to the $3D$ directional and temporal subspace of the radiance field that exists for each position in an environment, a new approach to directional and temporal sampling and reconstruction within complex, globally illuminated, animated environments. The sampling scheme was built around an unstructured, randomized, space-time ray tracing algorithm, a resolution-independent temporal file format and a Delaunay complex reconstruction method. Results show the achievement of believable motion with a small number of directional and temporal samples, about the number required to produce a typical high quality frame. Both first and higher order visual effects, illumination and visibility, are believably reconstructed as this information is pre-computed into the individual samples. Silhouette edges and other discontinuities are more difficult to track but can be addressed with a combination of triangle filtering and image post-processing.

Chapter 8

Conclusion and Discussion

"Love can sweep you off your feet and carry you along in a way you've never known before. But the ride always ends, and you end up feeling lonely and bitter. Wait. It's not love I'm describing. I'm thinking of a monorail."

Jack Handey – Deep Thoughts

Now that we have presented our sample-based framework for efficiently computing global illumination in animated environments, we have to address certain issues related to the design and implementation decisions we made and are going to make in the future. We need to critique the research up to the present and, more importantly, decide which avenues are best to pursue in the future.

8.1 Critiquing the Current State of the Framework

Our work is different from the previous work in global illumination for animated environments in that we pre-compute, into sample-based structures, as much information as possible about the illumination variances in our chosen environments. We were able to show the feasibility and efficiency of our work by deriving a series of representations based on range-images, image volumes, and low-dimensional Delaunay complexes but the answer to each question that was posed generated new questions.

8.1.1 Multi-stage rendering

One of the greatest assets of our framework is also one of the greatest weaknesses. We rely heavily on the progress of research into efficient illumination algorithms. This ultimately gives us a large body of techniques to choose from when constructing our ’accurate enough and fast enough’ global illumination procedures but does not allow us consistent progress. Illumination algorithms evolve slowly and we are, at the present time, tied to this pace. This is not as discouraging as it sounds due to our decision to separate the independent illumination modes, specular and diffuse, into distinct computations. Using separate computations allows us to utilize the current trends in both image space and object space illumination algorithms, not limiting us to one particular class of performance enhancements. As new efficiency techniques are created, we are already positioned to easily take advantage of them through our modular design. This modular design also allow easy integration between our rendering algorithms and our environment sampling methods.

8.1.2 Range-image sequences

Our radiance distribution approximation based on range-image sequences was extremely easy to implement and convergent, as shown in the results, but also highlighted the biggest flaw in our initial assumptions. Accuracy measures, or more specifically our class of coherence metrics, are not spatially and temporal equivalent. The methodologies for comparing “distance” between spatially and temporally varying information need to be very different. This was demonstrated by the extremely high sampling rates required to accurately capture and portray object motion using image blending. Our image space algorithm for tracking spatial illumination variances was very easily tied to an image-based comparison metric but our temporal direct illumination and visibility coherence algorithms faltered using a similar metric. We were not able to find a suitable metric beyond the standard ’place a temporal sample/image using the standard fifteen or thirty frame per second discretization’ used when generating traditional computer animations.

8.1.3 Temporal image volumes

The temporal sampling problems that plagued our range-image approach motivated our use of simple abstract geometric structures (Delaunay triangulation) and adaptive sampling in our volumetric image-based method. This new approach yielded very good results with a very small number of radiance samples but had one basic problem. Since the sampling patterns used were randomized, indexing information needed to be stored in the image volumes along with radiance information. This made the search for specific data values time consuming and drastically increased the storage requirements. Another issue, although not problematic in our $2D$ and $3D$ implementations, is that many abstract computational structures do not scale well into higher dimensions. As global illumination problems start to be represented more abstractly, directly incorporating viewer positioning and lighting configuration information within a single paradigm, the costs associated with our methods may outweigh their simplicity.

8.2 Planning the Future

The are many avenues of future work in the area of sample-based approaches to global illumination in animated environments. Besides correcting or extending our particular framework, there are promising lines of research that are not based on our methods.

8.2.1 Multi-stage rendering

As illumination methods become more efficient and fast hardware-based rendering methods become more realistic, there will be less of a need for intermediate radiance structures. The problem here is that most of the global illumination research is striving for a level of physical accuracy that is not necessary for many applications. The recent move into perceptual metrics [40] for computer-generated imagery [258] will undoubtedly, yield insights not only into how to best judge images, in an a posteriori manner, but also into how to devise accurate and efficient adaptive sampling patterns when performing global illumination.

When we move away from the production of individual images into the production of image sequences the problem is exacerbated. Image space metric are not adequate because it is not single-frame quality we are after as much as smooth, high quality animations. Metrics for image-sequence quality are much harder to construct. On a different note, it may be the case that we should be separating the notion of quality in the images from the quality in the illumination. Object space metrics for illumination quality might prove useful when trying to squeeze out the highest levels of accuracy and efficiency in both illumination computation and walk-through presentation.

8.2.2 Multi-dimensional sampling and reconstruction

As we begin to represent the global illumination and walk-through problem as a sampling and reconstruction problem in higher dimensions, a difficulty arises due to lack of any intuition regarding proximity relationships. It is easy in a Euclidean sense to compute the distance between two samples, and if we are working in a “spatial” domain might even seem meaningful. Should the dimensions, in general, be treated homogeneously? The answer becomes clear when we are trying to formulate a reconstruction mechanism for the samples whose are missing. The Delaunay triangulation connects “neighboring” cells using a proximity constraint based on distance. For a structure as complicated as an environment’s radiance distribution, “distance” is more difficult to measure. In this case the distance metric needs to relate the importance of spatial, directional, and temporal variances and treat the space heterogeneously. Investigating this possibility, the “density” of the different dimensions, is a very interesting and open problem, most likely relying on a framework similar to the theory of alpha shapes [94], weighted Delaunay triangulations designed to give the “shape” of a point set.

We can also aid our algorithms by augmenting the data that we store in our data structures. Motion and connectivity information will be useful when designing new reconstruction algorithms. This, along with derivative and average information should yield much sparser sample sets and a significant storage savings. Data quantization/codebook techniques will also help garner compression savings once the sample data has been stored externally in a file.

8.2.3 Geometry-assisted reconstruction

A problem that we saw with sample-based reconstruction algorithms, like view reconstruction, is that the methods require interpolation wherever data is missing in the destination structure. A powerful new approach is to combine sample-based techniques with geometric methods, creating a powerful mixed-mode walk-through mechanism with the geometry providing the structure and the radiance samples providing the illumination. The projection of the geometry onto the current view configuration can be used as a mask that guides sample-based reconstruction. If we store the unique object identifier with each sample in our radiance structure, then our reconstruction algorithms know exactly which radiance samples can be used to reconstruct which pixels, yielding finer reconstructed results.

8.2.4 Layered radiance textures - spherical harmonic radiance exitance distributions

The most interesting approach that combines geometry and pre-computed sample-based structures is that of the exitant radiance texture. One can picture the simple application of texturing the surfaces in an environment, as in a hardware walk-through system. What if the textures being applied contain illumination information, both direct and indirect? If we have a single texture representing the illumination leaving the surface, exitant radiance as a function of integrated incident radiance, a single texture value can be used to handle the diffuse illumination case.

We can do better if we are willing to repeatedly apply this procedure, to represent our radiance textures using layered images. For each layer of the texture, the value stored represents the integrated incident radiance that exits in a specific “basis” direction. With some chosen number of basis directions on the sphere, we can use spherical harmonics to construct a directional approximation to the exitant radiance function over all directions on the sphere from the layers in the texture. The problem now becomes one of producing the textures.

Producing the textures can be done by walking the object list in the chosen environment. For each object, walk the texture and solve the global illumination problem (integrating the incident radiance and reflecting it along the basis direction). Store this radiance in the appropriate layer of the texture. Do this for each required basis direction. Repeat the process for each cell on the texture on each object. Reconstruction involves a slight modification of the texturing hardware that accounts for exitant radiance construct from the layered texture.

Although this appears to be a very expensive process, the cost should be weighted against the fact that we are constructing a complete radiance representation for the environment allowing complete freedom of movement. Also, it is not known at this time what sampling rate needs to be maintained when constructing these textures. It might be the case that interpolation can be performed within each layer of the texture before using the spherical harmonic reconstruction to combine the layers.

8.3 Contributions Revisited

In this work we showed the feasibility of sample-based approaches to global illumination for animated environments. We derived a series of efficient representations based on time-sequences of range-images, and temporal image volumes and low-dimensional Delaunay complexes. These approximations lay the groundwork for future research into the use of pre-computed radiance data structures as an efficient method for walking through accurately illuminated, complex, time-varying environments.

This thesis first presented the previous work in global illumination research (Chapter 2). Simulating the propagation of light around an environment is an expensive process. An understanding of global illumination theory and its simulation algorithms is necessary if we are to understand why walk-through systems for our realistic environments are not yet commonplace. It also helps us judge the tradeoffs when it comes to choosing a rendering method and environment representation scheme.

We then progressed chronologically through the previous work in interaction/walk-through systems (Chapter 3). The quality of a walk-through system is based on being able to quickly and accurately compute the content of a series of images that are going to be presented to the user. Using the highest quality images produced by today's multi-pass illumination algorithms is not feasible but provides the basis for creating *effective* walk-through systems; those systems that meet the minimum criteria for quality and speed while still presenting the effects the user *needs* to see.

We then introduced our view of an environment's radiance distribution, and a framework for representing this distribution based on an analogy to human vision [2] (Chapter 4). This chapter laid the groundwork for our new rendering mechanism and our sample-based radiance distribution approximations.

In Chapter 5 we discussed a new rendering methodology for efficiently computing global illumination for complex, animated environments. The method we proposed is multi-stage and decouples the processing of the different illumination modes by the importance of the visual effects they compute to the overall presentation of the environment. Our approach is designed to be flexible allowing accuracy to be traded for efficiency and uses an object space radiosity algorithm to compute view-independent indirect illumination and an image space direct illumination algorithm, using the current view configuration to compute the image presented to the user.

Chapter 6 introduced our first approximation to the radiance distribution built around time-sequences of range-images and discussed a series of adaptive coherence algorithms designed to track illumination and visibility variances in an animated environment without computing any illumination information. We presented an object space decomposition algorithm that harnesses the power of the rendering algorithm to efficiently track indirect illumination coherence and compute or interpolate the necessary indirect illumination solutions. We then presented image space algorithms that quickly evaluate spatial and temporal direct illumination and visibility coherence without computing any illumination. These coherence algorithms are designed to produce a "recipe" that tells us which points of view in the environment are important, and which times need to be accounted for in order to adequately represent the radiance field using image-sequences (sets of range-images). The recipe is then passed to our rendering algorithm to produce the images. Once the range-image sequences have been produced, simple and efficient algorithms for reconstructing arbitrary views (image-based temporal and view interpolation) of the environment at any desired times are used.

In Chapter 7, we replaced the time-sequences of range-images with temporal image volumes and showed the feasibility of a sparse spatio-temporal sampling scheme when trying to capture and reconstruct temporal radiance variances using sample-based structures. The method presented here uses sparse spatio-temporal environment sampling, a resolution-independent volumetric temporal file format, and Delaunay complex image reconstruction methods to produce images of the environment at the desired times.

