

University of Warwick institutional repository: <http://go.warwick.ac.uk/wrap>

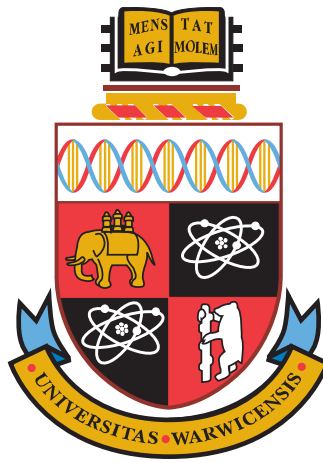
A Thesis Submitted for the Degree of PhD at the University of Warwick

<http://go.warwick.ac.uk/wrap/55482>

This thesis is made available online and is protected by original copyright.

Please scroll down to view the document itself.

Please refer to the repository record for this item for information to help you to cite it. Our policy information is available from the repository home page.



Interactive Global Illumination on the CPU

Piotr Bartłomiej Dubla
B.Sc. (Hons)

*A thesis submitted in partial fulfilment of the requirements for
the degree of
Doctor of Philosophy in Engineering*

*School of Engineering
University of Warwick
November 2010*

Contents

Acknowledgements	x
Declaration	xi
List of Publications	xii
Abstract	xiii
1 Introduction	1
1.1 Physically-based Rendering	1
1.2 Ray tracing	3
1.3 Interactive Global Illumination	4
1.4 Research Objectives	8
1.5 Thesis Outline	9
2 Background	11
2.1 Introduction to Rendering	11
2.1.1 Radiometry	11
2.1.2 Surface Interactions of Light	12
2.1.3 Light Transport	14
2.2 Primary Rendering Techniques	16
2.2.1 Rasterisation	16
2.2.2 Ray Tracing	17
2.2.3 Radiosity	18
2.3 Accelerating Rendering	19

2.3.1	Component-based Rendering	19
2.3.2	Selective-Rendering	21
2.3.2.1	Rasterisation	22
2.3.2.2	Ray tracing	22
2.3.2.3	Perception	23
2.3.3	Interleaved Sampling	24
2.3.4	Dynamic Acceleration Structures	25
2.3.5	Irradiance Caching	27
2.4	Synchronisation	30
2.4.1	Blocking	30
2.4.2	Busy-waiting	31
2.4.3	Non-blocking	31
2.4.4	Atomic Primitives	32
3	Interactive Global Illumination	33
3.1	Interactive Ray Tracing	33
3.1.1	CPU Algorithms	34
3.1.1.1	Systems	34
3.1.1.2	Algorithmic enhancements	36
3.1.2	GPU Algorithms	37
3.1.2.1	Systems	37
3.1.2.2	Acceleration Data Structures	38
3.2	Interactive Global Illumination	40
3.2.1	CPU Algorithms	41
3.2.1.1	Systems	41
3.2.1.2	Radiosity	43
3.2.1.3	Sparse sampling	44
3.2.2	GPU Algorithms	48
3.2.2.1	Radiosity	48
3.2.2.2	Instant Radiosity	50
3.2.2.3	Image-based methods	53
3.2.2.4	Photon Mapping	54

3.2.2.5	Precomputed Radiance Transfer	56
3.2.2.6	Rasterisation	56
3.3	Discussion	57
3.4	Summary	61
4	Impact of Selective Rendering on Interactive Ray Tracing	62
4.1	Introduction	62
4.2	Experimental Framework	63
4.3	Experiment	64
4.4	Results	68
4.5	Discussion	69
4.6	Summary	70
5	Adaptive Interleaved Sampling for Interactive Global Illumination	71
5.1	Introduction	71
5.2	Framework	72
5.2.1	Identification	73
5.2.2	Deconstruction	74
5.2.3	Pairing	76
5.2.4	Implementation	78
5.3	Adaptive Interleaved Sampling (AIS)	80
5.3.1	Algorithm	84
5.3.2	Indirect Diffuse Lighting	85
5.3.3	Soft Shadows	86
5.3.4	Single-scattering Participating Media	88
5.4	Results	90
5.4.1	Validation	93
5.5	Summary	97
6	Instant Caching for Interactive Global Illumination	99
6.1	Introduction	99
6.2	Instant Caching	100

6.2.1	Static Instant Caching	101
6.2.2	Temporal Instant Caching (TIC)	103
6.3	Results	108
6.3.1	Static images	108
6.3.2	Animations	111
6.3.3	Validation	112
6.4	Summary	114
7	Wait-Free Shared-Memory Irradiance Cache	116
7.1	Introduction	116
7.2	Algorithms	118
7.2.1	Lock-Based Irradiance Cache (LCK)	118
7.2.2	Local-Write Irradiance Cache (LW)	119
7.2.3	Wait-Free Irradiance Cache (WF)	120
7.3	Results	121
7.3.1	Still images	123
7.3.2	Animations	126
7.4	Summary	129
8	Conclusions and Future Work	131
8.1	Conclusions	131
8.2	Contributions	133
8.3	Impact	135
8.4	Limitations and Extensions	136
8.5	Directions for Future Work	138
8.6	Final Remarks	139
	References	141
A	Adaptive Interleaved Sampling	170

List of Figures

1.2.1 The concept of ray tracing.	4
1.2.2 Examples of images rendered using path-tracing.	5
1.3.1 Explanation of interpolation via adaptive sampling.	7
2.1.1 BRDF examples.	13
2.3.1 Component-based Rendering.	21
2.3.2 Aleph map.	24
2.3.3 Interleaved sampling.	25
2.3.4 Samples in an irradiance cache.	28
3.1.1 Manta interactive ray tracer.	36
3.2.1 Razor.	43
3.2.2 Progressive radiosity.	44
3.2.3 Holodeck simulation.	45
3.2.4 Render Cache.	46
3.2.5 Corrective splatting.	47
3.2.6 Antiradiance.	50
3.2.7 Real-time Indirect Illumination with Clustered Visibility.	52
3.2.8 Screen-space directional occlusion.	53
3.2.9 Photon mapping on the GPU.	55
3.2.10 Micro-Rendering.	57
4.2.1 Explanation how stride changes.	63
4.2.2 The four scenes used.	65

4.3.1 Primary rays only.	66
4.3.2 Primary and secondary rays.	66
4.3.3 Secondary rays only.	67
4.3.4 Normalised speed-up compared to 4096×4096	67
5.2.1 Framework flowchart	73
5.2.2 Explanation of framework goals.	76
5.2.3 AIS pipeline.	81
5.3.1 The three steps used in adaptive interleaved sampling.	83
5.3.2 ID adaptive guidance.	85
5.3.3 SS adaptive guidance.	87
5.3.4 PM adaptive guidance.	88
5.3.5 The scenes used for obtaining AIS results.	92
5.3.6 AIS participating media scenes.	93
5.4.1 Office scene (including the PM version).	95
6.2.1 Irradiance cache vs. instant cache.	103
6.2.2 The five cases.	104
6.2.3 The scenes used for all experiments.	109
6.3.1 HDR-VDP comparisons for the Office scene.	115
7.2.1 The five scenes utilised in the experiments.	123
7.3.1 Still Images: Results for all scenes.	128
7.3.2 Animation results for Cornell Box.	128
7.3.3 Animation results for Conference Room.	129
A.1 Indirect diffuse (ID).	171
A.2 Guidance ID.	172
A.3 Soft shadows (SS).	173
A.4 Guidance SS.	174
A.5 Participating Media (PM).	175
A.6 Guidance PM.	176
A.7 Instant Global Illumination (IGI).	177
A.8 AIS with maximum samples (A-MAX).	179

A.9 Path-traced reference (PT).	181
A.10 VDP results for AIS vs. PT	183
A.11 VDP results for A-MAX vs. PT	185
A.12 VDP results for IGI vs. PT	186

List of Tables

4.1	Scene details	64
5.1	Eleven scenes rendered with different components.	96
5.2	Speedup for the 11 scenes.	96
5.3	Results for HDR-VDP calculations in %.	97
6.1	Results for rendering the first frame.	110
6.2	Results for the diffuse interreflections only.	110
6.3	Results for rendering the animations.	111
6.4	Rendering times averaged over 100 frames.	112
6.5	Results of the HDR-VDP comparison.	113
7.1	Results for all scenes.	125

Acknowledgements

Firstly I would like to thank my supervisors, Alan and Kurt. Alan, if it wasn't for your laptop troubles at Afrigraph all those years ago and our chance meeting I would not be where I am now, and for that I will be forever grateful. You not only remembered me after all those years when I was ready to do a PhD but also went out of your way to ensure that I could join you in Bristol and become part of your group. Kurt, you've been both a great friend and a mentor, your extensive graphics knowledge and love of research pushed me to try new things and better my own knowledge as much as I could, this thesis would not be what it is without your guidance and patience. I am indebted to you both, and your constant support and motivation are reflected in this thesis and all my research.

To the other members of the research group, our collaborations not only made for more varied and interesting research but your friendships balanced out these four years and all our arguments, discussions and outings made the PhD a fantastic experience I will never forget. Our time together formed friendships that span most of Europe and beyond so I'd like to say thank you to Tom, Vedad, Jass, Vibhor, Mike, Alena, Carlo, Elmedin, Alessandro, Francesco, Belma, Ela, Gabriela, Matt, Remi and Sandro and I wish you all the best in your future endeavours, whatever they may be.

I would like to thank my father who didn't even blink an eye when I said I was leaving a comfortable and successful job to pursue a PhD and who happily bought me a one-ticket to England and told me to go chase my dreams. I also dedicate this PhD to both my mother and my grandfather, both of whom regrettably passed away during the last four years and never got the chance to see me complete it.

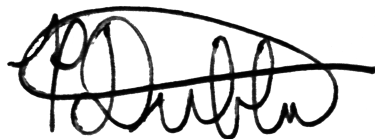
Finally I would like say thank you to all the people I have missed or not mentioned, there was so many others who were part of the whole PhD process and helped me in numerous ways during the past four years. I am also thankful to the University of Bristol and University of Warwick for all the support as well as the EPSRC for funding my research via the UK-EPSRC grant EP/D069874/2.

Declaration

The work in this thesis is original and no portion of work referred to here has been submitted in support of an application for another degree or qualification of this or any other university or institute of learning.

Signed:

Date: November 2010

A handwritten signature in black ink, appearing to read 'P. Dubla', with a large, sweeping horizontal stroke across the top.

Piotr Dubla

List of Publications

The following have been published as a result of the work contained within this thesis.

Journal papers

- **Dubla P.**, Debattista K., Chalmers A.: *Adaptive Interleaved Sampling for Interactive High Fidelity Rendering*. Computer Graphics Forum (Volume 28, Issue 8, December 2009). Eurographics Association.
- Debattista K., **Dubla P.**, Banterle F., Santos L.P., Chalmers A.: *Instant Caching for Interactive Global Illumination*. Computer Graphics Forum (Volume 28, Issue 8, December 2009). Eurographics Association.
- Debattista K., **Dubla P.**, Santos L.P., Chalmers A.: *Wait-Free Shared-Memory Irradiance Cache*. Computer Graphics and Applications (Preprint). IEEE.

Peer-reviewed Conference Papers

- **Dubla P.**, Debattista K., Santos L.P., Chalmers A.: *Wait-Free Shared-Memory Irradiance Cache*. In the 10th Eurographics Symposium on Parallel Graphics and Visualization (Munich, Germany, 2009) Eurographics Association.
- **Dubla P.**, Chalmers A., Debattista K.: *An Analysis of Cache Awareness for Interactive Selective Rendering*. In the 16th International Conference on Computer Graphics, Visualization and Computer Vision (Plzen, Czech Republic, 2008).

Abstract

Computing realistic physically-based global illumination in real-time remains one of the major goals in the fields of rendering and visualisation; one that has not yet been achieved due to its inherent computational complexity. This thesis focuses on CPU-based interactive global illumination approaches with an aim to develop generalisable hardware-agnostic algorithms. Interactive ray tracing is reliant on spatial and cache coherency to achieve interactive rates which conflicts with needs of global illumination solutions which require a large number of incoherent secondary rays to be computed. Methods that reduce the total number of rays that need to be processed, such as *Selective rendering*, were investigated to determine how best they can be utilised.

The impact that selective rendering has on interactive ray tracing was analysed and quantified and two novel global illumination algorithms were developed, with the structured methodology used presented as a framework. *Adaptive Interleaved Sampling*, is a generalisable approach that combines interleaved sampling with an adaptive approach, which uses efficient component-specific adaptive guidance methods to drive the computation. Results of up to 11 frames per second were demonstrated for multiple components including participating media. *Temporal Instant Caching*, is a caching scheme for accelerating the computation of diffuse interreflections to interactive rates. This approach achieved frame rates exceeding 9 frames per second for the majority of scenes. Validation of the results for both approaches showed little perceptual difference when comparing against a gold-standard path-traced image. Further research into caching led to the development of a new *wait-free data access control mechanism* for sharing the irradiance cache among multiple rendering threads on a shared memory parallel system. By not serialising accesses to the shared data structure the irradiance values were shared among all the threads without any overhead or contention, when reading and writing simultaneously. This new approach achieved efficiencies between 77% and 92% for 8 threads when calculating static images and animations. This work demonstrates that, due to the flexibility of the CPU, CPU-based algorithms remain a valid and competitive choice for achieving global illumination interactively, and an alternative to the generally brute-force GPU-centric algorithms.

CHAPTER 1

Introduction

Achieving realistic physically-based global illumination in real-time for dynamic scenes, running at or above 30 updates per second, remains one of the major goals in the fields of rendering and visualisation. This goal is a challenge due to its inherent computational complexity. Moving from a non-interactive to an interactive solution, one that updates at least once per second, is the first fundamental step towards achieving real-time results. Interactivity brings with it many fundamental challenges and constraints that must be addressed, but if achieved it provides near instantaneous feedback on all the complex light interactions that can occur. This feedback is essential while a user is designing a complex scene or changing and adjusting materials and lighting. There are many commercial fields that would benefit from this level of fidelity and interactivity such as architecture, light design, computer graphic animation, product design, special effects and many others. This thesis achieves the goal of interactive global illumination by developing a number of novel algorithms, demonstrating new CPU-based approaches, which not only achieve interactivity, but do so on a single multi-core desktop PC.

1.1 Physically-based Rendering

Predicting the appearance of a virtual environment consists of a number of stages, detailed in the framework presented by Greenberg [Gre99]. In this framework three distinct stages, detailed below, are presented: the goniometric model, the light transport model and the perceptual model. The focus of this thesis is primarily on the light transport model.

- **Goniometric model:** This stage deals with model acquisition and defin-

ing the geometric or parametric primitives that together define the scene, along with their behaviour with respect to the reflection, transmission and emission of light.

- **Light transport model:** The light model or simulation that defines how light energy that is emitted by a light source is scattered by various interactions with the scene and ultimately how much of it arrives at a particular receiver, normally a camera or an eye.
- **Perceptual model:** Given that the human visual system does not respond linearly to all wavelengths of light or levels of illumination, the physical values returned by the light transport model must be translated and displayed on a particular devices, such a monitor. This model may also be used to aid in the rendering process, as well as providing validation once processing is complete.

There are many methods that can be used to acquire a model of a scene, one being to capture the geometry and reflectance properties by direct measurements, another is to create the model manually. The first approach is useful when the goal is to re-create an existing scene while the second allows for the construction of new geometry and scenarios that, may for instance, be physically impossible to construct. This complex task is a field unto itself and outside of the scope of this thesis, for an overview refer to Watt [Wat93]. The behaviour of the materials in the model such as the reflection, transmissions and emission of light are formalised in the bi-directional reflectance function (BRDF) which is further detailed in Section 2.1.2. Once a model has been acquired the goal is then to visualise it, a process referred to as digital image synthesis or more commonly, rendering [Gla95].

Rendering makes use of lighting simulations to compute an image. Each simulation can be termed a lighting, or light transport, model, which is in turn composed of two components: local illumination which accounts for the emission and reflectance of light from a given surface and global illumination which accounts for the light transport between different surfaces. The mechanics of this are formulated as the rendering equation [Kaj86]. The concept of physically-based rendering refers to a light model where physically accurate measurements and techniques are used for the computation of the local and global illumination components. For further information on radiometry and light transport refer to

Sections 2.1.1 and 2.1.3 respectively.

In the perceptual model, tone mapping attempts to solve the problem of mapping the unbounded energy values produced by rendering, possibly in physical units, to the finite and discrete range supported by the display device being used, such as a monitor. This requires the use of a tone mapping operator, the development of which requires a thorough understanding of both the display technology being used as well as the inner workings of the human visual system, for a complete overview please refer to Reinhard *et al.* [RWPD05]. Perceptual metrics and visual attention models have also been developed to aid rendering in a number of ways. They can identify conditions where rendering can terminate [Mys98] and exploit low-level [YPG01] as well as high-level [CCW03] processes of the human visual system to guide the computation. Finally images can be evaluated once processed to determine how perceptual accurate they are [Mys98, MCTR98].

1.2 Ray tracing

To perform physically-based rendering a simulation of light must be used that allows for realistic light transport between surfaces in a virtual scene. While other methods have been used, as shall be discussed in Section 2, the most common method used is ray tracing. Ray tracing is a method in which the light is simulated by a number of rays, each one containing a discrete amount of energy. As can be seen in Figure 1.2.1 these rays are then cast into the scene and intersected against the scene geometry, with behaviour such as reflectance and absorption dictated by physically-based functions, called bi-directional reflectance functions or BRDFs [Nic65], defined for each surface. These rays are then used to calculate how much light would reach a given receiver, typically an eye or camera. For a visual representation of this process refer to Figure 1.2.1.

Ray tracing algorithms find their roots in the ray-casting methods of [App68] for computing surface visibility. This technique involved shooting rays from a virtual camera into a scene and returning the closest object hit thus accounting for surface visibility. Whitted [Whi80] presented the first ray-tracer for shading computations, which elegantly recursed the computation at the first surface hit for computing shading, reflections and refractions by shooting further rays from that point. This work was extended to account for the full range of global illumination computations, that could be physically-based, in the work of Cook [CPC84] and

Kajiya [Kaj86]. These extensions included effects such as shading from area light sources, complex BRDF modelling for correct specular and glossy reflection and refraction, indirect diffuse computations, caustics, participating media, motion blur, depth of field *etc.* all based on the simple recursive ray tracing computation. As can be seen in Figure 1.2.2 the results can be very realistic.

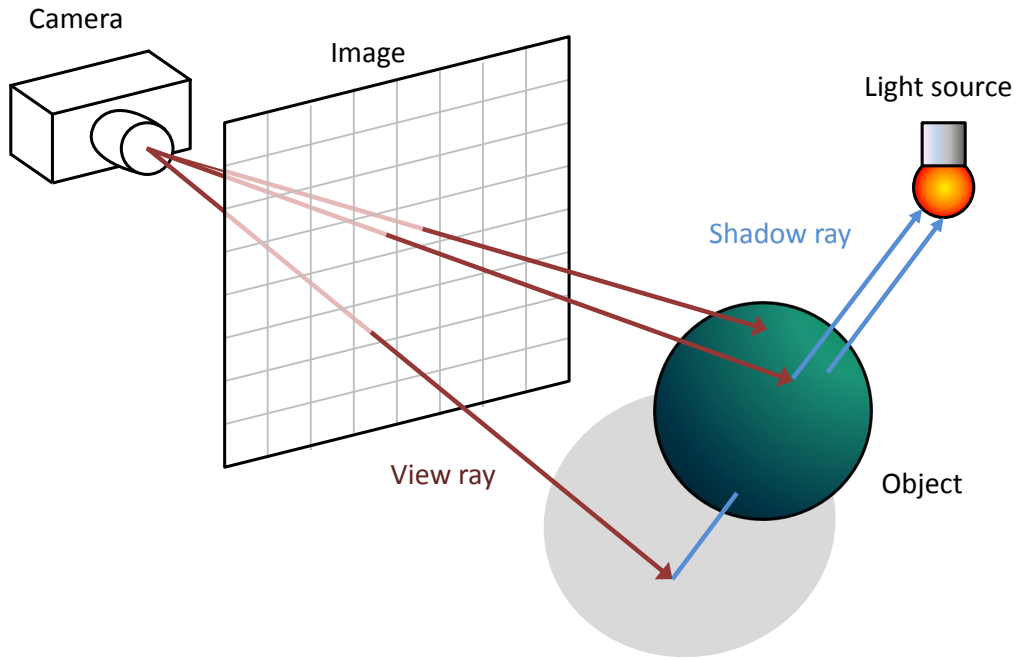


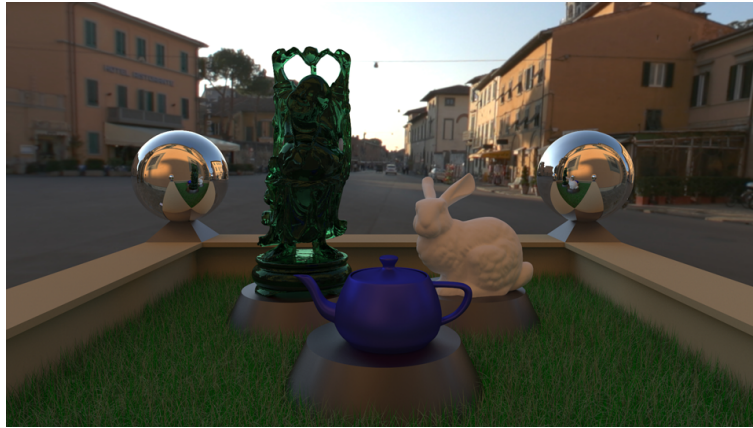
Figure 1.2.1: This figure demonstrates the concept of ray tracing. A ray is cast from the camera through the image plane, passing through a particular pixel. This ray can then intersect the scene at a point. A further ray is then cast from this point towards the light to determine how much energy from the light arrives via this path, assuming it is unobstructed, otherwise the point is in shadow.

1.3 Interactive Global Illumination

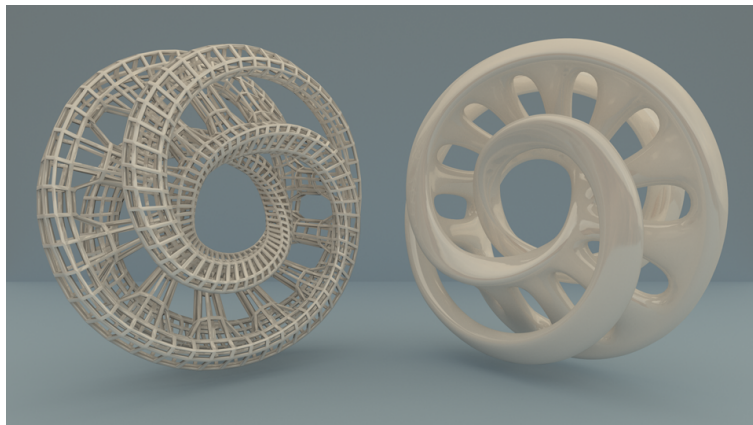
Whitted-style ray tracing solutions [BEL*07] are now running on consumer desktop PCs. The research focus of this thesis is to achieve similar interactive results for interactive global illumination, a much more computational intensive problem. This involves computing complex effects such as diffuse interreflections and



(a) A model of a real building, Sponza [Dab10], relit to simulate a specific time of day.



(b) A scene composed of a number of models: the Stanford bunny and Buddha, Utah teapot and custom grass and enclosure. The custom models were created by hand while the others were captured utilising laser scanning techniques. *Light Probe* courtesy of Paul Debevec [Deb10].



(c) A scene containing two mobius strip models

Figure 1.2.2: Examples of images rendered using path-tracing [Kaj86], a global illumination algorithm. Each image was rendered at a resolution of 1920×1080 and took between 4 to 8 hours to compute on a cluster of machines.

soft shadows, all on a single multi-core desktop PC. The research concentrates on CPU-based approaches due to the fact that ray tracing has many desirable properties and effectively, and until recently, has been mostly a CPU-based method. A wealth of previous research provides optimised data structures, traversal and shading all making full use of the hardware with novel approaches presented in this thesis, refer to Sections 5 - 7, as well as concurrent research efforts detailed in Chapter 3. A further target of this research is to develop generalisable hardware-agnostic algorithms and as can be seen in the literature, such as Section 3.2, the rapidly-evolving high-throughput specialised GPU, while already generalising, necessitates the development of algorithms that are very tailored to the hardware. This is demonstrated by the fact that current CPU-based methods achieve interactive rates comparable to GPU-based methods, even though the GPUs possess a higher computational throughput. Implementations of languages such as CUDA [GGN*08] and OpenCL [Mun08] expose the computational power of the GPU for general purpose development, but due to the evolving nature of the hardware these development platforms are not as stable as the CPU-based ones. While a number of improvements have been made in areas such as acceleration data structure construction and traversal, generalised GPU-based ray tracing is still hard to achieve. Recent systems, such as Optix [PBD*10], attempt to provide this functionality but do so in a relatively closed system due to the inherent complexities. A very limited number of specific methods and entry points for ray generation are provided and an internal acceleration data structure is used, which is not accessible. Even with such highly optimised GPU-based approaches there exist competitive CPU-based ones, such as those by Wald *et al.* [WKB*02] and our methods presented in Chapters 5 and 6, that offer a higher degree of flexibility, from a research and software engineering perspective. The viability of CPU-based algorithms, especially those that exploit parallelism, is further enhanced as the number of cores available, currently a maximum of thirty two in a single machine, continues to increase. This increase in computational power, combined with the rapid convergence of the two platforms, also demonstrates the need to exploit all the resources available when attempting to achieve interactive global illumination, be they CPU-based or GPU-based. CPU-based development offers the ability to focus on algorithmic research without the need to tailor any results to very specific hardware and development platforms. With the additions of new features to GPUs, such as caches, current issues affecting CPUs, such as cache coherency may also become relevant in the near future.

As we shall describe in detail in 3.1.1 and 3.1.2, interactive ray tracing has focused on the exploitation of coherency among primary rays, ones cast from or converging at a point in the scene. In contrast, to attain interactivity global illumination requires the computation of a large number of secondary rays, those that have already been reflected one or more times and therefore don't share a common origin or similar directions. The inherent complexity of the computation combined with limited computational resources requires that new approaches be considered that leverage all the available computational power and make optimal use of it.

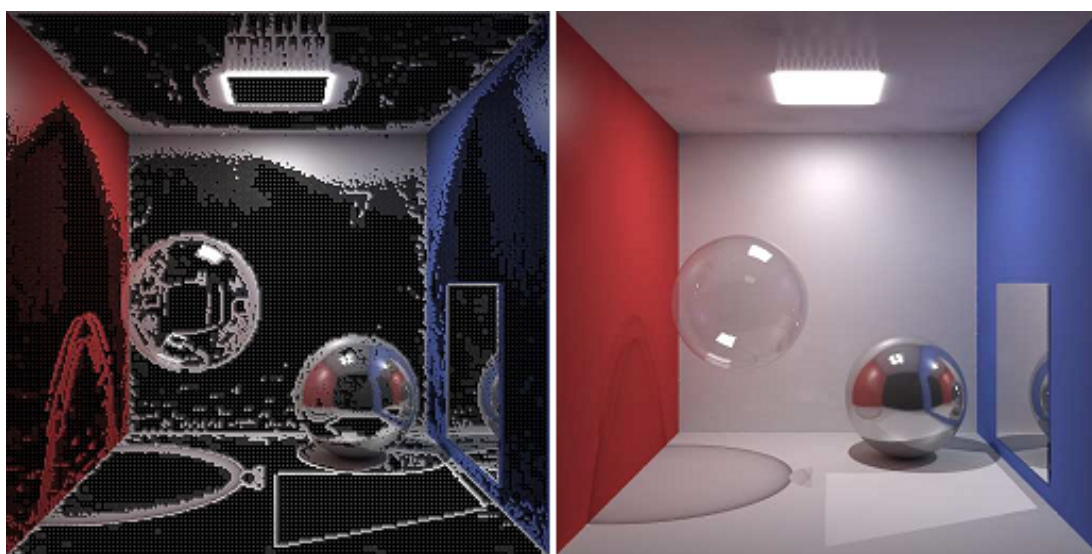


Figure 1.3.1: On the left side is an image that has been adaptively sampled, starting with tiles that are 4×4 pixels in size. The values at the four corners of each tile were computed and if deemed necessary the tile was further recursively subdivided, as needed, into four parts. This image was then interpolated, via bilinear interpolation, using the four corner values at each tile with the resulting image being shown on the right. *Images courtesy of Kurt Debattista [Deb06].*

One such approach, identified in Chapters 3 and 5, is selective rendering, which encompasses progressive [CCWG88], adaptive [Deb06], time-constrained and perceptually-driven techniques [CCW03, SCM04]. This approach focuses computational power on specific areas in a given image where the work would be most beneficial. This is done by not computing all the pixels in the image, but only those that are deemed necessary. In the case of perceptually-driven methods the pixels in the image that are classified as salient or perceptually important to an observer, are where computational resources are focused. The rest of the pixels that are not rendered are generally interpolated in some way, such as those

in Figure 1.3.1 where adaptive sampling was used and the interior pixels of each tile were filled in via bilinear interpolation [Deb06].

Combining interactive ray tracing methods and existing off-line global illumination techniques with selective rendering, to accelerate the computation sufficiently for interactive rates, seems like a natural choice but there exist some fundamental issues which this thesis addresses. As mentioned above, interactive ray tracing methods are dependent on high levels of coherency to achieve their interactive rates, which is in direct conflict with selective rendering which is naturally incoherent due to the fact that it processes only select pixels in an image that may be completely spatially incoherent.

This thesis analyses and quantifies the impact that selective rendering has on interactive ray tracing. These results are then utilised to develop two novel interactive global illumination solutions along with novel data access control mechanism. The interactive global illumination solutions don't rely on any pre-computation and are therefore suited for dynamic scenes where cameras, lights and objects can all be moved. Furthermore, a data access control mechanism is presented that presents a wait-free implementation geared towards highly parallel and interactive systems. These solutions are all designed to function in an interactive context and serve as important steps towards reaching the final goal of a real-time solution, while being of interest to a number of the commercial fields mentioned previously.

1.4 Research Objectives

This thesis aims to combine selective rendering techniques and state-of-the-art interactive global illumination algorithms. The main research objectives of this thesis are:

- to analyse and quantify the impact of combining selective rendering techniques and interactive global illumination algorithms.
- to utilise this information to develop a number of novel CPU-based interactive global illumination algorithms by combining these two approaches. Focusing on interactivity and taking into consideration both interactive and off-line global illumination algorithms.

- to create a rendering system to implement and test these novel algorithms again against current state-of-the-art approaches.

1.5 Thesis Outline

The layout of this thesis is as follows.

Chapter 2: Background presents an overview of the concepts pertaining to realistic image synthesis as well as providing background on all the algorithms that will be covered in this thesis.

Chapter 3: Interactive Global Illumination provides a detailed literature review of interactive ray tracing and global illumination on both the CPU and GPU, as well as a discussion on the problems that the reliance on coherency poses to interactive global illumination algorithms that attempt to adaptively calculate the solution.

Chapter 4: Impact of Selective Rendering on Interactive Ray Tracing analyses and quantifies the impact of selective rendering on interactive ray tracing and identifies where selective rendering can be applied to best effect.

Chapter 5: Adaptive Interleaved Sampling for Interactive Global Illumination builds on the work presented in the previous chapter to develop a novel global illumination algorithm called *Adaptive Interleaved Sampling*. A generalisable approach that combines interleaved sampling with an adaptive approach which uses efficient component-specific adaptive guidance methods to drive the computation. The methodology employed while developing this algorithm is also refined and presented as a framework that provides a structured approach when combining selective rendering methods and interactive global illumination techniques.

Chapter 6: Instant Caching for Interactive Global Illumination takes a previously off-line technique, irradiance caching, and develops an adaptive update method, using the methodology from the framework, to allow the solution to calculate diffuse interreflections at interactive rates. The design of the algorithm enables the extension of the spatial coherence of the irradiance cache into the temporal domain, exploiting the temporal coherence to avoid wasteful computation by using selective techniques.

Chapter 7: Wait-Free Shared-Memory Irradiance Cache develops a new wait-free data access control mechanism, prompted by the use of the irra-

diance cache in an interactive context in the previous chapter, that allows the irradiance cache to be shared among many threads on a shared memory parallel system without contention issues. This approach is evaluated to show its superiority over two traditional data access algorithms: a lock-based approach and a local write approach.

Chapter 8: Conclusions and Future Work concludes the thesis and discusses limitations, extensions, impact and future work.

CHAPTER 2

Background

This chapter introduces the fundamental concepts and knowledge needed to understand the processes and methods behind the generation of physically accurate representations of virtual scenes.

2.1 Introduction to Rendering

This section begins with Section 2.1.1 where the physical quantities utilised in radiometry are introduced. Subsequent sections, Sections 2.1.2 and 2.1.3, discuss the interaction of light with objects within the virtual scenes and light's transport through the scenes in the context of the rendering equation.

2.1.1 Radiometry

The goal of any local or global illumination algorithm is to calculate the distribution of light energy in a scene, be it the whole distribution or part thereof. To achieve this goal an understanding of the physical quantities that represent light energy is needed. Radiometry is the field of study that deals with the physical quantities and measurements of light. This section provides a brief overview of the radiometric units used in the rest of this thesis, following the terms and terminology utilised in Dutre *et al.* [DBB06]. It should be noted that photometric terms utilised in light perception are simply derived from their radiometric counterparts.

Radiant power or flux (Φ) is the primary radiometric quantity which is measured in *watts (joules/sec)*. Flux is not dependant on the size of the source or receiver or the distance between them. Flux simply expresses the total energy

flow at a surface per unit time.

Irradiance (E) is amount of incoming radiant flux per unit surface area, and is measured in *watts/m²*. Radiosity (B) or radiant exitance is the outgoing radiant power per unit surface area. It is also measured in *watts/m²*.

$$\Phi = \frac{dQ}{dt} \quad E = \frac{d\Phi}{dA} \quad B = \frac{d\Phi}{dA}$$

Radiance (L) is the most commonly used quantity in realistic image synthesis and denotes the flux per unit project area per unit solid angle, measured in *watts/(m²steradin)*. This is simply the flux coming in from a specific direction onto a small area.

$$L = \frac{d^2\Phi}{dw dA \cos\Theta}$$

Transport theory, dealing with the mathematics governing the transfer of energy between media, can also be utilised to express the radiometric quantities in terms of a point x and direction Θ where $L(x \rightarrow \Theta)$ denotes the radiance leaving a point x in direction Θ . The relationships between Φ , $E(x)$ and $B(x)$ is examined below, where A is the total surface area and Ω is the total solid angle at each point on the surface.

$$\begin{aligned} \Phi &= \int_A \int_{\Omega} L(x \rightarrow \Theta) \cos \theta dw_{\Theta} dA_x \\ E(x) &= \int_{\Omega} L(x \leftarrow \Theta) \cos \theta dw_{\Theta} \\ B(x) &= \int_{\Omega} L(x \rightarrow \Theta) \cos \theta dw_{\Theta} \end{aligned} \tag{2.1.1}$$

For the remainder of this thesis radiance leaving point x in direction Θ will be represented as $L(x \rightarrow \Theta)$ and radiance arriving at point x from direction Θ as $L(x \leftarrow \Theta)$.

2.1.2 Surface Interactions of Light

When light energy is emitted into a scene it interacts with a number of different objects by getting reflected, transmitted or absorbed at surfaces boundaries. A light reflectance model attempts to model these interactions, usually through the use of a function that attempts to reproduce the physical behaviour of the material.

The simplest and most often used light reflectance model is the Bidirectional

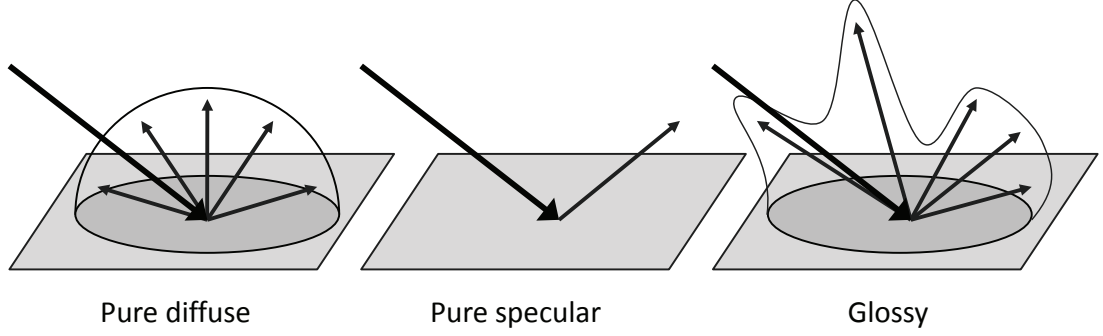


Figure 2.1.1: BRDF examples: The first case demonstrates pure diffuse or Lambertian reflection [Lam60], where light energy is scattered equally in all directions. The pure specular case reflects all incoming energy in one particular direction with no scattering. Finally the glossy case exhibits both diffuse and specular properties, scattering light but having a number of specific directions which are favoured.

Reflectance Distribution Function (BRDF). The BRDF (f_r) is a function that describes the ratio of reflected differential radiance at a point x from direction Ψ to the differential irradiance in the outgoing direction Θ . A more generalised version of the BRDF function is one that has both an entrance and exit point, and models not only reflectance but also transmittance allowing for effects such as subsurface scattering [JC98], this function is called the Bidirectional Surface Scattering Reflectance Distribution Functions (BSSRDF) [Jen01]. Focusing on the BRDF, utilising the definitions from Dutre *et al.* [DBB06], which is denoted as $f_r(x, \Psi \rightarrow \Theta)$:

$$f_r(x, \Psi \rightarrow \Theta) = \frac{dL_r(x \rightarrow \Theta)}{dE(x \leftarrow \Psi)} = \frac{dL_r(x \rightarrow \Theta)}{L(x \leftarrow \Psi) \cos(N_x, \Psi) dw_\Psi} \quad (2.1.2)$$

where $\cos(N_x, \Psi)$ is the cosine of the angle formed by the normal vector N_x and incoming direction Ψ at point x . A common property of BRDFs is reciprocity where the value of the BRDF remains unchanged if the incident and exitant directions are interchanged so that $f_r(x, \Psi \rightarrow \Theta) = f_r(x, \Theta \rightarrow \Psi)$.

Many BRDFs exist and are utilised in image synthesis, the simplest being the pure diffuse and pure specular BRDFs which can be seen in Figure 2.1.1. Real materials are much more complex, such as the glossy example in Figure 2.1.1, and many attempts have been made to capture this complexity. Cook and Torrance [CT81]; Ward [WH92] provides a comprehensive overview of BRDF models, including anisotropic and spectral representations.

2.1.3 Light Transport

Having dealt with surface interactions of light energy this section examines the light transport in a virtual scene. The goal of the light transport is to compute the steady-state distribution of light energy within the scene. This computation accounts for the global aspect of the illumination model and as such is fundamental to any global illumination algorithm. This model is represented as a mathematical equation termed the rendering equation [Kaj86]. For this equation it is assumed that participating media, such as smoke or fog, is not present and that light travels instantaneously. For each surface point x and each direction Θ the rendering equation returns the exitant radiance $L(x \rightarrow \Theta)$. The exitant radiance is the sum of the emitted radiance from a the point x , described as $L_e(x \rightarrow \Theta)$, and the reflected radiance described as $L_r(x \rightarrow \Theta)$. The derivation of the rendering equation below follows Dutre *et al.* [DBB06].

$$L(x \rightarrow \Theta) = L_e(x \rightarrow \Theta) + L_r(x \rightarrow \Theta) \quad (2.1.3)$$

Utilising the definition from Equation 2.1.2 and integrating through, to take into light incoming from all directions on the hemisphere, gives:

$$L_r(x \rightarrow \Theta) = \int_{\Omega_x} f_r(x, \Theta \leftrightarrow \Psi) L(x \leftarrow \Psi) \cos(N_x, \Psi) dw_\Psi \quad (2.1.4)$$

Equation 2.1.3 is substituted in to reach the final form of the rendering equation:

$$L(x \rightarrow \Theta) = L_e(x \rightarrow \Theta) + \int_{\Omega_x} f_r(x, \Theta \leftrightarrow \Psi) L(x \leftarrow \Psi) \cos(N_x, \Psi) dw_\Psi \quad (2.1.5)$$

Equation 2.1.5 is called the hemispherical formulation of the rendering equation. Another formulation which is widely used is the formulation that expresses the rendering equation in terms of the contribution all surfaces make to the reflected radiance. For this formulation, a visibility function $V(x, y)$ is utilised, where $V(x, y)$ is 1 if y is directly visible from x and 0 otherwise. Referring back to Equation 2.1.5, incoming radiance at x from direction Ψ is the same as outgoing radiance from y in direction $-\Psi$, $L(x \leftarrow \Psi) = L(y \rightarrow -\Psi)$. The solid angle dw_Ψ

can reformulated as $\cos(N_y, -\Psi) \frac{dA}{r_{xy}^2}$. This leads to the following formulation:

$$L(x \rightarrow \Theta) = L_e(x \rightarrow \Theta) + \int_A f_r(x, \Theta \leftrightarrow \Psi) L(y \leftarrow -\Psi) V(x, y) \frac{\cos(N_x, \Psi) \cos(N_y, -\Psi)}{r_{xy}^2} dA_y$$

This is more commonly written as:

$$L(x \rightarrow \Theta) = L_e(x \rightarrow \Theta) + \int_A f_r(x, \Theta \leftrightarrow \Psi) L(y \leftarrow -\Psi) V(x, y) G(x, y) dA_y \quad (2.1.6)$$

where:

$$G(x, y) = \frac{\cos(N_x, \Psi) \cos(N_y, -\Psi)}{r_{xy}^2}$$

The rendering equation, or parts of it, are utilised by most rendering approaches when attempting to get realistic results. From classical ray-tracing [Whi80] which would only sample along pure specular and transmitted paths to classical radiosity that only used a perfectly diffuse BRDF [GTGB84] these approaches solved a part of the rendering equation.

While the algorithms mentioned above only solve a part of the rendering equation, there also exist a number of light transport algorithms that attempt to solve it fully. These global illumination algorithms can be divided into two broad categories. The first attempt to solve the light transport problem in an unbiased manner using Monte Carlo ray tracing methods. Foremost among these are path-tracing [Kaj86] and distributed ray tracing [CPC84] which produce unbiased results but suffer due to the large computational cost required to reduce noise in the solutions. Bi-directional techniques [LW93, VG94, Pat93] that traced rays from both the eye and the light sources reduced noise and enhanced overall performance. Metropolis light transport [VG97] took the bi-directional concept further by intelligently sampling the multi-dimensional space of the integral by locating paths proportional to their contribution to the eye. The second category relies on biased techniques, both consistent and inconsistent, such as irradiance caching [WRC88], instant radiosity [Kel97], photon mapping [Jen01] and lightcuts [WFA*05] later extended to multi-dimensional lightcuts [WABG06]. These methods sacrificed some accuracy for a greatly decreased computational time as well as a much more converged and noise-free result. While biased if an algo-

rithm is consistent, such as photon mapping, it converges towards the correct result given enough computational time.

While there are many other aspects of light transport, such as the influence of participating media, that affect the rendering equation these are simply out of the scope of this thesis. For a thorough overview of this area please consult Dutre *et al.* [DBB06]; Shirley and Morley [SM03].