8.4 Summary

Besides presenting and critiquing our approach to using sample-based structures for global illumination in animated environments, we have hypothesized on many different future approaches to using these structures. As fast as the geometric walk-through mechanisms are getting, they still lose when it comes to processing realistic illumination effects. As accurate as today's global illumination methods are, they still lose when it comes to processing efficiency. Since accuracy is ultimately the goal when trying to immerse the user in a synthetic world that is indistinguishable from the real thing, the global illumination solution will not be replaced by a hardware only approach anytime soon. This means that sample-based structures are going to be around for a while. We need to use them judiciously, isolating those regions containing important information from those regions that can be “overlooked.” The future is bright for these pre-computed structures.

Appendix A

Mathematical Preliminaries

In order to perform adequate spatial and temporal reconstruction from image samples, certain mathematical machinery is necessary. Our algorithms are designed to use simple, straightforward combinations of sample values and require a foundation of splines (piecewise cubic Bézier curves), linear mappings and projection, and nearest-neighbor data structures.

A.1 Linear Form Curves - Image Blending (Temporal Reconstruction)

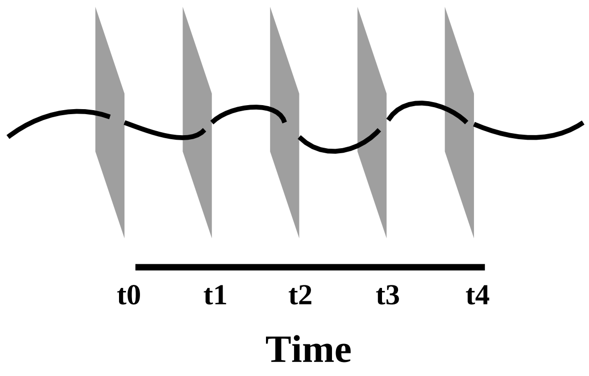


Figure A.1: An image sequence placed along the time axis.

To achieve smooth object motion, animators displace frames by very small increments of time. This high sampling rate allows for all but the most minute of motions, those taking less than $1/30th$ or $1/15th$ of a second, to be adequately captured. No explicit reconstruction mechanism is needed as our persistence of vision blends together the frames when they are being played back. When a sparser sampling exists (Figure A.1), an explicit reconstruction method, blending operation, is required if we want to have any chance of accurately portraying the object motion that resides in the frame sequence.

In order to perform image blending with no a priori information about the movement of the objects being seen, pixel-wise operators are the easiest and most general to use. Pixel-wise image operations combine equivalent pixel locations (r, c) in small frame sequences $(f_{i-j}, \ldots, f_{i+k})$ to generate the appropriate pixel value in the blended frame. Any intra-image processing would be done first to compute the pixel values in the f_i's before applying any inter-image blending. Combining images this way is equivalent to running $2D$ splines, $(time \times color)$, through the corresponding pixel locations in the sequences being blended.

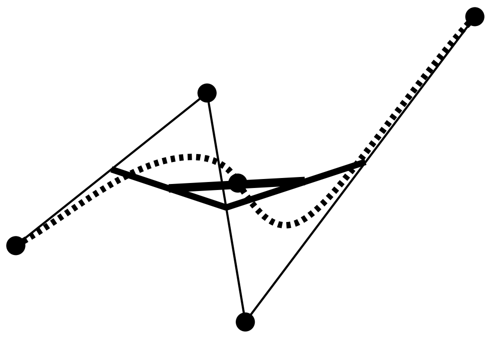


Figure A.2: A cubic Bézier curve computed from linear segments.

Our piecewise cubic Bézier formulation [95] is an implementation of linear form curves [278]. Linear form curves use multilinearity and symmetry to build simple general curves out of linear segments (Figure A.2). If we define a *line* function as,

$$line(\mathbf{t}; p_i, p_{i+1}) = [\ p_i \quad p_{i+1}\ ] \cdot \mathbf{t} \ \ ,$$

with the constraints that $\mathbf{t} = [\ 1-t \quad t\ ]^T$ and $t \,\epsilon\, [p_i, p_{i+1}]$, we can use a barycentric formulation,

$$bary(t; a_i, a_{i+1}) = \frac{1}{a_{i+1} - a_i} \begin{bmatrix} a_{i+1} - t \\ t - a_i \end{bmatrix} \ \ ,$$

to map t onto the interval $[a_i, a_{i+1}]$. Combining the *line* and *bary* functions yields a method for generating a parametric line between p_i and p_{i+1} for a parameter running from a_i to a_{i+1},

$$line(bary(t; a_i, a_{i+1}, 0); p_i, p_{i+1}) \ \ .$$

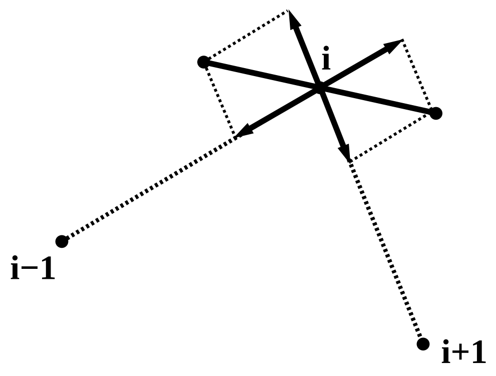


Figure A.3: Computing the tangents (interior points) around a control point of a cubic Bézier curve.

The piecewise cubic Bézier formulation is useful in that the endpoints of the segments are interpolated and the complete curve can be easily constructed by computing all the segments of the desired interval. The problem with using the formulation is that more data is needed than just the values to be interpolated. Two interior points for each segment (Figure A.3) are needed to define the directions that the curve will take exiting the initial point and entering the final point of the segment. These interior points cannot be chosen arbitrarily. Care must be taken in order to maintain continuity constraints, either functional (magnitude and direction of the derivative) or geometric (direction of the derivative), as the pieces are "sewn" together, so as to not introduce visual artifacts during blending. We construct the interior Bézier points ($p_i^{incoming}$ and $p_i^{outgoing}$) using the vectors connecting p_{i-1} to p_i and p_i to p_{i+1},

$$\begin{aligned} p_i^{incoming} &= p_i - \frac{1}{3}(p_i - p_{i-1}) - \frac{1}{3}(p_{i+1} - p_i) \\ p_i^{outgoing} &= p_i + \frac{1}{3}(p_i - p_{i-1}) + \frac{1}{3}(p_{i+1} - p_i) \ . \end{aligned}$$

The end conditions for the curve ($p_0^{outgoing}$ and $p_n^{incoming}$) are computed using

$$\begin{aligned} p_0^{outgoing} &= p_0 + \frac{1}{3}(p_1 - p_0) - \frac{1}{3}(p_2 - p_1) \\ p_n^{incoming} &= p_n - \frac{1}{3}(p_n - p_{n-1}) + \frac{1}{3}(p_{n-1} - p_{n-2}) \ . \end{aligned}$$

Once the interior points are chosen for each segment of our Bézier blending curve, the interior values for any time in the desired time interval can be constructed using

$$line(bary(t; a_i, a_{i+1}),$$
$$\quad line(bary(t; a_i, a_{i+1}),$$

$$
\begin{aligned}
&\qquad line(bary(t; a_i, a_{i+1}), p_i, p_i^{outgoing}),\\
&\qquad line(bary(t; a_i, a_{i+1}), p_i^{outgoing}, p_{i+1}^{incoming})),\\
&line(bary(t; a_i, a_{i+1}),\\
&\qquad line(bary(t; a_i, a_{i+1}), p_i^{outgoing}, p_{i+1}^{incoming}),\\
&\qquad line(bary(t; a_i, a_{i+1}), p_{i+1}^{incoming}, p_{i+1})))
\end{aligned}
$$

A.2 Projection - View Interpolation (Spatial Reconstruction)

Usually, images are presented as the end result of the rendering/global illumination process. For our particular application, the individual images are just a means to the end. In order to perform spatial reconstruction, to compute walk-throughs of the desired environment, we need to unify the spatial relationship between the images (*image space*) we have computed, and the world coordinate system (*world space*) of the environment that the images capture [293]. The images, then, can be used to present/re-create the information that would be seen in a desired view of the environment without resorting back to a geometric representation. To do this, we need to describe the position and orientation of the synthetic camera used to generate the image as well as any depth information associated with the individual pixels in the image. The depth information allows us to unambiguously resolve perspective foreshortening, to undo the perspective, and provides our link between the space of the image and the space of the environment. Our job, then, is convert pixel locations into world space locations, and vice-versa. This is facilitated by using an intermediate representation, *camera space*.

A.2.1 Change of Basis - Camera $\Leftrightarrow$ World

Camera space is represented, without loss of generality, by an orthonormal set of left-handed axes, $\mathbf{u} = (u_x, u_y, u_z)$, $\mathbf{v} = (v_x, v_y, v_z)$, and $\mathbf{n} = (n_x, n_y, n_z)$, where $\mathbf{u}$ and $\mathbf{v}$ parameterize the image plane and $\mathbf{n}$ parameterizes depth. This coordinate frame is then placed in the environment relative to $\mathbf{r} = (r_x, r_y, r_z)$, the *view reference point*. The only piece of information missing from this representation is $\mathbf{e} = (e_u, e_v, e_n)$, the *eye point* or *center of projection*, which is often placed on the $\mathbf{n}$-axis at some signed distance E from the view reference point (Figure A.4).

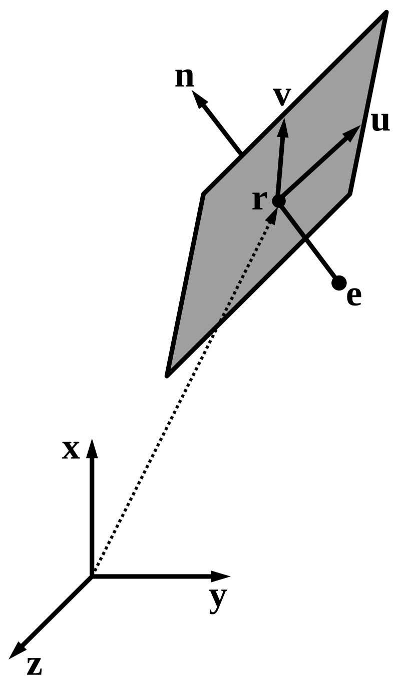


Figure A.4: The relationship between world and camera space.

So now we have a coordinate system that is specific to any camera we can place in the environment. How do we convert between this arbitrary camera space, C and the world space, W? The simplest way is to construct a linear mapping, M_{33}, that aligns the axes of the proposed camera space to the axes of world space:

$$\begin{aligned} W &= CM_{33}, \\ \begin{bmatrix} \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \end{bmatrix} &= \begin{bmatrix} \begin{bmatrix} u_x & u_y & u_z \end{bmatrix} \\ \begin{bmatrix} v_x & v_y & v_z \end{bmatrix} \\ \begin{bmatrix} n_x & n_y & n_z \end{bmatrix} \end{bmatrix} \begin{bmatrix} M_{33}[0,0] & M_{33}[0,1] & M_{33}[0,2] \\ M_{33}[1,0] & M_{33}[1,1] & M_{33}[1,2] \\ M_{33}[2,0] & M_{33}[2,1] & M_{33}[2,2] \end{bmatrix}. \end{aligned}$$

Since $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{n}$ are orthonormal, the mapping M_{33} is just C^T:

$$\begin{bmatrix} \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} & \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} & \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} \end{bmatrix}.$$

The only bit of information missing from this derivation is that the coordinate system origins are not coincident. The mapping M_{33} goes from camera space to world space using a (3×3)

transformation matrix. These matrices are insufficient for representing translations in $3D$. The matrix M_{33} represents a pure rotation and as such must be augmented with a separate translation vector to make the coordinate origins coincident (an *affine mapping*). This yields the following mapping from a camera space position, $\mathbf{c} = (u, v, n)$ to a world space position, $\mathbf{w} = (x, y, z)$:

$$\begin{aligned} \mathbf{w} &= \mathbf{c}M_{33} + \mathbf{r}, \\ (\, x, y, z \,) &= (\, u, v, n \,)\, M_{33} + \mathbf{r}. \end{aligned}$$

Homogeneous coordinates unify the rotational and translational aspects of our camera-to-world mapping allowing us to represent it with a (4×4) matrix, M_{44}:

$$(\, x, y, z, 1 \,) = (\, u, v, n, 1 \,) \begin{bmatrix} \begin{bmatrix} u_x \\ u_y \\ u_z \\ r_x \end{bmatrix} & \begin{bmatrix} v_x \\ v_y \\ v_z \\ r_y \end{bmatrix} & \begin{bmatrix} n_x \\ n_y \\ n_z \\ r_z \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \end{bmatrix}.$$

To construct the mapping from world space to camera space, need to invert the process:

$$\begin{aligned} \mathbf{c} &= (\mathbf{w} - \mathbf{r})M_{33}^{-1}, \\ (\, u, v, n \,) &= ((\, x, y, z \,) - \mathbf{r})M_{33}^{-1}, \end{aligned}$$

which in homogeneous coordinates is represented as

$$(\, u, v, n, 1 \,) = (\, x, y, z, 1 \,) \begin{bmatrix} \begin{bmatrix} u_x \\ u_y \\ u_z \\ r_x \end{bmatrix} & \begin{bmatrix} v_x \\ v_y \\ v_z \\ r_y \end{bmatrix} & \begin{bmatrix} n_x \\ n_y \\ n_z \\ r_z \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \end{bmatrix}^{-1}.$$

A.2.2 $3D$ Projected Space - Camera ⇔ Image

When working with images, we do not really have a simple camera space representation of the visible surfaces in the environment but instead have a projective representation of camera space after a perspective transformation has been applied (image space). We can, however, undo the perspective projection if we have depth information at each pixel and know the eye point. The process involves following individual projectors from the eye point through the pixel locations on

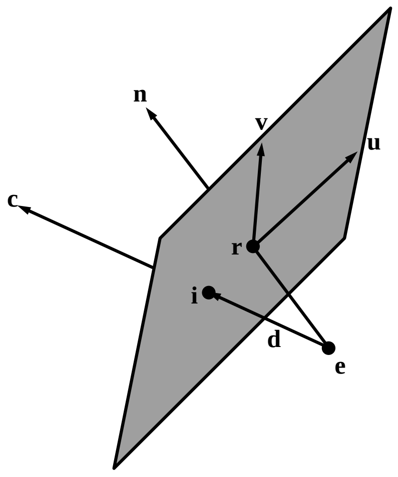


Figure A.5: The relationship between camera and image space.

the image plane (in image space) into camera space until a distance equal to the depth for the current pixel has been reached.

A camera space location, $\mathbf{c} = (u, v, n)$ is represented on the image plane, which is situated in the $n = 0$ plane, as $\mathbf{i} = (u^*, v^*, 0)$ where (u^*, v^*) is the standard image space indexing scheme (Figure A.5). A projector from the eye point, $\mathbf{e} = (e_u, e_v, e_n)$, through the point on the image plane is given by $\mathbf{d} = (u^* - e_u, v^* - e_v, -e_n)$. Since the depth information, t, included with each pixel is computed in camera space units, we need to normalize this vector,$\hat{\mathbf{d}}$, in order to project to the correct location in camera space. This yields the following mapping:

$$\begin{aligned} \mathbf{c} &= \mathbf{e} + t\hat{\mathbf{d}}, \\ (\,u, v, n\,) &= (\,e_u, e_v, e_n\,) + t\,(\,u^* - e_u, v^* - e_v, -e_n\,)\,. \end{aligned}$$

If we invert this procedure, to go from camera space to image space, we are left with the standard perspective problem. This mapping is found in most computer graphics texts [100] and requires a simple “similar triangle” argument to construct.

A.2.3 World $\Leftrightarrow$ Camera $\Leftrightarrow$ Image

We can combine the two mappings into a procedure for converting pixel locations with depth information into a set of world space locations, with associated illumination information. For any desired view, these world space locations can be projected, using the inverse mappings described, onto this new view as if they were "geometry" in the environment. This procedure forms the basis for our view interpolation method which is a simple combination of undoing the projections onto the pre-computed image planes, re-projecting the locations/values onto the desired view/image plane, and filtering the result.

A.3 Delaunay Triangulation (Spatial and Temporal Reconstruction)

Reconstructing from our temporal image volumes requires a method for connecting points that are inherently unstructured. This topological information (connectedness/compactness/nearness) is created using a triangulation algorithm that converts the points into a set of n-dimensional *simplices* (triangles, tetrahedra, etc.). This set of simplices completely bounds the *convex hull* of the points (linear combinations of the points).

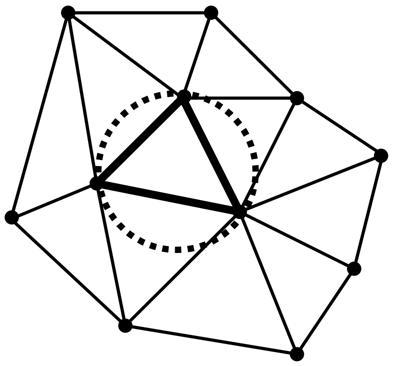


Figure A.6: A Delaunay triangulation and empty circumcircle property.

To be more specific, an n-dimensional triangulation of a point set $P = (p_1, \ldots, p_n)$ is a collection of n-dimensional simplices whose defining points lie in P. The simplices do not intersect one another and share only boundary features such as edges or faces (often referred to as *flats*). The *Delaunay* triangulation is particularly useful in that it has the property that any n-dimensional simplex contains no other points of P except the $n + 1$ defining points of the simplex (Figure A.6).

The Delaunay triangulation (Figure A.6) has many interesting properties. In two dimensions, the Delaunay triangulation has been shown to be optimal relative to the minimum interior angle metric. The minimum interior angle of a simplex in a Delaunay triangulation is greater than or equal to the minimum interior angle of any other possible triangulation. The Delaunay triangulation is the *dual* of another important construction in computational geometry, the *Voronoi*

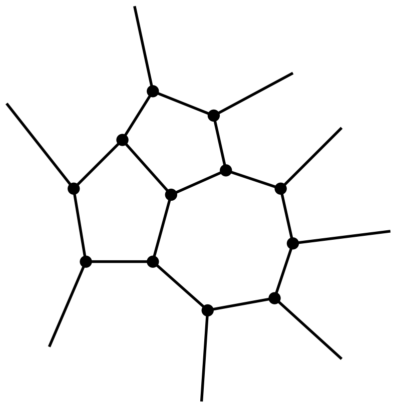


Figure A.7: The Voronoi diagram of the preceding Delaunay triangulation.

diagram/tessellation (Figure A.7). The Voronoi tessellation is a spatial decomposition where each tile represents the space closest to a point p_i. An n-dimensional Delaunay triangulation can be constructed from the Voronoi diagram by creating edges between Voronoi cells that share common $n-1$-dimensional boundaries (e.g. faces in $3D$ and edges in $2D$). Conversely, the vertices of the Voronoi diagram are located at the centers of the Delaunay circumcircles.

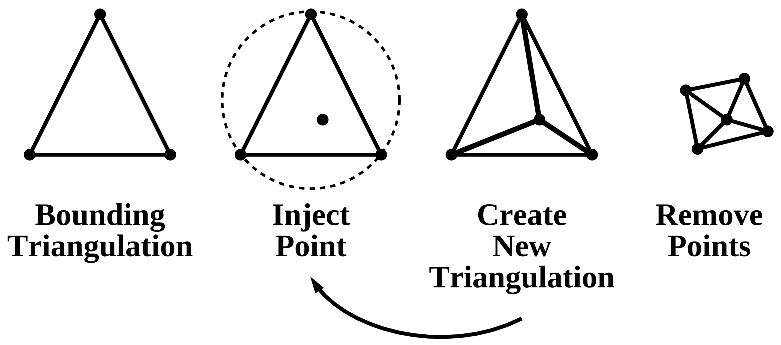
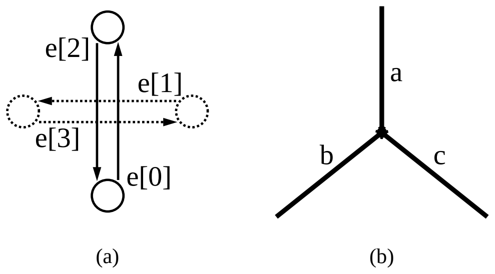


Figure A.8: The Watson/Bowyer method of constructing the Delaunay triangulation.

The Delaunay triangulation can be constructed using a variety techniques. A particularly elegant technique, introduced independently in 1981 by Watson [329] and Bowyer [49] (Figure A.8) begins by initially constructing a Delaunay triangulation that strictly bounds the point set P (a *bounding* triangulation). This bounding simplex is usually a triangle ($2D$) or tetrahedron ($3D$). Each point that has to be introduced into the structure is found within the circumcircles of a particular set of simplices. These simplices are then deleted leaving a disconnected region in the triangulation. After deleting the simplices, the $n-1$-dimensional faces on the boundary of the hole and the newly introduced point are used to construct a modified triangulation to connect up the disconnected region.



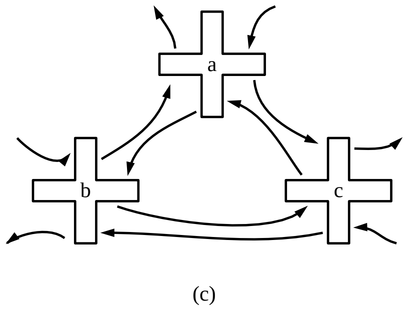


Figure A.9: The quad-edge data structure.

A more efficient $2D$ incremental Delaunay triangulation algorithm is based on the $quad-edge$ data structure presented by Guibas and Stolfi [127, 184]. The quad-edge data structure was designed for representing general subdivisions of orientable manifolds. It is similar to the winged-edge data structure [24], but it simultaneously represents both the subdivision and its dual. Each quad-edge record groups together four directly edges corresponding to a single undirected edge in the subdivision and to its dual edge (Figure A.9a). Pointers are used to store the next counter-clockwise edge around the origin of the current edge as well as any geometrical and non-topological information that is necessary. The quad-edge structure is illustrated in Figure A.9b and Figure A.9c. These figures show how three edges incident on the same vertex are represented using the quad-edge structure. The vertex itself corresponds to the inner cycle of pointers in Figure A.9c. The remaining three cycles correspond to the three faces meeting at the vertex.

The complexity of the quad-edge data is hidden in two intuitive operators. A primitive to create an edge (*make-edge*) is needed, along with a topological operator (*splice*) which can be used to link disjoint edges together as well as to break two linked edges apart. It is important to note that the *splice* operator is its own inverse. Guibas and Stolfi show that two operators together can be used to construct any subdivision.