2.2 Primary Rendering Techniques

Sections 2.2.1 - 2.2.3 introduce the three primary rendering techniques utilised in digital image synthesis: rasterisation, ray tracing and radiosity.

2.2.1 Rasterisation

Interactive computer graphics has largely been dominated by rasterisation algorithms. Rasterisation supports scenes composed of geometric primitives that are computed as polygons, directly or through on-demand polygonisation from other geometric primitives such as Non-Uniform Rational Basis Spline (NURBs). The rasterisation pipeline begins with a series of computations that transform the geometry into the view space, followed by a projection onto a 2D image view plane. Lighting is calculated and visibility is usually computed using z-buffer algorithms [Cat74]. Integrated within the z-buffer algorithm, to complete the pipeline is the rasterisation process. This is typically composed of polygon-fill algorithms, texturing and sometimes aspects of the lighting computation. Rasterisation has been favoured by the gaming industry and film industry [CCC87] due to its simplicity and rendering speed. Over the past decade real-time speeds have been accentuated further by dedicated graphics hardware such as the graphics processing unit (GPU). The GPU has also been utilised to compute ray tracing as well as global illumination solutions as discussed in Sections 3.1 and 3.2.

Initially rasterisation-based renderers struggled to compute more complex lighting effects. With the introduction of shaders, user-defined programs or kernels, that replace a specific stage of the rendering pipeline such as the vertex or pixel stage, more advanced effects became possible. Still even simple effects, such as shadowing for point light sources, required another computation pass of the rasterisation pipeline [Cro77, Wil78]. Other algorithms, and typically extra pipeline passes, are required for specular [OR98] and glossy reflections [DB97],

area lights [HLHS03], caustics [Wym08] and other lighting computations, although very often these are not physically-based but physically-inspired. Other methods can be applied for pre-computing non-specular effects, such as radiosity [GTGB84] or PRT [SKS02] and its extensions [ZHL*05, PWL*07, IDYN07]. While these methods now support rigid-body dynamic scenes they, as yet, do not handle deforming objects.

Recently with the increase in computational power of the GPU, as well as the ability for shaders to generate new geometry, a number of more complex effects, such as depth of field [BK08], dynamic indirect lighting [SIMP06b, RGKS08, ML09], soft shadows [DL06, SGNS07], complex reflections [YYM05, EMDT06] and refractions [Wym05, OB07] have been added to the rasterisation pipeline. A comprehensive overview on interactive rendering methods using rasterisation is provided by Akeine-Moller *et al.* [AMH02].

2.2.2 Ray Tracing

The class of algorithms termed ray-tracing, introduced in Section 1.2 find their roots in the ray-casting methods of [App68] for computing surface visibility. This technique was used for hidden surface removal and involved shooting rays from a virtual camera into a scene and returning the closest object hit thus accounting for surface visibility.

Whitted [Whi80] presented the first method to compute illumination via ray tracing. This method is now known as classical or Whitted-style ray tracing. This view dependant method shot rays from the camera out into the scene identifying the first object hit. At this point shading was calculated by shooting extra rays towards all light sources to determine visibility. If the intersection was with a specular object one or two rays (to account for reflection and refraction) were recursively shot. While this method solved the rendering equation for specular surfaces, in the same way that radiosity solved it for diffuse interactions, extensions were needed for ray tracing to solve all aspects of the rendering equations. These extensions were initially provided by distribution ray tracing [CPC84] and path tracing [Kaj86] which utilised Monte Carlo techniques to provide a full global illumination solution. Further extensions to this work would eventually include effects such as shading from area light sources, complex BRDF modelling for correct specular and glossy reflection and refraction, indirect diffuse computations, caustics, participating media, motion blur, depth of field *etc.* all based

on the simple recursive ray tracing computation.

Furthermore, ray tracing offers the option to be able to compute intersections directly with many different geometric types (without tessellation) and is easy to parallelise. The ability for ray-tracing to produce realistic images meant that software based around these algorithms was developed for use in realistic lighting simulations, see for example Ward [WRC88] and more recently has begun to be used in film production [Her04, Chr06]. For a comprehensive overviews of ray tracing methods refer to Glassner [Gla95]; Shirley and Morley [SM03].

2.2.3 Radiosity

Classical radiosity was introduced into the computer graphics field from thermal engineering by Goral *et al.* [GTGB84]. This view-independent finite-element method solves the global illumination component of the rendering equation for perfectly diffuse surfaces and only handles diffuse to diffuse interactions. It does this by discretising the scene into a series of patches and computing the radiosity for each patch, which is the total power leaving the surface of the patch. This computation is a system of N simultaneous linear equations where the fraction of power arriving at one patch from another is called the form factor. The computation of the visibility between patches is potentially the most expensive computation in the radiosity pipeline and a number of approaches have been proposed to tackle the problem.

The most popular approach to solve the visibility problem is to use projection. The form factor with any surface, calculated from a particular point, is simply the projection of that surface onto the hemisphere around the point. The hemi-cube method [CCWG88] uses rasterisation to project the hemisphere on the faces of a hemicube using techniques such as hidden surface removal to accelerate the computation. Other methods to solve the visibility problem use different projections and ray casting techniques, further details can be found in Sillion and Puech [SP94]; Ashdown [Ash95].

Solving the radiosity equation, once the form factors have been determined, is usually done by expressing the system of linear equations as a matrix. Iterative methods, such as the Jacobi or Gauss-Siedel, can then be used but these solutions have a complexity of $O(N^3)$ when dealing with N elements. This can be reduced by managing patch complexity adaptively [HSA91] or by using faster converging techniques [CCWG88]. Another approach is to use stochastic meth-

ods that replace form factor computation with form factor sampling, improving both the computational speed and reducing memory usage [Bek99, DBB06].

Solving the radiosity equation for other surface interactions, such as glossy or specular, has been attempted [AH93] but the solution required a more complex surface subdivision approach and form factor calculation, increasing computational time substantially. Most other solutions that utilise radiosity to solve the rendering equation [Kaj86] do so by using a hybrid solution that incorporates ray tracing to compute the specular components. Some of these methods are covered in more detail in Sections 3.2.1.2 and 3.2.2.1.

View independent radiosity has a number of advantage over ray tracing. Lighting only needs to be computed once, and can then be re-used for interactive walkthroughs of static scenes, where the geometry and lighting do not change.

2.3 Accelerating Rendering

In this section a number of techniques are examined that are used to accelerate some aspect of rendering. Section 2.3.1 examines how the rendering equation can be broken up into a number of components and how algorithms can target specific components to reduce the overall computational time. In Section 2.3.2 adaptive and progressive approaches are investigated which focus computational power, using some form of guidance, in areas of highest benefit. Section 2.3.4 examines dynamic acceleration data structures and their application and contribution to interactive ray tracing. Finally Section 2.3.5 introduces the irradiance cache, extended in Chapters 6 and 7, and its current extensions and uses.

2.3.1 Component-based Rendering

The subdivision of the rendering process into components has been proposed a number of times in order to make the computation more efficient. The work that paved the way for component-based approaches was shade trees [CPC84] introduced by Cook and used for the REYES renderer [CCC87]. Shade trees, later extended by Perlin [Per85] to include control structures and flow, allowed different BRDFs, see Section 2.1.2, to be used on surfaces in the same scenes requiring the computation of both specular and diffuse components.

Initial approaches that computed components separately, as a means of solving the rendering equation [Kaj86], were termed multi-pass algorithms. These algorithms combined different approaches, mostly classic radiosity and ray tracing. In Wallace *et al.* [WCG87] a multi-pass algorithm was presented that utilised the z-buffer algorithm to calculate view dependant planar reflections and a rendering pass to compute the diffuse component. This work was extended by Sillion and Puech [SP89] where ray tracing was utilised for computing the specular component and the form factors of the remaining non-planar objects. Ward [WRC88] decoupled and cached the expensive indirect diffuse component in his distributed ray tracer, using standard ray tracing for the direct diffuse and specular components. A three-pass method was used by Shirley [Shi90] to calculate various components: path tracing from the light source was used to calculate caustics, radiosity was used for indirect illumination and stochastic ray tracing calculated the remaining components. Another such algorithm was presented by Heckbert [Hec90] where an approach similar to adaptive radiosity was used for the indirect illumination component and the result stored in textures, which were adaptively subdivided based on their screen size. This approach used ray tracing to compute the remaining components adaptively sampling both light rays as well as eyes rays, in the same fashion as Whitted [Whi80]. Chen *et al.* [CRMT91] introduced an entirely progressive algorithm which used a number of different approaches for the components. The indirect diffuse component was computed using a progressive radiosity pass, using ray tracing for the specular components. Caustics were computed using light tracing with direct lighting computed using standard ray tracing. The approach allowed for the progressive radiosity computation to be halted, a view selected and the high frequency caustics and direct lighting to be computed. The low frequency lighting could then be further refined by either continuing the progressive radiosity solution or by using path tracing. The concept of a lighting networks was introduced by Slusallek [SSH*98], and could be viewed as an evolution of shading trees [CPC84]. This framework formalised many aspects of the multi-pass methods, allowing different algorithms to compute distinct parts of the rendering problem and arranging these in a network. This network could have many topologies, including loops and allowed users to specify their own networks where only parts of the scene were computed with a given algorithm. This approach, due to its complexity, developed a regular expression system to express these networks, using similar notation to Heckbert [Hec90], which allowed for the detection of redundant and missing light

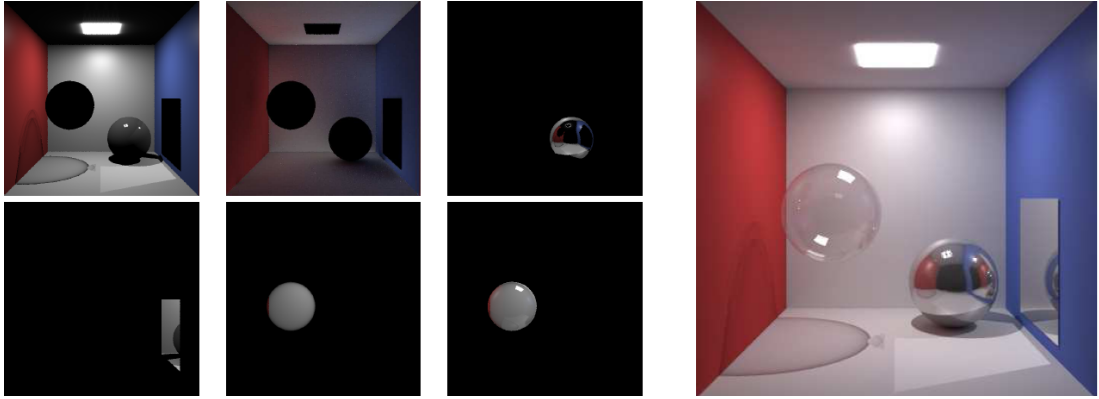


Figure 2.3.1: A Cornell Box scene split into a number of components: (top-left) direct, (top-middle) indirect diffuse (top-right) pure specular, (bottom-left) glossy (bottom-middle) transmitted (bottom-right) reflected. *Images courtesy of Kurt Debattista [DSSC05].*

paths.

More recently approaches such as Stokes *et al.* [SFWG04] and Debattista *et al.* [DSSC05] have combined component-based approaches with perceptual rendering, covered in Section 2.3.2. Stokes *et al.* [SFWG04] presented a perceptual metric, which was combined with path tracing, to predict the importance of the components for a given scene. The metric used the primary rays of the path tracing computation to collect information about the scene, this information combined with the metric was then used to allocate resources based on perceptual importance. In Debattista *et al.* [DSSC05] a component regular expression (*crex*) was introduced that allowed for fine grained control with regard to the components that were computed and the order they were computed in, see Figure 2.3.1. Combining the *crex* with a perceptual metric allowed for the reduction of computational complexity while maintaining perceived visual quality.

2.3.2 Selective-Rendering

Selective rendering is defined by Debattista [Deb06] as “*those techniques which require a number of rendering quality decisions to be taken and acted upon prior to, or even dynamically during the actual computation of any image or frame of an animation*”. This covers a large number of approaches and algorithms including those that are adaptive, progressive and time-constrained.

2.3.2.1 Rasterisation

Techniques applicable to off-line selective rendering go back to early work done on level of detail by Clark [Cla76]. While most techniques stored a distinct version of the model for each level of detail and discretely switched between them the work presented in Luebke and Hallen [LH01] utilised an adaptive perceptual metric to select an appropriate level of detail. Bergman *et al.* [BFGS86] presented a different approach in which the complexity of the shading was adaptively adjusted on a per polygon basis, from displaying only vertices to using Phong shading with anti aliasing. The focus of this thesis is on ray tracing and therefore for further comprehensive overviews of rasterisation-based approaches please refer to Akenine-Moller and Haines [AMH02]; Luebke *et al.* [LWC*02]

2.3.2.2 Ray tracing

Adaptive and progressive methods for ray tracing were presented in the earliest approaches, such as Whitted [Whi80], where the removal of aliasing was performed by recursively subdividing based on the radiance at the corners of pixels. Mitchell [Mit87] presented a ray tracer which first coarsely sampled the image plane, using non-uniform Poisson sampling, and then refined the samples based on an adaptive heuristic which took into account the relative intensity of samples as perceived by the non-linear response of the human visual system. Painter and Sloan [PS89] presented an algorithm that was both adaptive and progressive, using a kd-tree for both storing samples and identifying where new samples should be generated based on a number of heuristics that took into account the area of the kd-tree node as well as if the node was on an edge of an object. Like Mitchell [Mit87] they suggested exploiting the non-linear response of the human visual system but the suggestion was implemented by Meyer and Liu [ML92]. A general rendering approach which has the ability to adaptively adjust its computation when it identifies areas of high importance, based on their radiance contribution, is the Metropolis Light Transport [VG94]. Guo presented a progressive rendering approach based on the use of a directional coherence map (DCM) [Guo98]. The image plane was regularly and then recursively subdivided into blocks of four, using a quadtree, and the radiance at the vertices of each block was calculated. Blocks were then progressively subdivided into two groups, smooth and edge blocks. Further subclassification was then performed on edge blocks to mark them as complex if more than one edge was detected

within the block. This occurred recursively but at any point the image could be reconstructed using the DCM. The work was further extended in Farrugia and Péroche [FP04] where a perceptual metric rather than the DCM was used. Direct lighting can also be selectively calculated using techniques such as those presented in Ward [War91b] and Shirley *et al.* [SSW*06]. For Ward [War91b] the contribution of multiple lights was sorted based on contribution potential, which was based on criteria such as: distance of light from surface, light intensity and size of light source. The lights with the highest potential had shadow rays calculated for them until a certain threshold was reached after which a statistical simulation, based on previous visibility tests, was used to approximate the rest of the contributions. In Shirley *et al.* [SSW*06] the scene was divided into cells that stored differentiated between lights that considered important and ones that weren't, which were then sampled accordingly. Jin *et al.* [JIC*09] provided an adaptive and selective approach to tackle supersampling in an interactive ray tracing context where both image-space and object-space attributes were used to calculate a priority which then guided the computation.

2.3.2.3 Perception

Perceptually assisted selective rendering algorithms that apply perceptual considerations when computing progressive or adaptive refinement, such as Mitchell [Mit87] reviewed above, exploit the human visual system to drive the computation. Bolin and Meyer [BM98] and Ferwerda *et al.* [FSPG97] produced frequency-based ray tracers that used very complete models of the human visual system and incorporated aspects such as spatial processing and visual masking. Visual difference predictors have also been used to direct samples in stochastic ray tracing as well as determine stopping conditions [Mys98, BM98]. These visual difference predictors were costly to compute though as they had to be re-calculated many times each frame until Ramasubramanian *et al.* [RPG99] decoupled their spatially-dependant saliency component from the luminance dependant component. This led to many selective rendering implementations that used saliency models such as Yee *et al.* [YPG01] where a saliency model they term the *Aleph Map* was used to influence the search radius for samples when performing the indirect-diffuse lighting calculations from an irradiance cache, refer to Figure 2.3.2 for an example. Haber *et al.* [HMYS01] utilised saliency maps and task objects to identify the most salient objects on which the glossy and specular

components where rendered in higher detail. In Cater *et al.* [CCW03]; Sundstedt *et al.* [SCM04] saliency maps and task maps where used to vary the number of samples calculated in a global illumination framework based on the *Radiance* renderer by Ward [War94]. Sparse sampling methods such as Walter *et al.* [WDP99], all covered in detail in Section 3.2.1, also used adaptive techniques to focus computation in areas of importance. For an comprehensive overview of recent work in perceptually adaptive graphics please refer to O’Sullivan *et al.* [OHM*04]; De-battista [Deb06].

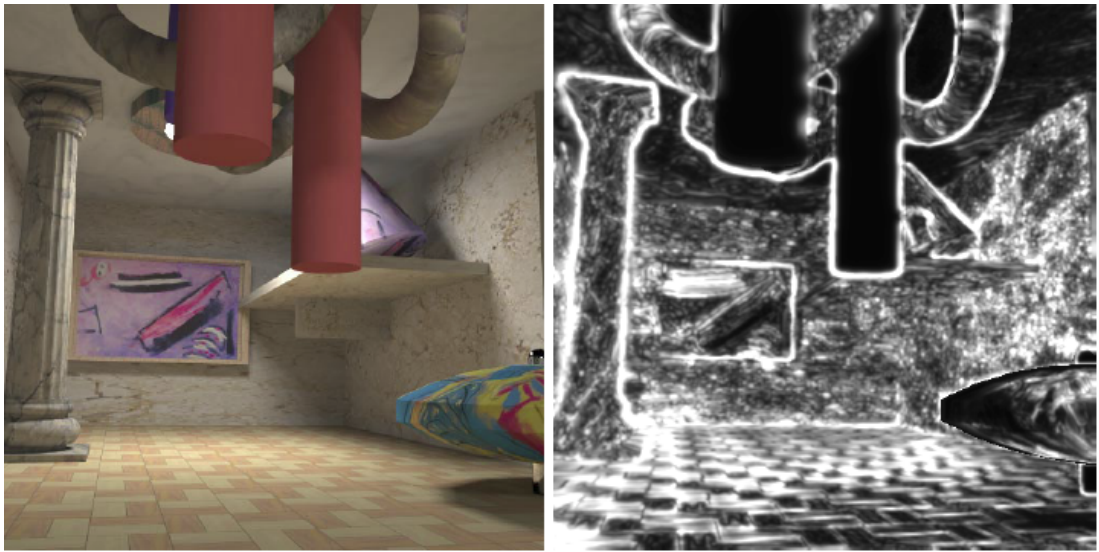


Figure 2.3.2: An adaptively rendered scene, showing on the left, utilising an Aleph map [YPG01], shown on the right, as guidance. *Images courtesy of Hector Yee [YPG01].*

2.3.3 Interleaved Sampling

When correlated samples are utilised they are faster to generate and more coherent but this is at the expense of visible sampling patterns and structured noise. Decorrelation of these samples on the other hand increases variance and therefore generates random noise. Interleaved sampling [KH01] combats this by creating an interleaved sampling pattern. This is done by blending smoothly between both regular and irregular sampling, interleaving the samples of a regular grid, which are correlated, in an irregular way to maintain coherency but reduce the aliasing of the sampled image.

Interleaved sampling has been used to accelerate both ray racing and rasteri-

sation. In the paper by Wald *et al.* [WKB*02] it was used to accelerate the global illumination solution by allowing only a subset of the virtual point lights that were generated to be sampled per pixel. Segovia *et al.* [SIMP06b] also proposed a real-time method for interleaved sampling where they introduced a technique to maintain coherency between neighbouring pixels by splitting the rendered image into a number of sub-buffers. These sub-buffers contain only pixels from a specific part of the interleaved sampling pattern. These are then later recombined and filtered taking into account discontinuities, see Figure 2.3.3. In Sloan *et al.* [SGNS07] and Forest *et al.* [FBP08] interleaved sampling is used to accelerate shadow generation on the GPU by reducing the overall number of samples that need to be calculated.

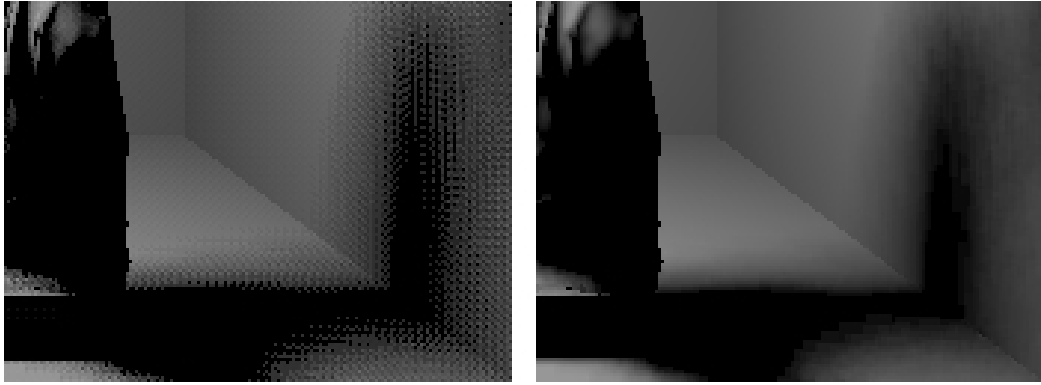


Figure 2.3.3: An example of the structured noise created by interleaved sampling is shown on the left, this noise can be greatly reduced, as can be seen on the right. This is achieved by utilising a correctly-sized filter kernel, same size as that of the interleaved sampling pattern, along with discontinuity buffering [WKB*02].

2.3.4 Dynamic Acceleration Structures

Since ray tracing calculations for computing visibility must intersect the geometry within a scene, if each ray were to intersect each geometric primitive the algorithm would be intractable. Much research has gone into the construction of efficient data structures for ray traversal to minimise the computational cost of ray-object intersections. These data structures, commonly known as acceleration data structures, are usually based on subdividing space uniformly or adaptively. An overview of all acceleration structures techniques is beyond the scope of this thesis therefore the focus of this section is on the creation of dynamic acceleration data structures. For a full and comprehensive overview of acceleration structures

please refer to Havran [Hav01]; *Hunt et al.* [HMS06]; Havran *et al.* [HHS06]; Wald *et al.* [WBS07].

When ray tracing developed the ability to ray trace scenes at interactive rates on single machines [RSH05, Wal04] a significant factor, besides faster hardware, was more effective acceleration structures and enhanced traversal algorithms. At the time kd-trees were observed to give the best performance [Hav01] especially when using the Surface Area Heuristic (SAH) [MB90] during construction.

Three primary strategies were identified at that time that enabled interactive ray tracing:

1. Avoid rebuilding the kd-tree [LLAm01, WKB*02, GFW*06].
2. Utilise and optimise other acceleration structures such as:
 - Grids [RSH00, WIK*06].
 - Bounding Volume Hierarchies (BVHs) [LYTM06, WBS07].
 - Hybrid structures [WK06].
3. Optimise the construction of SAH based kd-trees, primarily by parallelising and optimising the evaluation of the expensive SAH cost function [HMS06, PGSS06, CKL*10].

Shevtsov *et al.* presented a fast and highly parallel kd-tree construction algorithm that allowed for a full kd-tree rebuild of an entire scene every frame [SSK07], Wald *et al.* published two papers that redirected a large portion of the research towards BVHs. The first presented an optimised method for firing large ray packets (several times bigger than the SIMD width) through a BVH [WBS07], while the second showed how fast kd-tree rebuilding techniques could be applied to BVHs and how these methods provided almost ten times more speed-up when utilised in this context [Wal07]. At the same time Yoon *et al.* presented an algorithm to selectively restructure a BVH based on the output of two new metrics that measured the restructuring benefit and culling efficiency [YCM07]. Finally Wald *et al.* presented three new methods that allowed for fast, parallel and asynchronous construction of BVHs; a full rebuild using a fast binning approach when evaluating the SAH cost function; a parallel version of the binned build; and, a asynchronous build that occurs over multiple frames [WIP08].

Up until this point the focus has been in tracing large packets of rays through the BVH [WKB*02] and the construction of the structures had reflected this.

More recently a number of publications have investigated building a BVH so that SIMD utilisation can be exploited not by tracing multiple rays against one node, but by tracing one ray against multiple nodes of the structure. Ernst *et al.* [EG08], Dammertz *et al.* [DHK08] and Wald *et al.* [WBB08] have all recently presented techniques for the construction of a BVH with more than two branches, generally four, for each node.

All the work above has focused on construction of acceleration structures on the CPU. Recently new work has dealt with construction acceleration structures entirely on the GPU. A technique for construction a full kd-tree entirely on the GPU was presented by Zhou *et al.* [ZHWG08]. This method constructs nodes in a breadth-first order to exploit the significant streaming performance that the GPU provides. In the same vain Lauterbach *et al.* presented an algorithm for BVH construction on the GPU, providing two versions: one general version for use with methods such as collision detection, and a specialised version tuned for ray tracing [LGS*09]. This approach was extended in Pantaleoni and Luebke [PL10] where two construction approaches were presented, a hierarchical variation of the construction approach along with an SAH-optimised variation.

2.3.5 Irradiance Caching

The irradiance cache was first presented in Ward [WRC88] and is a data structure used to accelerate the computation of indirect diffuse interreflections in a view driven fashion. Traditionally the indirect diffuse component of the lighting is calculated by sampling the hemisphere around a particular point, via recursive ray tracing, to calculate the total irradiance. This is a computational expensive process due to the recursive nature of the algorithm and the sampling density required. Ward noted and exploited the fact that the indirect diffuse component is generally a continuous function that does not suffer the high frequency changes common to direct lighting and the specular component. When an indirect diffuse computation occurs the irradiance cache is first queried to determine if any samples exist within a given search radius, if they do they are then extrapolated/interpolated from to produce the required value. If a sample is not found the full computation occurs, and the result is stored in the cache for future computations. The range searches, performed to locate valid samples within the cache, are accelerated by the use of an octree [Gla84] which is incrementally built every time a new sample is added by storing the new irradiance value as well as

updating the octree topology. An example of the resulting sample locations can be seen in Figure 2.3.4. By caching the results of the expensive indirect diffuse computations an order of magnitude decrease in the overall computation time was observed. This caching approach has been extended for accelerating the computation of other components such as subsurface scattering [KLC06] as well as participating media [JDZJ08].

The irradiance cache has been extended in many ways, the first of these was the use of irradiance gradients [WH92]. These gradients, both rotational and translational, were stored along with the cache samples and used to improve performance and reduce artifacts by adjusting the shape of the search radius which was valid during range searches. A new approach for calculating the error metric, which dictates when new samples are created and when they are extrapolated/interpolated, was introduced by Tabellion and Lamorlette [TL04]. They also demonstrated how the irradiance cache had been utilised in accelerating global illumination in a framework used for rendering imagery for computer generated movies. Krivanek *et al.* [KGBP05] presented the radiance cache, which extended the caching of diffuse interreflections by allowing glossy interreflections to be cached as well. This work was then further ex-

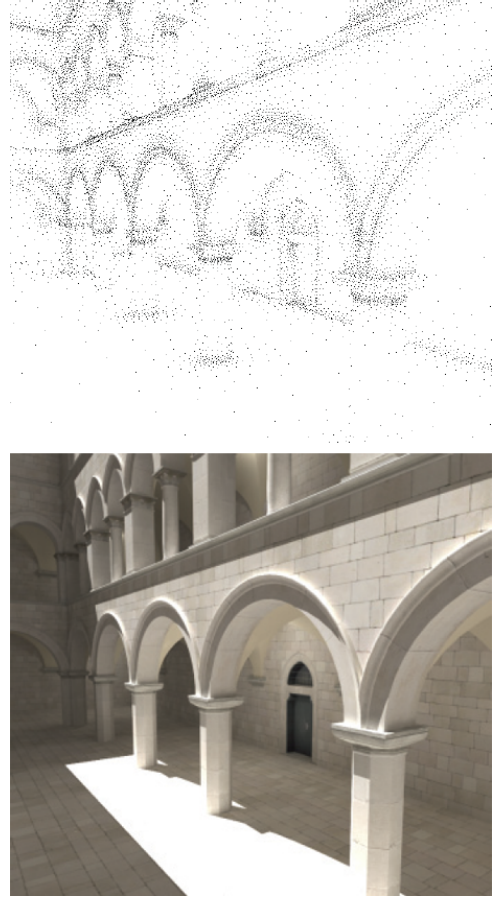


Figure 2.3.4: The top image show a visualisation of samples present in an irradiance cache, with the resulting rendered image at the bottom. *Images courtesy of Jaroslav Krivánek [KGW*07].*

extended by adapting the search radius during computation to improve the performance and quality of the caching [KBPv06]. Arikan *et al.* presented an approach in which they decoupled the final gathering of distant and local geometry to enhance performance. A GPU implementation was also presented, and while faster than the traditional implementation it was limited to computing single bounce

indirect lighting [GKBP05].

Another approach used to accelerate the irradiance cache is to run it on a parallel system. In these systems each thread or process might evaluate new irradiance values and add them to the cache. To increase efficiency the cache must be shared among all the processes, to avoid work replication, making it a shared data structure. This requires some form of access control mechanism, see Section 2.4, which ensures that the data is accessible, updatable, doesn't become corrupted and that whose overheads do not compromise performance or efficiency. In a distributed memory system multiple copies of the structure are maintained and need to be synchronised. In the Radiance distribution [War94] this was achieved by using the Network File System (NFS) for concurrent access to the cache. It was important to use an efficient file locking manager as otherwise contention would lead to poor performance. Another approach presented by Koholka *et al.* [KMG99] was to broadcast cache values amongst processors, each time 50 samples were calculated by a slave node. Robertson *et al.* [RCLL99] presented a centralised parallel version of Radiance where the calculated cache values were sent to a master node whenever a threshold was met. Each slave node then collected the values from the master node at regular intervals. Finally Debattista [Deb06] proposed restricting diffuse irradiance evaluations to a subset of the available nodes and synchronising the cache among these at a higher frequency than with the remaining nodes.

Due to the fact that the cache was view independent the cache could be re-used on subsequent frames as long as the scene remained static and only the camera moved. Attempts have been made at temporal caching for use in dynamic scenes such as the work in Tawara *et al.* [TMS04] where the irradiance cache was extended into the temporal domain by updating the cache samples visibility rays, used when computing the final gathering, over time. The ageing scheme used identified the oldest samples and updated them, causing invalid samples to be used while objects were in motion. Smyk *et al.* [SKDM05] presented another temporal method based on caching the final gathering rays in photon mapping along with a new anchor data structure. Anchors were used to group final gathering rays, used to compute the cached sample, with the photons. The final gathering rays were linked with the closest anchor and when unavailable a new anchor was created on demand. Changes in the irradiance value at an anchor would trigger updating the irradiance cache strata linked with it. This method required the use of an extra abstract data structure (the anchor kd-tree) to bind

the photons with the final gathering rays and all the mechanisms related with it to manage anchor creation, update and deletion, on top of photon mapping and irradiance caching. Finally Gautron *et al.* [GBP07] presented a temporal solution that catered for changes in the temporal domain and included computation of a temporal gradient. This method predicted the incoming lighting and how it would affect the cached samples, with the drawback being that paths through the cache needed to be known before hand to correctly predict illumination. It should be noted that all the methods presented above, the original irradiance cache, the extensions as well as the distributed and temporal approaches all worked off-line and were not utilised in any interactive systems.

2.4 Synchronisation

This section will examine the different approaches to access control mechanisms for shared data structures, and their relative advantages and disadvantages. Traditionally, access control to shared memory data structures is maintained via mutual exclusion, a property that dictates that only one thread or process can access a particular piece of data at a time. The area of code where mutual exclusivity must be maintained, and therefore concurrent access cannot occur, is termed a critical section.

2.4.1 Blocking

Blocking occurs when locking mechanisms, such as semaphores, mutexes and monitors, are used to guard critical sections [Dij68]. When one process attempts to acquire a lock that is already held it will block until the lock is free.

Blocking is undesirable for many reasons, while a process is blocked it cannot perform any work which may be unacceptable in a real-time or high priority system. Certain lock interactions can also lead to conditions such as deadlock, livelock and priority inversion. Deadlock is a situation where a process blocks acquiring a lock but never acquires it due to another process never releasing it. Livelock, a special case of resource starvation [Dij68], involves two processes changing their states continually but never progressing. Finally priority inversion is a situation where a lower priority process will run blocking a higher priority process waiting for a locked resource. Blocking also entails expensive context switches and increases contention as the number of processes increases.

2.4.2 Busy-waiting

When frequent access to a shared data structure may be required, the cost of blocking may be prohibitive. In this case if another process lies within the critical section a process is made to busy-wait instead of block, by continuously checking if the lock is available, until access is allowed. An example of a busy-wait or spinning technique is the spin lock. Such control mechanisms incur overheads, such as serialisation of accesses to the shared data structure, but avoid expensive context switching or re-scheduling that occurs for blocking mechanisms. Busy waiting of frequently-accessed resources leads to contention which can drastically reduce performance as the number of threads increases [ALL89]

2.4.3 Non-blocking

An alternative is algorithms that utilise non-blocking synchronisation, an approach which avoids mutual exclusion by carefully ordering instructions. These algorithms can eliminate code serialisation by removing all critical sections and also reduce contention [HS08]. Non-blocking algorithms are classified in three main categories: obstruction-free, lock-free and wait-free.

The weakest form of the non-blocking approaches take the form of obstruction-free methods. An algorithm is obstruction-free if it can guarantee that a thread can complete in finite time if it operated in isolation (i.e. with all other threads suspended). This requires that any partially-completed operation can be aborted and the changes made rolled back at any time.

Lock-free algorithms guarantee that at least one among a set of concurrent threads will complete in finite time. This allows individual threads to starve but guarantees system-wide throughput. This process generally occurs in four phases: completing one's work, assisting an obstruction, aborting an obstruction and waiting. Completion of assigned work is complicated by the possibility of assistance and abortion, but is generally the fastest path to complete execution. The decision whether to assist, abort or wait if an obstruction is encountered is typically the most complex part of a lock-free algorithm. All lock-free algorithms are obstruction-free.

Lock-free and obstruction free methods rely on retries and cannot guarantee an upper bound on the number of executed instructions. When all threads are guaranteed to complete in finite time the algorithm is said to be wait-free. With a guaranteed upperbound on the number of instructions wait-free algorithms

Listing 2.1: “Fetch and Add” (XADD) and “Compare and Swap” (CAS)

```
1  atomic XADD(address location)
2  {
3      int value = *location;
4      *location = value + 1;
5      return value
6  }
7
8  atomic CAS(address location, value cmpVal, value newVal)
9  {
10     if ( *location == cmpVal )
11     {
12         *location = newVal;
13         return true;
14     } else return false;
15 }
```

avoid starvation, deadlock and livelock and priority inversion and are ideal for multiprogrammed multiprocessors. All wait-free algorithms are also lock free. It has been shown that that all algorithms can be implemented in a wait-free manner [Her88]. While many approaches that transform serial code, called universal constructions, have been presented these approaches generally result in reduced performance, even when compared to a standard blocking approach.

2.4.4 Atomic Primitives

Atomic primitives are also key in the development of non-blocking algorithms. These single instruction functions can be executed without any interruptions on modern hardware. They can be seen as reducing the critical section of an algorithm into individual indivisible machine instructions. Herlihy [Her91] provides a hierarchy of the effectiveness of such primitives, the most effective being those that can be used to implement any wait-free data structure which he described as being compare and swap (CAS) or the load-link store-conditional (LL/SC) instruction pair, an alternative to compare and swap found on some architectures. While effective, it has been shown that CAS or LL/SC primitives cannot provide starvation-free implementations of many common data structures without memory costs growing linearly in the number of processes [FHS04]. This makes implementation of wait-free algorithms challenging. Pseudo code (Listings 2.1) is shown for the two atomic instructions, fetch and add (XADD) and compare and swap (CAS), which are used later in Chapter 7.

CHAPTER 3

Interactive Global Illumination

This chapter provides a thorough review of both interactive ray tracing and global illumination on both the CPU and GPU. This work builds on the rendering concepts introduced in Chapter 2 and details the primary area of research that pertains to this thesis, interactive global illumination. The information presented here pertaining to the developments in this field is then utilised during the development and presentation of a number of novel interactive global illumination algorithms further detailed in Chapters 5 - 7.

This chapter begins by introducing interactive ray tracing (IRT) and examines how the CPU (Section 3.1.1 and GPU 3.1.2) are utilised in computing interactive ray tracing solutions. In Section 3.2 interactive global illumination algorithms are examined in detail investigating both CPU (Section 3.2.1) and GPU (Section 3.2.2) methods. Section 3.3 examines the role coherence plays in interactive ray tracing and how this coherence is exploited to increase computational throughput and how this can be combined with sparse sampling methods. Finally Section 3.4 provides a summary of the chapter.

3.1 Interactive Ray Tracing

This section examines interactive ray tracing which encompasses systems and algorithms that produce Whitted-style [Whi80] results at interactive rates, see Section 2.2.2 for an overview of Whitted-style ray tracing. Interactive in this context refers to the ability of the system to complete the computation of the solution and update it at least once per second. These systems focus on coherent primary rays, those that leave the camera and strike the scene, as well as a very limited number of coherent secondary rays for effects such as hard shadows,

reflections and refractions.

These systems, generally, form the basis or structure for interactive global illumination systems. These global illumination systems require even a larger number of rays, additional secondary rays, to be traced. These allow for effects such as diffuse interreflections, soft shadows, motion blur and many others to be calculated. The ability to generate the primary rays at interactive rates is therefore critical for allowing interactive global illumination systems to function.

3.1.1 CPU Algorithms

This section focuses on the detailed examination of CPU-based interactive ray tracing systems and algorithms. These systems are an important part, and form the basis, of interactive global illumination systems. This is due to the fact that primary rays must be generated at interactive rates if the system is to be interactive as a whole.

3.1.1.1 Systems

Initial work on interactive ray tracing occurred as early as 1995 when distributed multiprocessor machines, referred to as supercomputers, were utilised in an attempt to achieve interactive rates for ray tracing systems. To evaluate the suitability of a custom-built 64 processor machine as an interactive ray tracing system Keates and Hubbard utilised a custom ray tracer employing a regular grid as an acceleration structure [KH95]. Interactive rates of one to five frames per second (fps) were achieved on as few as 32 processors but only by reverting to simple ray casting by disabling secondary rays (such as shadow, reflection and refraction rays) and using progressive rendering. Unlike Keates and Hubbard, Muuss [Muu95] utilised a system that contained 96 processors, along with a commercial ray tracing package that used a nonuniform binary space partitioning (BSP) tree [FKN80]. Interactive and near-interactive, between a half and two frames per second, rates were achieved for a resolution of 720×486 when rendering three distinct spectral bands.

Parker *et al.* presented work that allowed for interactive isosurface rendering using a 128 processor distributed shared-memory machine [PSL*98, PPL*99] and a custom ray tracer. Using brute-force ray tracing along with simple optimisations such as volume bricking, where the dataset is broken up into smaller volumes or bricks for easier transport between nodes, combined with a shallow hi-

erarchy they demonstrated interactive rates, between one and twenty frames per second. The results presented showed a highly scalable system that could render a one gigabyte volume dataset, including shadows, at ten frames per second. This work was then extended into *-Ray by Parker *et al.* where the ability to render objects other than isosurfaces (such as spheres, polygons and spline models) was added [PMS*99]. Frameless rendering [BFMZ94], an approach where the display is no longer presented as a time series of frames but rather as a single frame where different regions are updated over time, was also employed to speed-up the computation and allow for interactive rendering on as few as eight processors.

While the previous approaches utilised custom or high-end shared-memory supercomputers, with costs in the millions of dollars, Wald *et al.* presented a system running on a cluster of four commodity desktop machines [WSBW01]. By carefully exploiting spatial coherence at an object and image level along with the use of SIMD instructions [Int03] and optimised traversal and intersection routines this work presented interactive rates for scenes of up to 8 million triangles. In Wald *et al.* [WSB01] this work was further extended to handle much larger scenes, up to 12.5 million triangles. This was achieved via simple pre-process step in which a scene database is generated and distributed to all the clients, which then fetch and cache BSP voxels as required. This combined with latency hiding and load balancing allowed for interactive rates on a cluster of nine dual-core machines. Another system that presented the visualisation of large datasets was from Demarle [DPH*03]. This system was derived from the earlier mentioned *-Ray architecture [PMS*99] and combined this with techniques from [WSB01] while building a better system for storage of the distributed data. This allowed interactive rendering of a 7.5GB dataset.

With the increase of computational power and the advancement of algorithms new frameworks and implementations were developed that moved away from Whitted-style ray tracing [Whi80] and focused on more complex global effects. Boulos *et al.* focused on the implementation of distributed ray tracing [CPC84] for effects such as soft shadows, depth-of-field, glossy reflections, participating media and motion blur [BEL*06, BEL*07]. The Razor ray tracing rendering architecture [DHW*07] used three basic methods to achieve speedup: multiresolution geometry, dynamic acceleration data structures and decoupling shading from visibility. This was done in order to provide a framework that could compete with high-end rasterisation renderers. Due to the focus on more complex effects the implementations covered above, such as those by Boulos *et al.* and

Djeu *et al.* are covered in greater details in Section 3.2.1 when interactive global illumination is examined.

Bigler *et al.* [BSP06] presented the software architecture of their interactive ray tracer, Manta, and described its application in engineering and scientific visualisation, for an example refer to Figure 3.1.1. Their emphasis was on design considerations and differences between an interactive and batch rendering system with a focus on a high degree of parallelism and flexibility along with specific optimisations such as instruction-level parallelism via SIMD [Int03] and packet-based acceleration structures. This work was also the basis of OptiX a real-time GPU-based ray-tracer which is covered in greater detail in Section 3.2.2. RTfact developed by Georgiev *et al.* focused on a template-based library consisting of packet-centric components combined into an efficient ray tracing framework. Generic design approach with loosely coupled algorithms and data structures allows for easy integration of new algorithms with maximum runtime performance [GS08]. Here, unlike Manta which relied on very low-level optimisations throughout the code-base, RTfact hid this complexity with the use of templates and higher level primitives.