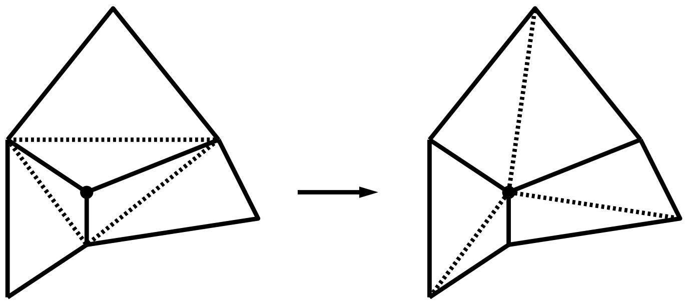


Figure A.10: The Delaunay Triangulation. Candidate edges flipped after point insertion.

As with the previously mentioned algorithm, this incremental Delaunay triangulation algorithm starts with triangle large enough to contain all of the points in the input. Points are added into the triangulation one at a time, maintaining the *Delaunay criteria*. Figure A.10 illustrates the point insertion process. First, the triangle containing the new point p is located. New edges are created to connect p to the vertices of the containing triangle. The old edges of the triangle are inspected to verify that they still satisfy the empty circumcircle condition. If the condition is satisfied, the edge in question remains unchanged. If the condition is violated, the offending edge is flipped. Flipping an offending edge creates two new candidates for inspection and the inspection process repeats.

Th complexity of this algorithm depends largely on the underlying structures used to perform the geometric search. Naive searching to find the containing triangle for a particular point can be performed in $O(n)$. If complicated structures are used, this search can be performed in optimal $O(\lg n)$ time. Once the containing triangle is located, the insertion of the new point can require $O(n)$ edges to be flipped. The expected running times of these phases can be reduced to $O(\sqrt{n})$ and $O(1)$ respectively if guarantees can be made about the distribution of the data values.

Bibliography

[1] CIE Techinical Committee 4.2. Standardization of luminance distribution on clear skies. CIE Publication No. 22, Commission International de L'Eclairaze, Paris, 1973.

[2] E. Adelson and J. Bergen. The plenoptic function and the elements of early vision. In *Computational Models of Visual Processing*, pages 1–20. MIT Press, Cambridge, MA, 1991.

[3] J. Airey, J. Rohlf, and F. Brooks, Jr. Towards image realism with interactive update rates in complex virtual building environments. *Computer Graphics (1990 Symposium on Interactive 3D Graphics)*, 24(2):41–50, March 1990.

[4] K. Akeley. Realityengine graphics. *Computer Graphics (SIGGRAPH '93 Proceedings)*, 27(4):109–116, August 1993.

[5] K. Akeley and T. Jermoluk. High-performance polygon rendering. *Computer Graphics (SIGGRAPH '88 Proceedings)*, 22(4):239–246, August 1988.

[6] A. Appel. Some techniques for shading machine renderings of solids. In *AFIPS 1968 Spring Joint Computer Conference*, volume 32, pages 37–45, 1968.

[7] J. Arvo. The irradiance jacobian for partially occluded polyhedral sources. *Computer Graphics (SIGGRAPH '94 Proceedings)*, 28(4):343–350, July 1994.

[8] J. Arvo. *Analytic Methods for Simulated Light Transport*. PhD thesis, Department of Computer Science, Yale University, December 1995.

[9] J. Arvo, K. Torrance, and B. Smits. A framework for the analysis of error in global illumination algorithms. *Computer Graphics (SIGGRAPH '94 Proceedings)*, 28(4):75–84, July 1994.

[10] P. Atherton, K. Weiler, and D. Greenberg. Polygon shadow generation. *Computer Graphics (SIGGRAPH '78 Proceedings)*, 12(3):275–281, 1978.

[11] L. Aupperle and P. Hanrahan. A hierarchical illumination algorithm for surfaces with glossy reflection. *Computer Graphics (SIGGRAPH '93 Proceedings)*, 27(4):155–162, August 1993.

[12] L. Aupperle and P. Hanrahan. Importance and discrete three point transport. *Fourth Eurographics Workshop on Rendering*, pages 85–94, June 1993.

[13] N. Badler, B. Webber, J. Kalita, and J. Esakov. Animation from instructions. In N. Badler, B. Barsky, and D. Zeltzer, editors, *Making them move: mechanics, control, and animation of articulated figures*, pages 51–93. Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1991.

[14] H. Baltes. *Inverse Source Problems in Optics*. Springer-Verlag, New York, NY, 1978.

[15] B. Barber, D. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hulls. Submitted to the ACM Transactions on Mathematical Software, May 1995.

[16] J. Barnes and P. Hut. A hierarchical $o(n \log n)$ force-calculation algorithm. *Nature*, 324:446–449, 1986.

[17] B. Barsky. An intuitive description of parametric splines in computer graphics. In *Proceedings of Graphics Interface '90*, pages 252–266. Canadian Information Processing Society, May 1990.

[18] R. Bartels, J. Beatty, and B. Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1987.

[19] R. Bartels and I. Hardtke. Speed adjustment for key-frame interpolation. In *Proceedings of Graphics Interface '89*, pages 14–19. Canadian Information Processing Society, June 1989.

[20] D. Baum, S. Mann, K. Smith, and J. Winget. Making radiosity usable: Automatic preprocessing and meshing techniques for the generation of accurate radiosity solutions. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):51–60, July 1991.

[21] D. Baum, H. Rushmeier, and J. Winget. Improving radiosity solutions through the use of analytically determined form-factors. *Computer Graphics (SIGGRAPH '89 Proceedings)*, 23(3):325–334, July 1989.

[22] D. Baum, J. Wallace, and M. Cohen. The back-buffer algorithm: an extension of the radiosity method to dynamic environments. *The Visual Computer*, 2(5):298–306, 1986.

[23] D. Baum and J. Winget. Real time radiosity through parallel processing and hardware acceleration. *Computer Graphics (1990 Symposium on Interactive 3D Graphics)*, 24(2):67–75, March 1990.

[24] B. Baumgart. Geometric modeling for computer vision. AIM-249, STA CS-74-463, CS Dept, Stanford U., October 1974.

[25] E. Becker, G. Carey, and J. Oden. *Finite Elements, An Introduction, Volume 1*. Prentice-Hall Publishing Company, Englewood Cliffs, NJ, 1981.

[26] T. Beier and S. Neely. Feature-based image metamorphosis. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(2):35–42, July 1992.

[27] J. Beran-Koehn and M. Pavicic. A cubic tetrahedral adaption of the hemi-cube algorithm. In J. R. Arvo, editor, *Graphics Gems II*, pages 299–302. Academic Press, San Diego, CA, 1991.

[28] J. Beran-Koehn and M. Pavicic. Delta form-factor calculations for the cubic tetrahedral algorithm. In D. B. Kirk, editor, *Graphics Gems III*, pages 324–328. Academic Press, San Diego, CA, 1992.

[29] M. Berger. *Geometry I*. Springer-Verlag, 1987.

[30] L. Bergman, H. Fuchs, E. Grant, and S. Spach. Image rendering by adaptive refinement. *Computer Graphics (SIGGRAPH '88 Proceedings)*, 22(4):29–38, August 1988.

[31] G. Beylkin, R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms. *Communications on Pure and Applied Mathematics*, 44:141–183, 1991.

[32] G. Beylkin, R. Coifman, and V. Rokhlin. Wavelets in numerical analysis. In G. Beylkin, R. Coifman, I. Daubechies, S. Mallet, Y. Meyer, L. Raphael, and B. Rushkei, editors, *Wavelets and Their Applications*, pages 181–210. Jones and Bartlett, Cambridge, MA, 1992.

[33] N. Bhate and A. Tokuta. Photorealistic volume rendering of media with directional scattering. *Third Eurographics Workshop on Rendering*, pages 227–245, May 1992.

[34] K. Blanton. A new approach for flight simulator visual systems. In *Simulators IV, Proceedings of the SCCS Simulators Conference*, pages 229–233, 1987.

[35] J. Blinn. Models of light reflection for computer synthesized pictures. *Computer Graphics (SIGGRAPH '77 Proceedings)*, 11(2):192–198, July 1977.

[36] J. Blinn. Simulation of wrinkled surfaces. *Computer Graphics (SIGGRAPH '78 Proceedings)*, 12(3):286–292, August 1978.

[37] J. Blinn. How to write a paper for SIGGRAPH. *IEEE Computer Graphics and Applications*, 7(12):62–64, December 1987.

[38] J. Blinn. Triage tables. *IEEE Computer Graphics and Applications*, 10(1):70–75, January 1990.

[39] J. Blinn and M. Newell. Texture and reflection in computer generated images. *Communications of the ACM*, 19(10):542–547, 1976.

[40] K. Boff and J. Lincoln. *Engineering Data Compendium: Human Perception and Performance, Vol. 1*. Wright-Patterson Air Force Base, 1988.

[41] C. Borel and S. Gerstl. Simulation of partially obscured scenes using the radiosity method. In *Proceedings SPIE on Characterization, Propagation, and Simulation of Sources and Backgrounds*, volume 1486, pages 271–277, April 1991.

[42] C. Borel, S. Gerstl, and B. Powers. The radiosity method in optical remote sensing of structured 3-D surfaces. *Remote Sensing of Environment*, 36(1):13–44, 1991.

[43] K. Bouatouch and P. Tellier. A two-pass physics-based global lighting model. In *Proceedings of Graphics Interface '92*, pages 319–328. Canadian Information Processing Society, May 1992.

[44] W. Bouknight. A procedure for generation of three-dimensional half-toned computer graphics presentations. *Communications of the ACM*, 13(9):292–301, September 1970.

[45] W. Bouknight and K. Kelley. An algorithm for producing half-tone computer graphics presentations with shadows and moveable light sources. In *Proceedings of the AFIPS 1970 Spring Joint Computer Conference*, volume 36, pages 1–10, 1970.

[46] C. Bouville, K. Bouatouch, P. Tellier, and X. Pueyo. A theoretical analysis of global illumination models. *First Eurographics Workshop on Rendering*, June 1990.

[47] C. Bouville, J.Dubois, I. Marchal, and M. Viaud. Monte-carlo integration applied to an illumination model. *Eurographics '88*, pages 483–98, September 1988.

[48] C. Bouville, P. Tellier, and K. Bouatouch. Low sampling densities using a psychovisual approach. In *Eurographics '91*, pages 167–182. North-Holland, September 1991.

[49] A. Bowyer. Computing dirichlet tessellations. *The Computer Journal*, 24(2):162–166, 1981.

[50] F. Brooks, Jr. Walkthrough — a dynamic graphics system for simulating virtual buildings. *Computer Graphics (1986 Symposium on Interactive 3D Graphics)*, 20(2):9–21, October 1986.

[51] C. Buckalew and D. Fussell. Illumination networks: Fast realistic rendering with general reflectance functions. *Computer Graphics (SIGGRAPH '89 Proceedings)*, 23(3):89–98, July 1989.

[52] I. Busbridge. *The Mathematics of Radiative Transfer*. Cambridge University Press, Bristol, 1960.

[53] J. Caird. The perception of spatial layout in visually simulated enviroments. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume 2, pages 1159–1162, 1992.

[54] A. Campbell, III and Donald Fussell. Adaptive mesh generation for global diffuse illumination. *Computer Graphics (SIGGRAPH '90 Proceedings)*, 24(4):155–164, August 1990.

[55] E. Catmull. *A Subdivision Algorithm for Computer Display of Curved Surfaces*. Ph.d. thesis, Department of Computer Science, University of Utah, December 1974.

[56] S. E. Chen. A progressive radiosity method and its implementation in a distributed processing environment. Masters thesis, Program of Computer Graphics, Cornell University, January 1989.

[57] S. E. Chen. Incremental radiosity: An extension of progressive radiosity to an interactive image synthesis system. *Computer Graphics (SIGGRAPH '90 Proceedings)*, 24(4):135–144, August 1990.

[58] S. E. Chen. Quicktime vr - an image-based approach to virtual environment navigation. *Computer Graphics (SIGGRAPH '95 Proceedings)*, 29(4):29–38, July 1995.

[59] S. E. Chen, H. Rushmeier, G. Miller, and D. Turner. A progressive multi-pass method for global illumination. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):165–174, July 1991.

[60] S. E. Chen and L. Williams. View interpolation for image synthesis. *Computer Graphics (SIGGRAPH '93 Proceedings)*, 27(4):279–288, August 1993.

[61] P. Christensen, D. Salesin, and T. DeRose. A continuous adjoint formulation for radiance transport. *Fourth Eurographics Workshop on Rendering*, pages 95–104, June 1993.

[62] P. Christensen, E. Stollnitz, D. Salesin, and T. DeRose. Wavelet radiance. *Fifth Eurographics Workshop on Rendering*, pages 0–0, June 1994.

[63] J. Clark. The geometry engine: A vlsi geometry system for graphics. *Computer Graphics (SIGGRAPH '82 Proceedings)*, 16(2):127–133, July 1982.

[64] K. Clarkson and P. Shor. Applications of random sampling in computational geometry, ii. *Discrete Computational Geometry*, 4:387–421, 1989.

[65] M. Cohen. A radiosity method for the realistic image synthesis of complex diffuse environments. Masters thesis, Program of Computer Graphics, Cornell University, August 1985.

[66] M. Cohen. Radiosity. In D. E. Rogers and R. A. Earnshaw, editors, *State of the Art in Computer Graphics: Visualization and Modeling*, pages 59–90. Springer-Verlag, New York, NY, 1991.

[67] M. Cohen, S. E. Chen, J. Wallace, and D. Greenberg. A progressive refinement approach to fast radiosity image generation. *Computer Graphics (SIGGRAPH ’88 Proceedings)*, 22(3):75–84, August 1988.

[68] M. Cohen, D. Greenberg, D. Immel, and P. Brock. An efficient radiosity approach for realistic image synthesis. *IEEE Computer Graphics and Applications*, 6(3):26–35, March 1986.

[69] M. Cohen and Donald Greenberg. The Hemi-Cube: A radiosity solution for complex environments. *Computer Graphics (SIGGRAPH ’85 Proceedings)*, 19(3):31–40, August 1985.

[70] M. Cohen and J. Wallace. *Radiosity and Realistic Image Synthesis*. Academic Press, Cambridge, MA, 1993.

[71] IES Daylighting Committee. Recommended practice of daylighting. *Lighting Design and Application*, 9(2):45–58, 1979.

[72] S. Conte and C. de Boor. *Elementary Numerical Analysis*. McGraw-Hill, New York, NY, 1980.

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

[74] R. Cook, T. Porter, and L. Carpenter. Distributed ray tracing. *Computer Graphics (SIGGRAPH ’84 Proceedings)*, 18(3):137–145, July 1984.

[75] R. Cook and K. Torrance. A reflectance model for computer graphics. *ACM Transactions on Graphics*, 1(1):7–24, January 1982.

[76] S. Coons. Transformations and matrices. Notes for summer school on Computer Graphics for Design ers, U. of Michigan, June 1967.

[77] S. Coons. Constrained least squares. *Computers & Graphics*, 3:43–47, 1978.

[78] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. McGraw-Hill, New York, NY, 1990.

[79] H. Coxeter. *Introduction to Geometry*. John Wiley & Sons, Inc., 1961.

[80] C. Csuri. Computer animation. *Computer Graphics (SIGGRAPH ’75 Proceedings)*, 9(2):92–101, June 1975.

[81] C. Csuri. Art and animation. *IEEE Computer Graphics and Applications*, 11(1):30–35, January 1991.

[82] C. de Boor. *A Practical Guide to Splines*. Springer-Verlag, 1978.

[83] L. Delves and J. Mohammed. *Computational Methods for Integral Equations*. Cambridge University Press, Cambridge, UK, 1985.

[84] P. Diefenbach. *Multi-pass Pipeline Rendering: Interaction and Realism through Hardware Provisions*. PhD thesis, University of Pennsylvania, March 1996.

[85] M. Dippé and E. Wold. Antialiasing through stochastic sampling. *Computer Graphics (SIGGRAPH ’85 Proceedings)*, 19(3):69–78, July 1985.

[86] M. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall Publishing Company, 1976.

[87] J. Dorsey. *Computer Graphics Techniques for Opera Lighting Design and Simulation*. Ph.d. thesis, Program of Computer Graphics, Cornell University, January 1993.

[88] J. Dorsey, J. Arvo, and D. Greenberg. Interactive design of complex time-dependent lighting. *IEEE Computer Graphics and Applications*, 15(2):26–36, March 1995.

[89] J. Dorsey, F. Sillion, and D. Greenberg. Design and simulation of opera lighting and projection effects. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):41–50, July 1991.

[90] G. Drettakis and E. Fiume. Structure-directed sampling, reconstruction, and data representation for global illumination. *Second Eurographics Workshop on Rendering*, pages 189–201, May 1991.

[91] S. Drucker and P. Schröder. Fast radiosity using a data parallel architecture. *Third Eurographics Workshop on Rendering*, pages 247–258, May 1992.

[92] W. Eddy. A new convex hull algorithm for planar sets. *ACM Transactions on Mathematical Software*, 1977.

[93] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer-Verlag, New York, NY, 1987.

[94] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29(4):551–559, July 1983.

[95] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, San Diego, CA, 1990.

[96] I. Faux and M. Pratt. *Computational Geometry for Design and Manufacture*. Ellis Horwood Ltd., 1979.

[97] M. Feda and W. Purgathofer. Progressive refinement radiosity on a transputer network. *Second Eurographics Workshop on Rendering*, May 1991.

[98] M. Feda and W. Purgathofer. Accelerating radiosity by overshooting. *Third Eurographics Workshop on Rendering*, pages 21–32, May 1992.

[99] R. Feynman, R. Leighton, and M. Sands. *The Feynman Lectures on Physics*. Addison-Wesley, 1963.

[100] J. Foley, A. van Dam, S. Feiner, and J. Hughes. *Computer Graphics: Principles and Practices (2nd Edition)*. Addison-Wesley, 1990.

[101] D. Forsyth, C. Yang, and K. Teo. Efficient radiosity in dynamic environments. *Fifth Eurographics Workshop on Rendering*, pages 313–324, June 1994.

[102] A. Fournier, A. Gunawan, and C. Romanzin. Common illumination between real and computer generated scenes. In *Proceedings of Graphics Interface '93*, pages 254–262. Canadian Information Processing Society, 1993.

[103] W. Freeman and E. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, September 1991.

[104] H. Fuchs, Z. Kedem, and B. Naylor. On visible surface generation by a priori tree structures. *Computer Graphics (SIGGRAPH '80 Proceedings)*, 14(3):124–133, July 1980.

[105] T. Funkhouser and C. Sequin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. *Computer Graphics (SIGGRAPH '93 Proceedings)*, pages 247–254, July 1993.

[106] T. Funkhouser, C. Sequin, and S. Teller. Management of large amounts of data in interactive building walkthroughs. *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, pages 11–20, July 1992.

[107] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, Inc., New York, NY, 1979.

[108] D. George, F. Sillion, and D. Greenberg. Radiosity redistribution for dynamic environments. *IEEE Computer Graphics and Applications*, 10(4):26–34, July 1990.

[109] P. George. *Automatic Mesh Generation*. John Wiley & Sons, Inc., New York, NY, 1991.

[110] R. Gershbein, P. Schröeder, and P. Hanrahan. Textures and radiosity: Controlling emission and reflection with texture maps. *Computer Graphics (SIGGRAPH '94 Proceedings)*, 28(4):0–0, July 1994.

[111] A. Glassner. Spacetime ray tracing for animation. *IEEE Computer Graphics and Applications*, 8(2):60–70, March 1988.

[112] A. Glassner. How to derive a spectrum from an RGB triplet. *IEEE Computer Graphics and Applications*, 9(4):95–99, July 1989.

[113] A. Glassner, editor. *Introduction to Ray Tracing*. Academic Press, Palo Alto, CA, 1989.

[114] A. Glassner, editor. *Graphics Gems*. Academic Press, San Diego, CA, 1990.

[115] A. Glassner. Maintaining winged-edge models. In D. Kirk, editor, *Graphics Gems III*, pages 191–201. Academic Press, San Diego, CA, 1992.

[116] A. Glassner. *Principles of Digital Image Synthesis*, volume 2. Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1995.

[117] A. Glassner. *Principles of Digital Image Synthesis*, volume 1. Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1995.

[118] G. Golub and C. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 1989.

[119] C. Goral, K. Torrance, D. Greenberg, and B. Battaile. Modeling the interaction of light between diffuse surfaces. *Computer Graphics (SIGGRAPH '84 Proceedings)*, 18(3):212–22, July 1984.

[120] S. Gortler and M. Cohen. Radiosity and relaxation methods. Technical Report TR 408-93, Princeton University, 1993.

[121] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen. The lumigraph. *Computer Graphics (SIGGRAPH '96 Proceedings)*, 30:0–0, August 1996.

[122] S. Gortler, P. Schröder, M. Cohen, and P. Hanrahan. Wavelet radiosity. *Computer Graphics (SIGGRAPH '93 Proceedings)*, 27(4):221–230, August 1993.

[123] H. Gouraud. Continuous shading of curved surfaces. *IEEE Transactions on Computers*, C-20(6):623–629, June 1971.

[124] D. Greenberg. More accurate simulations at faster rates. *IEEE Computer Graphics and Applications*, 11(1):23–29, January 1991.

[125] D. Greenberg. A ray tracing simulation of a radiosity simulation. *IEEE Computer Graphics and Applications*, 11(1):6–7, January 1991.

[126] D. Greenberg, M. Cohen, and K. Torrance. Radiosity: a method for computing global illumination. *The Visual Computer*, 2(5):291–297, September 1986.

[127] Leonidas Guibas and Jorge Stolfi. Primitives for the manipulation of general subdivisions and computation of voronoi diagrams. *ACM Transactions on Graphics*, 4(2):74–123, April 1985.

[128] P. Haeberli and K. Akeley. The accumulation buffer: Hardware support for high-quality rendering. *Computer Graphics (SIGGRAPH '90 Proceedings)*, 24(4):309–318, August 1990.

[129] E. Haines. Beams O' light: Confessions of a hacker. In *SIGGRAPH '91 Frontiers in Rendering Course Notes*. Addison-Wesley, July 1991.

[130] E. Haines. Ronchamp: a case study for radiosity. In *SIGGRAPH '91 Frontiers in Rendering Course Notes*. Addison-Wesley, July 1991.

[131] E. Haines and D. Greenberg. The light buffer: a shadow testing accelerator. *IEEE Computer Graphics and Applications*, 6(9):6–16, September 1986.