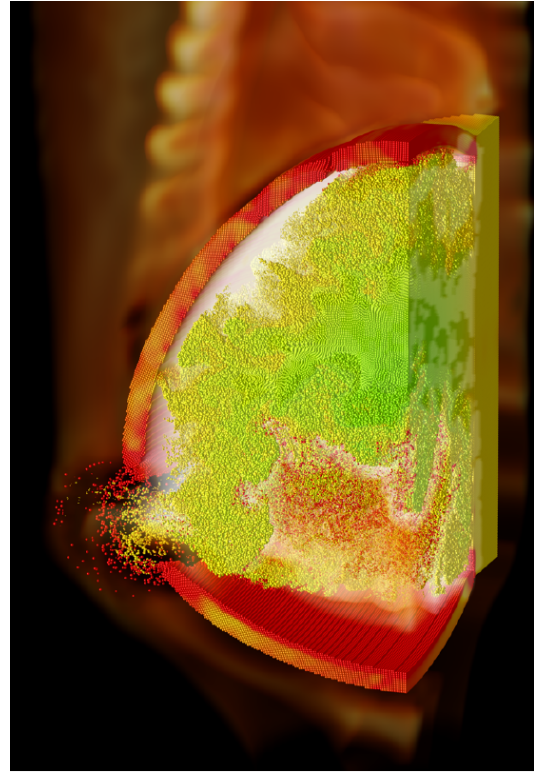


Figure 3.1.1: The Manta interactive ray tracer rendering 2.8 million particles as individual spheres along with the temperature field as a volume, all at 15 to 20fps on a single multi-core desktop PC. *Image courtesy of James Bigler [BSP06].*

3.1.1.2 Algorithmic enhancements

While work was being presented on the implementation of entire systems, algorithmic improvements were also underway. In Dmitriev *et al.* rays were scheduled to be fired as pyramidal shafts [DHS04]. This combined with certain conditions allowed for an entire shaft's visibility to be determined by simply

evaluating the corner rays, using SIMD instructions to further accelerate this. Other systems such as the one presented by Reshetov *et al.* [Res06] used similar concept of combining rays into beams or shafts. Intersecting these beams allowed for individual rays to start traversing the accelerating structure at some node deep inside the tree, saving computation and traversal time. While the previous approaches only worked for coherent groups of rays with shared origins Reshetov presented a new traversal algorithm for incoherent groups of rays. This approach, unlike splitting the rays into coherent subgroups, allowed for rays with completely different directions to be traced together reducing intersections by up to 50%. Other methods for culling entire ray packets using geometric and interval arithmetic were presented by Boulos *et al.* [BWS06]. Approaches for culling ray packets against triangles, axis-aligned bounding boxes (AABB) and spheres using interval arithmetic, corner rays and bounding planes were shown. Reshetov *et al.* presented further work on packet culling by proposing a system where special transient frusta were generated every time a leaf of an acceleration structure was traversed by a packet of rays [Res07]. These frusta contained the intersection of active rays with the leaf node allowing for the elimination of 90% of all potential intersection tests along with a tenfold reduction in the size of the acceleration structure while achieving better overall performance.

3.1.2 GPU Algorithms

GPU-based algorithms share a number of similarities to their CPU-based counterparts but due to the rapidly evolving GPU architecture are much more reliant on specific hardware optimisations and features to achieve interactive results. While borrowing heavily from the CPU-based systems they need to be examined in conjunction with the CPU-based approaches to have a full overview of the field of interactive ray tracing.

3.1.2.1 Systems

One of the first GPU-based ray tracer was called the "Ray Engine" and was presented by Carr *et al.* [CHH02]. This implementation utilised the GPU to perform ray-triangle intersections, by using a fragment program, while setup was performed on the CPU and shading utilised the standard rasterisation pipeline. A "ray cache" was also implemented to make full use of the parallelism of the GPU by batching similar rays so that they would intersect collections of spa-

tially coherent triangles, due to this scenes with highly incoherent ray-trees were problematic. Also the communication between CPU and GPU proved to be a bottleneck as triangle and ray data needed to be converted and sent to the GPU and results read back over the AGP bus, which is limited to 250 MB/sec.

At the same time Purcell *et al.* [PBMH02] had also developed a GPU-based ray tracer, the primary difference being this implementation performed more work on the GPU, specifically eye-ray generation and acceleration structure traversal. The acceleration structure chosen was a grid, which was stored in a 3D texture and traversal was performed using the 3D-DDA [FTI86] with a multi-pass approach. The shading did not use the rasterisation pipeline directly but implemented more complex secondary effects such as shadow casting and 2-bounce path tracing. While this approach presented a more complete GPU-based ray tracing pipeline the approach assumed that the acceleration structure for the geometry was already generated before rendering began, making the method unsuitable for dynamic scenes. Also due to the fact that four different pixel shaders: for ray spawning, ray traversal, ray-triangle intersection and shading were used and that the rays are were different phases, peak GPU performance was limited to 10% [CHCH06]. Further research has extended this work [Chr04, TSr05] but all the attempts suffered from the same drawback and did not exploit the full computational capabilities of the GPU.

Due to the recent developments involving complex data structures, adaptive techniques and complex shading work has once again focused on the creation of systems. OptiX is such a system, based on the Manta ray tracer [BSP06], it is a general purpose ray tracing framework designed to run on state-of-the-art GPUs [PBD*10]. Using a domain-specific just-in-time compiler it generates custom ray tracing kernels by combining user-supplied programs for functions such as ray generation, material shading and scene traversal. These kernels are highly optimised at compile time allowing for interactive rates even when complex shading models are used. The drawbacks of this framework are that the functionality is limited to NVIDIA GPUs, has a relatively fixed-stage pipeline and in-built acceleration structure and the code must be written in CUDA [GGN*08].

3.1.2.2 Acceleration Data Structures

Up until early 2004 all GPU-based ray tracers had relied on a uniform grid as an acceleration structure. While easy to generate recent work had shown

that other structures, such as kd-trees and BVHs, had numerous advantages in a number of different scenarios [Hav01, WKB*02]. Ernst *et al.* [EVG04] were the first to implement a kd-tree traversal on the GPU, their implementations main limitation was that it required a fixed maximum stack depth. Foley and Sugerman [FS05] extended this work and demonstrated two new approaches for stack-less traversal of kd-trees on the GPU. Their results showed that for scenes with objects at different scales the kd-tree approach was up to 8 times faster than a uniform grid but still an order of magnitude slower than the best CPU approaches. They identified load balancing and data recirculation as the core issues that were behind this disparity in performance. Horn *et al.* [HSHH07] further extended this approach to use a single pass by using GPU branching and looping. Further work on stack-less kd-tree traversal was demonstrated by Popov *et al.* in which they eliminated stack usage entirely and reduced the number of traversal steps required [PGSS07]. This work, while only benefiting CPUs moderately, improved GPU performance greatly allowing for over 16 million rays per second with moderately complex scenes involving secondary rays and complex shading.

Early work on BVH traversal and construction was performed by Thrane and Simonsen [TSr05] and Carr *et al.* [CHCH06]. In Thrane *et al.* [TSr05] the algorithms suffered from the same drawbacks experienced in Purcell *et al.* [PBMH02], as they used the same approaches as they did for the kd-tree traversal with the hierarchy still being constructed on the CPU. Carr *et al.* [CHCH06] constructed and traversed their hierarchy on GPU but it required a large amount of off-line pre-processing of the meshes. To optimise the traversal for large scenes [GPSS07] demonstrated a parallel packet-based algorithm using a shared stack that allowed for the ray tracing of a 12.7 million triangle scene on the GPU at interactive rates, including the generation of shadows and shading. The traversal was CPU-based but accurately approximated the surface area heuristic using streamed binning while still being one order of magnitude faster than previously published results. At the same time Roger *et al.* [RAH07] presented an acceleration structure that fully supported dynamic animated scenes. It combined concepts from beam tracing [HH84] and the hierarchical approaches such as the ones used in Arvo and Kirk [AK87] and Ghazanfarpour and Hasenfratz [GH98] using a ray-space hierarchy combined with breadth-first ray tracing [NO97].

More modern approaches have tackled the efficiency of construction as well as traversal of data structures on the GPU. In Lauterbach *et al.* [LGS*09] two

novel parallel algorithms were presented for the construction of a BVH and then combined into a hybrid approach to remove the already identified bottlenecks experienced by other approaches. This fast construction allowed for re-building of the BVH every frame and therefore supported fully dynamic scenes at interactive rates. Similarly in Zhou *et al.* [ZHWG08] a kd-tree construction algorithm was presented that built in entire tree in breadth-first order with a scheme to evaluate node split costs. To further exploit the fine-grained parallelism of the GPU at the upper tree levels, where the nodes were large, the algorithm parallelised the computation over all geometric primitives instead of nodes at that level. This approach was competitive with multi-core CPU algorithms and was fast enough to rebuild a full kd-tree each frame. For traversal an in-depth analysis was performed by Aila *et al.* [AL09] analysing structure traversal and primitive intersections methods on the GPUs. Comparing current methods to their theoretical upper-bounds, using a simulator to attain these, they showed that most current methods were $1.5 - 2.5\times$ slower and that these inefficiencies were due to previously unidentified work distribution problems. Their proposed solution fixed these issues and provided results for both coherent primary and incoherent secondary rays. In Laine [Lai10] the concept of a restart trail was introduced, a simple algorithmic method that made restarts, traversals from the root of the tree so that the already processed part of the tree is not entered again, possible regardless of the type of hierarchy. This was achieved by storing only one bit of data per level thus enabling stackless and short stack traversal for BVHs.

3.2 Interactive Global Illumination

As has been shown, a lot of progress has been made in interactive ray tracing. Yet, most of these efforts have focused on simple ray tracing generally computing only visibility, limited specular reflections, basic shading and shadow rays from point light sources. The majority of these algorithms and systems managed to obtain real time performance by exploiting the natural coherence of primary rays. While these rays are naturally coherent, the same cannot be said for secondary rays. The efficient computation of secondary rays is required if interactive global illumination is to be achieved. Global illumination includes effects such as diffuse interreflections, motion blur, depth-of-field, soft shadows, participating media and glossy reflections. All these effects are dependant heavily on secondary

rays, which are highly incoherent. This lack of coherency means that, unlike the algorithms in Sections 3.1.1 and 3.1.2, different methods must be developed to accelerate the computation of these rays.

3.2.1 CPU Algorithms

This section reviews the systems and methods used in accelerating the computation of global illumination to interactive rates. The systems, presented in chronological order, are examined to determine the combinations of approaches and methods they utilise to achieve interactivity. The individual algorithms and methods are examined in detail and grouped by the higher-level approaches they share.

3.2.1.1 Systems

Arguably, the first CPU-based interactive system to attempt to exploit coherence in secondary rays was the Instant Global Illumination (IGI) system [WKB*02] based on instant radiosity [Kel97]. For each frame a pre-processing phase generated a number of virtual point light sources (VPLs) across the scene using random walks. At the rendering stage, shadow rays would be shot to the light sources for computing direct lighting and to the VPLs for computing irradiance. In order to achieve interactive frame rates, instead of using the same set of VPLs for each pixel, nine sets were computed individually using quasi-Monte Carlo methods and every pixel in a 3×3 grid computed the irradiance from a different set. This method improved performance at the cost of structured noise in the image which was mostly removed by a filtering stage using a discontinuity buffer [Kel98]. Ray tracing was used for the computation of specular effects and a photon map was used for computing caustics. In Benthin *et al.* [BWS03] they improved the system by tracing the shadow rays to the VPLs using SIMD calculations, computing the shading with SIMD and anti-aliasing by supersampling the number of rays per pixel for the visibility and shading, while just using a single subset of VPLs for that pixel.

The anti-aliasing was performed by subdividing the subset of VPLs, N , for that specific pixels into a number of small subsets N/M where M was the number of primary rays used to supersample that pixel. Each primary ray then used the smaller subset of VPLs ensuring that overall the number of VPLs utilised remained constant. In Wald *et al.* [WBS03] the system was further extended to

account for complex scenes with large numbers of light sources. For such scenes, if interactive rates were to be achieved, care had to be taken when choosing the light sources to shoot the VPLs from. Their system used an initial sparsely sampled path tracing pass with direct lighting to all light sources to identify the importance of the light sources for the given frame. The importance was transformed into a PDF and combined with PDFs from previous frames, to improve temporal coherence, and was used to select which light to emit VPLs from. Their system showed interactive results for scenes with large number of light sources, in the order of thousands, while only utilising a small fraction of the light sources at any given time. The drawback of the approach was that when complex scenes were used pixels, within the interleaved pattern, can potentially receive no contribution from their particular subset of VPLs. This resulted in temporal noise while moving where previously completely unlit pixels were suddenly lit.

Boulos *et al.* [BEL*07] presented an overview of rendering using packets for Whitted-style ray tracing [Whi80] and distribution ray tracing [CPC84] paying particular attention to secondary rays. Whitted-style ray tracing (WRT) allows for reflections and refractions to be computed. Distribution ray tracing allows for the computation of complex non-singular effects such as depth of field, glossy reflections, motion blur and soft shadows. For primary rays they used a BVH system [WBS07]. They demonstrated how Whitted-style ray tracing was currently achievable by grouping secondary rays by type of ray (*e.g.* shadow, reflected and refracted) and coherency. For distribution ray tracing the amount of rays that needed to be shot was prohibitive to achieve real-time rates for current hardware. Using interleaved quasi-Monte Carlo sampling methods similar to those in Kollig and Keller [KK02] they demonstrated how they could achieve reasonable performance even for distribution ray tracing [BEL*06]. While focusing on important secondary effects they ignored diffuse interreflections, the most incoherent of these effects, due to the fact that their system still relied on the exploitation of coherence to maintain interactive rates.

For the Razor ray tracing rendering architecture [DHW*07], the authors suggested a number of contributions to ray tracing which would enable it to compete with high-end rasterisation renderers. Their system used three basic methods to achieve speedup: multiresolution geometry, dynamic acceleration data structures and decoupling shading from visibility. Using a method similar to that proposed by Christensen [CLF*03], used for caching in off-line rendering, the authors recommended using multiresolution geometry for computations whereby secondary

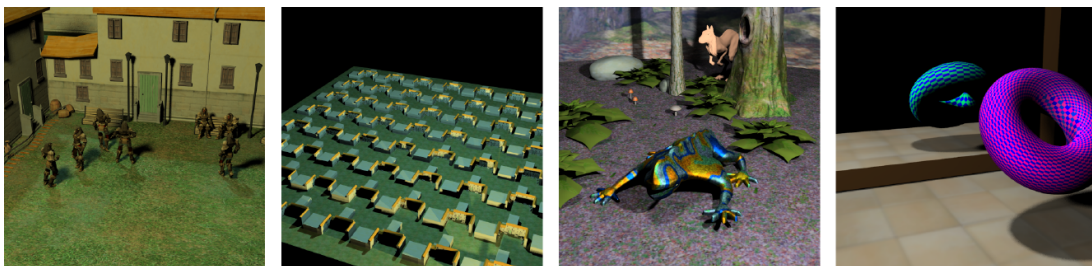


Figure 3.2.1: A number of scenes rendered by the Razor system [DHW*07] at near interactive rates. *Images courtesy of Peter Djeu [DHW*07].*

non-coherent rays intersect coarser versions of the geometry. This suggestion was based on the observation that most of the secondary rays have large ray differentials [Ige99] and therefore would probably intersect geometry which is non-coherent. Computing divergent rays on coarser geometry would therefore increase the memory access for visibility and shading. When rendering with LOD, a phenomena called tunnelling may occur when a ray is about to intersect an object but the level of detail of that object changes such that the ray has already passed (or tunneled through it), effectively missing the object which was meant to be hit. In order to avoid the problem of tunnelling the authors proposed intersecting the ray with geometry interpolated from two discrete levels. The multiresolution geometry was supported by a multi-scale kd-tree which maintained the geometry at multiple levels. Due to the complexities of building kd-trees for multiple geometric levels the kd-tree was evaluated lazily. The authors further suggested decoupling visibility from shading. The shading was evaluated lazily, the first stage only evaluated the view independent calculations and did so at the vertices, caching the results. The rest of the shading computation was performed in traditional ray tracing fashion. Results, as can be seen in Figure 3.2.1, demonstrated performance close to that of the coherent grid [WIK*06] for rays traced per second. While these systems integrated a number of methods it is also important to examine different classes of algorithms more closely.

3.2.1.2 Radiosity

Radiosity is a classical global illumination solution, see Section 2.2.3. It was one of the first global illumination techniques to be made interactive, due to the development of a progressive refinement solution [CCWG88]. Two similar approaches, Chen [Che90] and Drettakis and Sillion [DS97] made use of the progressive re-

finement to allow for interactive updates of the radiosity solution, but this was limited to simple scenes without much geometric complexity. In Chen [Che90] a view-independent and incremental update was proposed where a change to the scene such as moving a light, changing a surface property or moving an object would result in a minimal update, exploiting the coherency that remained within the system. Drettakis and Sillion [DS97] utilised a similar approach but generated a line-space hierarchy, a visualisation of which can be seen in Figure 3.2.2. Traversing this data structure then allowed for easy identification of the links that were affected by a dynamic change to the scene, and thus allowed the radiosity solution to be quickly updated. Both approaches suffered from tessellation artifacts, a common radiosity issue and did not scale well to complex scenes with large amounts of geometry. Further development of interactive radiosity solution moved over to using graphics hardware which is detailed in Section 3.2.2.

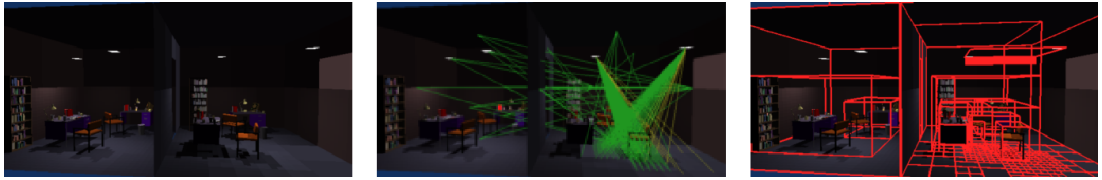


Figure 3.2.2: On the left is an example of a progressive radiosity solution from Drettakis *et al.* [DS97]. In the middle image a chair has been moved, with the green lines indicating links that have been affected, while the image on the right highlights the parts of the hierarchy that require updating. *Images courtesy of George Drettakis [DS97].*

3.2.1.3 Sparse sampling

Since the computational complexity of ray tracing is a function of the number of primary rays traced, as can be seen above, it is not surprising that there have been a number of rendering systems that have attempted to minimise the number of rays shot by sparsely sampling the image plane and then either interpolating or using caching mechanism to retrieve previously computed information, an example can be seen in Figure 3.2.3. The first such system was proposed by Ward and Simmons [WS99] and was called "The Holodeck". For this system the entire scene was subdivided into a three-dimensional regular grid. In each cell a number of rays that had been previously generated, along with their radiance contributions, were cached and later re-used. This light-field like approach enabled the ray tracing computation to be interactive and progressive while remaining view

independent. Decoupling the sampling from the display method also allowed the system to make effective use of graphics hardware when using the cache samples for re-rendering. Unlike progressive radiosity solutions the system was not limited to simple diffuse-only shading models and while light-field like, it did not need to compute the entire holodeck before visualisation could begin.



Figure 3.2.3: The image on the left is an example of a precalculated holodeck simulation, while the image on the right is the same simulation after 30 seconds of computation on 21 processors. *Images courtesy of Greg Ward [WS99].*

Like the Holodeck, the render cache [WDP99, WDG02] was one of the first sparse sampling rendering systems. This rendering calculated the illumination for a number of samples which it then stored in the render cache. The sampling calculation and illumination ran asynchronously to a viewing process, which re-projected cached samples onto the display. An example of the process can be seen in Figure 3.2.4. This enabled the system to maintain interactive frame rates. Unfortunately, the sample re-projection resulted in artifacts such as tearing. Bala *et al.* [BWG03] extended the render cache with an edge and point image (EPI) which ensured that interpolation did not occur over silhouette and shadow discontinuities. The EPI was generated via projection of shadows and geometry onto an image. GPU versions of the render cache and the EPI were presented in Velazques *et al.* [VALBW06]. These improvements do not eliminate all the geometric artifacts. In particular, geometric edges had to be reconstructed using a very large number of point samples and still suffered from distortions when the camera moved.

Stamminger *et al.*'s rendering system [SHSS00] was clearly inspired by the render cache. Similarly to the render cache, the viewing process was decoupled from the ray tracing computation. Rather than re-projecting all the individ-

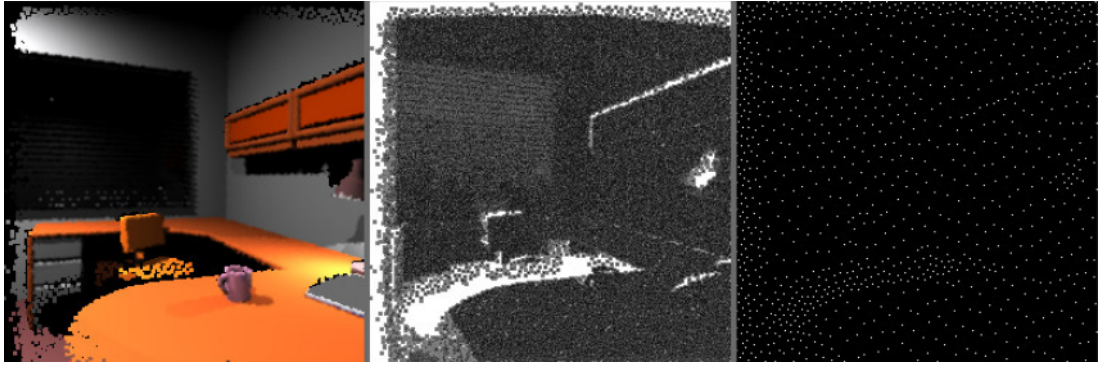


Figure 3.2.4: On the left is an example of an image produced using the render cache. In the middle is the corresponding priority image used while the dithered binary image on the right specifies which sample locations will be requested next from the renderer. *Images courtesy of Bruce Walter [WDP99].*

ual pixels in the render cache, rendering was performed using rasterisation and certain objects maintained data structures called corrective textures from which point samples were traced asynchronously using ray tracing and updated whenever possible. This system partially removed the artifacts associated with the render cache. Haber *et al.* [HMY01] extended this method further using a technique called corrective splatting, whereby non-diffuse objects were ray traced based on the computation of a saliency map [IKN98] which predicted which of the non-diffuse objects is being attended to, see Figure 3.2.5. While not suffering from geometric distortions like Walter *et al.* [WDP99] the use of camera-local projective textures introduced reprojection artifacts in the shading. Also, object-local textures must have been of an extremely high resolution in order to reconstruct sharp shading features such as hard shadow boundaries. While both Walter *et al.* [WDP99] and Stamminger *et al.* [SHSS00] were designed to facilitate global illumination walkthroughs, extending these systems to handle dynamic scenes would be non-trivial.

The shading cache [TPWG02] extended the concept of the render cache but rather than compute reconstruction in image space, they used an object space data structure based on the vertices of meshes on the rendered geometry for computing samples from and eventually storing samples to. They used a method similar to hierarchical radiosity [HSA91] for adaptively computing the mesh for improving the quality of the sampling based on a priority map. They used an asynchronous frameworks similar to that in the render cache. Their structures were composed of geometrical primitives and they used rasterisation for the view-



Figure 3.2.5: Example of a solution produced by Perceptually Guided Corrective Splatting [HMYS01]. On the left is an input image, consisting of a pre-processed view-independent global illumination solution. The middle image shows a saliency map which is used in the image on the right to produce fully converged solution after corrective splatting has been applied. *Images courtesy of Joerg Haber [HMYS01].*

ing thus leveraging the computational power of graphics hardware.

Wolfe *et al.* [WLWD03] presented a rendering concept for ray tracing and rasterisation using level-of-detail they termed interruptible rendering. The interest is in their ray tracing method. Their system used two error metrics a spatial and a temporal error. The rays sampled areas of the image plane using a quadtree for progressive rendering similar in concept to the ray tracing in Painter and Sloan [PS89]. The spatial error was defined by the size of the quadtree node that was being sampled. The temporal error was described by the amount the object moves. Rendering was computed and the results were written to the back framebuffer. Results were swapped to the front framebuffer when a specific condition was satisfied. The condition was that the combined spatial and temporal error for the back framebuffer had to be less than that of the front framebuffer and that the spatial error in the back framebuffer was exceeded by the temporal error. The advantages of such a system are that when rendering interactively with large dynamic movements coarse images were produced at high frame rates but when there was little motion rendering produced lower frame rates with more spatial detail in each frame.

Dayal *et al.* [DWWL05] presented an adaptive frameless rendering framework based on the concept of frameless rendering, previously introduced in Bishop *et al.* [BFMZ94], which rendered without the concept of double buffering by constantly updating the framebuffer. Their rendering method cached samples both spatially and temporally in a kd-tree representing the image plane and used

a spatiotemporal Gaussian for filtering and reconstructed images on the GPU through a splatting technique. Similar to the work of interruptible rendering, a fine balance was drawn between high temporal detail and low resolution and low temporal detail and high spatial detail. They showed interactive results with few artifacts for non-complex ray traced scenes with few artifacts when compared to high spatial detail only and high temporal detail only.

3.2.2 GPU Algorithms

This section, like Section 3.1.2, examines GPU-based approaches to provide a comprehensive overview of the field of interactive global illumination. The individual algorithms are examined in detail and are grouped by the higher-level methods they have in common.

3.2.2.1 Radiosity

Much research has occurred in the last few years to fully exploit the parallelism offered by high-end GPUs to accelerate global illumination. The initial focus was on radiosity [GTGB84] and finite element methods. Initial attempts to integrate graphics hardware into the computational pipeline where hybrid approaches where the GPU was utilised to accelerate very specific aspects of the pipeline.

The hemi-cube approach [CG85] proposed an adaptive approach that allowed for the visualisation of the solution as it was converging. The graphics hardware was utilised for image composition to allow for interactive an walk-through. The solution from Sillion and Puech [SP89] also used the hemi-cube approach but utilised the graphics hardware to also assist in approximating planar mirrors, to add some specular effects to the solution.

Unlike the previous hybrid methods both Carr *et al.* [CHH03] and Coombe *et al.* [CHL04] proposed GPU-based radiosity solutions that utilised the GPU to accelerate the computation of the radiosity solution itself. In Carr *et al.* [CHH03] the GPU is used to find a solution to the radiosity matrix, initially calculated on the CPU, using its massively parallel pipeline as a general processing unit. In Coombe *et al.* [CHL04] the GPU is used to accelerate visibility by rendering the scene for each polygon, using stereograph projection instead of hemi-cube projection. A quad-tree hierarchy is also used for the textures to allow for adaptive subdivision. This method is restricted to scenes with low complexity, 10 000 or fewer elements, and the use of textures to store the radiosity data also has a high

memory cost. In *Nijasure et al.* [NPG03,NPG04] the incoming radiance function at a number of fixed locations, based on a regular grid, is sampled and projected into the spherical harmonics basis. Then the incoming radiance at any surface point is estimated by interpolating the incoming radiance at nearby sample locations. The main drawback of this method is the choice of sample points as they are placed on regular grid inside the volume of the scene, therefore not adapting to the lighting complexity.

If visibility is neglected when computing indirect illumination then real-time global illumination is achievable as can be seen in Dachsbacher *et al.* [DS05,DS06]. Dachsbacher *et al.* [DS05] presented the concept of "reflective shadow maps" (RSMs) where rendering of a shadow map also involved storing not just the depth but the normals, world position and flux. This approach was a combination of methods Dachsbacher and Stamminger [DS03] and Tabellion and Lamorlette [TL04] which enabled each element in the shadow map to be considered as an indirect light source to allow for one-bounce indirect illumination in the scene. This work was further extended in Dachsbacher *et al.* [DS06] where these indirect elements are converted to point light sources which are then importance sampled, their shape adjusted based on the glossiness of the surface and then splatted directly to the screen when using a deferred shading approach. Multiple bounces using this approach are inefficient as many new shadow maps must be generated for each bounce.

In cases where visibility was considered the scene complexity was usually limited to a few thousand polygons in order to maintain interactivity [Bun05, DKTS07, DSDD07, RGKS08]. Bunnell [Bun05] introduced a hierarchical link structure where visibility was approximated by ambient occlusion. Similarly [DKTS07] also proposed an implicit visibility solution by constructing a hierarchical link structure between surface elements. This method relied on a certain amount of pre-computation on the CPU, only showing result for single-bounce indirect illumination and like other radiosity solutions was limited to very simple scenes to minimise the number of surface elements. For Dachsbacher *et al.* [DSDD07] the need for explicit visibility computation was removed by reformulation of the rendering equation, for results see Figure 3.2.6. With implicit visibility both radiance and anti-radiance were propagated for each element to compensate for light transmitted erroneously. While this method allowed for dynamic objects and lights it still suffered from general radiosity problems, such as discretization which necessitated filtering. The algorithm was also not entirely

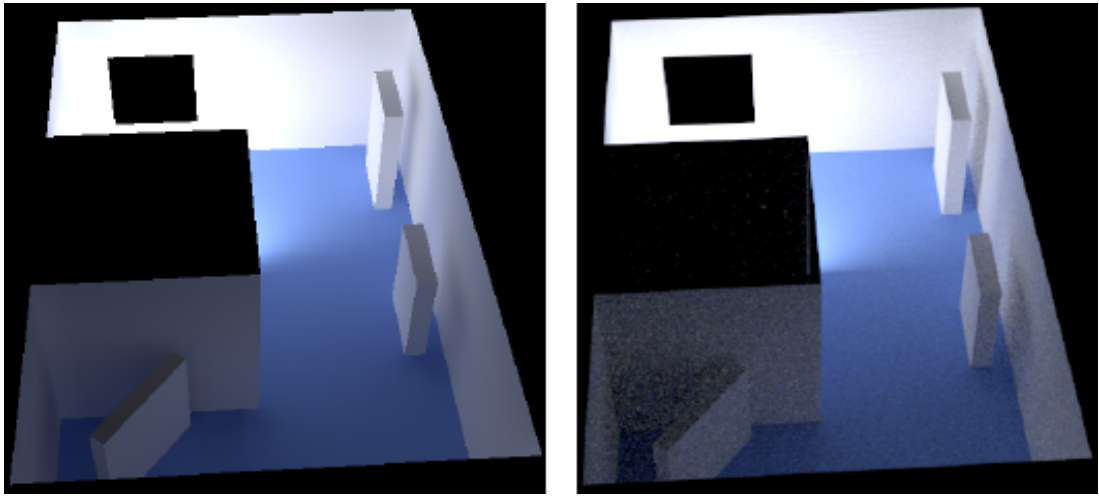


Figure 3.2.6: On the left an example of a solution produced by the radiosity algorithm introduced by Dachsbacher *et al.* [DSDD07], with a path-tracing reference solution on the right. The scene is an entirely indirectly lit, with a directional light entering via a window at the top of the scene and all shadows being indirect. Images courtesy of Carsten Dachsbacher [DSDD07].

GPU-based, requiring CPU setup and the use of dynamic objects was restricted due to the need for highly tessellated surfaces near indirect shadows. Finally in Ritschel *et al.* [RGKS08] a large number coherent surface shadow maps (CC-SMs) were computed and compressed, via a novel scheme, as a means to pre-compute visibility. This visibility approach was then used with a combination of the lightcuts algorithms [WFA*05] and hierarchical radiosity for an n-bounce global illumination solution. The approach also allowed for a final glossy bounce, high frequency effects and general BRDFs but required rigid geometry, therefore no deformations, and due to the pre-computation of visibility was very memory heavy.

3.2.2.2 Instant Radiosity

Unlike finite element methods a number of methods exist that are not directly dependent on scene complexity. A group of these use the concept of virtual point lights (VPLs) first introduced in "Instant Radiosity" [Kel97]. In this approach a small set of virtual point lights (VPLs) is computed using a quasi random walk from the light sources in the scene. During the walk, originating at the light source, if a diffuse surface is hit a VPL is deposited until some termination criteria is met. These VPLs are then used to illuminate the scene in combination with a

shadowing algorithm. For each VPL the scene is re-rendered lit by that specific light and all images are averaged together using the accumulation buffer. This allows most of the computation, other than the VPL generation, to be performed on the GPU. It also avoids many of the typical tessellation artifacts seen in radiosity-based methods, instead of discretizing the geometry, instant radiosity performs discretization by using only a small number of discrete virtual point light positions. Specular effects are achievable, by using spotlights to simulate glossy reflections, but this increases the computational and memory cost as more information about the light needs to be stored. Also dynamic objects are not considered at all. The reliance on graphics hardware to re-draw the scene many times also limits the scene complexity somewhat. This limitation is addressed in “Instant Global Illumination” [WKB*02] by using deferred shading [DWS*88] combined with interleaved sampling [KH01], this is covered in greater detail in Section 3.2.1. The approach is also optimised in Segovia *et al.* [SIMP06b] where the GPU version intelligently divides the image in such a way as to increase coherency and maximise bandwidth usage when using interleaved sampling but unlike the original version [Kel97] no occlusion checking is performed for the VPLs.

In Laine *et al.* [LSK*07] real-time rendering of static scenes, with one-bounce illumination is achieved, by using a variant of instant radiosity where VPLs and their respective shadow maps are re-used over multiple frames to lower computation costs, primarily those incurred when rendering the shadow maps. VPLs are also more carefully generated to ensure a good distribution and like Segovia *et al.* [SIMP06b] deferred rendering is used to speed-up light accumulation.

A slightly different approach is proposed in Ki and Oh [KO08] where similar VPLs are clustered together in a view independent manner to generate a hierarchical light representation, termed an “image pyramid” which is then used to light the scene. All tree generation and traversal occurs on the GPU and like Laine *et al.* [LSK*07] visibility is not calculated for the VPLs.

For Ritschel *et al.* [RGK*08] a crude hierarchical point representation of the scene is used to generate “imperfect shadow maps” (ISMs) for each of the VPL, any holes present are filled via a pull-push pass. These ISMs are much faster to compute due to their approximate nature and lower resolution. These maps are extended into “imperfect reflective shadow maps” (IRSMs) that then allow multiple bounces of light to be computed. While not entirely limited by scene complexity the point representation does increase in density as the scene gets

more complex which is a limiting factor. The limited resolution also means that small objects do not appear in the shadow maps. Finally a number of parameters need to be set that are scene dependant, meaning the method is not fully automatic.

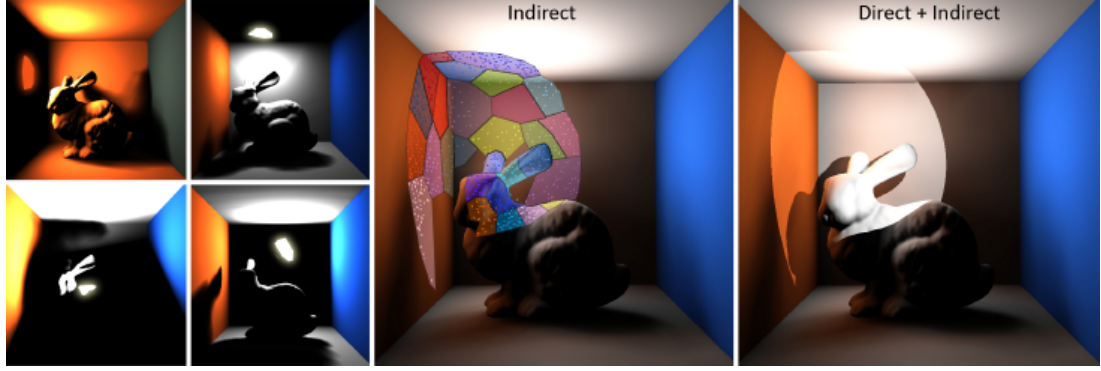


Figure 3.2.7: The images on the left show individual soft shadows from selected VALs. The complete clustering, for 30 VALs, is shown in the centre image while the full global illumination solution is shown on the right. *Images courtesy of Zhao Dong [DGR*09].*

Dong *et al.* [DGR*09] presents an approach where VPLs, generated using RSMs, are grouped into virtual area lights (VALs) via k-means clustering. During the computation of indirect diffuse contribution interleaved sampling is utilised, following a similar approach to Laine *et al.* [LSK*07]. Temporal coherence is maintained by restarting the k-means clustering each frame from an identical, initial cluster assignment as the VPL positions are coherent frame to frame. Each VAL contains N/M VPLs, where M is the number of VALs and N is the total VPLs generated. For the k-means clustering utilised, similar to Wang *et al.* [WZPB09] in Section 3.2.2.4, both euclidean distance and normals are taken into account to generate planar clusters of VPLs. Visibility is then calculated not for each VPL but for each VAL, providing a fractional visibility result, using convolution soft shadow maps (CSSMs) [ADM*08] with a parabolic projection. When shading is performed visibility is calculated based on the M clusters but all N contributions from the VPLs are utilised. For a visualisation of individual VAL contributions, the VAL distribution and a final combined rendering refer to Figure 3.2.7. The parabolic CSSMs inherit problems from CSSMs such as issues with contact shadows and MIP discretisation. For very large senders ringing artifacts can appear and penumbra regions are curved when viewed from a grazing angle. Due to the number of VALs needed the CSSMs are low-resolution which

means that very thin shadows cannot be resolved correctly. Due to the parabolic projection being non-linear, the squared filter region used to approximate the projection of the light source is no longer correct, resulting in artifacts. These artifacts mentioned for parabolic CSSMs are masked to a certain extent due to the overlap of the indirect shadows in the final image. Finally the approach is limited to single-bounce global illumination and is also dependent on scene complexity as each VAL must render all the scene geometry.

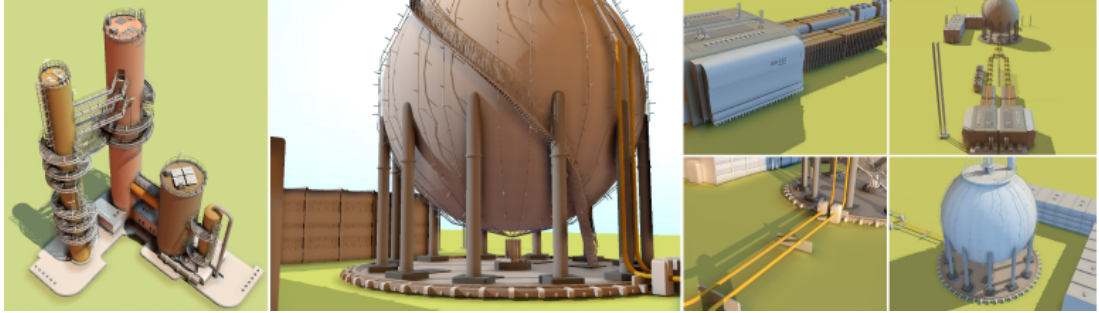


Figure 3.2.8: Screens-space directional occlusion (SSDO) along with one additional diffuse indirect bounce of light. The scene contains 537k polygons and runs at 20.4 fps at a resolution of 1600×1200 pixels with fully dynamic geometry and lighting. *Images courtesy of Tobias Ritschel [RGS09].*

3.2.2.3 Image-based methods

Sloan *et al.* [SGNS07] proposed a solution that demonstrated real-time indirect illumination and soft shadowing but only for low frequency distant lighting. The proposed solution calculated the lighting by replacing the scene geometry with spherical proxies which were then splatted into low resolution buffers (typically a half or quarter screen resolution) which stored both proxy information and low-order spherical harmonics. Once all proxies were added the results were combined with the lighting information and upsampled using joint bilateral up-sampling [KCLU07] to attain the final result. For Nichols and Wyman [NW09] a reflective shadow map was generated [DS05] and each element in the map was then splatted to the screen. The splatting used a multiresolution approach where discontinuities were detected, using min-max mipmaps generated for the depth and normals. Starting at the lowest resolution mip data is written into the relevant level of the hierarchy when no discontinuities are found or the highest level is reached. This multiresolution buffer is then upsampled in a such as way as to avoid haloring and ringing in the final result. This method is highly dependent on

the number of VPLs as each one needs to be splatted into the buffer and visual complexity, as an increase in discontinuities in the scene will require a high level of refinement. Finally in Ritschel *et al.* [RGS09] the global illumination is calculated entirely in image-space, utilising the concepts introduced in screen-space ambient occlusion (SSAO) [Mit07] and extending them to include directional shadows and one bounce diffuse interreflections, see Figure 3.2.8. The method is limited by information present in the depth buffer and overlaid objects are not treated correctly unless a method such as a depth-peeling is utilised [Eve01].

3.2.2.4 Photon Mapping

A further group of methods that have been optimised for an interactive context are based on photon mapping [Jen01]. Both Ma *et al.* [MM02] and Purcell *et al.* [PDC*03] utilised graphics hardware to accelerate the photon mapping process. This was achieved by breadth-first photon tracing to create the photon distributions and then storing them in an acceleration structure entirely generated on the GPU. For Ma *et al.* [MM02] a hash table was used and while effective, its generation and lookup were very costly. In Purcell *et al.* [PDC*03] a regular grid was utilised and two algorithms for creation of this grid were presented, a multi-pass and a single pass approach. Both approaches suffered from the same drawback: the GPU architecture was not designed, at the time, to handle complex data structures such as trees therefore necessitating the use of simpler structures. This in turn results in the nearest neighbour queries being simplified to meet these constraints resulting in both quality and performance impacts. A different approach was proposed by Larsen and Christensen [LC04] where the photon map was generated on the CPU and "approximate illumination maps" were constructed for each surface. The GPU was then used to perform final gathering and caustics filtering. More recent methods have utilised a newer generation of GPUs to create complex structures and make use of adaptive sampling [FD09, ML09, WZPB09, YWC*10].

In Fabianowski and Dingliana [FD09] photon differentials were created for every hit allowing for density estimation without nearest neighbour searching. The method supported a dynamic camera and lights and was updated every frame but suffered from temporal issues and was limited to two bounces to maintain interactivity. McGuire and Luebke [ML09] made use of an image-space method for the initial and final photon bounces, which was mapped to the GPU, while the

remainder of the photon tracing was performed traditionally on the CPU. This method was used for both caustics and diffuse interreflections and while fast was limited to point-light emitters and a pinhole camera. A further implementation was presented in [PBD*10] using the OptiX framework.

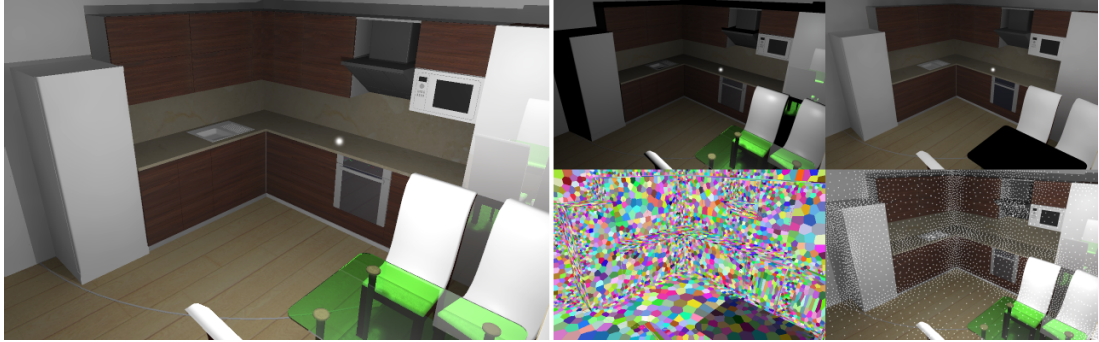


Figure 3.2.9: The left image shows the global illumination result of a kitchen scene with the four sub-images on the right showing: direct lighting and caustics, indirect lighting, clusters of the shading points, and the distribution of cluster centers. *Images courtesy of Rui Wang [WZPB09].*

Wang *et al.* [WZPB09] presented an approach that allowed for multi-bounce indirect lighting, glossy reflections, caustics and arbitrary specular paths. Their approach generated the photon map on the GPU and then simplified it based on seeding and k-means clustering to create a photon tree, stored as a kd-tree. This tree was then treated as a compact illumination cut, similar to the lightcuts approach [WFA*05], when determining illumination. Caching and interpolation, similar to that of the irradiance cache [WRC88], and adaptive sampling were utilised to enable interactivity while exploiting spatial coherence. See Figure 3.2.9 for a visualisation of the k-means clustering and sample distribution along with a final combined image and the individual components. Due to the use of a limited number of irradiance samples small geometric details were lost when computing indirect lighting. The use of low-order spherical harmonics also limited the final gather step to low-frequency glossy materials as highly glossy materials would have required a much longer illumination cut, impacting negatively on the search speed in the kd-tree. Furthermore, even with the attempts at maintaining temporal coherence within clusters the approach suffered from temporal artifacts as the irradiance samples are all recalculated each frame.

Finally in Yao *et al.* [YWC*10] an approach was presented that addressed current issues with image-based photon mapping techniques, primarily the lack of information for occluded geometry and light leaking due to the use of a single

image. This approach generated multiple environment attempting to maximise coverage of the scene. Once the photon distribution was computed the photons were then splatted to the screen with a variable splat sizes allowing for glossy BRDFs. Caustics were computed in a similar fashion to Wyman [Wym08] using a separate texture and specular reflections were not supported. The solution suffered from bias, caused by the photon mapping and upsampling, resulting in light blurring, boundary bias, light leakage and overly dark areas for some of the scenes. Finally, the solution was also limited by the size of the scene and scene complexity, to an extent, as a larger and more complex scenes required a larger number of environments maps which increased overall memory usage.

3.2.2.5 Precomputed Radiance Transfer

A completely different approach that involved a large amount of pre-computation but allowed for real-time global illumination was introduced by Sloan *et al.* [SKS02] as Precomputed Radiance Transfer (PRT), a global illumination solution specifically tailored to the GPU. It used off-line precomputation to calculate the radiance transfer between the surfaces of an object, this information was represented either via spherical harmonics [SKS02, SHHS03] or wavelets [LSSS04, WTL06] and then stored. While this approach only supported static non-deforming meshes and low-frequency distant lighting it allowed for real-time changes in the lighting and generated effects such as soft shadows, diffuse interreflections and caustics. Recent extensions to PRT [PWL*07, IDYN07] based on initial work by Zhou *et al.* [ZHL*05], which precomputed the radiance field for individual objects, allowed for dynamic moving object as long as the objects remained rigid and did not deform. Support was also extended to allow for high frequency local lights, at a cost, where low frequency lighting was computed in real-time and high-frequency lighting at interactive rates.

3.2.2.6 Rasterisation

While most of the approaches discussed above attempted to fit methods, that rely on ray tracing and were initially designed for the CPU, onto the GPU there exist methods such as Ritschel *et al.* [REG*09] which make full use of the GPU and its rasterisation capabilities. This approach utilised previous work done by the author [RGK*08] by using a point-based hierarchy for the scene representation. In the method a number of final gathering locations were chosen and "micro-

rendering” was performed by traversing and rasterising the point-hierarchy into a micro-buffer, which is also appropriately importance-warped based on BRDF importance sampling, see Figure 3.2.10. The final radiance values for each point were then calculated and stored in image space and upsampled using bilateral up-sampling [KCLU07]. While arbitrary BRDF types were supported glossy BRDFs did require a higher density of final gathering points, which the regular sample distribution was not suited for. Noise was also introduced to eliminate banding and was visible. More than two specular bounces was not supported, nor were transparent or refractive objects.

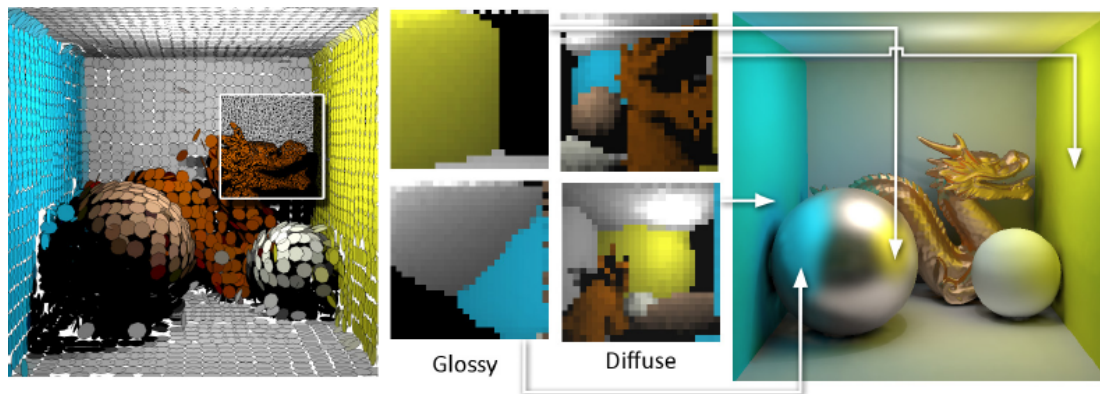


Figure 3.2.10: The image on the left shows two interior levels of the point hierarchy, with over one million points, used in rendering the image on the right. The middle images visualise the micro-buffers, for both glossy and diffuse BRDFs, generated as a result of the rendering process. *Images courtesy of Tobias Ritschel [REG*09].*

3.3 Discussion

As was discussed in Section 3.2.1, to achieve interactive rates it has been important for ray tracing systems to exploit coherency on all levels in the computational pipeline, making use of both instruction level and memory level coherency. How this coherency is exploited needs to be examined in further detail to understand its effects.

This first such mechanism to exploit coherency is packetisation and it does this by using SIMD computation along with coherent memory access for groups of rays. Packet tracing is a technique that was initially proposed in Wald *et al.* [WSBW01] where intersection, traversal and shading are all executed in parallel

on “ray packets” which are groups of coherent rays. Wald *et al.* initially proposed this technique for triangular scenes and kd-trees but it has been extended to a host of different acceleration structures and primitive types as well as being mapped to a variety of hardware architectures. Initial packetisation of rays was limited to packet sizes that were equal to the SIMD width of the host processors. SIMD instructions are available on most modern processors and facilitate the speed-up of computation when performing the same basic operation on multiple data elements. Ray tracing makes use of SIMD operations for all of its fundamental operations.

A key step to increased throughput was utilising packets greater than just the SIMD width and using frustum or interval arithmetic (IA) techniques to produce speed-up over the simple SIMD processing by amortising the cost of the computation by utilising much larger ray packets of up to 64 rays. This was used for group rays into pyramidal shafts to rapidly cull triangles [DHS04] and for enhanced traversal [RSH05] of kd-trees. The use of large packets was later extended to grids [WIK*06] and BVHs [WBS07]. See Boulos *et al.* [BWS06] for a comprehensive overview. All these approaches relied on high ray coherence to deliver benefits over SIMD packet tracing.

As rays become incoherent, SIMD has less utilisation and it is possible for large packets do more computation per ray than optimised single-ray code paths. Coherent packets can be generated for primary rays, hard shadows and perfectly specular reflections [Sch06]. By carefully considering the order in which rays are cast Whitted and Distribution ray-tracing [BEL*06] and indirect diffuse illumination [WKB*02], to some extent, can be efficiently computed using packet-tracing. Usage is limited however as Reshtov has shown that after several perfectly specular bounces most packets have only a single active ray in them [Res06]. Performance for diffuse bounces and ambient occlusion rays is even worse, sustaining the same performance loss after only one or two bounces [WGBK07].

Further work has been done by Overbeck *et al.* [ORM08] in examining algorithms for large packet traversal and culling in the context of Whitted-style ray tracing. The work demonstrated a new traversal algorithm that responds well to degradation in coherency but only when dealing with perfectly specular reflection and refractions. It was also limited to supporting point light sources so that light rays generated were coherent.

In order to regain coherence for a given ray distribution re-ordering of the rays in a packet has been investigated. Work performed in Pharr *et al.* [PKG97] re-

ordered the rays using a coarse scheduling grid and while it greatly reduced disk I/O in an out-of-core ray tracer, the benefits for SIMD parallelism were unclear. Similar to this approach was one by Navratil *et al.* [NFLM07] where re-ordering was proposed based on the acceleration structure. The focus of this investigation was on geometric and ray bandwidth, however, no absolute performance numbers were provided. Coherence could also be gained by increasing SIMD utilisation, as was shown in Wald *et al.* [WGBK07], where large packets were traced in a breadth-first manner. In Mansson *et al.* [MMAM07], rays were grouped into batches larger than their packet size and then shot in packet-sized subsets sequentially. Rays were put into specific groups based on a number of different heuristics but none of the approaches performed better than SIMD packet tracing. The limitation of all these techniques was that they required a large number of rays to be available from which coherence could be extracted.

An alternative to tracing packets is to utilise a single ray and intersect it with N objects, be they nodes in an acceleration structure or primitives such as triangles in a leaf node. This was argued against in Wald *et al.* [WSBW01] due to the fact that this approach would not provide any benefit for shading and kd-trees, the primary acceleration data structure at the time. Kd-trees favour small leaf nodes and are binary in nature, not allowing for more than two children, which prevented the idea from being extended to traversal. While the argument was valid at the time, the subsequent resurgence of BVHs has meant that node-parallel ray tracing has become a viable option. BVHs naturally support higher branching and the initial ray tracers that used them [RW80, GS87] made use of this fact. Current research is moving away from packetisation due to importance of secondary rays, and their naturally incoherent nature. A number of researchers have proposed BVH solutions that use a branch factor equal to that of the SIMD width of the host processor to allow for the efficient traversal of single rays [WBB08, DHK08, EG08, Tsa09]. In Wald *et al.* [WBB08], two construction methods were examined for building BVHs with a branching factor greater than two, an SAH-based collapsing method and a top-down recursive splitting method. Both methods showed a reduction in memory usage (due to a decrease in the overall number of nodes) and a higher SIMD usage during traversal. One disadvantage is that the intersection test gathered and swizzled data for sixteen triangles from different memory locations which affected cache coherence. Dammertz *et al.* [DHK08], Ernst and Greiner [EG08] and Tsakok [Tsa09] utilised a very similar approach to Wald *et al.* [WBB08], focusing on a collapsing method

to build their hierarchies. All dealt with the gather/swizzle problem by caching the results of these operations in a SIMD friendly manner (storing the data for four triangles in SoA fashion) thereby reducing the impact of multiple fetches.

These new methods, which favour single rays, take a step towards removing the reliance that interactive ray tracing has on packetisation to maintain coherence and achieve interactive rates. Even with the reliance on packetisation diminished the large number of secondary rays, even if traced efficiently, still pose significant challenges in achieving global illumination at interactive rates. Methods that reduce the total number of rays that need to be computed must therefore be examined to determine how best they can be utilised. Selective rendering, detailed in Section 2.3.2, although until now limited to off-line rendering, contains characteristics similar to sparse sampling (Section 3.2.1.3), another method that has already been used to accelerate global illumination. As can be seen in Section 3.2.1.3, sparse sampling's and selective rendering's main advantage lies in ray tracing a significantly lower number of pixels than the displayed resolution, which is the major problem with ray traced images. Unfortunately, both these approaches by definition do not exploit the spatial coherence which has made interactive ray tracing methods possible. The results of sparse sampling systems seem to be more effective when performed on only parts of the computation such as in Stamminger *et al.* [SHSS00]; Tole *et al.* [TPWG02], this is partially due to fact that these systems decouple the primary visibility from the shading, a property also mentioned by the authors of the Razor system [DHW*07] (described in Section 3.2.1).

This thesis will first proceed to examine the impact of combining selective rendering and current interactive ray tracing approaches, further detailed in Chapter 4, to quantify the effects on coherence when packetisation is used. These results will allow for further insight into what components the calculation can be divided into and how these two approaches can be combined in the development of novel interactive global illumination solutions. These solutions will need to make full use of the computational resources available, via these selective methods, while at the same time attempting to maintain a level of coherence so that interactive rates can be achieved.

3.4 Summary

This chapter has presented work related to interactive ray tracing, interactive global illumination and the exploitation of coherence in interactive ray tracing. It provides a thorough background of interactive ray-tracing, for both the CPU and GPU, covering both past approaches and current state-of-the art methods. A comprehensive look at field of interactive global illumination follows, using a similar approach as the interactive ray tracing section, first focusing on CPU-based and then GPU-based techniques. Each section is focused on the different groups of algorithms that have been employed as well as the overall systems that have attained interactivity and the methods these systems used. Finally the role of coherence in interactive ray tracing is further examined as well as its effects on shaping certain approaches such as packetisation and stream processing. This examination showed how the focus has shifted from tracing bundles or packets of coherent rays to providing methods that enable single incoherent rays to be ray traced quickly, an important step towards acceleration global illumination solutions where the rays are mostly incoherent, for example when computing diffuse interreflections.

CHAPTER 4

Impact of Selective Rendering on Interactive Ray Tracing

4.1 Introduction

As mentioned before the main focus of this thesis is on accelerating the computation of high-fidelity graphics, and specifically global illumination effects, to interactive rates by utilising current state-of-the-art algorithms, that exploit cache coherency, and combining them with selective rendering techniques. Both selective rendering methods [OHM*04] and techniques that exploit cache coherency [BWS06] have been independently shown to greatly reduce the overall computational time of a global illumination solution but very little research has been performed on how these two methods interact, especially in an interactive setting, and if computing less pixels translates into less overall computational time.

This chapter identifies and quantifies the impact of this interaction by conducting a series of experiments in which different levels of selective rendering are simulated in a state-of-the-art interactive ray tracing framework. The goal is to determine if the speed-up offered by selective rendering is not offset by the decrease in cache and spatial coherency that naturally occurs when only specific pixels in the image are being rendered. This is of great importance as current packet-based ray tracing techniques are heavily reliant on very coherent packets of rays to achieve the levels of performance that offer interactivity on consumer desktop PCs, the primary goal.

4.2 Experimental Framework

To test the interactions of selective rendering with current interactive ray tracing techniques, a modified a state-of-the-art interactive ray tracer is used. Manta [SBB*06] from the University of Utah was chosen for this task due to its, at the time, up to date algorithms, optimised acceleration structures [Wal07], the packetised nature of its rendering pipeline as well its extensibility due to the fact it is an open source project. Manta also served as the basis for Optix [PBD*10], NVIDIA’s real-time ray tracer for the GPU.

A detailed breakdown of the Manta pipeline is beyond the scope of this paper but a detailed overview is given in Stephens *et al.* [SBB*06]. The performed experiment modified the pipeline at the stage where it would have minimal impact on the performance of the rest of the pipeline. At this stage the pixels that need to be traced are specified and then converted into ray packets. A stride parameter is then provided which enabled us to simulate selective rendering by sub-sampling the image in a pre-defined way.

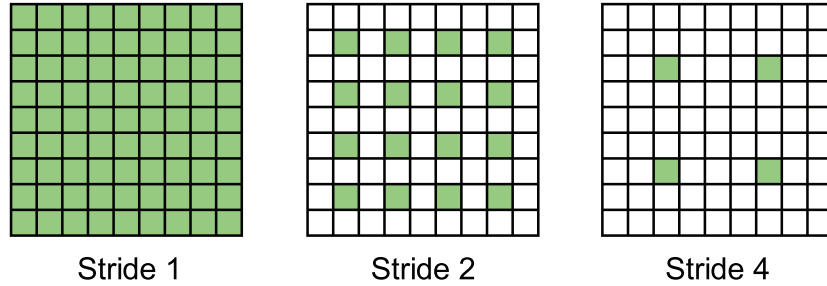


Figure 4.2.1: Explanation how stride changes.

Stride in the context of this experiment is a parameter that defines how the selective rendering is performed. While simply rendering a lower resolution image would produce the same amount of pixels as utilising a stride this would not truly simulate selective rendering. This is because while selective rendering only renders a subset of the pixels in the image, the image is still created at full resolution. This has implications at many stages of the rendering pipeline and in order to produce a superior simulation it was decided to render the image at full resolution with a stride instead of just rendering the image at progressively lower resolutions. The stride itself simply defines how many steps one must take away

from the current pixel before a new one is marked for processing. A stride of one simply means move over to the next pixel and mark it for processing, while a stride of two is move over twice, skipping one pixel and marking every second one for processing. Therefore a stride of X would render a $1/X^2$ of the total pixels in the image simulating the incoherent nature of selective rendering but allowing precise control over the amount of sub-sampling that took place. A ray is then generated per pixel and grouped into packets that are then traced through the scene. Figure 4.2.1 provides a graphical example.

Once the selective pixel sampler was added to Manta four test scenes were selected (these are detailed in Section 4.4) and the experiment was setup in such a way to minimise unintentional cache coherence. This could occur through the re-use of the same scene while testing multiple strides where it was possible that some data may have remained in cache from the previous run and therefore would skew results. All four test scenes were run after each other for each of the strides to ensure that the cache contained data from a previous scene and not data from the same scene that could be reused.

	Objects	Materials	Lights
Default	2	2	3
Sphere Grid	65	2	1
Complex 1	698,274	5	3
Complex 2	586,466	5	2

Table 4.1: Scene details

4.3 Experiment

For the experiments a resolution of 4096×4096 was selected to simulate a scene of 1024×1024 with 16 rays per pixel, this was done as using 16 rays per pixel may have led to results being skewed by certain code optimisations, such as shading being performed on multiple rays at once, within the framework utilised. Each scene was run with thirteen different strides, from 1 to 4096, doubling the stride each time. This produced a range of images where, in the first image, every pixel

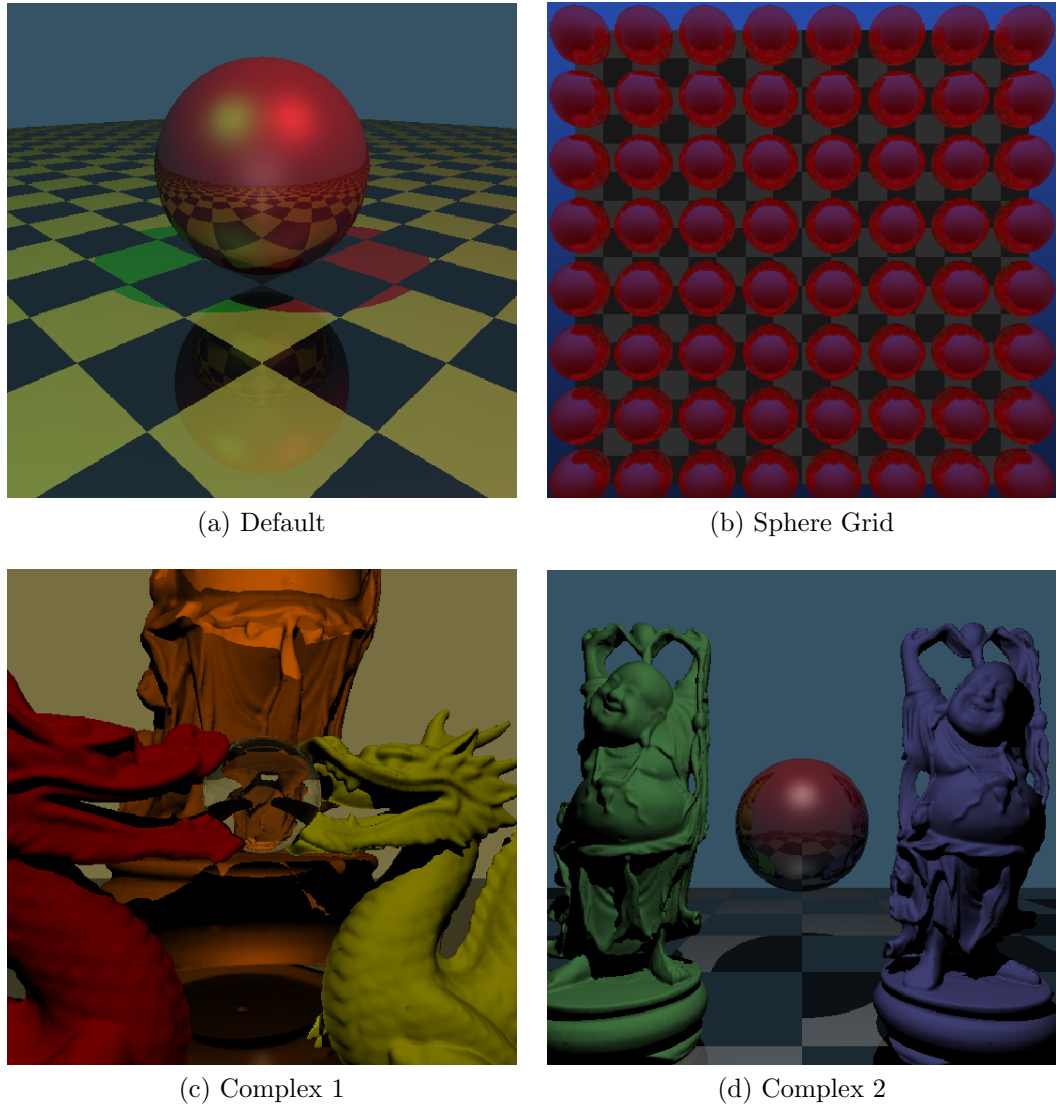
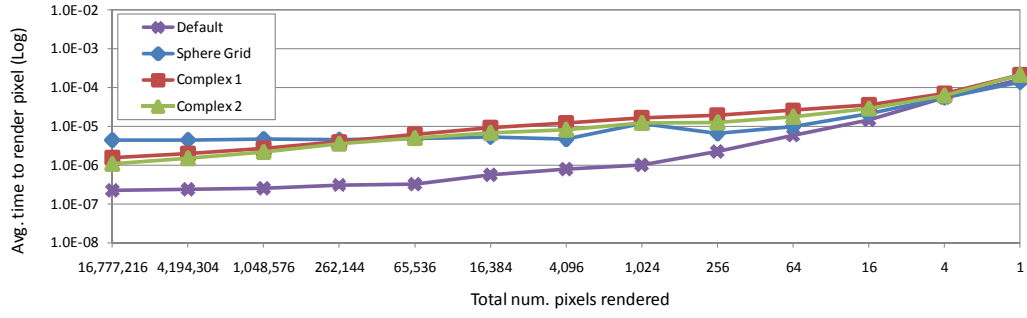


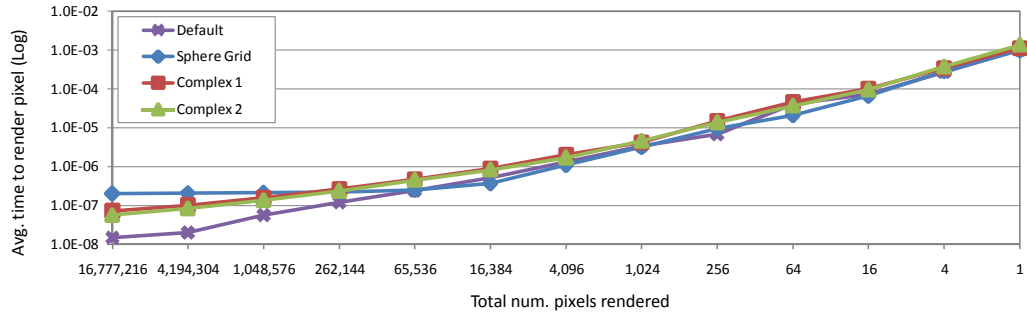
Figure 4.2.2: The four scenes used.

in the image was being rendered to the last image where only one pixel was being rendered.

For the experiment the standard Manta default of 64 rays per packet was used. Therefore for any results where less than 64 rays were processed the ray packet in Manta was not completely filled. This led to an increased cost per ray as the shading and other functions performed on the ray packet were no longer amortised over all 64 rays, but still amortised for every 4 rays when processing via SIMD [Int03].

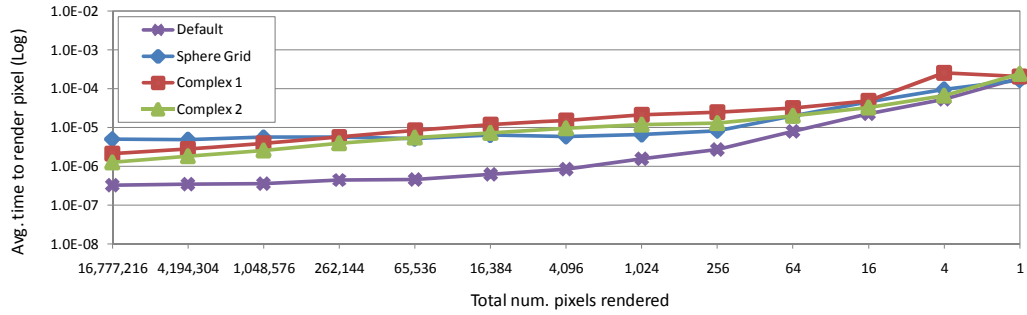


(a) Single-threaded

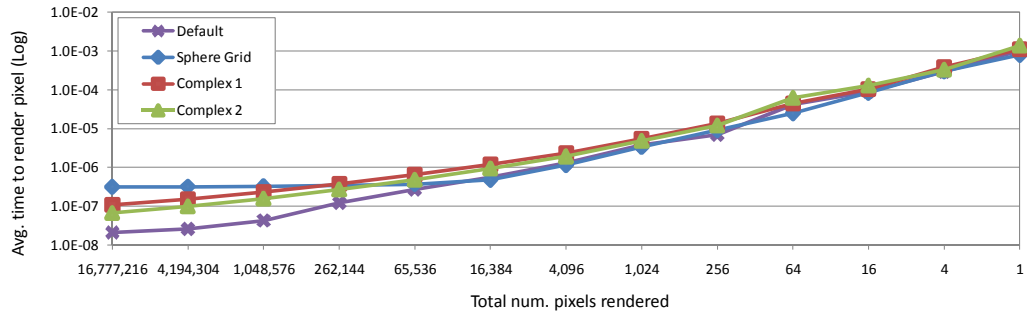


(b) Multi-threaded (8 threads)

Figure 4.3.1: Primary rays only.

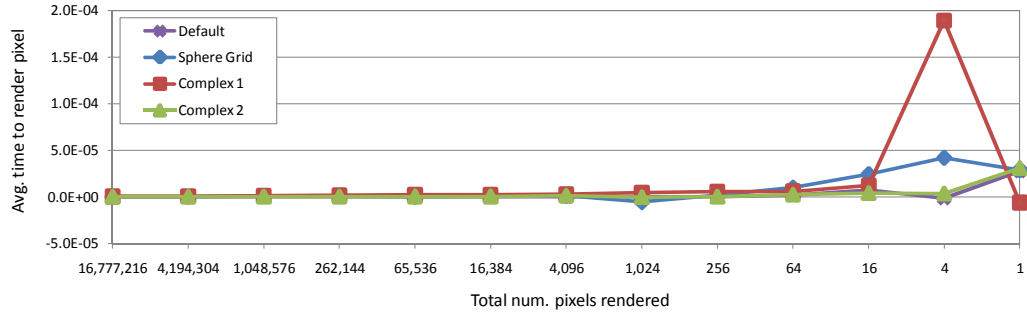


(a) Single-threaded

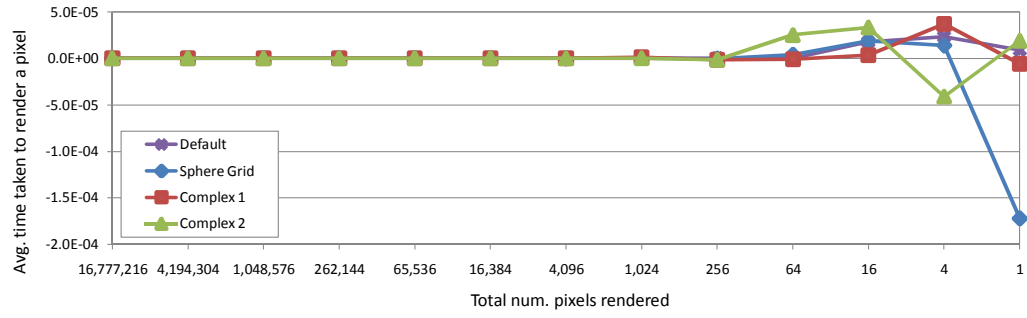


(b) Multi-threaded (8 threads)

Figure 4.3.2: Primary and secondary rays.

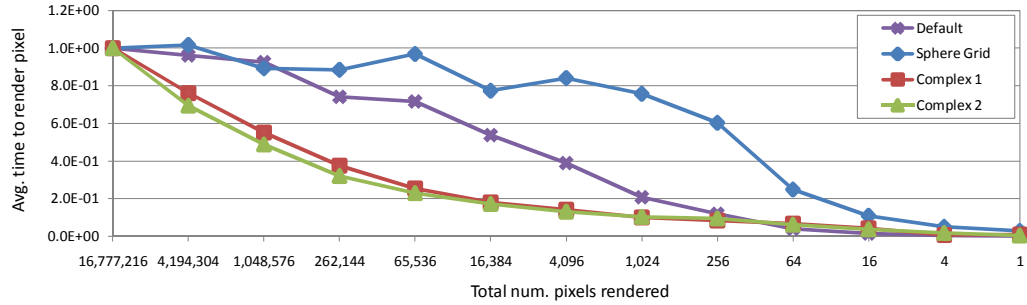


(a) Single-threaded

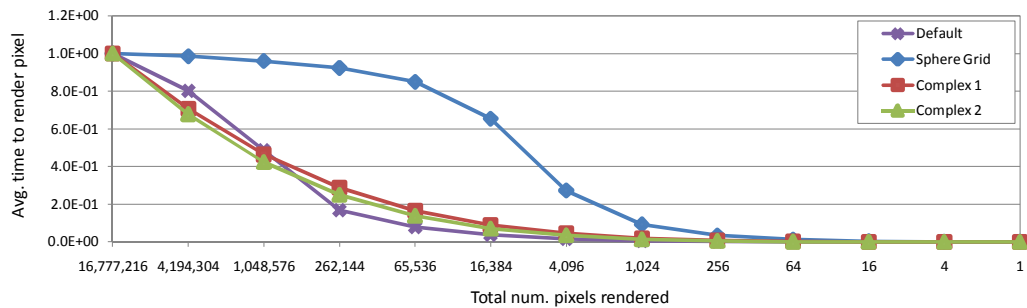


(b) Multi-threaded (8 threads)

Figure 4.3.3: Secondary rays only.



(a) Single-threaded



(b) Multi-threaded (8 threads)

Figure 4.3.4: Normalised speed-up compared to 4096×4096 .

4.4 Results

Figure 4.3.1 and 4.3.2 show results for primary rays only and primary and secondary rays respectively. The data displayed on the Y axis is the average time taken to render a pixel using a logarithmic scale versus the total number of pixels rendered on the X axis using an exponential scale. Figure 4.3.3 contains data for secondary rays only, where the Y axis is the average time taken to render a pixel using a linear scale versus the total number of pixels rendered on the X axis using an exponential scale. Figure 4.3.4 shows the normalised speed-up for different strides when rendering both primary and secondary rays where the Y axis is the normalised speed-up using a linear scale versus the total number of pixels rendered on the X axis using an exponential scale.

One can see in Figures 4.3.1a and 4.3.2a that for the single-threaded results there is an overall increase in the average time taken to render a pixel as the total number of pixels rendered decreases, although it is not strictly a logarithmic increase and for certain scenes such as *SPHERE GRID* and *DEFAULT* is linear in certain areas. Examining the multi-threaded results one can see much less disparity between the different scenes and an almost perfectly logarithmic increase in *average time taken to render a pixel* as *total number of pixels rendered* decreases. This can most likely be attributed to the increased amount of cache, 16MB of L2 cache on the dual Quad-core vs. 2MB on the single-core Prescott, and as cache coherency decreases a higher overall cost per pixel is observed, than was noted with the single-threaded results, for each ray.

The results in Figure 4.3.4 indicate, for a given stride, how close the results are to an ideal speed-up with the results being normalised to make comparison easier. An ideal speed-up being where the amount of time taken to render the given pixels is $T_1/(S_n^2)$ where T_1 is the time taken to render the image at a stride of 1 and S_n is the stride. For all scenes one can observe a decrease in speed-up as the stride is increased, and for all scenes other than *SPHERE GRID* the speed-up is only 20% of the ideal when the stride reaches thirty two. *SPHERE GRID* shows a much slower decrease in speed-up mostly due to its regular nature and higher coherence when calling shading and intersection routines. For both *DEFAULT* and *SPHERE GRID* it can be seen that the decrease in speed-up is much faster in the multi-threaded results (Figure 4.3.4b) and while not as pronounced both *COMPLEX 1* and *COMPLEX 2* also show a decrease in speed-up when comparing the single-threaded and multi-threaded results. This, like the

other results, shows that an increase in the amount of cache available adversely effects the performance of selective rendering.

4.5 Discussion

The implications of the results as pertaining to the interaction of selective rendering and interactive ray tracing are very interesting. As can be seen from Figure 4.3.1, 4.3.2 and 4.3.3 for the majority of scenes as the total number of pixels computed decreases the average time taken to compute a pixel increases logarithmically. One can see an order of magnitude increase in the average time taken to compute a pixel as one goes from a stride of one to a stride of thirty two, the majority of this increase can be attributed to a decrease in cache coherency as the width of the packet increases with the growing stride.

Further evidence that this occurs as a result of cache misses and poor spatial coherency can be seen in the results of Figures 4.3.1 and 4.3.3. In Figure 4.3.1 the highly coherent primary rays are timed separately and show a constant and logarithmic increase in the average time taken to compute a pixel as the total number of pixels rendered decreases due to the stride increase. Only specular secondary rays are timed in Figure 4.3.3 and show that as the total number of pixels rendered decreases due to increased stride. There is very little measurable change in the average time taken to compute a pixel until only 256 samples are being calculated out of a possible 16,777,216. At this level any variance can be attributed to coincidental coherence or just not traversing complex parts of the scene. This shows that secondary rays are playing a very insignificant part in the overall deterioration of speed and that it is the very spatially coherent primary rays that are responsible for this speed loss. The loss in spatial coherence between the primary rays as stride increases is being directly translated into a loss of cache coherency and therefore an overall increase in average time taken to render a pixel. This is most apparent in multi-threaded results (Figure 4.3.1b) as the amount of L2 cache available to the CPUs increase.

As all the current interactive ray tracing techniques [SBB*06, RSH05] and optimised accelerating structures [SSK07, Wal07] rely on spatially coherent rays to amortise the cost of tracing large packets and when this spatial coherency is no longer present, as is the case with selective ray tracing, many of the speedups gained from these techniques are lost. On the other hand, secondary rays are

naturally incoherent and due to this fact selective rendering has very little negative impact on the component of the ray tracing calculation as can be seen in Figure 4.3.3.

The discussion above along with results from Figure 4.3.4, which contain less than ideal speed-up that drops off sharply as stride increases for even the simplest scene, shows that selective rendering is useful but not as useful as it should be given the overall increase in average time taken to render a pixel. From this one can potentially draw two conclusions. One can either selectively render primary rays coherently using some form of adaptive or progressive approach or just not utilise selective rendering for primary rays and apply it just to secondary rays.

4.6 Summary

This chapter has quantified and provided an analysis of the effects that selective rendering has on interactive ray tracing. An existing state of the art interactive ray tracer (Manta from the University of Utah [SBB*06]) was modified to allow us to simulate different levels of selective rendering. The results show that the average time taken to render a pixel increases as the stride that was used is increased at and that this is mainly due to poor spatial coherence and its effects on primary rays. Primary rays rely on packets of spatially coherent rays for the traversal of acceleration structures and intersection of objects without this spatial performance deteriorates. This is problematic as cache coherence is heavily relied on by a large number of modern algorithms to provide the speed-up necessary for interactivity. It is shown that secondary rays, when rendered with a stride, contribute almost nothing to the increase in average rendering time for a pixel, as opposed to primary rays. This is because secondary rays are naturally incoherent and aren't effected by poor spatial coherence and a lack of cache coherence.

From the results it can be concluded that while selective rendering can be utilised in an interactive context the penalties incurred must be carefully managed or new algorithms devised to handle them. Adaptive or progress methods that maintain spatial coherency and perform selective calculations solely on the secondary rays should be further examined. Chapter 5 makes use of this information when constructing a framework that allows us create novel interactive global illumination solutions, by pairing existing algorithms with selective rendering techniques, so that they exhibit the features identified in this chapter.

CHAPTER 5

Adaptive Interleaved Sampling for Interactive Global Illumination

While interactive ray tracing methods handle primary rays extremely efficiently, mainly via optimised data structures and algorithms that exploit coherency, these methods do not apply to incoherent secondary rays. These are mainly used to compute effects such as diffuse interreflections, soft shadows and participating media, all important components of a global illumination solution. This inability for current techniques to deal with secondary rays and the use of selective rendering, via sparse sampling, to accelerate the computation of global illumination solutions was identified and discussed in detail in Section 3.3. This chapter examines potential solutions to this problem and how combining selective rendering techniques and interactive ray tracing and global illumination methods can lead to novel interactive global illumination solutions.

5.1 Introduction

Interleaved sampling methods (See Section 2.3.3) can provide relatively efficient ways of sampling for global illumination effects. Traditionally the sampling methods chosen are consistent, and not adaptive, thus do not take into account possible variations in the scene being rendered. Certain areas in an image will require more computation than others to achieve a good result, therefore to obtain a satisfactory image many samples would be wasted using this traditional approach. By adapting the interleaved sampling to only shoot more samples when required, reasonable computational gains are achieved without a perceived loss in quality. Furthermore, due to the linearity of the rendering equation, global illumination

effects can be computed and sampled independently. These individual effects are referred to as components [SFWG04]. The guidance for the adaptive interleaved sampling is tailored for each individual component further improving the rendering times of the method. The technique is demonstrated using three components: soft shadows, indirect diffuse lighting and participating media. The computation of other high-fidelity rendering features such as glossy effects and depth of field could follow a similar strategy.

This chapter first presents a framework, encompassing the methodology used along with its implementation details (Section 5.2). This section introduces the framework developed to identify, deconstruct and pair an existing interactive ray tracing and interactive global illumination methods with selective rendering techniques to produce novel interactive global illumination solutions. Three major stages are identified in the process and provide detailed information on what these stages entail. The framework described here is utilised in this chapter, as well as the subsequent (Chapter 6) chapters, to develop and implement two new interactive global illumination solutions. Sections 5.2.1 - 5.2.3 provide more detail on the three main stages of the framework while Section 5.2.4 provides implementation details. The focus is then shifted to the algorithm itself which extends interleaved sampling to be adaptive, computing further samples only when necessary (Section 5.3) into adaptive interleaved sampling (AIS). This algorithm is then used in the computation of a number of components, using custom adaptive heuristics for each one, such as soft shadows (Section 5.3.3), indirect diffuse illumination (Section 5.3.2) and participating media (Section 5.3.4). Results (Section 5.4) along with validation data comparing the approach to the original Instant Global Illumination (IGI) algorithm, based on Wald *et al.* [WKB*02], as well as reference path traced images are then presented.

5.2 Framework

Following on from the results in Chapter 4 this section presents the methodology and processes utilised during the selection of selective rendering techniques and interactive ray tracing / global illumination methods. It then examines the structured approach used to determine which selective techniques are applied to specific components of the computation. The methodology is refined into a framework which is used in this, and subsequent chapters (Chapters 6 and 7),

to guide the selection of algorithms that are considered when developing these novel approaches as well as the approaches utilised to make them interactive. Below in Figure 5.2.1 the process flow of the framework is visualised with Sections 5.2.1 - 5.2.3 providing further details on each of the primary steps: Identification, Deconstruction and Pairing.

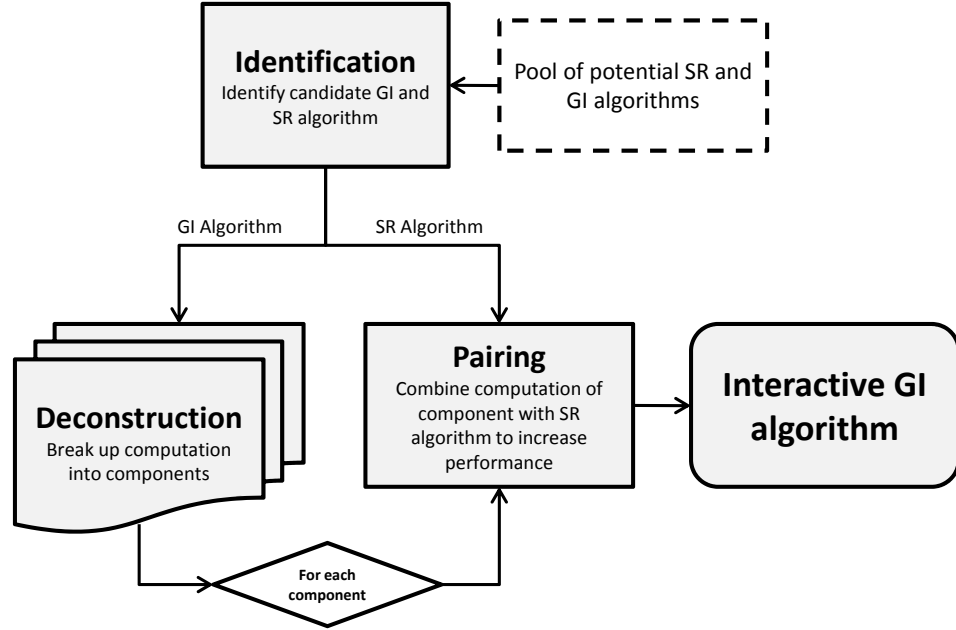


Figure 5.2.1: This figure visualises the process flow of the framework.

5.2.1 Identification

Due to the large number of suitable algorithms, encompassing both interactive ray tracing algorithms and interactive and off-line global illumination algorithms, that could be combined with selective rendering techniques, a more structured approach was developed to improve selection criteria. For the rendering algorithm the criteria utilised to identify which algorithm to utilise focus on algorithms that had already achieved interactive or near interactive rates while computing diffuse interreflections or a sufficiently large number of rays which would make this computation possible. For the selective technique the focus was placed on making the selected rendering algorithm adaptive or progressive to reduce the total computational cost.

For this specific implementation utilising the literature reviewed in Chapter 3 interleaved sampling [KH01] was chosen as the base algorithm to extend. As mentioned previously in Section 3.2.1 it had already been utilised in an interactive global illumination solution [WKB*02], by being combined with Instant Radiosity [Kel97], as well as a number of other areas [SGNS07, FBP08] to reduce overall computational costs. Due to its nature as a sampling algorithm it could potentially be extended to a number of different components, not just diffuse interreflections, and therefore held further potential to reduce the overall computational costs.