[132] E. Haines and J. Wallace. Shaft culling for efficient ray-traced radiosity. *Second Eurographics Workshop on Rendering*, pages 0–0, May 1991.

[133] D. Hall and H. Rushmeier. Improved explicit radiosity method for calculating non-lambertian reflections. *The Visual Computer*, 9(5):278–288, 1993.

[134] R. Hall. *Illumination and Color in Computer Generated Imagery*. Springer-Verlag, 1989.

[135] R. Hall, M. Bussan, P. Georgiades, and D. Greenberg. A testbed for architectural modeling. In *Eurographics '91*, pages 47–58. North-Holland, September 1991.

[136] P. Hanrahan. Rapid hierarchical global illumination algorithms. In *SIGGRAPH '91 Frontiers in Rendering Course Notes*. Addison-Wesley, July 1991.

[137] P. Hanrahan and W. Krueger. Reflection from layered surfaces due to subsurface scattering. *Computer Graphics (SIGGRAPH '93 Proceedings)*, 27(4), August 1993.

[138] P. Hanrahan and D. Salzman. A rapid hierarchical radiosity algorithm for unoccluded environments. *First Eurographics Workshop on Rendering*, pages 151–171, June 1990.

[139] P. Hanrahan, D. Salzman, and L. Aupperle. A rapid hierarchical radiosity algorithm. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):197–206, July 1991.

[140] X. He, K. Torrence, F. Sillion, and D. Greenberg. A comprehensive model for light reflection. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):175–186, July 1991.

[141] P. Heckbert. Adaptive radiosity textures for bidirectional ray tracing. *Computer Graphics (SIGGRAPH '90 Proceedings)*, 24(4):145–154, August 1990.

[142] P. Heckbert. *Simulating Global Illumination Using Adaptive Meshing*. Ph.d. thesis, Computer Science Division (EECS), University of California, Berkeley, June 1991.

[143] P. Heckbert. Discontinuity meshing for radiosity. *Third Eurographics Workshop on Rendering*, pages 203–226, May 1992.

[144] P. Heckbert. Radiosity in flatland. In *Eurographics '92*, pages 181–192. North-Holland, September 1992.

[145] P. Heckbert, editor. *Graphics Gems IV*. Academic Press, San Diego, CA, 1993.

[146] P. Heckbert and P. Hanrahan. Beam tracing polygonal objects. *Computer Graphics (SIGGRAPH '84 Proceedings)*, 18(3):119–127, July 1984.

[147] P. Heckbert and J. Winget. Finite element methods for global illumination. Technical Report UCB/CSD 91/643, Computer Science Division (EECS), University of California, Berkeley, July 1991.

[148] D. Hilbert and S. Cohn-Vossen. *Geometry and the Imagination*. Chelsea, New York, NY, 1952.

[149] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. *Computer Graphics (SIGGRAPH '93 Proceedings)*, 27(4):19–26, August 1993.

[150] H. Hottel and A. Sarofim. *Radiative Transfer*. McGraw-Hill, New York, NY, 1967.

[151] T. S. Huang, editor. *Image Sequence Processing and Dynamic Scene Analysis*. NATO Advanced Study Institute. Springer-Verlag, Berlin-Heidelberg, Germany, 1982.

[152] D. Immel, M. Cohen, and D. Greenberg. A radiosity method for non-diffuse environments. *Computer Graphics (SIGGRAPH '86 Proceedings)*, 20(4):133–142, August 1986.

[153] D. Judd and G. Wyszecki. *Color in Business, Science, and Industry*. John Wiley & Sons, Inc., 1975.

[154] J. Kajiya. Anisotropic reflection models. *Computer Graphics (SIGGRAPH '85 Proceedings)*, 19(3):15–21, July 1985.

[155] J. Kajiya. The rendering equation. *Computer Graphics (SIGGRAPH '86 Proceedings)*, 20(4):143–150, August 1986.

[156] J. Kajiya. Radiometry and photometry for computer graphics. In *SIGGRAPH '90 Advanced Topics in Ray Tracing Course Notes*. Addison-Wesley, August 1990.

[157] J. Kawai, J. Painter, and M. Cohen. Radioptimization – goal-based rendering. *Computer Graphics (SIGGRAPH '93 Proceedings)*, 27(4):147–154, August 1993.

[158] B. Kernighan and D. Ritchie. *The C Programming Language*. Prentice-Hall Publishing Company, Englewood Cliffs, NJ, 2nd edition, 1988.

[159] D. Kirk, editor. *Graphics Gems*. Academic Press, San Diego, CA, 1992.

[160] D. Kirk and J. Arvo. Unbiased sampling techniques for image synthesis. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):153–156, July 1991.

[161] J. Kleiss and D. Hubbard. Effects of three types of flight simulator visual scene detail on detection of altitude change. *Human Factors*, 35(4):653–671, 1993.

[162] D. Knuth. *Fundamental Algorithms*, volume 1 of *The Art of Computer Programming*. Addison-Wesley, Reading, MA, 1968.

[163] D. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. Addison-Wesley, Reading, MA, 1969.

[164] D. Knuth. *Sorting and Searching*, volume 3 of *The Art of Computer Programming*. Addison-Wesley, Reading, MA, 1973.

[165] D. Kochanek and R. Bartels. Interpolating splines with local tension, continuity, and bias control. *Computer Graphics (SIGGRAPH '84 Proceedings)*, 18(3):33–41, July 1984.

[166] J. Koenderink and A. van Doorn. The structure of images. *Biological Cybernetics*, 50:363–370, 1984.

[167] J. Koenderink and A. van Doorn. Representation of local geometry in the visual system. *Biological Cybernetics*, 55:367–375, 1987.

[168] A. Kok, C. Yilmaz, and L. Bierens. A two-pass radiosity method for bézier patches. *First Eurographics Workshop on Rendering*, pages 115–124, June 1990.

[169] C. Kolb. Rayshade user's guide and reference manual. Draft 0.4, January 1992.

[170] E. Languenou and P. Tellier. Including physical light sources and daylight in global illumination. *Third Eurographics Workshop on Rendering*, pages 217–225, May 1992.

[171] D. Laur and P. Hanrahan. Hierarchical splatting: A progressive refinement algorithm for volume rendering. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):285–288, July 1991.

[172] S. Laveau and O. Faugeras. 3-d scene representation as a collection of images and fundamental matrices. Technical Report 2205, INRIA, February 1994.

[173] K. Le. Finite element mesh generation methods: A review and classification˙ *Computer-Aided Design*, 20:27–38, 1988.

[174] B. LeSaec and C. Schlick. A progressive ray-tracing-based radiosity with general reflectance functions. *First Eurographics Workshop on Rendering*, pages 101–114, June 1990.

[175] M. Levoy and P. Hanrahan. Light field rendering. *Computer Graphics (SIGGRAPH '96 Proceedings)*, 30:0–0, August 1996.

[176] J. Lewins. *Importance, the Adjoint Function: The Physical Basis of Variational and Perturbation Theory in Transport and Diffusion Problems*. Pergamon Press, New York, NY, 1965.

[177] R. Lewis. Solving the classic radiosity equation using multigrid techniques. In *Proceedings of the 1992 Western Computer Graphics Symposium*, pages 157–164, April 1992.

[178] R. Van Liere. Divide and conquer radiosity. *Second Eurographics Workshop on Rendering*, pages 0–0, May 1991.

[179] A. Lippman. Movie maps: An application of the optical videodisc to computer graphics. *Computer Graphics (SIGGRAPH '80 Proceedings)*, 14(2):32–43, July 1980.

[180] D. Lischinski, B. Smits, and D. Greenberg. Bounds and error estimates for radiosity. *Computer Graphics (SIGGRAPH '94 Proceedings)*, 28(4):67–74, July 1994.

[181] D. Lischinski, F. Tampieri, and D. Greenberg. Improving sampling and reconstruction techniques for radiosity. Technical Report TR 91-1202, Program of Computer Graphics, Cornell University, August 1991.

[182] D. Lischinski, F. Tampieri, and D. Greenberg. Discontinuity meshing for accurate radiosity. *IEEE Computer Graphics and Applications*, 12(6):25–39, November 1992.

[183] D. Lischinski, F. Tampieri, and D. Greenberg. Combining hierarchical radiosity and discontinuity meshing. *Computer Graphics (SIGGRAPH '93 Proceedings)*, 27(4):199–208, August 1993.

[184] Dani Lischinski. Incremental delaunay triangulation. In P. Heckbert, editor, *Graphics Gems IV*. Academic Press, San Diego, CA, 1993.

[185] D. MacAdam. *Sources of Color Science*. MIT Press, Cambridge, MA, 1970.

[186] E. Maisel and G. Hégron. A realistic image synthesis of animation sequences based on temporal coherence. In *Third Eurographics Workshop on Animation and Simulation*, Cambridge, UK, September 1992.

[187] S. Mallet. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, July 1989.

[188] T. Malley. A shading model for computer generated images. Masters thesis, Department of Computer Science, University of Utah, 1988.

[189] J. Marks, R. Walsh, J. Christiensen, and M. Friedell. Image and intervisibility coherence in rendering. In *Proceedings of Graphics Interface '90*, pages 17–30. Canadian Information Processing Society, May 1990.

[190] N. Max and M. Allison. Linear radiosity approximation using vertex-to-vertex form factors. In *Graphics Gems III*, pages 318–323. Academic Press, San Diego, CA, 1992.

[191] N. Max and E. Getzoff. Spherical harmonic molecular surfaces. *IEEE Computer Graphics and Applications*, 8(4):42–50, July 1988.

[192] N. Max and R. Troutman. Optimal hemicube sampling. *Fourth Eurographics Workshop on Rendering*, pages 185–200, June 1993.

[193] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. *Computer Graphics (SIGGRAPH '95 Proceedings)*, 29(4):39–46, July 1995.

[194] M. Melkemi and L. Melkemi. An algorithm for detecting dot patterns. In *Computer Graphics: Developments in Virtual Environments*, pages 161–169. Academic Press, San Diego, CA, 1995.

[195] G. Meyer, H. Rushmeier, M. Cohen, D. Greenberg, and K. Torrance. An experimental evaluation of computer graphics imagery. *ACM Transactions on Graphics*, 5(1):30–50, January 1986.

[196] G. Miller, E. Hoffert, S. E. Chen, E. Patterson, D. Blackletter, S. Rubin, S. A. Applin, D. Yim, and J. Hanan. The virtual museum: Interactive 3d navigation of a multimedia database. *The Journal of Visualization and Computer Animation*, 3(3):183–197, 1992.

[197] D. Mitchell. The antialiasing problem in ray tracing. In *SIGGRAPH '90 Advanced Topics in Ray Tracing Course Notes*. Addison-Wesley, August 1990.

[198] D. Mitchell and P. Hanrahan. Illumination from curved reflectors. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(4):283–291, July 1992.

[199] P. Moon. *The Scientific Basis of Illuminating Engineering*. McGraw-Hill, New York, NY, 1936.

[200] M. Mortenson. *Geometric Modeling*. John Wiley & Sons, Inc., New York, NY, 1985.

[201] C. Muller. *Spherical Harmonics*. Springer-Verlag, New York, NY, 1966.

[202] S. Müller and F. Schöffel. Fast radiosity repropagation for interactive virtual environments using a shadow-form-factor list. In *Fifth Eurographics Workshop on Rendering*, pages 325–342, Darmstadt, Germany, June 1994.

[203] K. Myszkowski and T. Kunii. Texture mapping as an alternative for meshing during walkthrough. In *Fifth Eurographics Workshop on Rendering*, Darmstadt, Germany, June 1994.

[204] B. Naylor. Binary space partitioning trees: An alternative representation of polytopes. *Computer-Aided Design*, 22(2):250–253, March 1990.

[205] J. Neider, T. Davis, and M. Woo. *OpenGL Programming Guide*. Addison-Wesley, 1993.

[206] L. Neumann, M. Feda, M. Kopp, and W. Purgethofer. A new stochastic radiosity method for highly complex scenes. *Fifth Eurographics Workshop on Rendering*, pages 195–206, June 1994.

[207] L. Neumann and C. Kelemen. Solution of interreflection problem for very complex environments by transillumination method. *Second Eurographics Workshop on Rendering*, pages 0–0, May 1991.

[208] M. Newell, R. Newell, and T. Sancha. A solution to the hidden surface problem. *Proceedings of the ACM National Meeting*, 1972.

[209] W. Newman and R. Sproull. *Principles of Interactive Computer Graphics*. McGraw-Hill, New York, NY, 1979.

[210] F. Nicodemus. Radiance. *American Journal of Physics*, 31(5):368–377, 1963.

[211] F. Nicodemus, J. Richmond, J. Hsia, I. Ginsberg, and T. Limperis. Geometrical considerations and nomenclature for reflectance. NBS Monograph 160, National Bureau of Standards, U.S. Dept. of Commerce, Washington, D.C., October 1977.

[212] J. Nimeroff. Personal communication, 1994. I actually talked to myself.

[213] J. Nimeroff. Personal communication, 1995. I actually talked to myself.

[214] J. Nimeroff. Personal communication, 1996. I actually talked to myself.

[215] J. Nimeroff. A temporal image-based approach to motion reconstruction for globally illuminated animated environments. In *Seventh Eurographics Workshop on Rendering*, pages 176–185, Porto, Portugal, June 1996.

[216] J. Nimeroff. *An Image-Based Framework for Global Illumination in Animated Environments*. PhD thesis, University of Pennsylvania, March 1997.

[217] J. Nimeroff, J. Dorsey, and H. Rushmeier. A framework for global illumination in animated environments. In *Sixth Eurographics Workshop on Rendering*, pages 223–236, Dublin, Ireland, June 1995.

[218] J. Nimeroff, J. Dorsey, and H. Rushmeier. Implemention and analysis of a global illumination framework for animated environments. *IEEE Transactions on Visualization and Computer Graphics*, 2(4):0–0, December 1996.

[219] J. Nimeroff, J. Dorsey, E. Simoncelli, and N. Badler. Rendering spaces for architectural environments. *Presence, the Journal of Virtual Reality and Teleoperators*, 4(3):286–296, August 1995.

[220] J. Nimeroff, E. Simoncelli, and J. Dorsey. Efficient re-rendering of naturally illuminated environments. In *Fifth Eurographics Workshop on Rendering*, pages 359–374, Darmstadt, Germany, June 1994.

[221] T. Nishita and E. Nakamae. Continuous tone representation of three-dimensional objects taking account of shadows and interreflection. *Computer Graphics (SIGGRAPH '85 Proceedings)*, 19(3):23–30, July 1985.

[222] T. Nishita and E. Nakamae. Continuous tone representation of three-dimensional objects illuminated by sky light. *Computer Graphics (SIGGRAPH '86 Proceedings)*, 20(4):125–132, August 1986.

[223] A. Oppenheim and R. Schafer. *Digital Signal Processing*. Prentice-Hall Publishing Company, Englewood Cliffs, NJ, 1975.

[224] Committee on Colorimetry Optical Society of America. *The Science of Color*. Optical Society of America, 1963.

[225] Committee on Colorimetry Optical Society of America. *The Science of Color*. Optical Society of America, Washington, DC, 1973.

[226] J. O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, New York, NY, 1987.

[227] J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, New York, NY, 1994.

[228] J. Ousterhout. *Tcl and the Tk Toolkit*. Addison-Wesley, 1994.

[229] P. Padmos and M. Milders. Quality criteria for simulator images: A literature review. *Human Factors*, 34(6):727–748, 1992.

[230] A. Paeth, editor. *Graphics Gems V*. Academic Press, San Diego, CA, 1994.

[231] S. Pattanaik. *Computational Methods for Global Illumination and Visualisation of Complex 3D Environments*. PhD thesis, Birla Institute of Technology & Science, Computer Science Department, Pilani, India, February 1993.

[232] S. Pattanaik and S. Mudur. Computation of global illumination by monte carlo simulation of the particle model of light. *Third Eurographics Workshop on Rendering*, pages 71–83, May 1992.

[233] S. Pattanaik and S. Mudur. The potential equation and importance in illumination computations. *Computer Graphics Forum*, 12(2):131–136, March 1993.

[234] D. Pearson. *Transmission and Display of Pictorial Information*. Halsted Press, New York, NY, 1975.

[235] D. Pedoe. *Geometry, A Comprehensive Course*. Dover Publications, New York, NY, 1970.

[236] M. Peercy. Linear color representations for full spectral rendering. *Computer Graphics (SIGGRAPH '93 Proceedings)*, 27(4):191–198, August 1993.

[237] A. Penna and R. Patterson. *Projective Geometry and its Applications to Computer Graphics*. Prentice-Hall Publishing Company, 1986.

[238] P. Perona. Deformable kernels for early vision. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 222–227, Maui, Hawaii, 1991.

[239] B. T. Phong. Illumination for computer-generated images. Technical report, University of Utah, Computer Science Department, 1973.

[240] B. T. Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, June 1975.

[241] K. Picott. Extensions of the linear and area lighting models. *IEEE Computer Graphics and Applications*, 12(2):31–38, March 1992.

[242] P. Poulin and A. Fournier. Lights from highlights and shadows. In *Proceedings of the 1992 Western Computer Graphics Symposium*, pages 141–145, April 1992.

[243] F. Preparata and M. Shamos. *Computational Geometry - An Introduction*. Springer-Verlag, New York, NY, 1985.

[244] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, MA, 1988.

[245] C. Puech, F. Sillion, and C. Vedel. Improving interaction with radiosity-based lighting simulation programs. In *Computer Graphics (1990 Symposium on Interactive 3D Graphics)*, pages 51–57, March 1990.

[246] X. Pueyo. Diffuse interreflections. Techniques for form-factor computation: A survey. *The Visual Computer*, 7(4):200–209, July 1991.

[247] W. Purgathofer and M. Zeiller. Fast radiosity by parallelization. *First Eurographics Workshop on Rendering*, pages 171–181, June 1990.

[248] R. Recker, D. George, and D. Greenberg. Acceleration techniques for progressive radiosity. In *Computer Graphics (1990 Symposium on Interactive 3D Graphics)*, pages 59–66, March 1990.

[249] M. Regan and R. Pose. Priority rendering with a virtual reality address recalculation pipeline. *Computer Graphics (SIGGRAPH '94 Proceedings)*, 28(4):155–162, July 1994.

[250] M. Reichert. A two-pass radiosity method driven by lights and view position. Masters thesis, Program of Computer Graphics, Cornell University, January 1992.

[251] J. Rohlf and J. Helman. Iris performer: A high performance toolkit for real-time 3d graphics. *Computer Graphics (SIGGRAPH '94 Proceedings)*, 28(4):381–395, July 1994.

[252] H. Rushmeier. Extending the radiosity method to transmitting and specularly reflecting surfaces. Masters thesis, Program of Computer Graphics, Cornell University, 1986.

[253] H. Rushmeier. *Realistic Image Synthesis for Scenes with Radiatively Participating Media*. Ph.d. thesis, Program of Computer Graphics, Cornell University, 1988.

[254] H. Rushmeier, D. Baum, and D. E. Hall. Accelerating the hemi-cube algorithm for calculating radiation form factors. In *5th AIAA/ASME Thermophysics and Heat Transfer Conference*, Seattle, Washington, June 1990.

[255] H. Rushmeier, C. Patterson, and A. Veerasamy. Geometric simplification for indirect illumination calculations. In *Proceedings of Graphics Interface '93*, pages 227–236. Canadian Information Processing Society, May 1993.

[256] H. Rushmeier and K. Torrance. The zonal method for calculating light intensities in the presence of a participating medium. *Computer Graphics (SIGGRAPH '87 Proceedings)*, 21(4):293–302, July 1987.

[257] H. Rushmeier and K. Torrance. Extending the radiosity method to include specularly reflecting and translucent materials. *ACM Transactions on Graphics*, 9(1):1–27, January 1990.

[258] H. Rushmeier, G. Ward, C. Piatko, P. Sanders, and B. Rust. Comparing real and synthetic images: Some ideas about metrics. In *Sixth Eurographics Workshop on Rendering*, pages 213–222, Dublin, Ireland, June 1995.

[259] H. E. Rushmeier. Radiosity input/output. In *SIGGRAPH '92 Radiosity Course Notes*, pages 152–168. Addison-Wesley, July 1992.

[260] D. Salesin, D. Lischinski, and T. DeRose. Reconstructing illumination functions with selected discontinuities. *Third Eurographics Workshop on Rendering*, pages 99–112, May 1992.

[261] H. Samet. *Design and Analysis of Spatial Data Structures*. Addison-Wesley, Reading, MA, 1990.

[262] M. Sbert. An integral geometry based method for fast form-factor computation. In *Eurographics '93*, pages 409–420, 1993.

[263] M. Sbert, F. Perez, and X. Pueyo. Global monte carlo. A progressive solution. In *Sixth Eurographics Workshop on Rendering*, pages 0–0, June 1995.

[264] C. Schoeneman, J. Dorsey, B. Smits, J. Arvo, and D. Greenberg. Painting with light. *Computer Graphics (SIGGRAPH '93 Proceedings)*, 27(4):143–146, August 1993.

[265] P. Schröder. Wavelets in computer graphics. *Proceedings of the IEEE*, to appear; 1996. Invited article for special issue on wavelets, Ingrid Daubechies and Jelena Kovacevic, Eds.

[266] P. Schröder, S. Gortler, M. Cohen, and P. Hanrahan. Wavelet projections for radiosity. *Fourth Eurographics Workshop on Rendering*, pages 105–114, June 1993.

[267] P. Schröder and P. Hanrahan. A closed form expression for the form factor between two polygons. Technical Report TR 404-93, Princeton University, 1993.

[268] P. Schröder and P. Hanrahan. On the form factor between two polygons. *Computer Graphics (SIGGRAPH '93 Proceedings)*, 27(4):163–164, August 1993.

[269] P. Schröder and P. Hanrahan. Wavelet methods for radiance calculations. *Fifth Eurographics Workshop on Rendering*, pages 0–0, June 1994.

[270] S. Seitz and C. Dyer. View morphing. *Computer Graphics (SIGGRAPH '96 Proceedings)*, 30:0–0, August 1996.

[271] C. Séquin and E. Smyrl. Parameterized ray tracing. *Computer Graphics (SIGGRAPH '89 Proceedings)*, 23(3):307–314, July 1989.

[272] M. Z. Shao, Q. S. Peng, and Y. D. Liang. A new radiosity approach by procedural refinements for realistic image synthesis. *Computer Graphics (SIGGRAPH '88 Proceedings)*, 22(4):93–101, August 1988.