5.2.2 Deconstruction

The deconstruction of the rendering equation into components occurs as follows. The radiance at a pixel (x, y) in the direction Θ which intersects an object in the scene at point x_b (L_t) is the sum of all emitted (L_e) and reflected (L_r) light. This is shown in the equation below

$$L(x, y) = L_t(x_b \rightarrow \Theta) = L_e(x_b \rightarrow \Theta) + L_r(x_b \rightarrow \Theta)$$

where

$$L_r(x_b \rightarrow \Theta) = \int_{\Omega_{x_b}} f_r(x_b, \Theta \leftrightarrow \Psi) L(x_b \leftarrow \Psi) \cos(N_{x_b}, \Psi) dw_\Psi$$

The standard rendering equation [KH84] is extended by also calculating the contribution of single-scatter homogenous participating media which extends the equation to:

$$\begin{aligned} L(x_b \rightarrow \Theta) = & e^{-\tau(x_a, x_b)} L_t(x_b \rightarrow \Theta) + \\ & \int_{x_a}^{x_b} e^{-\tau(x_a, x)} \kappa_a(x) L_e(x \rightarrow \Theta) dx + \\ & \int_{x_a}^{x_b} e^{-\tau(x_a, x)} \kappa_s(x) \int_{\Omega} P(x, x \rightarrow \Theta, x \leftarrow \Phi) L_i(x \rightarrow \Theta) d\Phi dx \end{aligned}$$

where x_a is the point in space where the sensor is located, κ_a is the absorption coefficient, κ_s is the scattering coefficient, the extinction coefficient is $\kappa_t = \kappa_a + \kappa_s$, $\tau(x_a, x_b) = \int_{x_a}^{x_b} \kappa_t(x) dx$ is the optical thickness, P is the phase function, L_e the emitted radiance and L_i the in-scattered radiance. For full details and

expansion of all terms please refer to Nishita *et al.* [NMN87]. The equation is then further deconstructed into direct lighting, indirect lighting and participating media components as follows

$$L(x_b \rightarrow \Theta) = Att(x_a, x_b)(L_d(x_b \rightarrow \Theta) + L_i(x_b \rightarrow \Theta)) + \int_{x_a}^{x_b} Att(x_a, x_b)L_p(x_a, x_b)$$

where L_d is the direct lighting component, L_i the indirect lighting component, L_p is the participating media component and $Att(x_a, x_b) = e^{-\tau(x_a, x_b)}$ is the attenuation factor due to the participating media. The direct lighting is evaluated using the standard area formulation (Equation 2.1.6):

$$L_d(x_b \rightarrow \Theta) = \int_A f_r(x_b, \Theta \leftrightarrow \Psi)L(x_l \leftarrow -\Psi)V(x, x_l)G(x, x_l)dA_{x_l}$$

Both the direct lighting and participating media components are evaluated using standard Monte Carlo integration while the indirect component is evaluated utilising Instant Radiosity [Kel97]. The indirect lighting evaluation is performed in two steps: shooting photons from light sources and creating VPLs and a subsequent gathering pass. The gathering pass evaluates indirect diffuse irradiance incoming from all VPLs to provide a final contribution:

$$L_{indirect}(x_b \rightarrow \Theta) = \sum_{k=1}^N f_r(x_b, \Theta \leftrightarrow \Gamma)L_{e,k}V(x, x_k)G'(x, x_k)$$

where N is the number of VPLs in the scene, $V(\cdot, \cdot)$ is the visibility function between two points, $\Theta \leftrightarrow \Gamma$ is the light vector for the k -th VPL, $L_{e,k}$ is the emitted radiance of the k -th VPL, y_k is the position of the k -th VPL, $f_r(x, \Theta \leftrightarrow \Gamma) = \rho/\pi$ in the case of diffuse component (ρ is the reflectance), and G' is the bounded geometry term [PH04].

While the deconstruction leaves us with three main components it should be noted that the deconstruction can be generalised to an arbitrary number of components where the granularity of deconstruction is dictated by the implementation requirement. Any of the components presented here can be further deconstructed according to any criteria, for example, the type of the bi-directional reflectance function or light source.

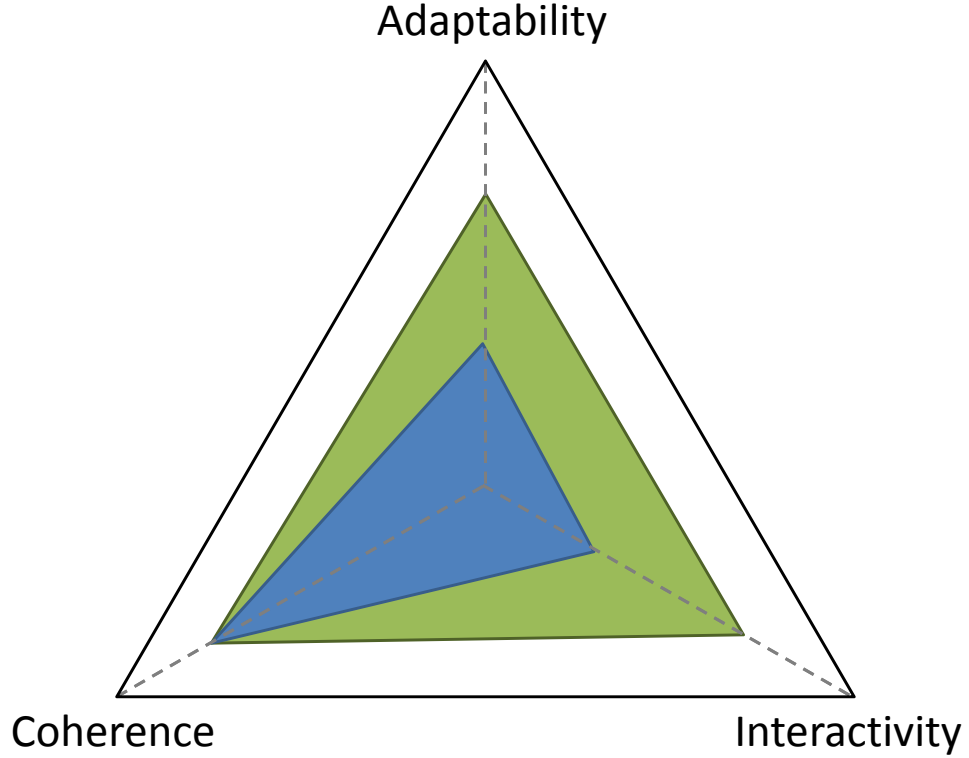


Figure 5.2.2: This figure visually demonstrates the goals of this framework. The blue plot indicates current algorithms, highly coherent but only slightly adaptable and interactive. The green plot visualises the type of algorithm this framework is aiming to create, a more balanced approach in which adaptability and therefore interactivity are increased while allowing for coherence to be exploited.

5.2.3 Pairing

Once deconstruction is complete each component is evaluated, in the context of the base algorithm used, to determine what potential selective rendering technique it can be paired with to increase performance. Due to the nature of the framework each pairing is tailored to each component and it is possible to use a number of different selective rendering techniques based on the result required. The three key criteria that drive selection of not just the selective rendering techniques but the base algorithm(s) are based on the need to maintain spatial and temporal coherency (*coherency*) while decreasing the computational cost of the solution by focusing the computation on areas of the image which will grant best visual fidelity (*adaptability*) all while interactive rates (*interactivity*). These criteria need to be balanced, on a per component basis, to produce an interac-

tive global illumination solution, with the added complexity of the interactions between the three criteria.

Interactivity is the fundamental goal of this framework, in this context it refers to both the timing of the update of the global illumination solution, imposing an upper limit on the amount of computational time that can be allocated for any given frame, as well the ability for the scene to be entirely dynamic without the need for pre-computation. Due to the interactive nature any algorithm chosen must also provide a stable approximation of the lighting solution that has temporal continuity from frame to frame as well as within a frame. As has been shown in Section 4.4 to increase interactivity both coherence and adaptability have to increase in order to decrease overall computational time. Due to the varying computational requirements and implementations for each component of the final computation, as can be seen in Sections 5.3.2-5.3.3, an optimal use of resources can be only achieved by applying optimisations on a per-component basis.

Coherence refers to the level of temporal and spatial coherence maintained while computing the interactive global illumination solution. As was demonstrated in the literature, Section 3.1.1, as well as the results in the previous chapter the majority of interactive ray tracing techniques require a high level of spatial coherence for the primary rays to be calculated effectively. The exploitation of temporal coherence is also an important aspect of the solution given its interactive nature. To exploit coherence a number of hardware specific mechanisms can be employed such as SIMD/SSE instructions [Int03], CPU caches and coherent memory access. By ensuring that data structures are correctly aligned and placed in memory in such a way that related data is grouped together correct utilisation of caches will also occur, further exploiting coherence, especially if data structures are small enough that large parts can reside in the processor cache. Algorithmic mechanisms such as packetisation also play an important role, this was discussed in detail in Section 3.3. Locality of reference also affects coherence, by ensuring that pixels to be processed are grouped together, in tiles for example, coherence is maintained. While an increase in coherence will indicate an increase in interactivity this, unless carefully managed, will generally indicate a reduction in adaptability.

Finally adaptability refers to the measure of how all available computational resources are optimally assigned to areas of the computation that require them most. This is not only linked to the selective rendering techniques utilised by them

adaptive, progressive or even time constrained but also how parallel an algorithm is. A high level of adaptability in the algorithm indicates the ability to make use of all the resources available, this means the ability for the algorithm to parallelise and scale well with an increase in available computational resources, such as making use of all the cores in a multi-core system, and doing so with high levels of efficiency. This not only affects the choice of algorithms but more importantly the data structures used due to the challenges parallelisation introduces, as was seen in Section 2.3.5.

These three criteria need to be balanced on a framework level to produce an efficient interactive global illumination solution, as can be seen in Figure 5.2.2 and they need to be balanced per component to achieve maximum results. A balance of all three criteria indicates an algorithm that makes full use of computational resources, focusing the resources to area of the computation that require it most while at the same time extracting maximum temporal and spatial coherence from the computation while maintaining interactivity at all times.

In this particular implementation, as can be seen in Sections 5.3.2 - 5.3.3 each component used in the framework was made adaptive by designing a custom independent local heuristic that only requires information from the current tile. By using tiling coherence was maintained during the computation, this was further enforced by ensuring all heuristics were local to the tile to exploit spatial coherence when accessing the tile data which itself was coherently computed using a deferred shading [DWS*88] approach. Temporal coherence was maintained by using quasi-monte sampling techniques which ensured that results computed for a given frame were consistent when everything was static. By eliminating dependencies between tiles and having all adaptive computations occur on a per component basis for each tile parallelism and adaptability were maintained, with multiple tiles being calculated simultaneously.

5.2.4 Implementation

The framework implementation was written using portable C++. The current version of the framework compiles and runs under MS Windows, Linux and Mac OSX.

The implementation was designed to be extensible and portable. The kernel is currently optimised for single ray traversal, with no packetisation having been implemented at this stage, to allow for a wider array of algorithms to be utilised.

The core of the ray tracing kernel consists of an implementation of dynamic bounding volume hierarchy (BVH) as described in Wald *et al.* [WBS07]. SIMD computation has been limited to a single aspect of the BVH traversal, the ray bounding volume intersection. The BVH currently only supports the updating of the bounding volumes of each node at each frame.

A deferred shading approach [DWS*88] is utilised by calculating primary-hit information and storing it in a number of buffers. The buffers hold the normals, depth, albedo colour and material type. This has the advantage that each component has access to all the relevant information stored in a spatially coherent manner to optimise memory access. Interleaved sampling is naturally tile-based, and a 3×3 pixel tile is utilised to produce a work packet (see below) that exploits this coherent memory access. For each tile all components are calculated by the same thread ensuring optimal use of processor cache when accessing needed information.

The renderer is designed to be multithreaded, with one thread per processor. At the start of every frame, each thread collects a work packet detailing the pixels to be rendered, from a centralised job queue. For each tile the thread computes the primary rays and all other components. After computing a work packet the job queue is accessed for further work packets. When no further work packets exist on the job queue, the rendering stage is considered complete. At this stage the threads wait on a barrier. After barrier synchronisation, each thread collects further work packets from the job queue. These work packets now represent pixels to be filtered. When filtering is completed, the threads wait on a barrier again before commencing the next frame. A central thread is responsible for work packet creation and allocation onto the job queue. It is also responsible for displaying the image.

Due to the continuous nature of the indirect diffuse lighting, which changes smoothly and does not contain sharp discontinuities, this is computed at lower resolution. The sub-sampled solution is updated using a guided bilinear filter [TM98] based on joint bilateral upsampling [KCLU07], similar to the up-sampling solution used in Sloan *et al.* [SGNS07]. The implementation uses the full resolution solution computed at the deferred shading stage as guidance. It utilises the depth at the individual pixels as well as normals to keep within object boundaries and to avoid up-sampling over discontinuities. For the guided bilinear filter a modified version of the joint bilateral filter is used and replaces the expensive gaussian filters with simple bilinear ones. Assuming V VPLs are generated

and an IS pattern of $N \times M$ pixels is used IGI will sample a single subset of V/NM VPLs per pixel. Between V/NM and V VPLs are sampled, IGIs maximum sampling rate being this implementations minimum one. Therefore while the indirect lighting computation occurs at a lower resolution, the per-pixel cost can be substantially higher than in IGI for pixels that require refinement due to more VPLs being sampled.

This helps to highlight one particular aspect of adaptive sampling. It can complement sub-sampling well since the number of samples used for the component computation, in this case indirect diffuse lighting, can adapt based on complementary heuristics. When upsampling the indirect diffuse component, see Section 5.3.2, using only one subset of the VPLs, as in IGI [WKB*02], there are occasions when none or very few of the VPLs in that subset are visible. These conditions often lead to flickering when upsampling, particularly in areas that are predominantly indirectly lit and highly occluded. AIS takes care of this problem naturally via the heuristic that detects these undersampled pixels and, if necessary, further samples more VPLs. While there are computational costs associated with the upsampling and extra VPL sampling the approach is still faster than IGI as can be seen in Section 5.4.

All the random sampling within the renderer relies on low discrepancy sampling [KK02] to produce well distributed sample points and maintain spatial coherence across the multiple threads as well as temporal coherence.

5.3 Adaptive Interleaved Sampling (AIS)

Similarly to the work presented by Debattista [Deb06] adaptive interleaved sampling decomposes the standard rendering equation as presented by Kajiya [Kaj86], with the participating media extensions as initially described by [KH84], into individual components that have distinct adaptive guidance mechanisms. These components (Figure 5.4.1) are direct illumination utilising not just point but area light sources that produce soft shadows (**SS**), indirect illumination or specifically indirect diffuse illumination (**ID**) and single scattering homogenous participating media (**PM**). It would be straightforward to add other components to this framework following a similar model. Furthermore, visibility and shading calculations are performed separately by utilising deferred shading [DWS*88]. This is done to maintain coherency when executing shading operations as the method is de-

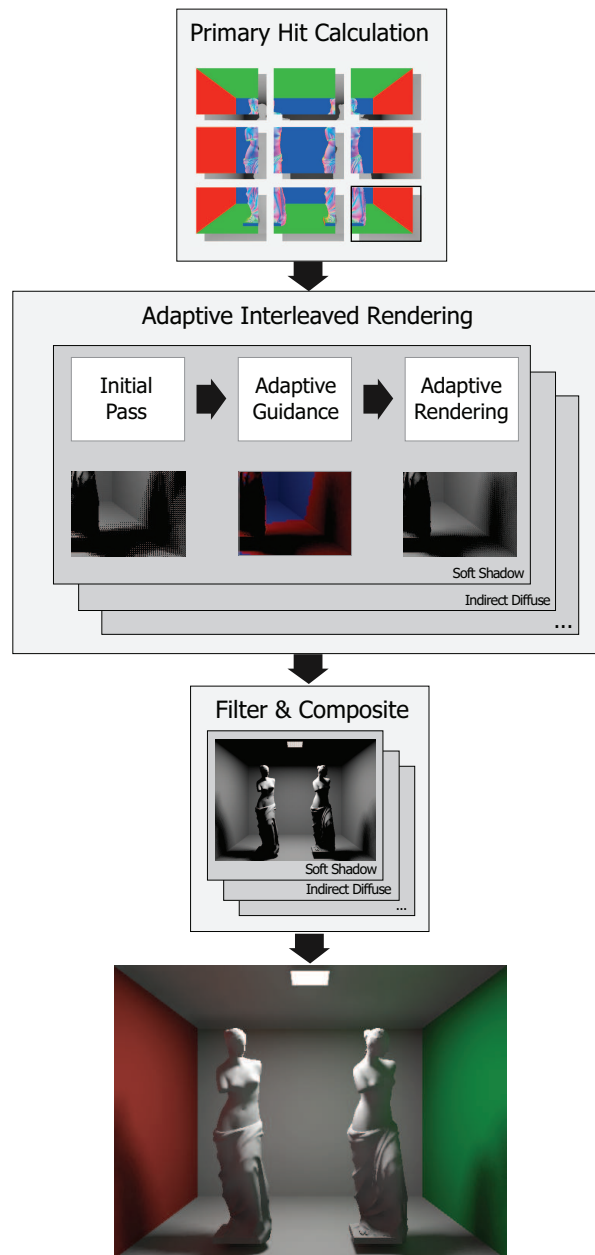


Figure 5.2.3: The pipeline for AIS. The primary hit calculation determines the object hit, depth and normal calculation. The tiles shown are for illustrative purposes. They are much larger than they would be in practice (a typical value is 3×3). Adaptive interleaved sampling is computed for each tile and for every component. Adaptive interleaved sampling of a soft shadow component for a given tile is shown. Once adaptive interleaved rendering occurs, all the components are filtered and composited.

signed predominantly with interactive environments in mind, where maintaining coherency is crucial.

The approach takes IS as presented in Keller *et al.* [KH01], and later utilised by Wald *et al.* [WKB*02] for indirect diffuse and soft shadow computation. The technique developed in Wald *et al.* [WKB*02] was chosen for a number of reasons. Firstly, it allows the reduction of the number of samples needed per pixel for each individual component, and given an interactive approach is being attempted this is crucial. The use of tiles in the technique increases spatial coherence when computing the solution and the final step of filtering removes the structural noise and produces a smooth and, more importantly, noise-free result unlike other monte carlo techniques. The lack of noise means the technique is more stable temporally producing results that don't flicker frame to frame. The IS when combined with adaptive sampling techniques results in Adaptive IS (AIS). This approach allows us to target components and create specific guidance metrics and heuristics that exploit the strengths of each algorithm.

For AIS the method links each pixel in the $N \times M$ tile to a particular subset of samples in a component specific sampler. This is done to allow for a reliable and repeatable sampling of the component space with the aim of multiple iterations utilising a disjoint subset of samples for each pixel. Each sampler has a number of disjoint sets of samples with each set used for a specific sampling iteration, and each pixel being linked to a set of samples, one from each disjoint set, which is always used for that specific pixel as the tile pattern is repeated. This means for the first sampling iteration the first pixel in the tile will always use the same sample even as the tiling IS pattern is repeated over the entire image. The results of each of the component calculations are then filtered, as per Wald *et al.* [WKB*02] as they contain structured noise due to the regular re-use of samples as the pattern is tiled, and later composited into a single resulting image.

Once the initial pass has been performed, an adaptive guidance metric analyses the $N \times M$ tile and determines if further refinement is necessary. This step is performed for each individual component utilising a component-specific heuristic. If the guidance identifies that the number of samples used is insufficient, a number of further iterations for each pixel in the tile are performed with each new iteration accessing the sample assigned to that pixel in the tile for that specific iteration. Figure 5.3.1 demonstrates the results produced by the three distinct stages for the **SS** component.

Once the initial and refinement pass have been completed for all tiles the

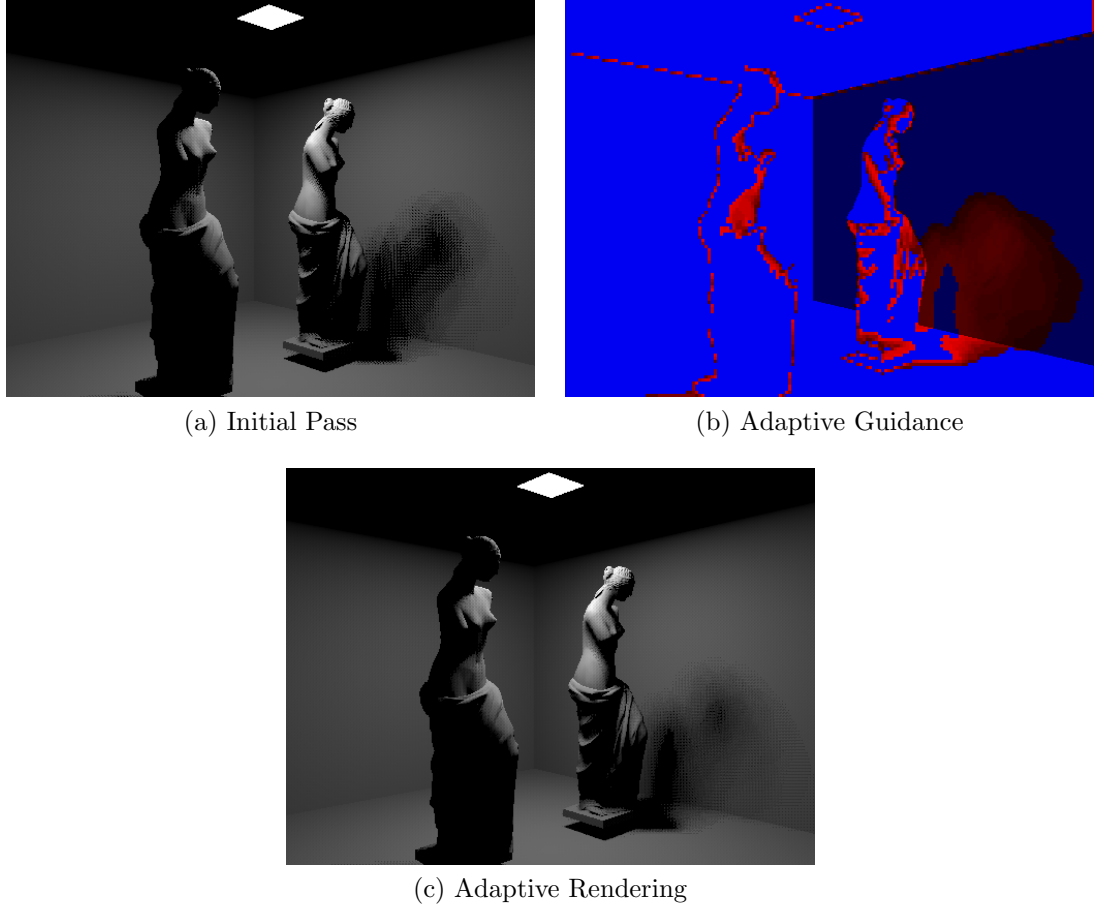


Figure 5.3.1: The three steps used in adaptive interleaved sampling.

resulting image is then filtered to remove the structured noise that arises from the IS sampling. This is done using an $N \times M$ discontinuity buffer [WKB*02]. Unlike Wald *et al.* [WKB*02] the approach filters each individual component separately. This is done as each components' contribution has a different structured noise pattern, due to their individual samplers, that when filtered individually produces a smoother result than if the filtering were to happen after composition step. The filtering kernel utilised for these operations is a simple box kernel, the same one utilised in Wald *et al.* [WKB*02]. It should be noted that the filtering occurs on the original luminance values of each component before any tonemapping or gamma correction has occurred.

In the end, for an $N \times M$ tile, each tile computes between NM and PNM samples. P being dependant on the guidance as well as the upper-bound allowed for each component. Finally, the individual components are composited together

with the albedo colour. The albedo colour of the objects is decoupled from the lighting calculations to ensure that high frequency detail, from textures for example, does not interfere when filtering the individual components.

The distinct adaptive guidance utilised for each component means that, unlike standard IS, more samples are taken in areas that contribute the most to the overall image. Figures 5.3.2, 5.3.3 and 5.3.4 show which parts of the image are adaptively sampled for the **SS**, **ID** and **PM** components respectively. Blue regions indicate areas that have not been adaptively sampled at all while red regions indicate the amount of iterations performed (the brighter red the region is, the more adaptive refinement has occurred).

Listing 5.1: Generalised AIS workflow

```

1  // Initial pass
2  for (each pixel in tile) {
3      pixel.col = Black
4      for 1 to initialIterations {
5          sample = getSample*(pixel)
6          pixel.col += contrib*(sample)
7          getRefinementInformation*()
8      }
9  }
10
11 // Adaptive guidance
12 refine = getAdaptiveGuidance*()
13
14 // Adaptive pass
15 if (refinementConditions*(refine) == true) {
16     numIterations = getIterations*(refine)
17
18     for (each pixel in tile)
19         for 1 to numIterations
20             sample = getSample*(pixel)
21             pixel.col += contrib*(sample)
22 }
```

5.3.1 Algorithm

Listing 5.1, shows the generalised approach that is undertaken when using AIS. This algorithm is an expansion of the second stage in the pipeline shown in Figure 5.2.3. Each function that ends with an asterisk denotes a component specific function which is further expanded upon in the upcoming sections. In

terms of the pseudo-code itself, the function *getSample()* returns a sample for the specific pixel. This sample depends on the component being calculated and the number of iterations that have already occurred. On lines 6 and 21 the function *contrib()* calculates the contribution for a specific component.

5.3.2 Indirect Diffuse Lighting

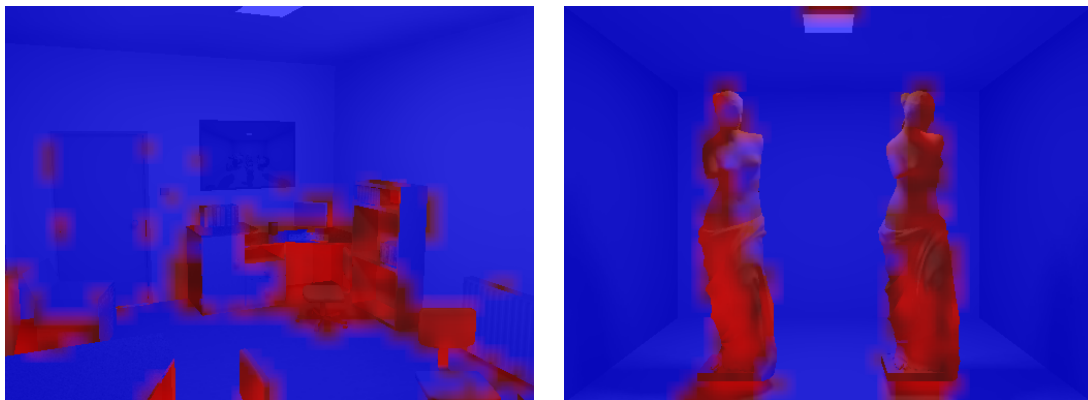


Figure 5.3.2: **ID** adaptive guidance.

The indirect diffuse lighting approach is based on the same model that is used in Wald *et al.* [WKB*02]. A set of VPLs is generated and then used to perform the **ID** calculation. Once the VPLs have been generated they are divided into subsets based on the tile size. Therefore for a $N \times M$ tile the number of subsets is NM . The sampler provides a sample to each pixel in the tile indicating which of the subsets it must utilise, ensuring that each pixel gets a unique subset. On subsequent iterations the pixels will receive a new subset of VPLs to sample, ensuring that each pixel does not utilise the same VPL subset more than once when refining. This careful break-up of VPLs into subsets ensures that the noise generated across the tile is structured in nature and can be eliminated in the filtering stage. If a larger number of VPLs and VPL subsets were generated, resulting in more sets than there are pixels in the tile, further refinement would be possible but would result in random noise. This would result in temporal noise across multiple frames.

As can be seen in Listing 5.2, the adaptive guidance runs an initial pass on the tile while keeping track of how many VPLs are occluded when they are sampled (Line 1). The average ratio of the occluded VPLs to the total number of

VPLs sampled is then utilised to determine how much refinement is needed. The amount of refinement is based on if any VPLs were actually occluded and how many sets of VPLs have been created (Lines 11 and 15). If no VPLs are deemed visible refinement is continued due to the fact that a light source is active and by its nature there should be some indirect lighting present. Figure 5.3.2 shows the amount of adaptive **ID** refinement for a selection of scenes, the original scenes can be seen in Figure 5.3.5b and Figure 5.3.5a.

Due to the fact that VPLs are utilised this heuristic is very dependant on the VPL distribution as well as the scene geometry. Because the heuristic increases the sampling rate based on VPL occlusion if a bad VPL distribution is generated or there is complex geometry present then a large number of VPLs, potentially all NM of them, will be sampled for each pixel. This can be rectified by making sure that the generation places a sufficient number of VPLs in such a way as to provide a good VPL distribution which takes into account the current viewpoint, such as in Segovia *et al.* [SIP07].

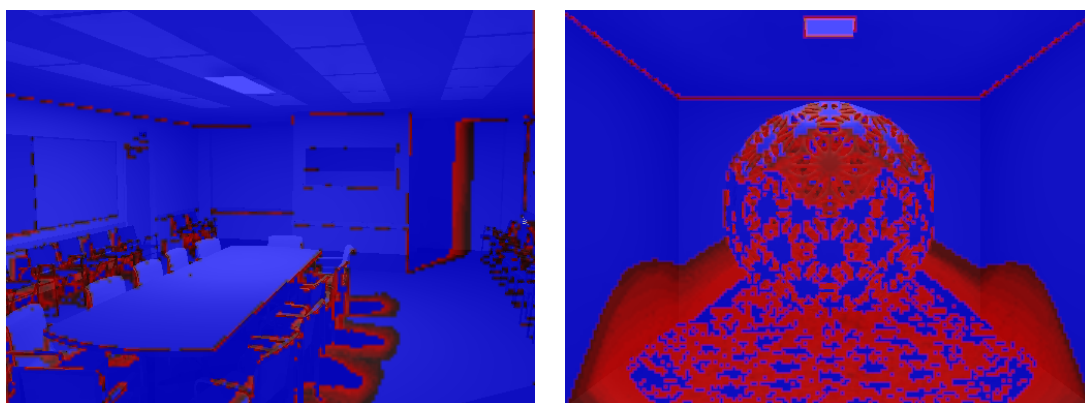
Listing 5.2: ID pseudo code

```

1  getRefinementInformationID() {
2      if (missedVPLs > 0)
3          totalMissedVPLs +=
4              missedVPLs / totalSampledVPLs
5  }
6
7  getAdaptiveGuidanceID() {
8      return(totalMissedVPLs / tile.size())
9  }
10
11 refinementConditionsID(refine) {
12     return(refine > 0.0)
13 }
14
15 getIterationsID(refine) {
16     return(refine * IDmaxSamples)
17 }
```

5.3.3 Soft Shadows

For the soft shadow computation, the sampler stores a distinct set of sample points on the light source for each pixel in the tile ensuring a good overall distribution without duplicating samples between pixels in the tile. As can be seen in

Figure 5.3.3: **SS** adaptive guidance.

Listing 5.3, the **SS** adaptive guidance, similar to the **ID** adaptive guidance, runs an initial pass over the tile calculating the contribution from the light for every pixel while storing how many of the samples on the light are occluded (Line 1). This information, the ratio of occluded samples to total samples taken, is then used to determine how much refinement is needed (Line 9). For any tile where all the pixels are not entirely lit or fully in shadow refinement is applied. The sampling employed allows for early termination of refinement if no samples are lit, unlike Section 5.3.2, as the tile is then considered to be completely in shadow and does not require further computation.

Through empirical observation and VDP comparisons (See Appendix A) it was determined that most errors occurred inside the penumbra of the shadow and this is where the refinement is focused. Figure 5.3.3 shows the level of adaptive **SS** refinement on two scenes (Figure 5.3.5c and Figure 5.3.5c).

For scenes with large area light sources, multiple area light sources and very distant light sources the penumbra of the shadows may make up a large part of the rendered image. This in turn would mean a degradation in performance as large parts of image would require refinement. While this is a limitation of the heuristic it should be noted that a non-adaptive approach, using a fixed number of samples, would simply provide very deteriorated shadows unless a very high sampling rate was utilised. But in the case of a high sampling rate a large number of samples would be wasted in any areas that were fully lit or fully in shadow.

Listing 5.3: SS pseudo code

```

1  getRefinementInformationSS() {
2      for (each light)
3          if occluded(lightSample)
4              missed++
5
6      missed /= numberLights
7  }
8
9  getAdaptiveGuidanceSS() {
10     return(missed / tile.size())
11 }
12
13 refinementConditionsSS(refine)
14 {
15     return(refine > 0.0 and refine < 1.0)
16 }
17
18 getIterationsSS(refine) {
19     return(refine * SSmaxSamples)
20 }

```

5.3.4 Single-scattering Participating Media

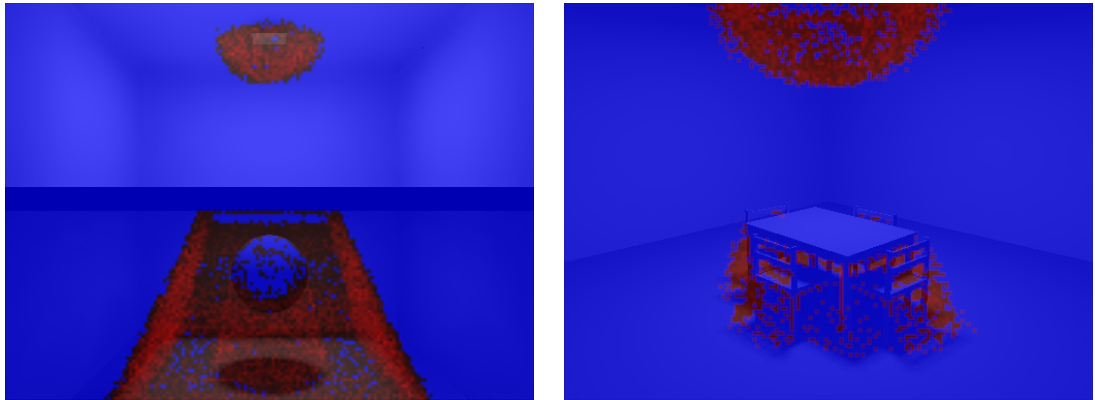


Figure 5.3.4: PM adaptive guidance.

PM is a single-scatter implementation that uses homogenous media. The PM utilises a sampler that returns a starting offset along the ray for that pixel. For a tile of $N \times M$ pixels each one is assigned a segment of space along the ray equal to the step-size of the ray marching divided it into NM equal parts. Each segment is then further equally subdivided according to the maximum amount of refinement

that will take place. Each pixel then utilises the starting point of these segments as offsets along the ray. This ensures that that each pixel in the tile will get a unique starting offset for all iterations, and these will be equally spaced apart, causing each ray to interleave its ray marching guaranteeing superior coverage of the sampling space.

As can be seen in Listing 5.4 the adaptive guidance runs an initial pass on the tile while keeping track of two values. The first pertains to how many sampling points along the ray were occluded from the light source while the ray marching was performed (Line 6) as this detects transitions between lit regions and regions in shadow. The second value records how many of the sampling points along the ray are within a certain distance of the light source. This distance is adjusted by the attenuation co-efficient of the medium (Line 9). This focuses on another prominent effect in PM which is the halo that develops around a light source when it is within a medium.

As each new pixel in the tile is calculated, these two values are combined into a single metric. This value is then compared to the value calculated for the previous pixel. If these two values differ this indicates a need for refinement (Line 14). The comparison with the previous value is arbitrary. The heuristic is trying to identify differences with the participating media contribution for the entire tile. This particular heuristic focuses computation in tiles that are on shadow volume boundaries and around objects within the volumes where the effects of participating media are most apparent. It also focuses on areas that are in the halo produced by any light sources. Figure 5.3.4 shows the **PM** adaptive guidance as applied to two of the scenes (Figure 5.3.5b and Figure 5.3.5d)

Due to the approach of this heuristic, a degradation of performance will occur if the scene contains geometry that creates a large number of irregular shadow volumes that overlap in a specific view. In this case, a large amount of refinement would occur affecting performance. This is a similar limitation to the one exhibited in the SS heuristic in Section 5.3.3.

Listing 5.4: PM pseudo code

```

1  getRefinementInformationPM() {
2      // N = number of steps when ray
3      // marching
4      for I = 1 to N
5          for (each light)
6              if occluded(lightSample)
7                  missed++
8  }
```

```

9         if (distLight < threshold - volume.attenCoef)
10             inHalo++
11
12     missed /= numberLights
13
14     // Combine values
15     value = 0, parts = 0
16
17     if (missed > 0 and missed < N)
18         value += (missed / N)
19         parts++
20
21     if (inHalo > 0 and inHalo < N)
22         value += (inHalo / N)
23         parts++
24
25     if (parts > 1)
26         value /= parts
27
28     // Check against previous
29     if (prevValue != value)
30         different++
31 }
32
33 getAdaptiveGuidancePM() {
34     return(different / tile.size())
35 }
36
37 refinementConditionsPM(refine) {
38     return(refine > 0)
39 }
40
41 getIterationsPM(refine) {
42     return(refine * PMmaxSamples)
43 }

```

5.4 Results

The following results were generated on an 8-core Mac Pro running at 3.2GHz, with a 2GB of memory. Results for eleven scenes are presented. For each scene the average frame rate for each component on its own, as well as an average frame rate for when all the components were being calculated simultaneously, is shown.

A total of eight unique scenes are used for the results, see Figure 5.3.5. The labels in Figure 5.3.5 correspond to how the scenes will be referred to in the text. Subdivided, Office and Shirely6 are also used for the participating media results by adding a participating media volume to those scenes, see Figure 5.3.6.

Each scene was rendered with the novel AIS algorithm and AIS where the

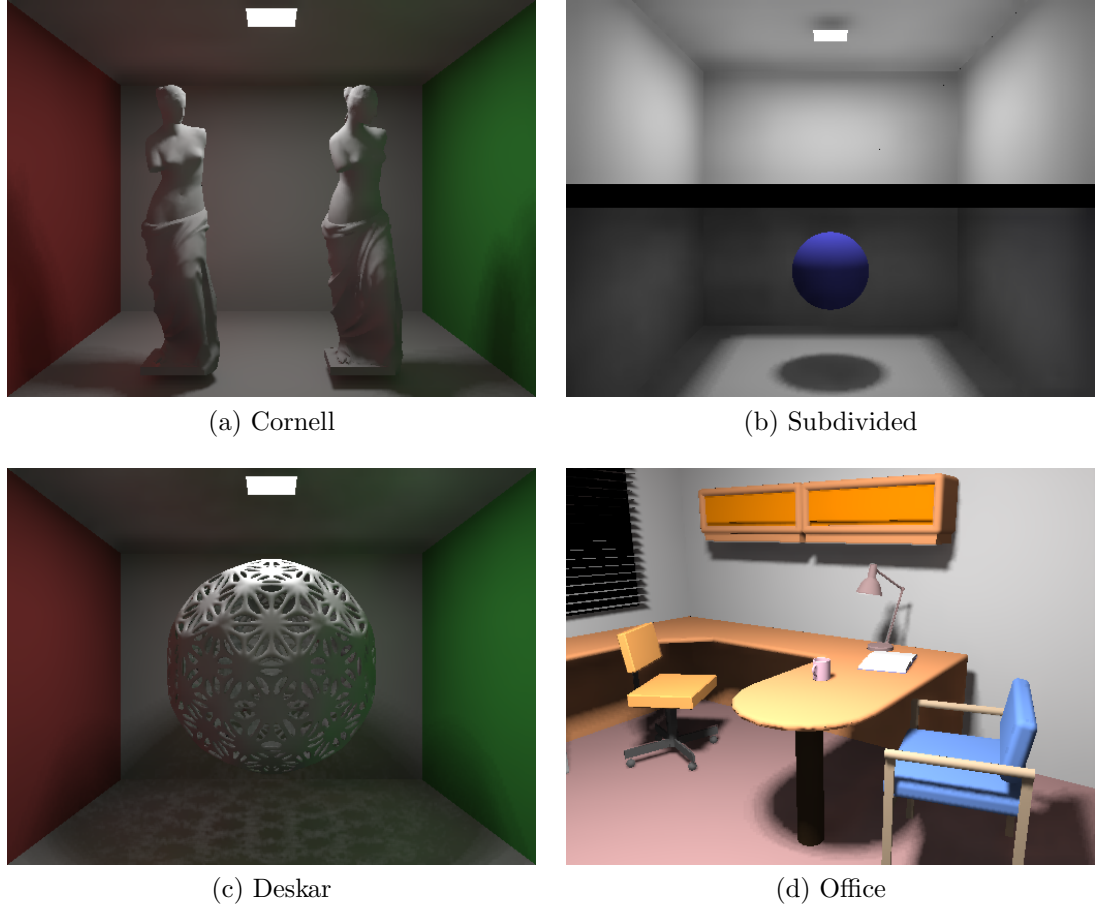


Figure 5.3.5: The scenes used for obtaining AIS results.

maximum amount of refinement possible was performed. These will be referred to as **A** and **A-MAX** respectively. A direction comparison with an IGI implementation based on Wald *et al.* [WKB*02] (referred to as **IGI**) is also provided, for the combined indirect lighting and soft shadow components. For all renderers the tile size was set to 3×3 . Each scene was rendered interactively and 100 frames were utilised in computing the average frame-rate which is presented in the results. For the **ID** scenes 256 VPLs were used. The base value for the initial pass was set to one for each component. **AIS** and **A-MAX** use sub-sampling at a quarter of the resolution, **IGI** is the standard version and does not use sub-sampling.

The specific thresholds and values utilised in the implementation, as pertaining to the heuristics, were as follows (these provided the best visual quality and computational performance). For **ID** the *IDmaxSamples* was set to the number

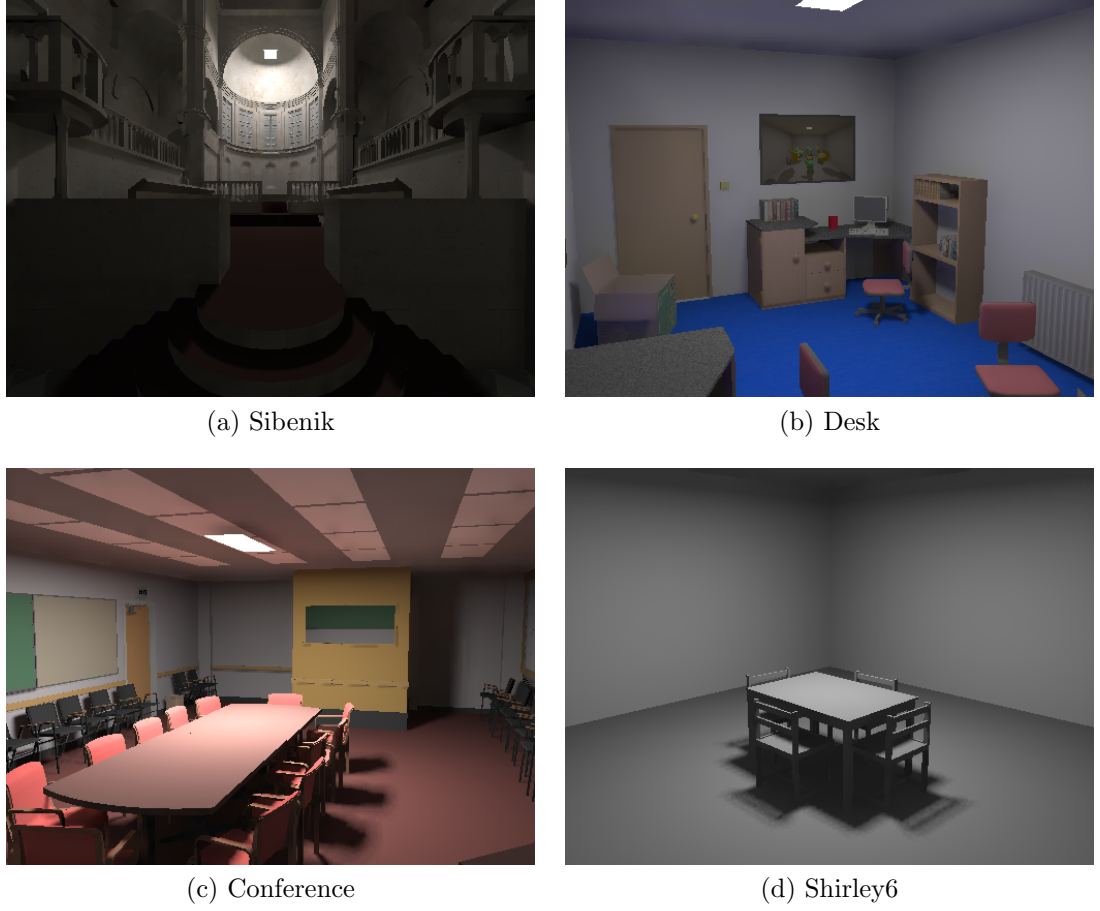


Figure 5.3.5: The scenes used for obtaining AIS results.

of VPL subsets which is 9. For **SS** the *SSmaxSamples* was also 9. In **PM** the distance *threshold* was set to 1.0 as when the distance to the light drops below one so does the divisor of the geometric term and this is where the majority of the PM halo appears. Finally the *PMmaxSamples* for **PM** was also set to 9.

Table 5.1 shows the timing for each of the renderers for the scenes and components in frames per seconds. Table 5.2 shows the same results tabulated as speedups for **AIS** against both **A-MAX** and **IGI**. The average speedup of **AIS** over **A-MAX** is 2.82 and over **IGI** is 3.53. As mentioned in Section 5.3 and Section 5.2.4 while the approach does share some common strategies with IGI significant changes have been made. A higher sampling rate is utilised for light sources as well as the different approach for calculating the indirect lighting solution. A comparison against IGI is included as it is the state-of-the-art interactive CPU-based GI solution.

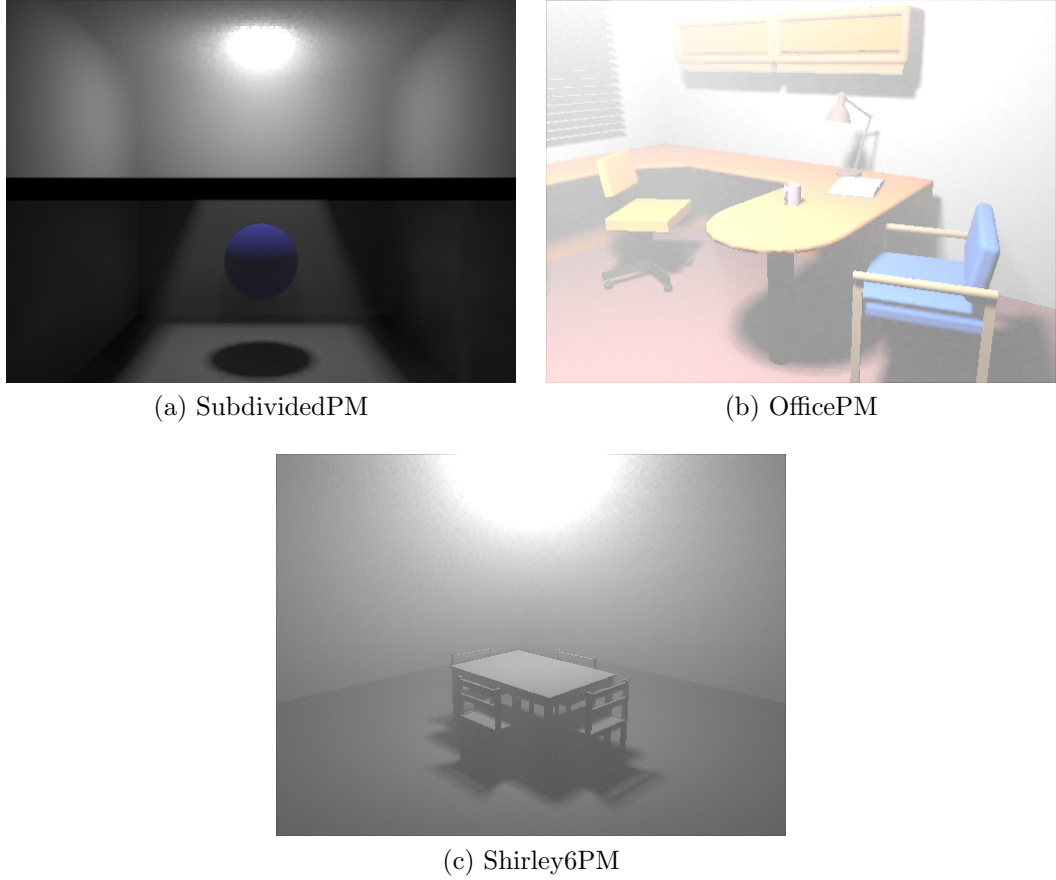


Figure 5.3.6: The scenes used for obtaining AIS participating media results.

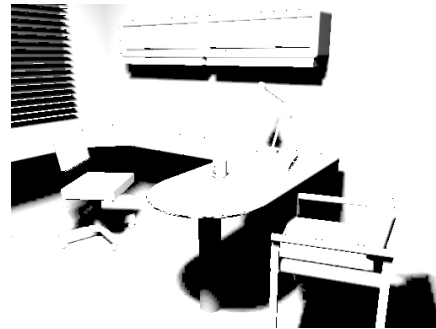
5.4.1 Validation

In order to validate the results, a single image (those shown in Figure 5.3.5 and Figure 5.3.6) was selected and rendered for each scene and validated against a reference path traced (**PT**) image. These images were compared using the High Dynamic Range (HDR) Visual Difference Predictor (VDP) [MDMS05], an HDR version of the VDP psychophysical metric, developed by Daly [Dal93]. HDR-VDP compares two images and identifies the perceptual differences between those two images by taking into account limitations in the human visual system. HDR-VDP results can be summarised by the percentage of different pixels detected with a certain probability. The results for two probabilities are presented: $P(X) \geq 75\%$ and $P(X) \geq 95\%$.

Results for **AIS**, **A-MAX** and **IGI**, for combined components, are shown as computed in the previous section against path traced images computed with



(a) ID



(b) SS



(c) PM



(d) Final



(e) Final PM



(f) Final A-MAX

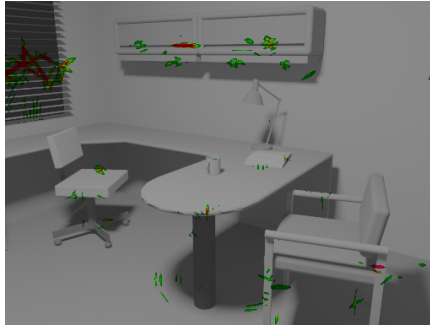


(g) Final A-MAX PM

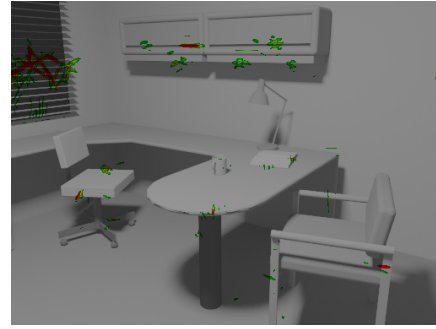


(h) Final IGI

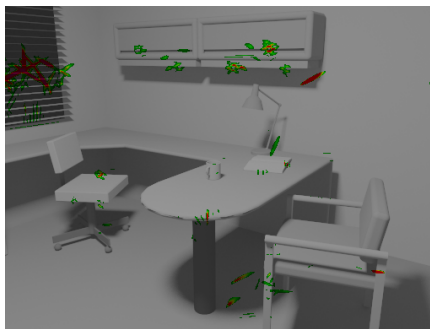
Figure 5.4.1: Office scene (including the PM version).



(a) VDP (AIS vs. PT)



(b) VDP (A-MAX vs. PT)



(c) VDP (IGI vs. PT)



(d) VDP (A-MAX PM vs. PT)



(e) VDP (AIS PM vs. PT)



(f) Office PT



(g) OfficePM PT

Figure 5.4.1: Office scene (including the PM version).

	ID		SS		PM		FULL		IGI
	A	A-MAX	A	A-MAX	A	A-MAX	A	A-MAX	
Cornell	10.17	3.57	9.56	5.24	-	-	6.25	2.35	1.75
Desk	5.29	1.78	5.12	3.46	-	-	3.21	1.27	0.95
Subdivided	15.19	5.05	17.74	8.16	-	-	11.03	3.55	2.89
Desk	9.99	3.11	12.02	4.92	-	-	6.94	2.15	1.71
Sibenik	3.14	1.45	5.03	1.20	-	-	2.45	1.01	0.66
Office	6.95	3.08	8.11	4.10	-	-	5.06	2.03	1.70
Conference	4.04	1.74	8.41	4.12	-	-	3.20	1.35	0.92
Shirley6	14.61	5.11	16.77	7.00	-	-	9.40	3.38	2.85
SubdividedPM	15.19	5.05	17.74	8.16	3.98	1.43	3.12	1.05	-
OfficePM	6.95	3.08	8.11	4.10	6.97	2.40	3.37	1.13	-
Shirley6PM	14.61	5.11	16.77	7.00	5.52	1.76	4.04	1.15	-
Average	9.65	3.46	11.40	5.22	5.49	1.86	5.28	1.86	1.68

Table 5.1: Frames per seconds for the eleven scenes rendered with different components.

	ID	SS	PM	FULL	
	A vs. A-MAX	A vs. A-MAX	A vs. A-MAX	A vs. A-MAX	A vs. IGI
Cornell	2.85	1.82	-	2.66	3.57
Desk	2.98	1.48	-	2.53	3.39
Subdivided	3.01	2.17	-	3.11	3.81
Desk	3.21	2.44	-	3.23	4.07
Sibenik	2.17	4.19	-	2.42	3.68
Office	2.26	1.98	-	2.49	2.98
Conference	2.32	2.04	-	2.38	3.47
Shirley6	2.86	2.40	-	2.78	3.29
SubdividedPM	3.01	2.17	2.78	2.97	-
OfficePM	2.26	1.98	2.90	2.98	-
Shirley6PM	2.86	2.40	3.14	3.53	-
Average	2.71	2.28	2.94	2.82	3.53

Table 5.2: Speedup for the 11 scenes rendered with different components comparing adaptive (**A**) with max adaptive (**A-MAX**) and IGI (**IGI**).

2,500 samples per pixel. Table 5.3 shows the results of the HDR-VDP calculations while an example of the comparison images can be found in Figure 5.4.1. As can be seen there is very little perceptual difference for the images obtained for **AIS**, **A-MAX** and **IGI** compared with the path traced images and the perceptual differences are comparable across the three renderers. These results indicate the

$P(X) \hat{\epsilon}$	AIS		A-MAX		IGI	
	75%	95%	75%	95%	75%	95%
Cornell	0.86	0.41	0.55	0.25	0.80	0.36
Deskcar	1.09	0.57	0.80	0.42	1.26	0.59
Subdivided	1.15	0.62	0.83	0.36	0.81	0.56
Desk	1.86	0.97	1.21	0.74	2.14	1.01
Sibenik	3.52	1.68	2.91	1.38	2.54	1.25
Office	0.41	0.17	0.12	0.03	0.41	0.14
Conference	1.48	0.97	1.46	1.02	1.66	1.11
Shirley6	0.08	0.03	0.02	0.00	0.18	0.06
SubdividedPM	1.90	1.07	1.12	0.69	-	-
OfficePM	0.26	0.11	0.25	0.09	-	-
Shirley6PM	2.10	0.96	1.35	0.57	-	-
Average	1.29	0.71	0.95	0.54	1.17	0.67

Table 5.3: Results for HDR-VDP calculations in %.

speedup demonstrated previously comes at little loss of perceptual quality, on average only 1.29% of the pixels were perceptually different when comparing AIS and the ground truth images. After analysis of the differences that were identified when comparing with ground truth it was found that they can largely be attributed to aliasing in the images (Refer to Appendix A to examine the VDP results for AIS vs. PT) which have been identified as an area for future work, see Section 8.5.

5.5 Summary

This chapter introduced a framework for combining existing interactive ray tracing and interactive global illumination algorithms with selective rendering techniques to form new interactive global illumination solutions. Three core stages in the methodology (identification, deconstruction and pairing) are identified and provide insight into how this process provides a structured approach when attempting to develop a novel interactive global illumination solution. While describing the framework three key criteria namely interactivity, coherence and adaptability are identified. It is shown how each of these three criteria interact and why each one must be considered on a per component basis to ensure maximal use of available computational resources when computing the global illumination solution.

This framework is utilised to introduce a novel global illumination solution

termed Adaptive Interleaved Sampling. By combining interleaved sampling with an adaptive approach, together with efficient component-specific adaptive guidance methods, it is shown how this leads to a significant reduction in computational costs while maintaining the same high perceptual quality of the resultant images. Detailed information is provided on each of the three custom heuristics developed for each of the components that are visualised. The components calculated are soft shadows, indirect diffuse lighting and participating media. It is also shown how this approach is generalisable and can be applied to any component as long as a custom heuristic can be written to guide the computation. A comparison between AIS, standard IGI [WBS02] and gold-standard path-traced images is performed. These results are validated using HDR-VDP [MDMS05], to show that the results are achieved while minimising the difference in perceptual quality.

A framework and an actual implementation, that results in a novel global illumination solution, has been presented. This framework is further used in Chapter 6 to explore other other potential algorithms. This extension shows the framework’s flexibility and the application of stages from Section 5.2 and the criteria specified in Section 5.2.3 to an entirely different algorithm.

CHAPTER 6

Instant Caching for Interactive Global Illumination

Traditionally caching of diffuse interreflections has been utilised primarily in off-line approaches due to its inherent complexity and costly operations such as radiance gathering. Continuing the work from Chapter 5 this chapter further explores the most computationally expensive component, diffuse interreflections, and an alternative novel approach to its computation. The concepts and framework developed in Chapter 5 are used to combine irradiance caching [WRC88], instant radiosity [Kel97] and the ability to identify and update all invalid indirect diffuse light paths resulting from geometric transformations into a novel algorithm. By exploiting both temporal and spatial cache coherence, along with a selective update to reduce computational complexity, interactive frame rates are achieved.

6.1 Introduction

Dynamic scenes, with changing geometry, lighting and materials, represent a challenge for interactive global illumination. Precomputed and caching methods are seldom used in this context because changes in the scene invalidate existing information; correctly updating the cached data is both complex and prone to temporal flickering artifacts. Consequently, certain methods, such as Instant Global Illumination (IGI) [WKB*02], recompute all light paths for every frame, thus ignoring any potential temporal coherence.

This chapter proposes a new interactive caching scheme for indirect diffuse interreflections within dynamic scenes based on exploiting spatial and tempo-

ral coherence enabling interactive global illumination on a single multicore PC. The method, referred to as Instant Caching, is based on computing illumination from virtual point light sources (VPLs), shot in a similar way to instant radiosity [Kel97], and caching the VPLs' contributions for spatial reuse across the same frame, thus exploiting spatial coherence similarly to the irradiance cache [WRC88]. The combination of instant radiosity and irradiance caching improves rendering time upon the former by interpolating over object space and achieves significant speedup upon the latter by having multiple light bounces handled by the VPL shooting stage.

Instant caching naturally adapts to dynamic geometry within an interactive application by exploiting temporal coherence among frames. VPLs are shot using a quasi-random distribution to ensure maximum temporal coherence; cached indirect diffuse light paths invalidated by geometric transformations are identified at each frame and only these need be recomputed. This approach is referred to as Temporal Instant Caching.

This algorithm reduces temporal noise and computational workload (as compared to brute force approaches) by exploiting temporal coherence, thus resulting in an improved frame rate; these results are achieved without the need for any additional data structures or any knowledge of the animation paths. Lighting and material changes are also accommodated within the same approach.

The chapter is organised as follows. Initially, Section 6.2 presents Instant Caching along with its Static (Section 6.2.1) and Temporal (Section 6.2.2) variants. Section 6.3 presents the comparative results of IC and IGI [WKB*02] for both static and dynamic scenarios along with a validation (Section 6.3.3) against a high-quality path-traced solution.

6.2 Instant Caching

Instant caching samples VPLs and caches the irradiance as opposed to using hemispherical sampling, as is the case with irradiance caching. This caching scheme serves a dual purpose accounting for increased performance in the spatial and temporal domains. Firstly, in the spatial domain, it is used to accelerate the computation of each frame by interpolating over object space, following an approach similar to the irradiance cache. Instant caching, however, requires shooting a single ray, referred to as a visibility ray, to evaluate each VPL con-

tribution. Multiple light bounces are handled by the VPL shooting stage. The irradiance cache requires either initiating a tree of rays or originating a random walk to evaluate each stratum contribution; either way, the computation is more than that of an instant cache visibility ray. Some methods have used irradiance caching in conjunction with photon mapping, thus not incurring the multiple bounces described above. However, the cost of extra visibility tests is replaced by photon shooting and density estimation. Additionally, the visibility test in instant caching may be more memory coherent since visibility rays are traced towards the same VPLs.

Secondly, in the temporal domain, updating each VPL contribution to a given cache sample due to scene transformations requires a single reevaluation of the respective visibility ray. Often, given the proposed optimisations (see Section 6.2.2), the visibility ray is intersected with only a subset of the geometry, improving interactive performance due to reuse of information from previous frames.

6.2.1 Static Instant Caching

The static algorithm, applied to the first frame of an animation, entails two steps: shooting photons from light sources and creating VPLs, in a manner similar to instant radiosity and IGI [Kel97, WKB*02], and a subsequent gathering pass. The gathering pass evaluates indirect diffuse irradiance incoming from VPLs at a sparse set of points and interpolates among them for the remaining ones, similarly to the irradiance cache.

The evaluation of the indirect diffuse component using instant radiosity at a point x requires the calculation of the contribution of each VPL, which equates to:

$$L_{indirect}(\mathbf{x}, \boldsymbol{\omega}_o) = \sum_{k=1}^N f_r(\mathbf{x}, \boldsymbol{\omega}_o, \boldsymbol{\omega}_k) L_{e,k} V(\mathbf{x}, \mathbf{y}_k) G'(\mathbf{x}, \mathbf{y}_k), \quad (6.2.1)$$

where N is the number of VPLs in the scene, $V(\cdot, \cdot)$ is the visibility function between two points, $\boldsymbol{\omega}_k$ is the light vector for the k -th VPL, $L_{e,k}$ is the emitted radiance of the k -th VPL, \mathbf{y}_k is the position of the k -th VPL, $f_r(\mathbf{x}, \boldsymbol{\omega}_o, \boldsymbol{\omega}_k) = \rho/\pi$ in the case of diffuse component (ρ is the reflectance), and G' is the bounded geometry term [PH04]. The bounded geometry term is defined as:

$$G'(\mathbf{x}, \mathbf{y}) = \frac{\cos^+ \theta_{\mathbf{x}} \cos^+ \theta_{\mathbf{y}}}{\|\mathbf{x} - \mathbf{y}\|_2^2} f(0.8min_d, 1.2min_d, \|\mathbf{x} - \mathbf{y}\|_2), \quad (6.2.2)$$

where \cos^+ is $\max(0, \cos \theta)$, $\cos^+ \theta_{\mathbf{x}}$ is the cosine of the angle between the light vector $\boldsymbol{\omega}_k$ and normal at \mathbf{x} , $\cos^+ \theta_{\mathbf{y}}$ is the cosine of the angle between $\boldsymbol{\omega}_k$ and the normal at \mathbf{y}_k , \min_d is the bounding distance, and f is the smoothing function:

$$f(a, b, x) = \begin{cases} 1 & \text{if } x > b \\ 3\left(\frac{x-a}{b-a}\right)^2 - 2\left(\frac{x-a}{b-a}\right)^3 & \text{if } a \leq x \leq b \\ 0 & \text{otherwise.} \end{cases} \quad (6.2.3)$$

Equation 6.2.1 is quite expensive to evaluate. To speed-up this calculation, caching, similar to Ward [WRC88], can be employed, where a set of cached samples, Ω , are used for interpolation:

$$L_{\text{indirect}}(\mathbf{x}, \boldsymbol{\omega}_o) \approx \frac{\rho}{\pi} \sum_{k \in S(\mathbf{x})} \frac{E_k w_k(\mathbf{x})}{\sum_{k \in S(\mathbf{x})} w_k(\mathbf{x})}, \quad (6.2.4)$$

where $S(\mathbf{x}) = \{k | k \in \Omega \wedge w_k(\mathbf{x}) > 1/a\}$, a is the caching radius, E_k is the cached irradiance for the k -th sample in the cache, and w_k is the weight for the k -th cached sample used which is calculated as:

$$w_k(\mathbf{x}) = (\|\mathbf{x} - \mathbf{x}_k\| + \sqrt{1 - (\mathbf{n} \cdot \mathbf{n}_k)})^{-1}, \quad (6.2.5)$$

where \mathbf{n} is the normal at \mathbf{x} , \mathbf{x}_k and \mathbf{n}_k are respectively the position and normal of the k -th sample. This metric differs from the original irradiance cache [WRC88], because the harmonic mean distance has been removed from the calculation of $w_k(\mathbf{x})$. The harmonic mean distance, when used with hemisphere sampling, is a value that gives a measure of the density of the geometry surrounding a given point. When using VPLs the visibility rays will seldom be well distributed over the hemisphere, thus the harmonic mean of the rays' length does not necessarily give an indication of the surrounding geometry density. This can be illustrated in the case of a point close to a corner, the computation of the harmonic mean distance with the well-distributed VPLs would not necessarily indicate the proximity to a corner. This rationale is illustrated in Figure 6.2.1. The assumption were further confirmed by the lack of any perceivable differences in quality between images computed with and without a harmonic mean. HDR-VDP comparisons [MDMS05] (HDR-VDP is a perceptual metric; see Section 6.3.3 for a detailed explanation) have been run using images generated with and without the harmonic mean for all the scenes shown in Figure 6.3. The greatest perceptual

difference recorded was 0.1% and for most images the result was 0.0%.

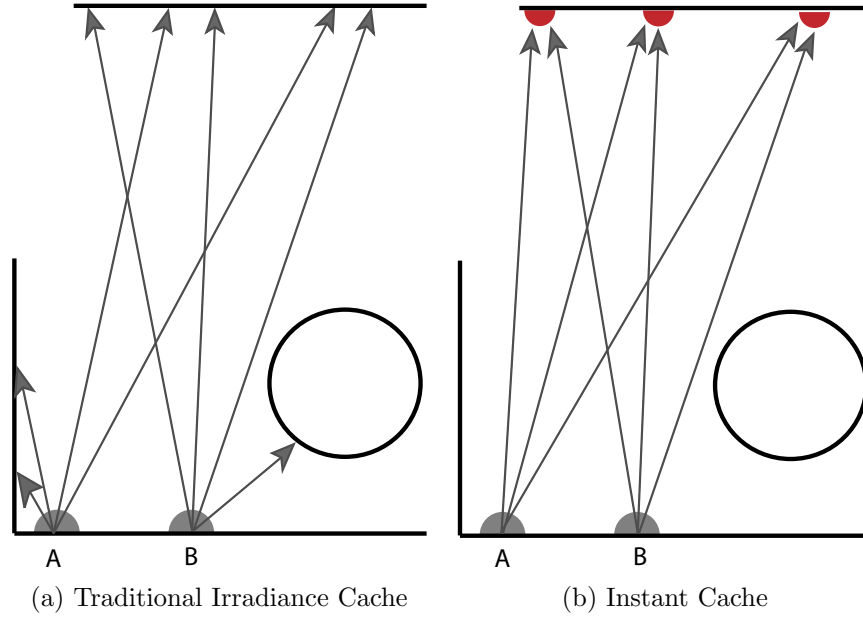


Figure 6.2.1: A comparison between the irradiance cache and the instant cache. In the irradiance cache, the harmonic mean distance for the cached sample A will be different from that in B, possibly resulting in further samples near the corner. In the instant cache, the harmonic mean distance would be independent from the geometry context but dependent on the VPLs distribution; in this particular example the harmonic mean of A and B is similar.

6.2.2 Temporal Instant Caching (TIC)

To maintain the correctness of the instant cache samples one needs to account for when objects move and invalidate the instant caching samples accordingly. The issues that affect the fidelity of the cache samples occur when an object occludes the path of a visibility ray or VPL, or when an object moves away from the visibility ray or VPL, commonly called deocclusion. Finally, when cached samples lie on dynamic objects the computation may be invalid. These possible situations are summarised into five cases as shown in Figure 6.2.2. Case 1 and Case 2 demonstrate the case when VPLs are occluded or deoccluded by moving objects. Case 3 and Case 4 demonstrate the cases when the visibility rays are occluded or deoccluded by dynamic objects. Case 5 demonstrates the case when the cache sample lies on a dynamic object.

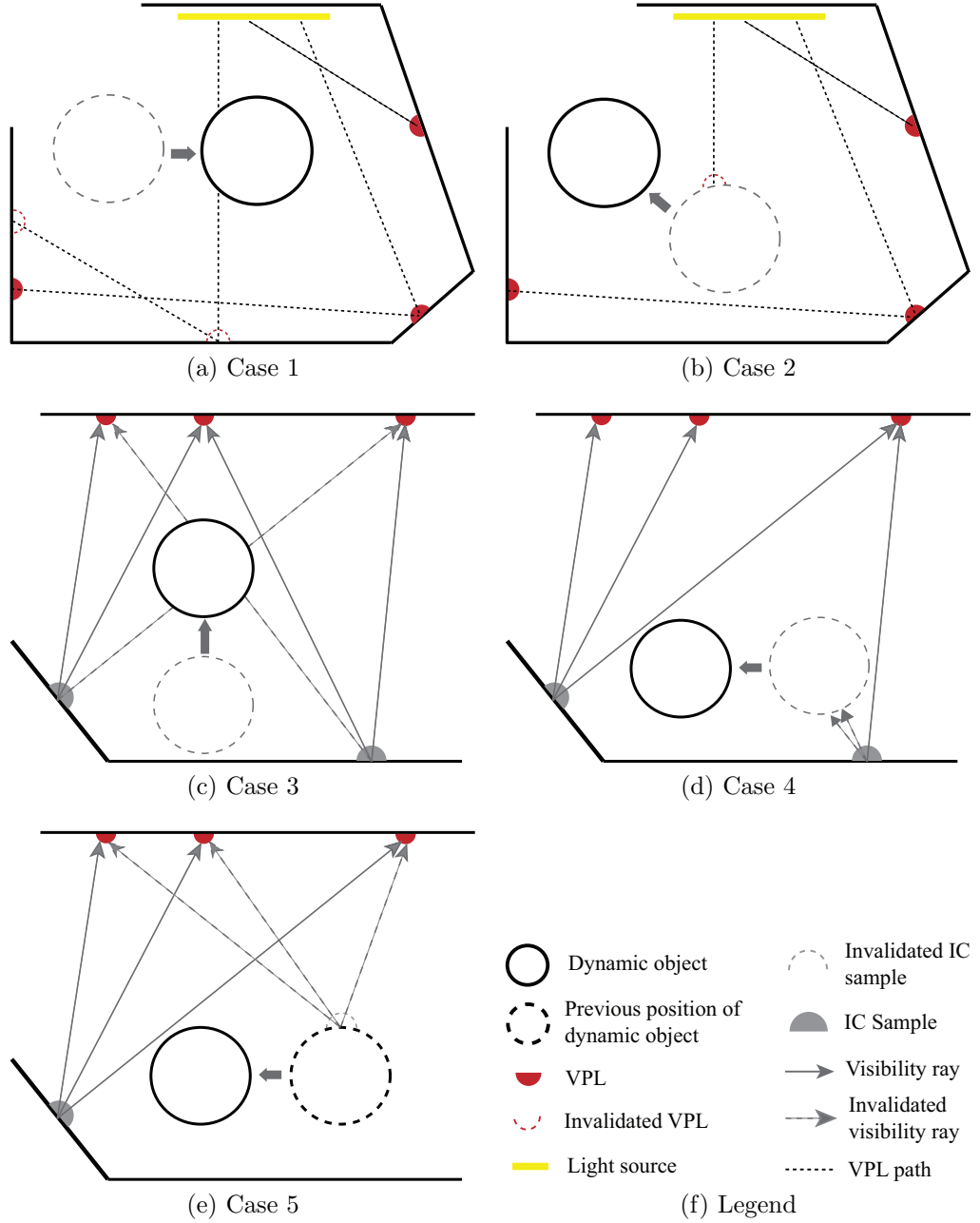


Figure 6.2.2: The five cases that trigger invalidation or updating of samples in the instant cache.

The instant cache maintains all the cached samples within an octree representation. These samples are placed only on diffuse surfaces and store a position, orientation, irradiance and an array for each VPL representing the contribution to each of the visibility rays. Orti *et al.* utilised a similar approach for accelerating

the computations of a radiosity solution in a dynamic environment [ORDP96]. This approach was more complex as it involved storing visibility information between patches and not simply points.

While this may seem like a large amount of memory, typically 256 VPLs are sufficient. Furthermore, a format similar to RGBE [War91a] can be employed for storage, resulting in four bytes per visibility ray.

In order to maintain the strong coupling between the visibility rays and VPLs, the VPLs are traced at the beginning of every frame using Quasi-Monte Carlo sampling. This ensures that the samples are always valid when nothing moves and coherence is maintained. It also makes it easy to identify which VPLs have been affected by moving objects. A simple comparison between new and old VPLs is enough to invalidate the visibility ray corresponding to those VPLs for each of the cached samples. This applies also for multiple VPL bounces. This handles both Case 1 and Case 2. The VPLs shot from light sources that have moved are naturally invalidated in this way.

Case 3 can be identified by a visibility test of the moved objects with the cached samples. While projection onto each cached sample could be used to identify objects that have moved, for simplicity and to avoid GPU access, visibility rays from the cached samples towards the corresponding VPLs are shot using ray tracing. Tests are performed only against the dynamic objects. This process could be computed against each of the objects individually or an acceleration data structure could be built just for the dynamic objects. If the VPL is occluded then its contribution is set to zero.

For Case 4, deocclusion, a more sophisticated approach is required. Since the method allows any object in the scene to be moved, deocclusion would potentially test against all objects in the scene. However, a number of optimisations are used to minimise the number of times tests need to occur for this case. Deocclusion only affects visibility rays which were previously occluded (in shadow). The first test selects those visibility rays for which the contribution on the previous frame was zero. For these, a further test with the previous position of the dynamic objects identifies those visibility rays that may need updating: if a ray does not hit any of the dynamic objects at their previous positions, then it must have struck a static object and no updating is required. The remaining visibility rays may potentially hit a VPL or may be occluded by other geometry. The visibility ray is then tested against the entire scene to ensure it is not occluded by a static object. If no occlusion occurs the new value for this visibility ray is

computed. The use of these multiple tests has a further advantage. They make it possible to test the first deocclusion against the old position of the dynamic object's bounding box. While this test is more conservative, it is ideal for when the object's vertices are changing, such as with deformable objects. A further optimisation that reduces the number of objects to be tested occurs if Case 3 is run immediately before Case 4. If a visibility ray is occluded then Case 4 need not be tested for that visibility ray. Finally, Case 5 is dealt with by identifying and removing cached samples within the bounds of the dynamic objects in the previous frame.

When identified, invalidated visibility rays can either be reevaluated immediately or flagged. Reevaluation requires removing the previous visibility ray's contribution to the cached sample and adding the new one. On the other hand, flagged cached samples can be updated on demand the moment an interpolation is required from them. Object space areas that are without cached samples in a new frame, because of such areas becoming visible or deletion of cache samples (due to Case 5) are naturally recomputed on demand by the algorithm in the traditional way.

The method does not need to know beforehand which objects are dynamic and which are not. The dynamic objects need only be identified in the frame in which they are moved. The only aspect that needs to be stored is their position in the previous frame. This feature makes the method well suited for interaction since any object could be moved at any time. As the number of dynamic objects increases, VPLs end up being created at different locations every frame, requiring tracing the corresponding visibility rays from each cache sample against the whole geometry. Additionally, even for those VPLs whose location does not change, the corresponding visibility rays have to be tested against an increasing number of dynamic geometric primitives, eventually approaching the totality of the scene. The performance of the algorithm degrades to that of the static instant cache.

Algorithm 1 demonstrates how all the tests are integrated in a single algorithm, in the case of immediate reevaluation, run at the beginning of each frame. Tracing rays against the entire scene is required when a given VPL is invalidated and for special cases as part of the Case 4 test. Case 1 and Case 2 are detected on Line 1 and updated for each VPL from Line 8. Case 5 is handled on Line 3. For Case 3 and Case 4, the tests can be combined into a block test, see Line 11 to Line 21. In this case the optimisation mentioned above for Case 4 can be applied when Case 3 is tested before Case 4. The order of the operations in the

Input: Instant Cache: IC, The VPLs: VPLs; The entire scene: Scene, The dynamics objects that have moved: dObjs

```

/* Function to trace ray across objs returning colour and a hit point
   hit                                                                    */
Function: hit = Trace(ray, objs, colour);
/* Function to recalculate the index visibility ray for ICsample
   based on colour                                                         */
Function: ReEvaluate(ICsample, index, colour);
1 InvalidateVPLs(); // Case 1 and Case 2
2 foreach ICsample in IC do
3   if ICsample.pos within bounds of previous position of dObjs then
4     Invalidate ICsample and remove from IC;
      // Case 5
5   else
6     foreach visibility ray in ICsample do
7       if VPLs[VPLindex] invalidated then // Case 1 and Case 2
8         Trace(visibility ray, Scene, Col);
9         ReEvaluate(ICsample, VPLindex, Col);
10      else // Case 3 and Case 4
11        hit = Trace(visibility ray, dObjs, Col);
12        if hit then // Case 3
13          ReEvaluate(ICsample, VPLindex, 0.0);
14        else // Case 4
15          if old visibility ray was in shadow then
16            hitOld = Trace(visibility ray, previous position of dObjs,
17                           Col);
18            if hitOld then
19              if !Trace(visibility ray, Scene, Col) then
20                ReEvaluate(ICsample, VPLindex, Col);
21              end
22            end
23          end
24        end
25 end

```

Algorithm 1: Integrating the invalidation (and possible update) into a single algorithm.

algorithm attempts to reduce the cost of the deocclusion operations which are more expensive. In particular the costly testing of the visibility ray against the entire scene in Line 18, is left as the final test, only executed when a number of conditions are not met, improving the overall performance.

Changes on the materials' BRDFs are naturally handled by the algorithm.

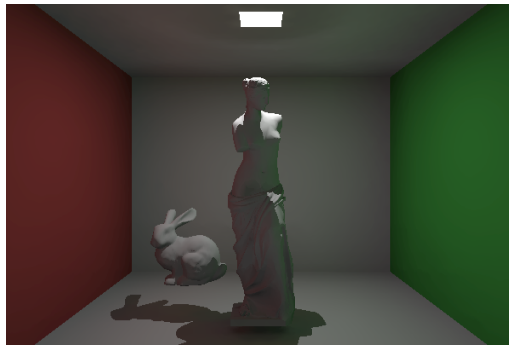
During the VPLs shooting stage a different BRDF will result in a different spatial distribution or in a different radiant flux for some of the VPLs. VPLs might thus be placed on different locations in the scene and/or their radiant flux might change. Both cases imply invalidation of the respective VPLs from the previous frame and are thus handled in the same manner as Case 1 or Case 2. Material changes on objects where cached samples lie do not require any updating since irradiance, not reflected or transmitted radiance, is cached. However, changes of non-Lambertian properties on objects directly visible from the eye are not handled by the algorithm, since this is currently used to shade diffuse surfaces only.

6.3 Results

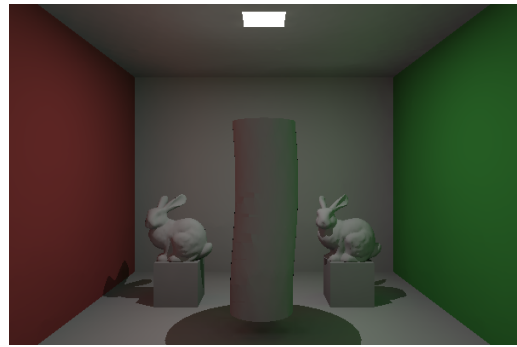
In order to demonstrate the effectiveness of instant caching some comparative results are presented. Besides implementing the instant caching algorithm, IGI [WKB*02] was also implemented. The acceleration structure used by the ray tracing kernel is a BVH implementation based on the method in Wald *et al.* [WBS07]. None of the implementations makes use of packetisation or SIMD operations except for the ray-bounding volume intersection within the BVH traversal [WBS07]. Included are the results achieved when using Whitted-style ray tracing to give an indication of the ray tracing performance. For Whitted-style ray tracing diffuse interreflections are not computed but specular interreflections are along with direct lighting. All images are rendered at a resolution of 600×400 with no super/sub-sampling. Direct lighting is computed for hard shadows only (one shadow ray per light source). IGI and instant caching shoot 256 VPLs. No form of precomputation was used in any of the timings. All results were generated using the scenes shown in Figure 6.2.3 on an eight core (dual socket quad-core at 3GHz) with 2GB memory running Mac OS X.

6.3.1 Static images

Table 6.1 presents results for the computation of a single image, in order to demonstrate the performance of the static instant cache. Results include rendering times for Whitted ray tracing (**RT**), instant global illumination (**IGI**) [WKB*02] and instant caching (**IC**). Instant caching is the fastest of the tested implementations, achieving a speedup between 1.49 to 3.62 relative to IGI, even



(a) Cornell Box



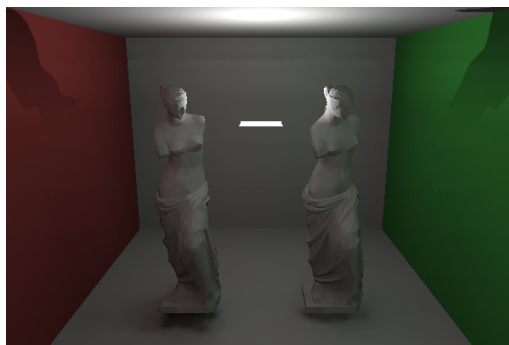
(b) Wobble



(c) Office



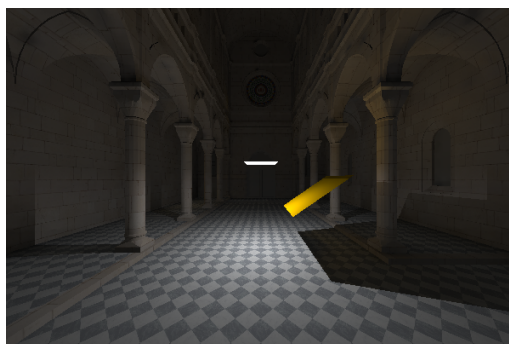
(d) Bunnies



(e) Cornell Indirect



(f) Conference



(g) Sibenik



(h) Desk

Figure 6.2.3: The scenes used for all experiments.

<i>Scenes</i>	RT	IGI	IC	Speedup
Cornell Box	0.05	0.59	0.21	2.81
Wobble	0.06	0.62	0.19	3.26
Desk	0.07	0.76	0.21	3.62
Bunnies	0.08	0.72	0.33	2.18
Cornell Indirect	0.06	0.77	0.24	3.21
Conference	0.14	1.58	0.45	3.51
Sibenik	0.19	1.27	0.85	1.49
Office	0.09	0.78	0.23	3.39

Table 6.1: Results, in seconds, for rendering the first frame. Each image was rendered from scratch without relying on any temporal coherence. Speedup compares IC vs. IGI.

<i>Scenes</i>	IGI	IC	Speedup
Cornell Box	0.43	0.08	5.38
Wobble	0.37	0.05	7.40
Desk	0.56	0.06	9.33
Bunnies	0.57	0.11	5.18
Cornell Indirect	0.55	0.08	6.87
Conference	1.23	0.15	8.27
Sibenik	0.97	0.38	2.55
Office	0.51	0.05	10.02

Table 6.2: Results, in seconds, for the diffuse interreflections only when rendering the first frame. Each image was rendered from scratch without relying on any temporal coherence. Speedup compares IC vs. IGI.

though the latter exploits image space coherence by using only a subset of the VPLs per pixel. The improvement in performance is more obvious when viewing the results of diffuse interreflections only; Table 6.2, shows the timings for these computations and for the same images. In this case for most scenes the speedup is greater than 5. This improvement is due to the exploitation of object space coherence via the interpolation and is expectable in approaches based

<i>Animation</i>	RT	IGI	IC	TIC	Speedup
Cornell Box	17.07	2.11	6.01	10.45	4.95
Wobble	15.84	1.63	2.79	8.95	5.49
Desk	12.33	1.31	4.92	8.94	6.82
Bunnies 1	13.02	1.49	3.42	9.13	6.13
Bunnies 2	12.96	1.45	3.55	8.04	5.54
Bunnies 4	12.81	1.46	3.51	5.81	3.98
Bunnies 9	12.69	1.47	3.54	3.56	2.42
Conference	6.96	1.54	3.37	7.78	5.05
Sibenik	8.23	0.70	1.25	2.63	3.76

Table 6.3: Results, in frames per seconds, for rendering the animations. Speedup compares TIC vs. IGI.

on caching and interpolation. The exploitation of coherence will further benefit instant caching when used in the temporal domain (see next section).

6.3.2 Animations

Results are presented for temporal instant caching with a number of animations. The animations are comprised of different objects moving around the scenes. For the Cornell Box, the bunny is rotated around the Venus statue. The deforming cylinder (Wobble) demonstrates that the proposed method can handle deforming objects; a sine wave is used for the deformation. For the Desk scene a chair moves along the floor. For the Bunnies scene results are presented for 1, 2, 4 and 9 bouncing bunnies, illustrating how the algorithm scales as the number of dynamic objects increases. For the Conference scene a green bunny moves across the table. Finally, for Sibenik a yellow rectangular object moves within the scene.

Table 6.3 shows the results, in frames per second, for rendering animations with Whitted ray tracing (**RT**), instant global illumination (**IGI**), static instant caching (**IC** - without temporal coherence, the cache is cleared after each frame), and temporal instant caching (**TIC**). These results show that temporal instant caching is between 2.42 and 6.82 times faster than IGI. Furthermore, the results show how temporal instant caching is faster than static instant caching. The performance of temporal instant caching decreases when there is not much temporal

<i>Animations</i>	IGI	IC	TIC	S_{IGI}	S_{IC}
Cornell Box	0.43	0.09	0.03	14.33	3.00
Wobble	0.37	0.09	0.03	12.33	3.00
Desk	0.56	0.09	0.03	18.67	3.00
Bunnies 1	0.56	0.12	0.03	18.67	4.00
Bunnies 2	0.55	0.13	0.04	13.75	3.25
Bunnies 4	0.55	0.13	0.05	11.00	2.60
Bunnies 9	0.57	0.12	0.09	6.33	1.33
Conference	1.23	0.17	0.05	24.60	3.40
Sibenik	0.97	0.42	0.05	19.40	8.40

Table 6.4: Rendering times, in seconds, averaged over 100 frames, for rendering the diffuse interreflections only. Speedup is shown for TIC vs. IGI (S_{IGI}) and TIC vs. IC (S_{IC}).

coherence to maintain, such as in the case of the multiple bouncing bunnies. The IGI and static instant cache remain constant as the number of dynamic objects increases. As expected, the performance of the temporal instant caching reduces to that of the static instant cache as this number increases (compare Bunnies 1 with Bunnies 9).

Since the proposed caching technique aims at speeding-up diffuse interreflections, Table 6.4 demonstrates the timings, in seconds, for this lighting component only, for IGI, IC and TIC. The comparison between static and temporal instant caching clearly shows the efficiency of temporal coherence exploitation, with speedups of 1.33 to 8.4 times faster than instant caching. The worst speedup is obtained with Bunnies 9, where given the large number of dynamic objects there is not much temporal coherence to be exploited. The differences in speedup relatively to the values presented in Table 6.3 are due to constant costs, such as primary rays, direct illumination and image display.