[273] P. Shirley. Physically based lighting calculations for computer graphics: a modern perspective. *First Eurographics Workshop on Rendering*, pages 67–81, June 1990.

[274] P. Shirley. A ray tracing method for illumination calculation in diffuse-specular scenes. In *Proceedings of Graphics Interface '90*, pages 205–212. Canadian Information Processing Society, May 1990.

[275] P. Shirley. *Physically Based Lighting Calculations for Computer Graphics*. Ph.d. thesis, Computer Science Department, University of Illinois at Urbana-Champaign, 1991.

[276] P. Shirley. Time complexity of monte carlo radiosity. *Computers and Graphics*, 16(1):117–120, 1992.

[277] P. Shirley and C. Wang. Direct lighting calculation by monte carlo integration. *Second Eurographics Workshop on Rendering*, pages 117–120, May 1991.

[278] K. Shoemake. Linear form curves. In A. Paeth, editor, *Graphics Gems V*. Academic Press, San Diego, CA, 1995.

[279] K. Shoemake and T. Duff. Matrix animation and polar decomposition. In *Proceedings of Graphics Interface '92*, pages 258–264. Canadian Information Processing Society, May 1992.

[280] R. Siegel and J. R. Howell. *Thermal Radiation Heat Transfer*. Hemisphere Publishing Corp., Washington, DC, 1992.

[281] F. Sillion. *Lighting Simulation for Image Synthesis: Realism and Interactivity*. Ph.d. thesis, University of Paris, 1989.

[282] F. Sillion. The state of the art in physically-based rendering and its impact on future applications. *Second Eurographics Workshop on Rendering*, May 1991.

[283] F. Sillion, J. Arvo, S. Westin, and D. Greenberg. A global illumination solution for general reflectance distributions. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):187–196, July 1991.

[284] F. Sillion and C. Puech. A general two-pass method integrating specular and diffuse reflection. *Computer Graphics (SIGGRAPH '89 Proceedings)*, 23(3):335–344, July 1989.

[285] F. Sillion and C. Puech. *Radiosity and Global Illumination*. Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1994.

[286] E. Simoncelli. *Distributed Analysis and Representation of Visual Motion*. PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, Cambridge, MA, January 1993.

[287] E. Simoncelli, W. Freeman, E. Adelson, and D. Heeger. Shiftable multi-scale transforms. *IEEE Transactions on Information Theory*, 38(2):587–607, March 1992. Special Issue on Wavelets.

[288] B. Smits, J. Arvo, and D. Greenberg. A clustering algorithm for radiosity in complex environments. *Computer Graphics (SIGGRAPH '94 Proceedings)*, 28(4):0–0, July 1994.

[289] B. Smits, J. Arvo, and D. Salesin. An importance-driven radiosity algorithm. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(4):273–282, July 1992.

[290] E. Sparrow and R. Cess. *Radiation Heat Transfer*. Hemisphere Publishing Corporation, Washington, DC, 1978.

[291] S. Spencer. The hemisphere radiosity algorithm: A tale of two algorithms. *First Eurographics Workshop on Rendering*, pages 127–135, June 1990.

[292] G. Strang. *Introduction to Applied Mathematics*. Wellesley-Cambridge Press, Wellesley, MA, 1986.

[293] G. Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, Wellesley, MA, 1993.

[294] W. Sturzlinger. Radiosity with voronai-diagrams. *Third Eurographics Workshop on Rendering*, pages 169–177, May 1992.

[295] M. Subbarao. *Interpretation of Visual Motion: A Computational Study*. Research Notes in Artificial Intelligence. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1988.

[296] I. Sutherland. Sketchpad: A man-machine graphical communication system. In *Proceedings of the Spring Joint Computer Conference*, 1963.

[297] R. Szeliski. Image mosaicing for tele-reality applications. Technical Report No. 94/2, Digital Equipment Corporation, Cambridge Research Lab, June 1994.

[298] A. Takagi, H. Takaoka, T. Oshima, and Y. Ogata. Accurate rendering technique based on colorimetric conception. *Computer Graphics (SIGGRAPH '90 Proceedings)*, 24(4):263–272, August 1990.

[299] F. Tampieri. Accurate form-factor computation. In D. Kirk, editor, *Graphics Gems III*, pages 329–333. Academic Press, San Diego, CA, 1992.

[300] F. Tampieri and D. Lischinski. The constant radiosity assumption syndrome. *Second Eurographics Workshop on Rendering*, pages 0–0, May 1991.

[301] S. Teller. Computing the antipenumbra of an area light source. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(4):139–148, July 1992.

[302] S. Teller, K. Balla, and J. Dorsey. Conservative radiance interpolants for ray tracing. In *Seventh Eurographics Workshop on Rendering*, pages 0–0, Porto, Portugal, June 1996.

[303] S. Teller and P. Hanrahan. Global visibility algorithms for illumination computations. *Computer Graphics (SIGGRAPH '93 Proceedings)*, 27(4):239–246, August 1993.

[304] S. Teller and C. Séquin. Visibility preprocessing for interactive walkthroughs. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):61–69, July 1991.

[305] W. Thibault and B. Naylor. Set operations on polyhedra using binary space partition trees. *Computer Graphics (SIGGRAPH '87 Proceedings)*, 21(4):153–162, July 1987.

[306] K. Torrance and E. Sparrow. Theory for off-specular reflection from roughened surfaces. *Journal of Optical Society of America*, 57(9), 1967.

[307] A. Varshney and J. Prins. An environment-projection approach to radiosity for mesh-connected computers. *Third Eurographics Workshop on Rendering*, pages 271–281, May 1992.

[308] E. Veach and L. Guibas. Bidirectional estimators for light transport. In *Fifth Eurographics Workshop on Rendering*, pages 147–162, Darmstadt, Germany, June 1994.

[309] E. Veach and L.Guibas. Optimally combining sampling techniques for monte carlo rendering. *Computer Graphics (SIGGRAPH '95 Proceedings)*, 29(4):419–428, August 1995.

[310] C. Vedel. Improved storage and reconstruction of light intensities on surfaces. *Third Eurographics Workshop on Rendering*, pages 113–121, May 1992.

[311] C. Vedel and C. Puech. A testbed for adaptive subdivision in progressive radiosity. *Second Eurographics Workshop on Rendering*, pages 0–0, May 1991.

[312] C. Verbeck and D. Greenberg. A comprehensive lightsource description for computer graphics. *IEEE Computer Graphics and Applications*, 4(7):66–75, July 1984.

[313] J. Vilaplana. Parallel radiosity solutions based on partial result messages. *Third Eurographics Workshop on Rendering*, pages 259–270, May 1992.

[314] J. Vilaplana and X. Pueyo. Exploiting coherence for clipping and view transformations in radiosity algorithms. *First Eurographics Workshop on Rendering*, pages 137–150, June 1990.

[315] J. Wallace. Radiosity and ray tracing: a comparison of shading strategies. In *SIGGRAPH '90 Advanced Topics in Ray Tracing Course Notes*. Addison-Wesley, August 1990.

[316] J. Wallace. Trends in radiosity for image synthesis. *First Eurographics Workshop on Rendering*, pages 1–14, June 1990.

[317] J. Wallace, M. Cohen, and D. Greenberg. A two-pass solution to the rendering equation: A synthesis of ray tracing and radiosity methods˙ *Computer Graphics (SIGGRAPH '87 Proceedings)*, 21(4):311–320, July 1987.

[318] J. Wallace, K. Elmquist, and E. Haines. A ray tracing algorithm for progressive radiosity. *Computer Graphics (SIGGRAPH '89 Proceedings)*, 23(3):315–324, July 1989.

[319] C. Wang. Physically correct direct lighting for distribution ray tracing. In D. Kirk, editor, *Graphics Gems III*, pages 307–313. Academic Press, San Diego, CA, 1992.

[320] J. Wang and E. Adelson. Layered representation for motion analysis. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 361–366, New York, NY, June 1993.

[321] Y. Wang and W. A. Davis. Octant priority for radiosity image rendering. In *Proceedings of Graphics Interface '90*, pages 83–91. Canadian Information Processing Society, May 1990.

[322] G. Ward. Evaluating a real lighting simulation. In *SIGGRAPH '90 Advanced Topics in Ray Tracing Course Notes*. Addison-Wesley, August 1990.

[323] G. Ward. Real pixels. In J. R. Arvo, editor, *Graphics Gems II*, pages 0–0. Academic Press, San Diego, CA, 1991.

[324] G. Ward. The radiance lighting simulation and rendering system. *Computer Graphics (SIGGRAPH '94 Proceedings)*, 28(4):459–472, July 1994.

[325] G. Ward and P. Heckbert. Irradiance gradients. *Third Eurographics Workshop on Rendering*, pages 85–98, May 1992.

[326] G. Ward, F. Rubinstein, and R. Clear. A ray tracing solution for diffuse interreflection. *Computer Graphics (SIGGRAPH '88 Proceedings)*, 22(4):85–92, August 1988.

[327] G. J. Ward. Measuring and modeling anisotropic reflection. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(4):265–272, July 1992.

[328] J. Warnock. A hidden-surface algorithm for computer generated half-tone pictures. Technical Report TR 4–15, NTIS AD-733 671, University of Utah, Computer Science Department, 1969.

[329] D. Watson. Computing the n-dimensional delaunay tessellation with applications to voronoi polytopes. *The Computing Journal*, 24(2):167–172, 1981.

[330] A. Watt and M. Watt. *Advanced Animation and Rendering Techniques: Theory and Practice*. Addison-Wesley, New York, NY, 1992.

[331] M. Watt. Light-water interaction using backward beam tracing. *Computer Graphics (SIGGRAPH '90 Proceedings)*, 24(4):377–385, August 1990.

[332] K. Weiler and P. Atherton. Hidden-surface removal using polygon area sorting. *Computer Graphics (SIGGRAPH '77 Proceedings)*, 11(2):0–0, 1977.

[333] J. Weng, T. S. Huang, and N. Ahuja. *Motion and Structure from Image Sequences*. Springer-Verlag, Berlin-Heidelberg, Germany, 1993.

[334] S. Westin, J. Arvo, and K. Torrance. Predicting reflectance functions for complex surfaces. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(4):255–264, July 1992.

[335] T. Whitted. An improved illumination model for shaded display. *Communications of the ACM*, 23(6):343–349, June 1980.

[336] L. Williams. Pyramidal parametrics. *Computer Graphics (SIGGRAPH '83 Proceedings)*, 17(3):1–11, July 1983.

[337] G. Wolberg. *Digital Image Warping*. IEEE Computer Society Press, 1990.

[338] A. Woo, P. Poulin, and A. Fournier. A survey of shadow algorithms. *IEEE Computer Graphics and Applications*, 10(6):13–32, November 1990.

[339] G. Wyszecki and W. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulae*. John Wiley & Sons, Inc., 1982.

[340] G. Wyvill, C. Jay, and D. McRobie. Pixel-independent ray tracing. In *Computer Graphics: Developments in Virtual Environments*, pages 43–55. Academic Press, San Diego, CA, 1995.

[341] H. Zatz. Galerkin radiosity: A higher order solution method for global illumination. *Computer Graphics (SIGGRAPH '93 Proceedings)*, 27(4):213–220, August 1993.

[342] N. Zhang. Two methods for speeding up form-factor calculation. *Second Eurographics Workshop on Rendering*, pages 85–98, May 1991.