6.3.3 Validation

In order to validate the results temporal instant caching is compared with IGI and a path traced image (shot with 2,500 samples per pixel) using the perceptual metric HDR-VDP [MDMS05]. The 100th frame was selected for the comparisons

VDP Results	$P(X) > 75\%$		$P(X) > 95\%$	
	PT	IGI	PT	IGI
Cornell Box	0.57	0.00	0.31	0.00
Wobble	0.13	0.00	0.05	0.00
Desk	2.11	0.02	1.01	0.01
Bunnies 9	0.21	0.00	0.06	0.00
Cornell specular	0.91	0.00	0.43	0.00
Cornell Indirect	1.86	0.33	0.88	0.19
Conference	1.61	0.16	0.91	0.06
Sibenik	0.18	0.06	0.03	0.00
Office	0.45	0.23	0.00	0.00

Table 6.5: Results of the HDR-VDP comparison between temporal instant caching and path tracing (PT), and temporal instant caching and IGI. Numbers indicate the percentage of pixels that might be perceived as different by a human observer with a probability larger than 75% or 95%, respectively.

of the animated scenes. HDR-VDP identifies the perceptual differences between two HDR images by taking into account limitations of the human visual system. HDR-VDP generates a certainty map detailing for each pixel the probability that a human observer will detect a difference. This may be visualised as an image that highlights the differences between the two images in false colour, with the areas most likely to be considered perceptually different in red, those less noticeable in green and those not considered perceptually different at all in grey. Figure 6.3.1 visualises the HDR-VDP results in false colour for the Office scene. Furthermore, the results can be summarised in a single numeric value that gives the percentage of pixels out of the entire image that are likely to be perceived as different within a given probability. Results are presented for $P(X) \geq 0.75$ and $P(X) \geq 0.95$, which indicate the percentage of pixels that are likely to be detected with a probability greater than or equal to 0.75 and 0.95 respectively. Table 6.5 shows results when comparing temporal instant caching with a path traced image and with IGI. When comparing with IGI the results showed no more than a 0.33% difference. This shows that although instant caching results in an improvement in performance there is no perceived loss in quality when compared

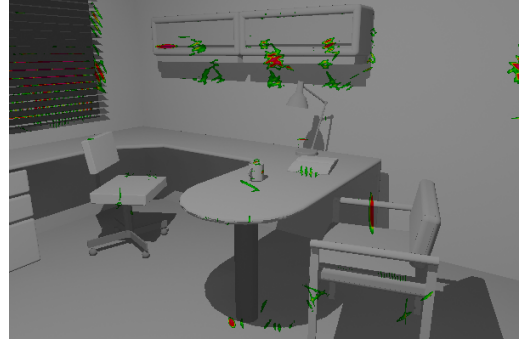
to IGI. Furthermore, as can be seen in the results, there is very little perceptual difference when compared to path traced images, that require several minutes of computation for each frame on modern multicore machines, with no more than 2.11% difference. It also must be noted that the temporal instant caching and IGI shoot only one ray per pixel while the path tracing's 2,500 samples per pixel reduce aliasing considerably and will account, in part, for the perceptual error detected, see Figure 6.3.1.

6.4 Summary

This chapter has demonstrated a caching scheme for accelerating ray tracing with indirect diffuse interreflections to interactive rates. This is achieved for dynamic scenes with no pre-computation where the materials, geometry and camera can change. The algorithmic design has made it straightforward to extend spatial coherence into the temporal domain. The temporal coherence used for temporal instant caching makes it possible to avoid computation that is wasteful by utilising selective techniques to update and compute only what is needed. The use of spatial and temporal coherence allows us to maintain high frame rates; however, when little temporal coherence remains, the spatial coherence is typically enough to maintain reasonable frame rates which are competitive with some of the best CPU methods such as IGI. Furthermore, in experiments, it was shown that the new method can be up to $24.6\times$ faster than IGI and has no perceivable loss in visual quality. At most 0.33% of pixels could be detected as perceptually different, with a 95% certainty, when comparing IGI and IC.



(a) Path traced



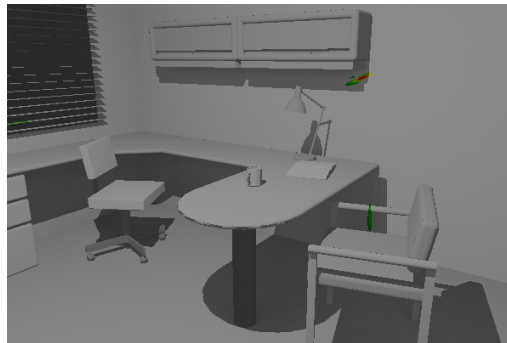
(b) VDP TIC vs. PT



(c) TIC



(d) IGI



(e) VDP TIC vs. IGI

Figure 6.3.1: HDR-VDP comparisons for the Office scene. Areas most likely to be considered perceptually different in red, those less noticeable in green and those considered perceptually equivalent in grey. Note that many errors appear around blinds, which are primarily caused due to the IGI and TIC using only one sample per pixel. The path traced image has many more samples per pixel (2,500 spp) which removes aliasing artifacts.

CHAPTER 7

Wait-Free Shared-Memory Irradiance Cache

This chapter further examines irradiance caching due to its viability as an interactive global illumination algorithm, as demonstrated in the previous chapter. In the previous chapter access into the irradiance cache was facilitated via a locking mechanism, as shown in this chapter this does not to scale when dealing with a large number of threads due to an increase in serialisation and contention, and is therefore not optimal. As was discussed in the Framework, in Chapter 5, parallelism is a key component in ensuring the ability of an algorithm to successfully reach and improve interactive frame rates. Based on this view, a wait-free algorithm which allows concurrent access by all threads to a shared irradiance cache without any locks or critical sections is presented.

7.1 Introduction

Efficient access to shared data structures becomes an important issue with the ever increasing number of processors available in local multi-core systems, with the potential to strongly impact rendering performance. Effective sharing of the irradiance cache in multithreaded systems is mandatory in order to achieve high efficiency levels, since computed irradiance values become readily available to all threads, thus avoiding work replication. This is especially relevant as the utilisation of the irradiance cache has increased significantly over the last few years, e.g., as a stand-alone algorithm for computing indirect (ir)radiance [SKDM05, TL04, KGPB05], as an acceleration data structure for rendering participating media phenomena [JDZJ08] or used with photon mapping [Jen01].

Listing 7.1: Traditional sequential IC

```

1  IrradianceCache IC;
2
3  ComputeIndirectDiffuse() {
4      //get irradiance from IC if there are valid records
5      inIC = IC.getIrradiance ();
6      if (!inIC) { // no valid records found
7          // compute it by sampling the hemisphere
8          ICsample = ComputeIrradianceRT ();
9          // insert new IC sample into the octree
10         IC.insert (ICsample);
11     }
12 }
13
14 IrradianceCache::getIrradiance() {
15     <Traverse the octree searching for valid records>
16     if (found) return true;
17     else return false;
18 }
19
20 IrradianceCache::insert (ICsample) {
21     // recursively traverse the octree
22     // starting at root
23     IC.root.insert (ICsample);
24 }
25
26 ICNode::insert (ICSample) {
27     if (correct insertion node) {
28         IClist.Add (ICsample);
29     } else {
30         // go deeper in the octree
31         xyz = EvaluateOctant();
32         if (children[xyz] == NULL)
33             children[xyz] = new ICNode ();
34         children[xyz].insert (ICsample);
35     }
36 }
37
38 ICList::Add (ICsample) {
39     // insert new record in head of list
40     IClist.records[head++] = ICsample;
41 }

```

This chapter proposes an efficient wait-free algorithm which allows concurrent access to a shared IC by all threads without using any locks or critical sections. This approach is then compared with two other mechanisms which share the irradiance cache among threads on a shared memory system. The first is based on traditional locking techniques and locks the shared IC every time a thread

accesses it, both for reading and writing. The second is a local copy method which avoids concurrent access control by maintaining a local IC, per thread, and merging at the end of each frame.

This chapter is structured as follows. Section 7.2 presents the algorithms for the three data access control mechanisms: Lock-Based, Local-Write and Wait-Free (Sections 7.2.1 - 7.2.3). Section 7.3 details a number of experiments utilising both static images and animations comparing all three approaches running on a varying number of threads.

7.2 Algorithms

In this section the algorithms for the three evaluated data access control mechanisms are presented. To begin with traditional single-threaded irradiance cache with no access control is shown in Listing 7.1. Subsequent sections demonstrate how the traditional approach can be modified to enable the different access control algorithms.

Listing 7.2: Lock-based IC

```

1  ComputeIndirectDiffuse()
2  {
3      //get irradiance from IC if there are valid records
4      IC.lock();
5      inIC = IC.getIrradiance ();
6      IC.unlock();
7
8      if (!inIC) { // no valid records found
9          // compute it by sampling the hemisphere
10         ICsample = ComputeIrradianceRT ();
11         // insert new IC sample into the octree
12         IC.lock();
13         IC.insert (ICsample);
14         IC.unlock();
15     }
16 }
```

7.2.1 Lock-Based Irradiance Cache (LCK)

The lock-based access control algorithm locks the IC whenever a read or write is made to it. However, the code responsible for hemisphere sampling, `ComputeIrradianceRT()`, is not a critical region (Listing 7.2 lines 4 - 6, 12 - 14), thus allowing

concurrent evaluation of irradiance. The primary disadvantage of the LCK approach is that it serialises all accesses, both reads and writes, to the shared IC. As the number of threads increases so does contention preventing performance from scaling appropriately with the degree of parallelism. Table 7.1 shows that the overhead associated with locks (time spent waiting to enter critical regions summed over all threads) increases substantially when going from two to eight threads.

Listing 7.3: Local-Write IC

```

1  IrradianceCache IClocal[number threads], ICglobal;
2
3  ComputeIndirectDiffuse()
4  {
5      //get irradiance from IC if there are valid records
6      inIC = ICglobal.getIrradiance ();
7
8      if (!inIC)
9          inIC = IClocal[current thread].getIrradiance ();
10
11     if (!inIC) { // no valid records found
12         // compute it by sampling the hemisphere
13         ICsample = ComputeIrradianceRT ();
14         // insert new sample into the local cache
15         IClocal[current thread].insert (ICsample);
16     }
17 }
```

7.2.2 Local-Write Irradiance Cache (LW)

An alternative approach, not dependant on locking the data structure each time it is modified, is to have a global IC readable by all threads and an additional local IC per thread; each thread writes only on its local IC but reads from both. At certain predefined execution points, such as the end of a frame, the local ICs are sequentially merged onto the global IC. This form of synchronisation uses an end of frame as a barrier, effectively this is a blocking approach to synchronisation.

The major drawback of this approach is that it does not allow for any sharing of IC samples within a single frame, thus resulting in work replication; this is reflected in Table 7.1, where the LW algorithm has a much higher IC sample count than the other two approaches. The time taken to sequentially merge the caches is not significant, as can be seen in the overheads section of Table 7.1 (at

least up to eight threads). Additionally, memory consumption is dictated by the number of threads being used and the complexity of the octree itself.

Listing 7.4: Wait-Free IC

```

1  ICNode::insert (ICSample) {
2      if (correct insertion node)
3          IClist.Add (ICsample);
4      else { // go deeper in the octree
5
6          xyz = EvaluateOctant();
7          if (children[xyz]==NULL) {
8              temp = new ICNode()
9              temp.insert (ICsample);
10             // Update new branch into the octree
11             // This only occurs on the first level of
12             // recursion subsequent levels just insert
13             // normally.
14             if (!CAS (children[xyz], NULL, temp))
15                 free temp;
16         }
17         else
18             children[xyz].insert (ICsample);
19     }
20 }
21
22 ICList::Add (ICSample) {
23     // get index of new sample in node list
24     int index = XADD (&head);
25     IClist.records[index] = ICsample;
26 }
```

7.2.3 Wait-Free Irradiance Cache (WF)

The wait-free algorithm does not rely on any critical sections to both read and write to the shared IC. When adding samples to an IC node the atomic XADD operator (Listing 7.4 line 24) is used, returning a unique index into the list of records, which ensures that samples are never over-written; simultaneously, the private index to the next free position is incremented. While it may seem the data structure remains at an inconsistent state this does not happen since IrradianceCache::getIrradiance() does not use `head` as a stopping condition; rather all elements of `IClist` are initialised to `NULL` and searching stops when a `NULL` pointer is found. Thus elements which have not been properly inserted into the data structure will never be used.

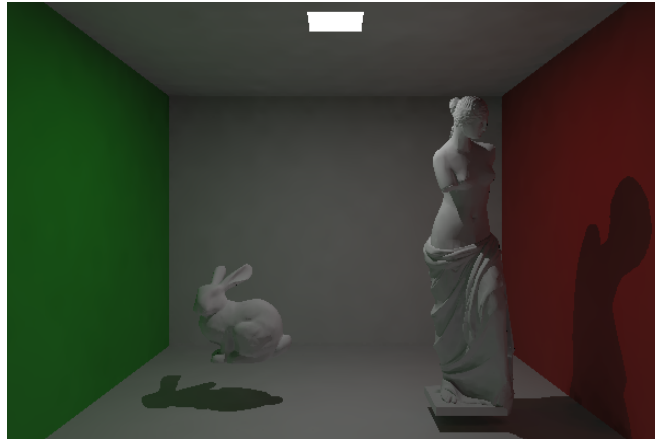
When adding a new child node to the octree, the new subtree is built using a temporary pointer. When fully built, the subtree is attached to the octree using the CAS operator (Listing 7.4 line 14). If the relevant child still does not exist, indicated by the pointer still being NULL, then CAS completes successfully. If, however, another thread wrote to the same child, then CAS will fail and this thread will discard both the created subtree and the associated sample after utilising it for the current computation. As can be seen in Table 1 the number of discarded samples is minimal, amounting to no more than 0.3% of the total samples.

The atomic primitives used in most wait-free algorithms still need a memory barrier in order to ensure out-of-order execution does not corrupt the shared data structure. Typically a memory barrier precedes the use of atomic primitives such as CAS and XADD. This can often be expensive since out-of-order execution typically accounts for an increase in performance. In the wait-free algorithm memory barriers are kept to a minimum by only calling them when inserting IC samples into the octree and not when accessing the cache for interpolation. This is how the much more frequent IC interpolation requests do not entail any overheads over the serial methods.

The wait-free approach ensures that the single shared IC can be accessed concurrently by all threads, and, as can be seen in Listing 7.4, requires little changes in the code from a traditional sequential irradiance cache. As shall be seen in the next section, this results in faster execution times both when interpolating and creating IC samples and also it does not suffer the larger memory requirements of the LW approach.

7.3 Results

All presented results have been generated on an 8-core (dual quad-core) Intel Xeon machine running at 2.5GHz with 8 gigabytes of RAM, using a custom written interactive ray tracer. Experiments were run under CentOS 4 with the code being compiled with ICC 9.0. The renderer utilised does not make use of packetisation or SIMD operations except for the ray-bounding volume intersection test used when traversing the acceleration data-structure, which is a BVH implementation based on Wald *et al.* [WBS07]. Five different scenes (Figure 7.2.1) were utilised in the experiments. These scenes were picked to provide a range



(a) Cornell (48k)



(b) Conference (190k)



(c) Desk (12k)

Figure 7.2.1: The five scenes utilised in the experiments. The polygon count for each scene is shown in brackets.



(a) Office (20k)



(b) Sponza (66k)

Figure 7.2.1: The five scenes utilised in the experiments. The polygon count for each scene is shown in brackets.

of geometric complexity, physical dimensions and lighting conditions. All scenes were rendered at a resolution of 600×400 .

7.3.1 Still images

Table 7.1 provides results for one, two, four and eight threads with the time taken to calculate the frame, the number of IC samples generated, overheads associated with each algorithm and speed-up. Results for one thread were obtained using the traditional sequential approach (TRA) and speed-up is computed with respect to these results. For the lock-based approach (LCK) the reported overhead is the aggregate time spent to enter critical regions summed over all threads. For

	1	2			4			8		
	TRA	LCK	LW	WF	LCK	LW	WF	LCK	LW	WF
Time (s)	3.152	1.633	2.096	1.614	0.863	1.178	0.814	0.656	0.700	0.473
IC samples	3463	2742	4339	2707	2483	4404	2473	2441	4440	2410
Overheads [†]	0	0.053	0.024	4	0.189	0.018	1	1.370	0.019	7
Speed-up	1.000	1.988	1.548	2.011	3.759	2.756	3.986	4.950	4.640	6.862

(a) Cornell

	1	2			4			8		
	TRA	LCK	LW	WF	LCK	LW	WF	LCK	LW	WF
Time (s)	3.749	1.971	2.444	1.965	1.104	1.385	1.049	0.753	0.798	0.607
IC samples	3477	3038	4282	2998	2748	4394	2775	2700	4378	2709
Overheads [†]	0	0.035	0.018	4	0.189	0.021	0	1.259	0.023	8
Speed-up	1.000	1.902	1.534	1.907	3.396	2.707	3.572	4.976	4.696	6.178

(b) Desk

	1	2			4			8		
	TRA	LCK	LW	WF	LCK	LW	WF	LCK	LW	WF
Time (s)	4.854	2.460	3.180	2.445	1.283	1.768	1.269	0.786	0.929	0.658
IC samples	3065	2517	3663	2524	2272	3817	2256	2130	3842	2170
Overheads [†]	0	0.064	0.028	1	0.192	0.021	3	1.127	0.026	3
Speed-up	1.000	1.973	1.526	1.985	3.783	2.745	3.826	6.176	5.223	7.381

(c) Conference Room

Table 7.0: Results for all scenes[†] - Overheads are all in seconds except for WF which is number of irradiance samples discarded

	1			2		4			8		
	TRA	LCK	LW	WF	LCK	LW	WF	LCK	LW	WF	
Time (s)	2.947	1.474	1.976	1.654	0.895	1.207	0.864	0.650	0.689	0.469	
IC samples	2089	1881	2650	1976	1803	3199	1802	1766	3290	1785	
Overheads [†]	0	0.042	0.019	1	0.337	0.018	3	1.397	0.022	1	
Speed-up	1.000	1.999	1.491	1.782	3.291	2.442	3.412	4.531	4.276	6.287	

(a) Office

	1			2			4			8		
	TRA	LCK	LW	WF	LCK	LW	WF	LCK	LW	WF		
Time (s)	7.330	3.802	4.672	3.676	2.100	2.563	1.935	1.143	1.505	1.083		
IC samples	3357	3113	4166	3113	3032	4286	2958	2929	4379	2942		
Overheads [†]	1	0.046	0.026	1	0.186	0.026	5	1.018	0.028	4		
Speed-up	1.000	1.928	1.569	1.994	3.779	2.860	3.707	6.413	4.872	6.766		

(b) Sponza

Table 7.1: Results for all scenes[†] - Overheads are all in seconds except for WF which is number of irradiance samples discarded

local-write (LW) the reported overhead is the time taken to sequentially merge all local caches into the global one at the end of the frame, for wait-free (WF) it is the number of samples discarded. Each image was calculated with an empty irradiance cache to show a worst-case scenario with maximal irradiance calculations occurring. Graphs of all this data are presented in Figure 7.3.1; the left Y-axis and the accompanying line graph shows the instantaneous framerate (reciprocal of time taken to render the frame) while the right Y-axis and the bar graph shows the speed-up compared to the traditional single-threaded irradiance cache with no access control.

LW performs and scales worse than the two other algorithms. This is because no sharing is actually occurring since only one frame is rendered and merging of the local caches only happens at the end of the frame. Each thread must

evaluate all irradiance samples that project into its assigned tiles of the image plane, leading to much work replication as can be seen by the number of evaluated irradiance samples.

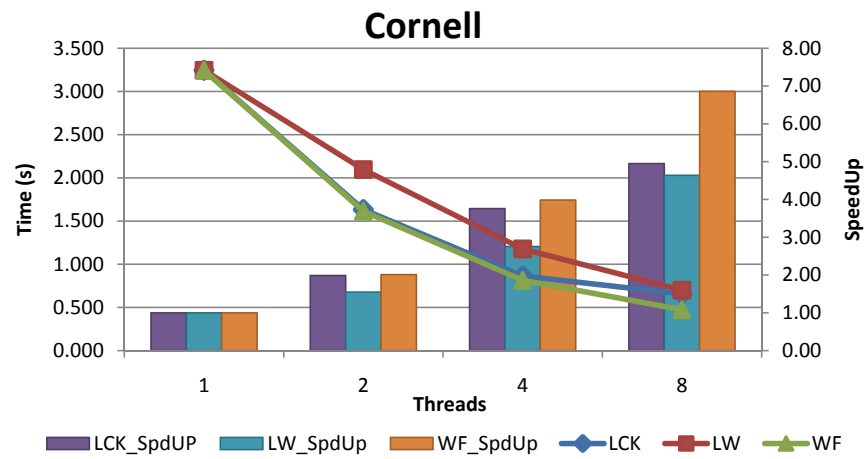
The performance difference between LCK and WF becomes evident as the number of threads increases. The aggregated time waiting for locks increases with the number of threads, resulting on a major performance loss. The wait-free algorithm scales much better because it does not serialise neither writes nor reads to the shared data structure. For a reduced number of threads LCK performs similar to WF since most of the time is spent evaluating new irradiance samples, which is not a critical region of the code. As the number of threads increases, more range searches are performed; since these are serialised in LCK, a performance penalty arises.

For eight threads an average speed-up of 6.66 for the WF algorithm, 4.71 for LW and 5.38 for LCK was recorded. The WF algorithm is clearly the fastest with LCK showing comparable results with less than 8 threads but showing inferior scaling as more threads are utilised.

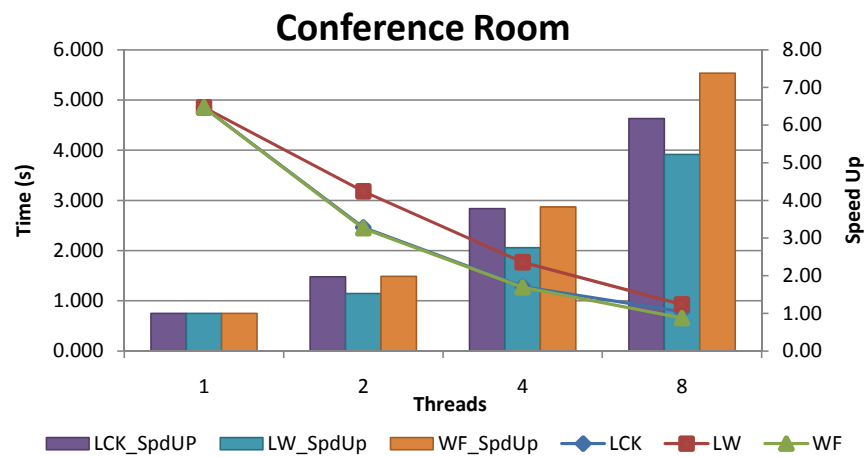
7.3.2 Animations

Two scenes, Cornell and Conference Room, were selected and each was rendered, using 8 threads, for 100 frames while the camera did a 360 degrees rotation around the scene. Each frame in the sequence re-utilised previously created cache samples while simultaneously calculating new ones. This provides an overview of performance when a more balanced mix of evaluation and interpolation is occurring, unlike the case for the still images. The results for these particular experiments are displayed in Figures 7.3.2 and 7.3.3, showing the time taken to render each of the 100 frames for the Cornell and Conference Room scenes respectively for each of the three algorithms. For each of the scenes the first frame is the equivalent of the still images above, where the cache is totally empty and all the samples needed to be generated.

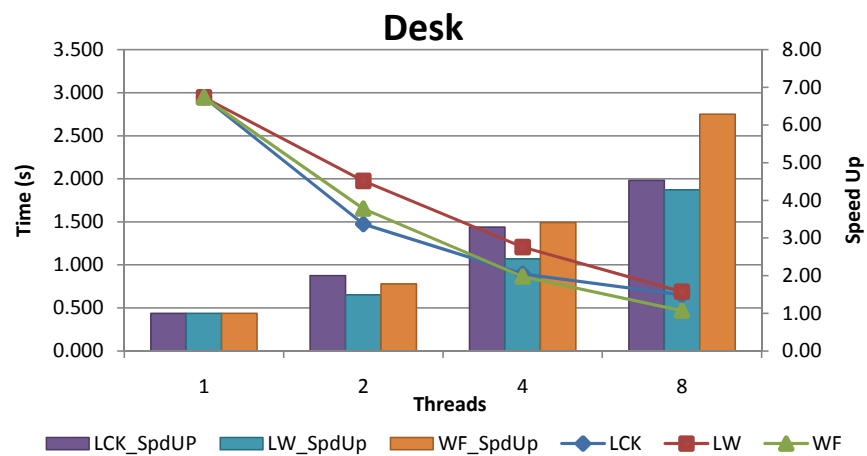
Clearly, LCK performs worse than LW and WF, except for the first frame. Since for the remaining frames the IC will not be empty, many irradiance samples can be reused; but LCK serialises all range searches performed to locate these samples, thus severely impacting on performance. WF outperforms LW because the former shares irradiance samples immediately without any extra overhead associated with reading, while the latter does not share samples within a frame,



(a) Cornell

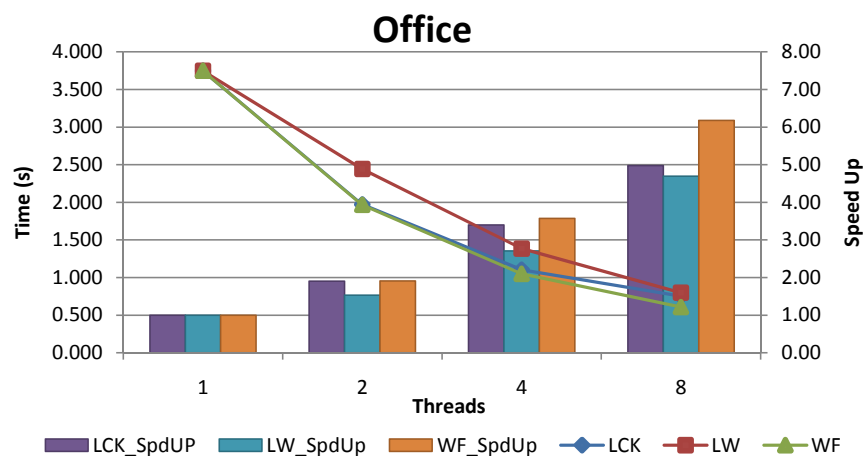


(b) Conference Room

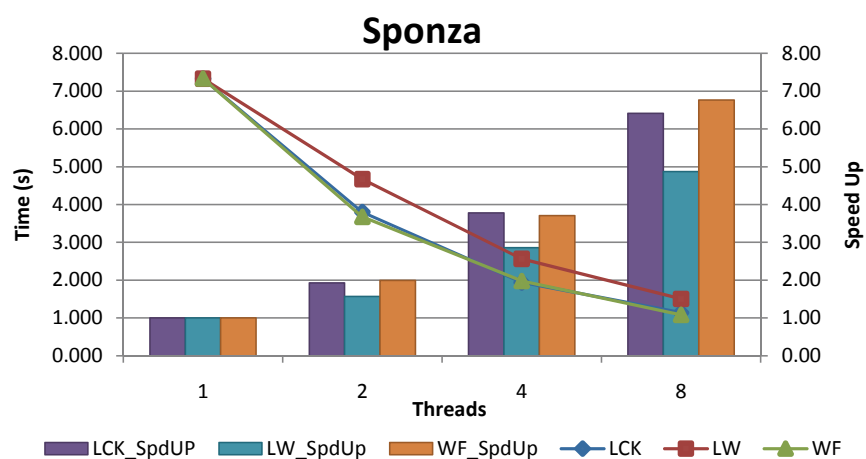


(c) Desk

Figure 7.3.1: Still Images: Results for all scenes



(a) Office



(b) Sponza

Figure 7.3.1: Still Images: Results for all scenes.

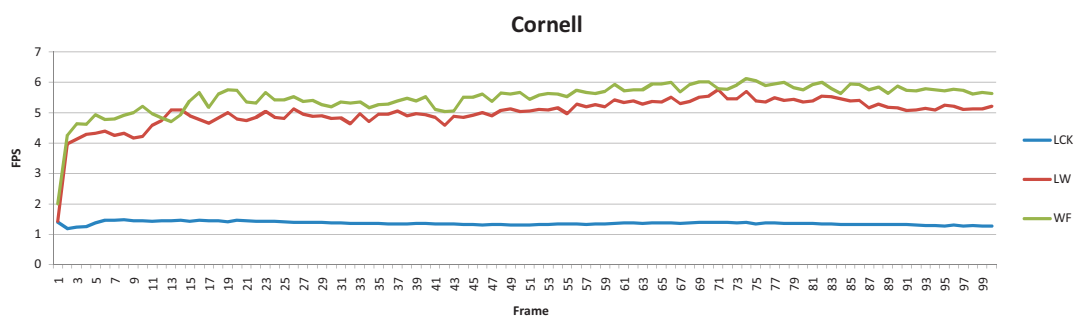


Figure 7.3.2: Animation results for Cornell Box.

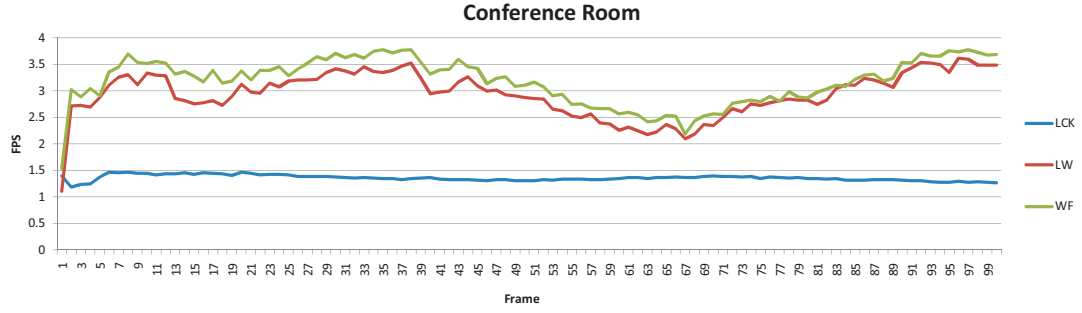


Figure 7.3.3: Animation results for Conference Room.

thus resulting on costly extensive evaluations of more indirect diffuse irradiance values. Summarising, LCK is mostly penalised by reading serialisation, whereas LW is penalised by work replication.

Excluding the first frame in the Cornell scene an average of 5.52 FPS for the WF algorithm, 5.03 FPS for LW and 1.3 FPS for LCK was attained. Even though LCK performed better than LW for the first frame, the cost of locking is high even when doing a mixture of reads and writes and this is reflected in its poor performance for the animation. These results are also reflected in the Conference Room scene where 3.2 FPS for WF, 2.96 FPS for LW and 1.03 FPS for LCK was achieved. WF performs the best out of the three approaches displaying, on average, a 9% speed-up over the LW approach and a 368% speed-up over the LCK approach.

7.4 Summary

This chapter proposes a new wait-free data access control mechanism for sharing the irradiance cache among multiple rendering threads on a shared memory parallel system and evaluate it against two traditional data access algorithms: a lock-based approach and a local write one. It is demonstrated that the proposed approach outperforms the others and scales better with the number of threads.

The lock-based algorithm serialises all access to the shared data structure, reads included. Range searches performed in the octree to locate valid irradiance samples are serialised, resulting in performance losses; this problem is aggravated with the number of threads and the resulting contention. The local write algorithm does not share any irradiance values evaluated within each frame, thus

suffering a performance penalty as a result of work replication. Neither of these two algorithms scales well as the number of threads increases.

The wait-free algorithm does not serialise accesses to the shared data structure and irradiance values are immediately shared among all threads without any overhead associated with reading. It exhibits the best frame rates for walkthroughs within static scenes and scales well with the number of threads, achieving an efficiency between 77% and 92% for 8 threads.

The proposed wait-free data access control mechanism is both efficient and simple to implement, requiring only minor modifications to a traditional sequential irradiance cache implementation. The relevance of efficient, scalable and reliable mechanisms to control access to shared data structures within shared memory systems is ever increasing with the advent of multi-core systems., which in the near future will have a degree of concurrency which that is expected to be larger than that on current machines.

CHAPTER 8

Conclusions and Future Work

Global illumination remains one of the major fields of research in computer graphics, with efforts to increase interactivity as well as overall fidelity ongoing. This thesis has introduced novel global illumination algorithms that allow for effects such as diffuse interreflections, soft shadows and participating media at interactive rates for fully dynamic scenes. The needs of numerous fields such as architecture, lighting design, special effects and game development have increased the demand for such algorithms that provide a high fidelity solution for complex scenes, and do so at interactive rates. While exploiting coherency, to achieve interactivity, had been successful for interactive ray tracing the need for full global illumination has required the development of new algorithms to focus limited computational power currently available where it is most needed. Selective rendering algorithms do just this by steering the computation, utilising a number of different selective criteria, while performing adaptive or progressive rendering. Our novel algorithms have met the challenges outlined above, exploiting both coherency on all levels in the rendering pipeline along with selectively focusing computation where it is most needed.

8.1 Conclusions

The aim of this thesis was to develop new algorithms for achieving interactive global illumination. This was achieved by combining selective techniques and interactive ray tracing and global illumination methods into two novel interactive global illumination algorithms. The methodology and structured approach utilised when researching, developing and testing these algorithms was also presented as a framework. The algorithms developed support dynamic scenes where

the camera, materials and objects could change and move freely without any pre-computation being required.

During the analysis of the impact that selective rendering has on interactive ray tracing it was shown that selective rendering, due to its poor spatial coherence, had an adverse effect mainly on primary rays. This lack of coherence can severely limit selective rendering within interactive ray tracing systems, which rely on cache coherence to achieve their high performance. A logarithmic increase in computational cost per pixel was observed as spatial coherence deteriorated. By identifying three core stages in the methodology employed (identification, deconstruction and pairing) a framework was created to combine existing interactive ray tracing and interactive global illumination algorithms with selective rendering techniques to form new interactive global illumination solutions. Such solutions needed to consider three key criteria: interactivity, coherence and adaptability and had to do so on a per component basis to ensure maximal use of available computational resources.

By combining interleaved sampling [Kel97] with an adaptive approach, that used efficient component-specific adaptive guidance methods (Section 5.3.3 - 5.3.4), a novel interactive global illumination algorithm was developed. Adaptive Interleaved Sampling achieved interactive frame rates of between three and eleven frames per second for a variety of scenes on a single eight-core machine. An average speed-up of 3.53 times over competing methods was demonstrated, with certain cases showing an order of magnitude improvement. This was achieved while maintaining high visual fidelity, perceptually equivalent to IGI [WBS02]. Results showed that on average only 1.37% of pixels were deemed perceptually different from high-quality path traced solution. This approach was shown to be generalisable and that it was applicable to any component as long as it utilised ray tracing and a custom heuristic could be written to guide the computation.

Irradiance caching was then examined and was modified and extended to support an interactive update as well as a more coherent indirect diffuse inter-reflection computation. An interactive caching scheme for indirect diffuse interreflections, Temporal Instant Caching, was developed to work with dynamic scenes. The algorithmic design allowed for the extension of the already existing spatial coherence into the temporal domain. The temporal coherence was then exploited to avoid wasteful computation by using selective techniques to only update relevant cache samples. This approach achieved frame rates exceeding nine frames per second for the majority of scenes on a single eight-core

machine. When only a small amount of temporal coherence existed the algorithm still maintained reasonable frame rates which were competitive with other algorithms, such as IGI [WBS02]. These results were achieved with no perceivable loss in visual quality when compared to IGI, with 0.23% of pixels being shown to be perceptually different, and minimal differences when compared to a gold-standard path-traced image where only 0.45% of pixels were detected as being perceptually different. Further research resulted in the development of a novel wait-free data access control mechanism allowed for the sharing of the irradiance cache among multiple rendering threads on a shared memory parallel system. It exhibited the best frame rates, out of the three approaches tested, for walkthroughs within static scenes, and scaled well with the number of threads, achieving an efficiency between 77% and 92% for 8 threads. For the static scenes an average speed-up of 6.66, over the traditional approach, was achieved. For dynamic scenes, examined over multiple frames as the camera was moved, the wait-free algorithm displayed, on average, 9% speed-up over the local-write algorithm and a 368% speed-up over the lock-based algorithm. While the wait-free algorithm was only 9% faster this was achieved without the extra memory usage or the increased sample count that local-write algorithm exhibited. This approach extended the irradiance cache for use in interactive contexts by entirely avoiding all the pitfalls of multi-threaded access to its underlying data structure. This proof-of-concept of a novel wait-free technique for a popular algorithm, such as irradiance caching, could also inspire other techniques to utilise this approach.

These algorithm demonstrated the potential of interactive global illumination on the CPU, with effects such as soft shadows, participating media and diffuse interreflections running on a single standard desktop PC for a number of different scenarios.

8.2 Contributions

This thesis explored the impact and application of selective rendering techniques on global illumination algorithms. Having built upon the existing knowledge in both these areas new algorithms were developed to exploit both area's unique strengths. The major novel contributions of this thesis are as follows:

- A comprehensive literature review of interactive ray tracing techniques and global illumination techniques for both the CPU and the GPU.

- An analysis and assessment of the effects that selective rendering has on interactive ray tracing by using a then state-of-the-art ray tracer from the University of Utah [BSP06]. These results and observations are used for the development of a framework detailed below. This work was published at the 16th International Conference on Computer Graphics, Visualization and Computer Vision [DCD08].
- A framework that encompasses the methodology and processes used during the development of two novel interactive global illumination algorithms. It provides a structured approach to combining selective rendering techniques and global illumination methods into new interactive global illumination solutions.
- A novel global illumination solution that combines interleaved sampling [Kel97] and instant global illumination [WKB*02] while making the final solution adaptive. The approach supports soft shadows, diffuse interreflections and participating media in fully dynamic scenes where the camera, lights and objects could be moved and materials modified. This work was published in the Computer Graphics Forum [DDC09].
- A further novel global illumination solution that combines irradiance caching [WRC88] with an adaptive temporal update. This method uses instant radiosity [Kel97] when computing the diffuse interreflections. The update only modifies samples that have changed since the previous frame and therefore allows for the computation of diffuse interreflections at interactive rates for dynamic scenes. This work was published in the Computer Graphics Forum [DDB*09].
- A new wait-free data access control mechanism for sharing the irradiance cache among multiple rendering threads on a shared memory parallel system. This proposed mechanism is both efficient and simple to implement, requiring only minor modifications to a traditional sequential irradiance cache implementation. This makes the irradiance cache suitable for highly parallel architectures. This work was published at the 10th Eurographics Symposium on Parallel Graphics and Visualisation [DDSC09].

8.3 Impact

The novel interactive global illumination algorithms developed in this thesis can possibly be used for various applications, both academic and commercial. The component-based nature of Adaptive Interleaved Sampling enables specific components to be picked out and used elsewhere, for other interactive applications that require global illumination, ranging from data visualisation to lighting design. The approach also allows for the addition of other components to further extend the approach as required to meet specific demands.

The extensions presented to irradiance caching, primarily used in off-line rendering till now, for use in an interactive context also have many potential applications. As has already been seen in Section 2.3.5, the irradiance cache is used in a number of industries and software applications to accelerate the computation of diffuse interreflections. Both PDI/Dreamworks [TL04] and Pixar [Chr08], two movie studios that produce computer animations, have shown that they make extensive use of the irradiance cache in a primarily CPU-based environment. The novel algorithms presented in this thesis, both the temporal update (Chapter 6) and the wait-free access control mechanism (Chapter 7), could further extend the usage of the irradiance cache in the production pipeline allowing for interactive previews within modelling software. This could be achieved using the same software architecture and rendering pipelines that are already in place. Furthermore, the wait-free approach could be utilised in off-line production rendering to improve the utilisation of computation resources and overall performance.

With an interactive global illumination solution in place animators, designers and artists would have the ability to test different lighting scenarios with immediate feedback. Without the long waits, which are currently required, this would increase productivity and allow for more iterations in a shorter space of time. This level of feedback would not only reduce the total number of hours required to produce production-ready assets, and hence reduce the cost, but would also minimise mistakes in the final production renders, which take many days or even weeks to compute. This increased accuracy and interactivity would allow potential problems to be spotted much sooner and costly mistakes could be avoided.

8.4 Limitations and Extensions

This section examines the work presented in this thesis and discuss the limitations of the current approaches as well as a number of possible extensions and directions for further research.

During the examination of the impact of selective rendering on state-of-the-art ray tracing algorithms the selective guidance consisted of sub-sampling the image with a regular stride, this was done to provide an easy and controlled mechanism with which to degrade the overall spatial coherence. Of further interest would be the examination of more complex selective guidance methods, such as one of those outlined in Section 2.3.2 such as the saliency map [YPG01] and task map [CCW03] to drive the computation. In certain cases, these methods would offer increased spatial coherence over the approach used in Chapter 4, this is due to the samples being unevenly distributed and forming groups of spatially coherent samples, potentially leading to improved performance. The addition of such a metric, which is not trivial to compute, would introduce a serial critical section into a highly parallel system. Utilising other computational resources, such as the GPU, to calculate these guidance metrics, as in Lee *et al.* [LKC09], and the trade off between the computation of an expensive selective guidance metric and more spatially coherent ray distribution would need to be examined.

For the implementation of adaptive interleaved sampling (Chapter 5) the ray tracing kernel could be enhanced with features such as packetisation [WSBW01]. Unlike most other adaptive approaches, the method should adapt well to a faster ray tracing kernel since the guidance and sampling for each tile are computed at the same time making it naturally coherent and suited for packetisation. Tracing the rays for the SS and ID components (Sections 5.3.3 and 5.3.2) could be packetised as multiple rays would share an origin on a light source or at a VPL due to the way the methods reuse samples for the same pixel in the IS pattern as it tiled. Another potential avenue of investigation would be more complex global heuristics. While only local heuristics that utilise information for a single tile and one component were implemented, global heuristics that use information from surrounding tiles as well as multiple components would be of interest. Finally the applicability of the framework for other algorithms such as GPU-based global illumination techniques, and how the criteria would need to be adjusted, would be worth examining.

The temporal instant cache has a number of possible extensions and avenues

of further research. One aspect common to most caching mechanisms is that the search for cached samples to interpolate from can impact the performance as the cache count increases. This problem is further accentuated since the cache count affects the update computation. Since samples computed in earlier frames might not be re-used (if they do not contribute to the current view point), ageing methods similar to those presented by Tawara *et al.* [TMS04], whereby cached samples that are infrequently used are discarded, would directly improve the performance. The test for ageing would not impact on the current temporal instant cache method and could be integrated as part of the update function when cycling through each cached sample (Line 2, Algorithm 1).

Temporal instance caching supports on-demand re-evaluation of the visibility rays, which although partially improving performance, still requires major parts of the computation for each sample to be executed every frame. An alternative approach would be to only update cached samples when requested for interpolation, which would entail that the update is performed on demand. This would require maintaining a structure consisting of those VPLs which are invalidated and those objects that would have moved at each frame. The on-demand cached sample update would need to query this structure to identify which visibility rays to update. Ageing would benefit this method by placing an upper-bound on the number of frames that the structure would need to store. Computing gradients for the instant cache, and possibly the temporal method also, which, as with similar gradient methods [WH92], could reduce the number of cached samples and help mitigate this issue.

Both adaptive interleaved sampling and temporal instant caching rely on VPLs in a similar manner to that of instant radiosity. One of the limitations of methods based on instant radiosity is that their rendering time is, for the most part, linearly dependent on the number of VPLs that are shot. For the case of the temporal instant cache, the situation is further aggravated since the temporal update is also dependent on the number of VPLs. Several approaches have been proposed to improve instant radiosity-based algorithms and alleviate this problem. Importance has been used in the past to direct VPL placement for complex environments [WBS03], thus reducing the number of VPLs that are required to be shot. Temporal-awareness was used to ensure that coherence between VPL placement was maintained. This importance would allow the algorithms to operate without large numbers of VPLs for complex highly occluded scenes. This could be integrated into the both algorithms as a pre-process before shooting the

VPLs, requiring little changes to the methods as presented, and few modifications to the algorithm presented by Wald *et al.* [WBS03]. Methods such as those presented in Segovia *et al.* [SIMP06a,SIP07]; Wald *et al.* [WBS03] could also be investigated to help improve VPL distribution, especially for complex scenes and those with high levels of occlusion. Lightcuts [WFA*05,WABG06] have been used to cluster point light sources (or VPLs) and reduce the VPL processing count. Their utilisation in conjunction with the temporal instant cache and adaptive interleaved sampling can be mutually beneficial. This may require adding some form of temporal criteria to lightcuts to ensure that the VPL clustering does not change drastically over frames, due to the interactivity of the systems. Lightcuts require building the clusters binary tree and selecting, for each shading point, the most appropriate cut. These operations incur non-negligible overheads that might compromise interactivity; careful optimisation and eventual relaxation of the clustering and cut selection criteria might be required.

Although the wait-free algorithm described in Chapter 7 has shown good scalability with up to eight threads, further investigation would be interesting to identify the limits of this trend by running the algorithm on machines with a larger number of processors sharing the same address space. Also the memory organisation might impact on the performance of the proposed algorithm, especially with an increased number of threads. Utilisation of the irradiance cache within dynamic environments, i.e., those where geometry might change between frames, would require the ability to remove from the shared data structure records which became invalid as well as those that are no longer being used. Assessment of a wait-free synchronisation algorithms supporting this removal operation would be of great interest.

8.5 Directions for Future Work

The current generation of GPUs are generalising rapidly, with the addition of caches and a choice of development languages such as CUDA [GGN*08] and OpenCL [Mun08]. They also offer an order of magnitude, or more, increase in computation power over CPUs. This generalisation will further encourage the trend, as was discussed in Sections 3.1.2 and 3.2.2, of interactive ray tracing and global illumination research moving towards GPU-based solutions. While initial research focused on GPU-specific algorithms, due to the highly specialised archi-

texture and the inability to use complex data structures, recent work is making use of algorithms developed on the CPU, such as photon mapping [Jen01] and path tracing [Kaj86]. These approaches, just like the framework in Section 5.2, will need to tackle the issues of balancing coherency and focusing computation due to the massively parallel nature of the GPUs where thousands, if not tens of thousands, of processes or threads are executing at any given time.

This presents a new set of challenges where rendering algorithms must take into consideration the highly parallel nature of the GPU when designing their data structures and access patterns to these structures. Algorithms such as the wait-free approach presented in this thesis, now viable on the GPU due to atomic operations such as CAS and XADD being available, will play a more important role as critical sections in code will have a much larger impact on the highly parallel GPUs. The problems identified with the lock-based, Section 7.2.1, and local-write, Section 7.2.2, algorithms would become substantially worse on the GPU. Even limited fine-grained locking would introduce a large amount of overhead due to contention while replication of the data structure would introduce large memory overheads. Potential solutions could be found in the fields of distributed rendering and parallel processing, as they have dealt with similar problems.

8.6 Final Remarks

Real-time accurate global illumination for fully interactive scenes remains the target for many in the rendering field. While it would seem that the ever increasing processing power provided by new generations of CPUs and GPUs would allow one to reach this goal by simply waiting for an sufficient amount of computational power, this is not the case. The increase in computational power also brings with it new demands for higher resolution images and more complex scenes, realistic materials and advanced effects. It is for this reason that algorithms must focus computation in areas of greatest gain while being aware of the hardware they are running on to make full use of the parallelism and coherency available. The algorithms and framework in this thesis make progress in the state-of-the-art by providing novel global illumination solutions that can run at interactive rates for fully dynamic scenes with no pre-computation required. Adaptive interleaved sampling, temporal instant caching and wait-free shared-memory irradiance caching show how design decisions are influenced, but

not dictated, by hardware and following a number of simple criteria can lead to novel solutions. The framework used to develop these algorithms focuses on the algorithmic not implementation details when combining selective rendering and global illumination techniques on a per component basis. In this way while the process is aware of higher level constraints, like the need for the algorithms to remain highly parallel, the improvements are not dependant on hardware trends or specific hardware platforms. The work in this thesis has highlighted the potential of combining selective rendering techniques and global illumination methods, on a per component basis, to form novel interactive global illumination solutions that fully exploit the ever increasing computational power available.

References

- [ADM*08] ANNEN T., DONG Z., MERTENS T., BEKAERT P., SEIDEL H.-P., KAUTZ J.: Real-time, all-frequency shadows in dynamic scenes. *ACM Trans. Graph.* 27, 3 (2008), 1–8.
- [AH93] AUPPERLE L., HANRAHAN P.: A hierarchical illumination algorithm for surfaces with glossy reflection. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1993), ACM Press, pp. 155–162.
- [AK87] ARVO J., KIRK D.: Fast ray tracing by ray classification. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1987), ACM, pp. 55–64.
- [AL09] AILA T., LAINE S.: Understanding the efficiency of ray traversal on gpus. In *HPG '09: Proceedings of the Conference on High Performance Graphics 2009* (New York, NY, USA, 2009), ACM, pp. 145–149.
- [ALL89] ANDERSON T. E., LAZOWSKA E. D., LEVY H. M.: The performance implications of thread management alternatives for shared-memory multiprocessors. *IEEE Trans. Computers* 38, 12 (1989), 1631–1644.
- [AMH02] AKENINE-MOLLER T., HAINES E.: *Real-Time Rendering*. A. K. Peters, Ltd., Natick, MA, USA, 2002.
- [App68] APPEL A.: Some techniques for shading machine renderings of

- solids. In *Proceedings of the Spring Joint Computer Conference* (1968), pp. 37–45.
- [Ash95] ASHDOWN I.: *Radiosity: a programmer's perspective*. John Wiley & Sons, Inc., New York, NY, USA, 1995.
- [Bek99] BEKAERT P.: *Hierarchical and Stochastic Algorithms for Radiosity*. PhD thesis, Department of Computer Science, Katholieke Universiteit Leuven, Leuven, Belgium, 1999.
- [BEL*06] BOULOS S., EDWARDS D., LACEWELL J. D., KNISS J., KAUTZ J., SHIRLEY P., WALD I.: Interactive Distribution Ray Tracing. *Technical Report, SCI Institute, University of Utah, No UUSCI-2006-022* (2006).
- [BEL*07] BOULOS S., EDWARDS D., LACEWELL J. D., KNISS J., KAUTZ J., SHIRLEY P., WALD I.: Packet-based Whitted and Distribution Ray Tracing. In *Proc. Graphics Interface* (May 2007).
- [BFGS86] BERGMAN L., FUCHS H., GRANT E., SPACH S.: Image rendering by adaptive refinement. In *SIGGRAPH '86* (1986), ACM Press, pp. 29–37.
- [BFMZ94] BISHOP G., FUCHS H., MCMILLAN L., ZAGIER E. J. S.: Frameless rendering: double buffering considered harmful. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1994), ACM Press, pp. 175–176.
- [BK08] BARSKY B. A., KOSLOFF T. J.: Algorithms for rendering depth of field effects in computer graphics. In *ICCOMP'08: Proceedings of the 12th WSEAS international conference on Computers* (Stevens Point, Wisconsin, USA, 2008), World Scientific and Engineering Academy and Society (WSEAS), pp. 999–1010.
- [BM98] BOLIN M. R., MEYER G. W.: A perceptually based adaptive sampling algorithm. In *SIGGRAPH '98* (1998), ACM Press, pp. 299–309.

- [BSP06] BIGLER J., STEPHENS A., PARKER S. G.: Design for parallel interactive ray tracing systems. In *in: Proceedings of IEEE Symposium on Interactive Ray Tracing* (2006), pp. 187–196.
- [Bun05] BUNNELL M.: Dynamic ambient occlusion and indirect lighting. *GPU Gems 2* (2005), 223–233.
- [BWG03] BALA K., WALTER B., GREENBERG D. P.: Combining edges and points for interactive high-quality rendering. *ACM Trans. Graph.* 22, 3 (2003), 631–640.
- [BWS03] BENTHIN C., WALD I., SLUSALLEK P.: A Scalable Approach to Interactive Global Illumination. *Computer Graphics Forum* 22, 3 (2003), 621–630. (Proceedings of Eurographics).
- [BWS06] BOULOS S., WALD I., SHIRLEY P.: *Geometric and Arithmetic Culling Methods for Entire Ray Packets*. Tech. rep., SCI Institute, University of Utah, 2006.
- [Cat74] CATMULL E. E.: *A subdivision algorithm for computer display of curved surfaces*. PhD thesis, The University of Utah, 1974.
- [CCC87] COOK R. L., CARPENTER L., CATMULL E.: The reyes image rendering architecture. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1987), ACM Press, pp. 95–102.
- [CCW03] CATER K., CHALMERS A., WARD G.: Detail to Attention: Exploiting Visual Tasks for Selective Rendering. In *Proceedings of the Eurographics Symposium on Rendering* (2003), pp. 270–280.
- [CCWG88] COHEN M. F., CHEN S. E., WALLACE J. R., GREENBERG D. P.: A progressive refinement approach to fast radiosity image generation. In *SIGGRAPH '88* (1988), ACM Press, pp. 75–84.
- [CG85] COHEN M. F., GREENBERG D. P.: The hemi-cube: a radiosity solution for complex environments. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1985), ACM Press, pp. 31–40.

- [CHCH06] CARR N. A., HOBEROCK J., CRANE K., HART J. C.: Fast gpu ray tracing of dynamic meshes using geometry images. In *GI '06: Proceedings of Graphics Interface 2006* (Toronto, Ont., Canada, Canada, 2006), Canadian Information Processing Society, pp. 203–209.
- [Che90] CHEN S. E.: Incremental radiosity: an extension of progressive radiosity to an interactive image synthesis system. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1990), ACM, pp. 135–144.
- [CHH02] CARR N. A., HALL J. D., HART J. C.: The ray engine. In *Graphics Hardware 2002* (Sept. 2002), pp. 37–46.
- [CHH03] CARR N. A., HALL J. D., HART J. C.: Gpu algorithms for radiosity and subsurface scattering. In *HWWS '03: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 51–59.
- [CHL04] COOMBE G., HARRIS M. J., LASTRA A.: Radiosity on graphics hardware. In *GI '04: Proceedings of Graphics Interface 2004* (School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004), Canadian Human-Computer Communications Society, pp. 161–168.
- [Chr04] CHRISTEN M.: *Implementing Ray Tracing on GPU*. Master's thesis, 2004.
- [Chr06] CHRISTENSEN P. H.: Ray tracing for the movie cars. In *Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing* (September 2006), pp. 1–6.
- [Chr08] CHRISTENSEN P.: Irradiance caching in pixar's renderman. In *SIGGRAPH '08: ACM SIGGRAPH 2008 classes* (New York, NY, USA, 2008), ACM, pp. 1–26.

- [CKL*10] CHOI B., KOMURAVELLI R., LU V., SUNG H., BOCCHINO R. L., ADVE S. V., HART J. C.: Parallel sah k-d tree construction. In *Proceedings of High-Performance Graphics 2010* (2010).
- [Cla76] CLARK J. H.: Hierarchical geometric models for visible surface algorithms. *Commun. ACM* 19, 10 (1976), 547–554.
- [CLF*03] CHRISTENSEN P. H., LAUR D. M., FONG J., WOOTEN W. L., BATALI D.: Ray differentials and multiresolution geometry caching for distribution ray tracing in complex scenes. *Comput. Graph. Forum* 22, 3 (2003), 543–552.
- [CPC84] COOK R. L., PORTER T., CARPENTER L.: Distributed ray tracing. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1984), ACM Press, pp. 137–145.
- [CRMT91] CHEN S. E., RUSHMEIER H. E., MILLER G., TURNER D.: A progressive multi-pass method for global illumination. In *SIGGRAPH '91* (1991), ACM Press, pp. 165–174.
- [Cro77] CROW F. C.: Shadow algorithms for computer graphics. In *SIGGRAPH '77: Proceedings of the 4th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1977), ACM Press, pp. 242–248.
- [CT81] COOK R. L., TORRANCE K. E.: A Reflectance Model for Computer Graphics. In *SIGGRAPH'81* (1981), ACM Press, pp. 307–316.
- [Dab10] DABROVIC M.: Sponza atrium, November 2010.
- [Dal93] DALY S.: The Visible Differences Predictor: An Algorithm for the Assessment of Image Fidelity. In *Digital Images and Human Vision* (1993), A.B. Watson, MIT Press, Cambridge, MA, pp. 179–206.
- [DB97] DIEFENBACH P. J., BADLER N. I.: Multi-pass pipeline rendering: realism for dynamic environments. In *SI3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics* (New York, NY, USA, 1997), ACM Press, pp. 59–ff.

- [DBB06] DUTRE P., BALA K., BEKAERT P.: *Advanced Global Illumination, 2nd Edition*. A K Peters, Natick, MA, 2006.
- [DCD08] DUBLA P., CHALMERS A., DEBATTISTA K.: An analysis of cache awareness for interactive selective rendering. In *Communications Papers proceedings* (2008), Skala V., (Ed.), WSCG.
- [DDB*09] DEBATTISTA K., DUBLA P., BANTERLE F., SANTOS L. P., CHALMERS A.: Instant caching for interactive global illumination. *Computer Graphics Forum* 28, 8 (2009), 2216–2228.
- [DDC09] DUBLA P., DEBATTISTA K., CHALMERS A.: Adaptive interleaved sampling for interactive high-fidelity rendering. *Computer Graphics Forum* (February 2009).
- [DDSC09] DUBLA P., DEBATTISTA K., SANTOS L. P., CHALMERS A.: Wait-Free Shared-Memory Irradiance Cache. pp. 57–64.
- [Deb06] DEBATTISTA K.: *Selective Rendering for High Fidelity Graphics*. PhD in Computer science, University of Bristol, 2006.
- [Deb10] DEBEVEC P.: High-resolution light probes, November 2010.
- [DGR*09] DONG Z., GROSCH T., RITSCHER T., KAUTZ J., SEIDEL H.-P.: Real-time indirect illumination with clustered visibility. In *Proceedings of the Vision, Modeling, and Visualization Workshop 2009 (VMV 2009)* (Braunschweig, Germany, 2009), Magnor M., Rosenhahn B., Theisel H., (Eds.), Otto-Von-Guericke-Universitt, pp. 187–196.
- [DHK08] DAMMERTZ H., HANIKA J., KELLER A.: Shallow bounding volume hierarchies for fast SIMD ray tracing of incoherent rays. In *Computer Graphics Forum (Proc. 19th Eurographics Symposium on Rendering)* (2008), pp. 1225–1234.
- [DHS04] DMITRIEV K., HAVRAN V., SEIDEL H.-P.: *Faster Ray Tracing with SIMD Shaft Culling*. Research Report MPI-I-2004-4-006, Max-Planck-Institut fr Informatik, Saarbrücken, Germany, December 2004.

- [DHW*07] DJEU P., HUNT W., WANG R., ELHASSAN I., STOLL G., MARK W. R.: Razor: An architecture for dynamic multiresolution ray tracing. *ACM Trans. Graph.* (2007).
- [Dij68] DIJKSTRA E. W.: The structure of the “the”-multiprogramming system. *Commun. ACM* 11, 5 (1968), 341–346.
- [DKTS07] DONG Z., KAUTZ J., THEOBALT C., SEIDEL H.-P.: Interactive global illumination using implicit visibility. In *Pacific Conference on Computer Graphics and Applications* (Washington, DC, USA, 2007), IEEE Computer Society.
- [DL06] DONNELLY W., LAURITZEN A.: Variance shadow maps. In *I3D ’06: Proceedings of the 2006 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2006), ACM, pp. 161–165.
- [DPH*03] DEMARLE D. E., PARKER S., HARTNER M., GRIBBLE C., HANSEN C.: Distributed interactive ray tracing for large volume visualization. In *PVG ’03: Proceedings of the 2003 IEEE Symposium on Parallel and Large-Data Visualization and Graphics* (Washington, DC, USA, 2003), IEEE Computer Society, p. 12.
- [DS97] DRETTAKIS G., SILLION F. X.: Interactive update of global illumination using a line-space hierarchy. In *Proceedings of SIGGRAPH 97* (Aug. 1997), Computer Graphics Proceedings, Annual Conference Series, pp. 57–64.
- [DS03] DACHSBACHER C., STAMMINGER M.: Translucent shadow maps. In *EGRW ’03: Proceedings of the 14th Eurographics workshop on Rendering* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 197–201.
- [DS05] DACHSBACHER C., STAMMINGER M.: Reflective shadow maps. In *I3D ’05: Proceedings of the 2005 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2005), ACM, pp. 203–231.
- [DS06] DACHSBACHER C., STAMMINGER M.: Splatting indirect illumination. In *I3D ’06: Proceedings of the 2006 symposium on Interactive*

- 3D graphics and games* (New York, NY, USA, 2006), ACM, pp. 93–100.
- [DSDD07] DACHSBACHER C., STAMMINGER M., DRETTAKIS G., DURAND F.: Implicit visibility and antiradiance for interactive global illumination. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers* (New York, NY, USA, 2007), ACM, p. 61.
- [DSSC05] DEBATTISTA K., SUNDSTEDT V., SANTOS L. P., CHALMERS A.: Selective component-based rendering. In *GRAPHITE, 3rd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia* (November 2005), ACM Press, pp. 13–22.
- [DWS*88] DEERING M., WINNER S., SCHEDIWY B., DUFFY C., HUNT N.: The triangle processor and normal vector shader: a vlsi system for high performance graphics. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1988), ACM, pp. 21–30.
- [DWWL05] DAYAL A., WOOLLEY C., WATSON B., LUEBKE D. P.: Adaptive frameless rendering. In *Rendering Techniques* (2005), pp. 265–275.
- [EG08] ERNST M., GREINER G.: Multi bounding volume hierarchies. In *Interactive Ray Tracing, 2008. RT 2008. IEEE Symposium on* (Aug. 2008), pp. 35–40.
- [EMDT06] ESTALELLA P., MARTIN I., DRETTAKIS G., TOST D.: A gpu-driven algorithm for accurate interactive reflections on curved objects. In *Rendering Techniques (Proceedings of the Eurographics Symposium on Rendering)* (June 2006), Akenine-Möller T., Heidrich W., (Eds.), Eurographics/ACM SIGGRAPH.
- [Eve01] EVERITT C.: Interactive order-independent transparency, 2001.
- [EVG04] ERNST M., VOGELGSANG C., GREINER G.: Stack implementation on programmable graphics hardware. In *VMV* (2004), pp. 255–262.

- [FBP08] FOREST V., BARTHE L., PAULIN M.: Accurate Shadows by Depth Complexity Sampling. *Computer Graphics Forum, Eurographics 2008 Proceedings* (2008).
- [FD09] FABIANOWSKI B., DINGLIANA J.: Interactive global photon mapping. *Computer Graphics Forum* 28, 4 (2009), 1151–1159.
- [FHS04] FICH F., HENDLER D., SHAVIT N.: On the inherent weakness of conditional synchronization primitives. In *In Proceedings of the 23rd Annual ACM Symposium on Principles of Distributed Computing* (2004), ACM Press, pp. 80–87.
- [FKN80] FUCHS H., KEDEM Z. M., NAYLOR B. F.: On visible surface generation by a priori tree structures. In *SIGGRAPH '80: Proceedings of the 7th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1980), ACM, pp. 124–133.
- [FP04] FARRUGIA J.-P., PÉROCHE B.: A progressive rendering algorithm using an adaptive perceptually based image metric. *Comput. Graph. Forum* 23, 3 (2004), 605–614.
- [FS05] FOLEY T., SUGERMAN J.: Kd-tree acceleration structures for a gpu raytracer. In *HWWS '05: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware* (New York, NY, USA, 2005), ACM, pp. 15–22.
- [FSPG97] FERWERDA J. A., SHIRLEY P., PATTANAIK S. N., GREENBERG D. P.: A model of visual masking for computer graphics. In *SIGGRAPH '97* (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 143–152.
- [FTI86] FUJIMOTO A., TANAKA T., IWATA K.: Arts: Accelerated ray-tracing system. *IEEE Computer Graphics and Applications* 6 (1986), 16–26.
- [GBP07] GAUTRON P., BOUATOUCH K., PATTANAIK S.: Temporal radiance caching. *IEEE Transactions on Visualization and Computer Graphics* 13, 5 (2007), 891–901.

- [GFW*06] GÜNTHER J., FRIEDRICH H., WALD I., SEIDEL H.-P., SLUSALLEK P.: Ray tracing animated scenes using motion decomposition. *Computer Graphics Forum* 25, 3 (Sept. 2006), 517–525. (Proceedings of Eurographics).
- [GGN*08] GARLAND M., GRAND S. L., NICKOLLS J., ANDERSON J., HARDWICK J., MORTON S., PHILLIPS E., ZHANG Y., VOLKOV V.: Parallel computing experiences with cuda. *IEEE Micro* 28 (2008), 13–27.
- [GH98] GHAZANFARPOUR D., HASENFRATZ J.-M.: A beam tracing with precise antialiasing for polyhedral scenes. *Computer Graphics* 22, 1 (1998), 103–115.
- [GKBP05] GAUTRON P., KŘIVÁNEK J., BOUATOUCH K., PATTANAIK S.: Radiance cache splatting: A GPU-friendly global illumination algorithm. In *Proceedings of Eurographics Symposium on Rendering* (June 2005).
- [Gla84] GLASSNER A. S.: Space subdivision for fast ray tracing. *IEEE Computer Graphics Applications* 4, 10 (Oct. 1984), 15–22.
- [Gla95] GLASSNER A.: *Principles of Digital Image Synthesis*. Morgan Kaufmann, 1995.
- [GPSS07] GÜNTHER J., POPOV S., SEIDEL H.-P., SLUSALLEK P.: Realtime ray tracing on GPU with BVH-based packet traversal. In *Proceedings of the IEEE/Eurographics Symposium on Interactive Ray Tracing 2007* (Sept. 2007), pp. 113–118.
- [Gre99] GREENBERG D. P.: A framework for realistic image synthesis. *Commun. ACM* 42, 8 (1999), 44–53.
- [GS87] GOLDSMITH J., SALMON J.: Automatic creation of object hierarchies for ray tracing. *Computer Graphics and Applications, IEEE* 7, 5 (May 1987), 14–20.
- [GS08] GEORGIEV I., SLUSALLEK P.: RTfact: Generic Concepts for Flexible and High Performance Ray Tracing. In *To appear in the*

- IEEE/Eurographics Symposium on Interactive Ray Tracing 2008* (Aug. 2008).
- [GTGB84] GORAL C. M., TORRANCE K. E., GREENBERG D. P., BATTAILE B.: Modeling the interaction of light between diffuse surfaces. *SIGGRAPH Comput. Graph.* 18, 3 (1984), 213–222.
- [Guo98] GUO B.: Progressive radiance evaluation using directional coherence maps. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), ACM Press, pp. 255–266.
- [Hav01] HAVRAN V.: *Heuristic Ray Shooting Algorithms*. Ph.d. thesis, Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, November 2001.
- [Hec90] HECKBERT P. S.: Adaptive radiosity textures for bidirectional ray tracing. In *SIGGRAPH '90* (1990), ACM Press, pp. 145–154.
- [Her88] HERLIHY M. P.: Impossibility and universality results for wait-free synchronization. In *PODC '88: Proceedings of the seventh annual ACM Symposium on Principles of distributed computing* (New York, NY, USA, 1988), ACM, pp. 276–290.
- [Her91] HERLIHY M.: Wait-free synchronization. *ACM Trans. Program. Lang. Syst.* 13, 1 (1991), 124–149.
- [Her04] HERY C.: Rendering evolution at industrial light & magic. In *Rendering Techniques* (2004), pp. 19–22.
- [HH84] HECKBERT P. S., HANRAHAN P.: Beam tracing polygonal objects. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1984), ACM Press, pp. 119–127.
- [HHS06] HAVRAN V., HERZOG R., SEIDEL H.-P.: On the fast construction of spatial data structures for ray tracing. *Proceedings of IEEE Symposium on Interactive Ray Tracing 2006* (Sept. 2006), 71–80.

- [HLHS03] HASENFRATZ J.-M., LAPIERRE M., HOLZSCHUCH N., SILLION F.: A survey of real-time soft shadows algorithms. In *Eurographics* (2003), Eurographics, Eurographics. State-of-the-Art Report.
- [HMS06] HUNT W., MARK W. R., STOLL G.: Fast kd-tree construction with an adaptive error-bounded heuristic. In *2006 IEEE Symposium on Interactive Ray Tracing* (Sept 2006), IEEE.
- [HMYS01] HABER J., MYSZKOWSKI K., YAMAUCHI H., SEIDEL H.-P.: Perceptually guided corrective splatting. In *EG 2001 Proceedings*, Chalmers A., Rhyne T.-M., (Eds.), vol. 20(3). Blackwell Publishing, 2001, pp. 142–152.
- [HS08] HERLIHY M., SHAVIT N.: *The Art of Multiprocessor Programming*. Morgan Kaufmann, March 2008.
- [HSA91] HANRAHAN P., SALZMAN D., AUPPERLE L.: A rapid hierarchical radiosity algorithm. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1991), ACM Press, pp. 197–206.
- [HSHH07] HORN D. R., SUGERMAN J., HOUSTON M., HANRAHAN P.: Interactive k-d tree gpu raytracing. In *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2007), ACM, pp. 167–174.
- [IDYN07] IWASAKI K., DOBASHI Y., YOSHIMOTO F., NISHITA T.: Pre-computed radiance transfer for dynamics scene taking into account light interreflection. In *Eurographics Symposium on Rendering 2007* (2007), Eurographics.
- [Ige99] IGEHY H.: Tracing ray differentials. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 179–186.
- [IKN98] ITTI L., KOCH C., NIEBUR E.: A model of Saliency-Based Visual Attention for Rapid Scene Analysis. In *Pattern Analysis and Machine Intelligence* (1998), vol. 20, pp. 1254–1259.

- [Int03] INTEL: *Software Developer's Manual Volume 1: Basic Architecture*. Tech. rep., Intel Corporation, 2003.
- [JC98] JENSEN H. W., CHRISTENSEN P. H.: Efficient simulation of light transport in scenes with participating media using photon maps. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), ACM, pp. 311–320.
- [JDZJ08] JAROSZ W., DONNER C., ZWICKER M., JENSEN H. W.: Radiance caching for participating media. *ACM Trans. Graph.* 27, 1 (2008), 1–11.
- [Jen01] JENSEN H. W.: *Realistic Image Synthesis Using Photon Mapping*. AK Peters, 2001.
- [JIC*09] JIN B., IHM I., CHANG B., PARK C., LEE W., JUNG S.: Selective and adaptive supersampling for real-time ray tracing. In *HPG '09: Proceedings of the Conference on High Performance Graphics 2009* (New York, NY, USA, 2009), ACM, pp. 117–125.
- [Kaj86] KAJIYA J. T.: The rendering equation. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1986), ACM Press, pp. 143–150.
- [KBPv06] KŘIVÁNEK J., BOUATOUCH K., PATTANAIK S. N., ŽÁRA J.: Making radiance and irradiance caching practical: Adaptive caching and neighbor clamping. In *Rendering Techniques 2006, Eurographics Symposium on Rendering* (Nicosia, Cyprus, June 2006), Akenine-Mller T., Heidrich W., (Eds.), Eurographics Association, Eurographics Association.
- [KCLU07] KOPF J., COHEN M., LISCHINSKI D., UYTENDAELE M.: Joint bilateral upsampling. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)* 26, 3 (2007), to appear.
- [Kel97] KELLER A.: Instant radiosity. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interac-*

- tive techniques* (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 49–56.
- [Kel98] KELLER A.: *Quasi-Monte Carlo Methods for Photorealistic Image Synthesis*. PhD thesis, Shaker Verlag Aachen, 1998.
- [KGBP05] KŘIVÁNEK J., GAUTRON P., BOUATOUCH K., PATTANAIK S.: Improved radiance gradient computation. In *SCCG '05: Proceedings of the 21st spring conference on Computer graphics* (New York, NY, USA, 2005), ACM Press, pp. 155–159.
- [KGPB05] KŘIVÁNEK J., GAUTRON P., PATTANAIK S., BOUATOUCH K.: Radiance caching for efficient global illumination computation. *IEEE Transactions on Visualization and Computer Graphics* 11, 5 (2005), 550–561.
- [KGW*07] KŘIVÁNEK J., GAUTRON P., WARD G., ARIKAN O., JENSEN H. W.: Practical global illumination with irradiance caching. In *ACM SIGGRAPH 2007 courses* (New York, NY, USA, 2007), SIGGRAPH '07, ACM.
- [KH84] KAJIYA J. T., HERZEN B. P. V.: Ray tracing volume densities. *SIGGRAPH Comput. Graph.* 18, 3 (1984), 165–174.
- [KH95] KEATES M. J., HUBBOLD R. J.: Interactive ray tracing on a virtual shared-memory parallel computer. *Comput. Graph. Forum* 14, 4 (1995), 189–202.
- [KH01] KELLER A., HEIDRICH W.: Interleaved sampling. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques* (London, UK, 2001), Springer-Verlag, pp. 269–276.
- [KK02] KOLLIG T., KELLER A.: Efficient multidimensional sampling. *Computer Graphics Forum* 21, 3 (Sept. 2002), 557–563.
- [KLC06] KENG S.-L., LEE W.-Y., CHUANG J.-H.: An efficient caching-based rendering of translucent materials. *Vis. Comput.* 23, 1 (2006), 59–69.

- [KMG99] KOHOLKA R., MAYER H., GOLLER A.: Mpi-parallelized radiance on sgi cow and smp. In *ParNum '99: Proceedings of the 4th International ACPC Conference Including Special Tracks on Parallel Numerics and Parallel Computing in Image Processing, Video Processing, and Multimedia* (1999), Springer-Verlag, pp. 549–558.
- [KO08] KI H., OH K.: A gpu-based light hierarchy for real-time approximate illumination. *The Visual Computer* 24, 7-9 (July 2008), 649–658.
- [Lai10] LAINE S.: Restart trail for stackless BVH traversal. In *Proceedings of High-Performance Graphics 2010* (2010).
- [Lam60] LAMBERT J. H.: *I.H. Lambert Photometria, sive, De mensura et gradibus luminis, colorum et umbrae [microform]*. V.E. Klett, Augustae Vindelicorum :, 1760.
- [LC04] LARSEN B. D., CHRISTENSEN N.: Simulating photon mapping for real-time applications. In *Eurographics Symposium on Rendering* (jun 2004), Henrik Wann Jensen A. K., (Ed.).
- [LGS*09] LAUTERBACH C., GARLAND M., SENGUPTA S., LUEBKE D., MANOCHA D.: Fast bvh construction on gpus. *Comput. Graph. Forum* 28, 2 (2009), 375–384.
- [LH01] LUEBKE D. P., HALLEN B.: Perceptually-driven simplification for interactive rendering. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques* (London, UK, 2001), Springer-Verlag, pp. 223–234.
- [LKC09] LEE S., KIM G. J., CHOI S.: Real-time tracking of visually attended objects in virtual environments and its application to lod. *IEEE Transactions on Visualization and Computer Graphics* 15 (2009), 6–19.
- [LLAm01] LEXT J., , LEXT J., AKENINE-MLLER T.: Eurographics 2001 / jonathan c. roberts short presentations towards rapid reconstruction for animated ray tracing, 2001.

- [LSK*07] LAINE S., SARANSAARI H., KONTKANEN J., LEHTINEN J., AILA T.: Incremental instant radiosity for real-time indirect illumination. In *Proceedings of Eurographics Symposium on Rendering 2007* (2007), Eurographics Association, pp. xx–yy.
- [LSS04] LIU X., SLOAN P.-P. J., SHUM H.-Y., SNYDER J.: All-frequency precomputed radiance transfer for glossy objects. In *Rendering Techniques* (2004), pp. 337–344.
- [LW93] LAFORTUNE E. P., WILLEMS Y. D.: Bidirectional Path Tracing. In *3rd International Conference on Computational Graphics and Visualization Techniques* (Alvor, Portugal, 1993), pp. 145–153.
- [LWC*02] LUEBKE D., WATSON B., COHEN J. D., REDDY M., VARSHNEY A.: *Level of Detail for 3D Graphics*. Elsevier Science Inc., New York, NY, USA, 2002.
- [LYTM06] LAUTERBACH C., YOON S.-E., TUFT D., MANOCHA D.: RT-DEFORM: Interactive Ray Tracing of Dynamic Scenes using BVHs. In *2006 IEEE Symposium on Interactive Ray Tracing* (2006), pp. 39–45.
- [MB90] MACDONALD D. J., BOOTH K. S.: Heuristics for ray tracing using space subdivision. *Vis. Comput.* 6, 3 (1990), 153–166.
- [MCTR98] McNAMARA A., CHALMERS A., TROSCIANKO T., REINHARD E.: Fidelity of graphics reconstructions: A psychophysical investigation. In *Proceedings of the 9th Eurographics Rendering Workshop* (June 1998), Springer Verlag, pp. 237–246.
- [MDMS05] MANTIUK R., DALY S., MYSZKOWSKI K., SEIDEL H.-P.: Predicting Visible Differences in High Dynamic Range Images - Model and its Calibration. *Human Vision and Electronic Imaging X, IS&T/SPIE's 17th Annual Symposium on Electronic Imaging* (2005), 204–214.
- [Mit87] MITCHELL D. P.: Generating antialiased images at low sampling densities. In *SIGGRAPH '87* (1987), ACM Press, pp. 65–72.

- [Mit07] MITTRING M.: Finding next gen: Cryengine 2. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses* (New York, NY, USA, 2007), ACM, pp. 97–121.
- [ML92] MEYER G., LIU A.: Color spatial acuity control of a screen subdivision image synthesis algorithm. In *In Human Vision, Visual Processing, and Digital Display III* (1992).
- [ML09] MCGUIRE M., LUEBKE D.: Hardware-accelerated global illumination by image space photon mapping. In *Proceedings of the 2009 ACM SIGGRAPH/EuroGraphics conference on High Performance Graphics* (New York, NY, USA, August 2009), ACM.
- [MM02] MA V. C. H., MCCOOL M. D.: Low latency photon mapping using block hashing. In *HWWS '02: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware* (Aire-la-Ville, Switzerland, Switzerland, 2002), Eurographics Association, pp. 89–99.
- [MMAM07] MANSSON E., MUNKBERG J., AKENINE-MOLLER T.: Deep coherent ray tracing. In *Interactive Ray Tracing, 2007. RT '07. IEEE Symposium on* (Sept. 2007), pp. 79–85.
- [Mun08] MUNSHI A.: OpenCL. *Parallel Computing on the GPU and CPU, SIGGRAPH* (2008).
- [Muu95] MUUSS M. J.: Towards real-time ray-tracing of combinatorial solid geometric models. In *Proceedings of BRL-CAD Symposium '95* (June 1995).
- [Mys98] MYSZKOWSKI K.: The visible differences predictor: Applications to global illumination problems. In *Rendering Techniques* (1998), pp. 223–236.
- [NFLM07] NAVRATIL P., FUSSELL D., LIN C., MARK W.: Dynamic ray scheduling to improve ray coherence and bandwidth utilization. In *Interactive Ray Tracing, 2007. RT '07. IEEE Symposium on* (Sept. 2007), pp. 95–104.

- [Nic65] NICODEMUS F. E.: Directional reflectance and emissivity of an opaque surface. *Appl. Opt.* 4, 7 (1965), 767–773.
- [NMN87] NISHITA T., MIYAWAKI Y., NAKAMAE E.: A shading model for atmospheric scattering considering luminous intensity distribution of light sources. *SIGGRAPH Comput. Graph.* 21, 4 (1987), 303–310.
- [NO97] NAKAMARU K., OHNO Y.: Breadth-first ray tracing utilizing uniform spatial subdivision. *IEEE Transactions on Visualization and Computer Graphics* 3, 4 (1997), 316–328.
- [NPG03] NIJASURE M., PATTANAIK S., GOEL V.: Interactive global illumination in dynamic environments using commodity graphics hardware. *Computer Graphics and Applications, Pacific Conference on 0* (2003), 450.
- [NPG04] NIJASURE M., PATTANAIK S., GOEL V.: Real-time global illumination on the gpu. *JOURNAL OF GRAPHICS TOOLS* 10 (2004), 55–71.
- [NW09] NICHOLS G., WYMAN C.: Multiresolution splatting for indirect illumination. In *I3D '09: Proceedings of the 2009 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2009), ACM, pp. 83–90.
- [OB07] OLIVEIRA M. M., BRAUWERS M.: Real-time refraction through deformable objects. In *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2007), ACM, pp. 89–96.
- [OHM*04] O’SULLIVAN C., HOWLETT S., McDONNELL R., MORVAN Y., O’CONOR K.: Perceptually adaptive graphics. In *Eurographics State of the Art Reports* (2004).
- [OR98] OFEK E., RAPPOPORT A.: Interactive reflections on curved objects. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), ACM Press, pp. 333–342.

- [ORDP96] ORTI R., RIVIÈRE S., DURAND F., PUECH C.: Radiosity for dynamic scenes in flatland with the visibility complex. In *Computer Graphics Forum (Proc. of Eurographics '96)* (Poitiers, France, Sep 1996), Rossignac J., Sillion F., (Eds.), vol. 16, pp. 237–249.
- [ORM08] OVERBECK R., RAMAMOORTHY R., MARK W. R.: Large Ray Packets for Real-time Whitted Ray Tracing. In *IEEE/EG Symposium on Interactive Ray Tracing (IRT)* (Aug 2008), pp. 41—48.
- [Pat93] PATTANAIK S. N.: *Computational Methods for Global Illumination and Visualisation of Complex 3D Environments*. PhD thesis, National Institute for Software Technology, Bombay, February 1993.
- [PBD*10] PARKER S. G., BIGLER J., DIETRICH A., FRIEDRICH H., HOBEROCK J., LUEBKE D., MCALLISTER D., MCGUIRE M., MORLEY K., ROBISON A., STICH M.: Optix: A general purpose ray tracing engine. *ACM Transactions on Graphics* (August 2010).
- [PBMH02] PURCELL T. J., BUCK I., MARK W. R., HANRAHAN P.: Ray tracing on programmable graphics hardware. *ACM Transactions on Graphics* 21, 3 (July 2002), 703–712. ISSN 0730-0301 (Proceedings of ACM SIGGRAPH 2002).
- [PDC*03] PURCELL T. J., DONNER C., CAMMARANO M., JENSEN H. W., HANRAHAN P.: Photon mapping on programmable graphics hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware* (2003), Eurographics Association, pp. 41–50.
- [Per85] PERLIN K.: An image synthesizer. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1985), ACM Press, pp. 287–296.
- [PGSS06] POPOV S., GÜNTHER J., SEIDEL H.-P., SLUSALLEK P.: Experiences with streaming construction of SAH KD-trees. In *Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing* (Sept. 2006), pp. 89–94.

- [PGSS07] POPOV S., GÜNTHER J., SEIDEL H.-P., SLUSALLEK P.: Stack-less kd-tree traversal for high performance GPU ray tracing. *Computer Graphics Forum* 26, 3 (Sept. 2007), 415–424. (Proceedings of Eurographics).
- [PH04] PHARR M., HUMPHREYS G.: *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.
- [PKG97] PHARR M., KOLB C., GERSHBEIN R., HANRAHAN P.: Rendering complex scenes with memory-coherent ray tracing. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 101–108.
- [PL10] PANTALEONI J., LUEBKE D.: Hlbvh: Hierarchical lbvh construction for real-time ray tracing of dynamic geometry. In *Proceedings of High-Performance Graphics 2010* (2010).
- [PMS*99] PARKER S., MARTIN W., SLOAN P.-P. J., SHIRLEY P., SMITS B., HANSEN C.: Interactive Ray Tracing. In *1999 Symposium Interactive 3D Computer Graphics* (1999), pp. 119–126.
- [PPL*99] PARKER S., PARKER M., LIVNAT Y., SLOAN P.-P., HANSEN C., SHIRLEY P. S.: Interactive ray tracing for volume visualization. *IEEE Transactions on Visualization and Computer Graphics* 5, 3 (July/Sept. 1999), 238–250.
- [PS89] PAINTER J., SLOAN K.: Antialiased ray tracing by adaptive progressive refinement. In *SIGGRAPH '89* (New York, NY, USA, 1989), ACM Press, pp. 281–288.
- [PSL*98] PARKER S., SHIRLEY P. S., LIVNAT Y., HANSEN C., SLOAN P.-P.: Interactive ray tracing for isosurface rendering. In *IEEE Visualization '98* (Oct. 1998), pp. 233–238.
- [PWL*07] PAN M., WANG R., LIU X., PENG Q., BAO H.: Precomputed radiance transfer field for rendering interreflections in dynamic scenes. *Computer Graphics Forum* 26, 3 (2007).

- [RAH07] ROGER D., ASSARSSON U., HOLZSCHUCH N.: Whitted ray-tracing for dynamic scenes using a ray-space hierarchy on the gpu. In *Rendering Techniques 2007 (Proceedings of the Eurographics Symposium on Rendering)* (jun 2007), Kautz J., Pattanaik S., (Eds.), Eurographics and ACM/SIGGRAPH, the Eurographics Association, pp. 99–110.
- [RCLL99] ROBERTSON D., CAMPBELL K., LAU S., LIGOCKI T.: Parallelization of radiance for real time interactive lighting visualization walkthroughs. In *Supercomputing '99: Proceedings of the 1999 ACM/IEEE conference on Supercomputing (CDROM)* (1999), ACM Press, p. 61.
- [REG*09] RITSCHER T., ENGELHARDT T., GROSCH T., SEIDEL H.-P., KAUTZ J., DACHSBACHER C.: Micro-rendering for scalable, parallel final gathering. *ACM Trans. Graph. (Proc. SIGGRAPH Asia 2009)* 28, 5 (2009).
- [Res06] RESHETOV A.: Omnidirectional ray tracing traversal algorithm for kd-trees. In *Interactive Ray Tracing 2006, IEEE Symposium on* (Sept. 2006), pp. 57–60.
- [Res07] RESHETOV A.: Faster ray packets - triangle intersection through vertex culling. In *Interactive Ray Tracing, 2007. RT '07. IEEE Symposium on* (Sept. 2007), pp. 105–112.
- [RGK*08] RITSCHER T., GROSCH T., KIM M. H., SEIDEL H.-P., DACHSBACHER C., KAUTZ J.: Imperfect shadow maps for efficient computation of indirect illumination. *ACM Trans. Graph.* 27, 5 (2008), 1–8.
- [RGKS08] RITSCHER T., GROSCH T., KAUTZ J., SEIDEL H.-P.: Interactive global illumination based on coherent surface shadow maps. In *GI '08: Proceedings of graphics interface 2008* (Toronto, Ont., Canada, Canada, 2008), Canadian Information Processing Society, pp. 185–192.
- [RGS09] RITSCHER T., GROSCH T., SEIDEL H.-P.: Approximating dynamic global illumination in image space. In *I3D '09: Proceedings*

- of the 2009 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2009), ACM, pp. 75–82.
- [RPG99] RAMASUBRAMANIAN M., PATTANAIK S. N., GREENBERG D. P.: A perceptually based physical error metric for realistic image synthesis. In *SIGGRAPH '99* (1999), ACM Press/Addison-Wesley Publishing Co., pp. 73–82.
- [RSH00] REINHARD E., SMITS B. E., HANSEN C.: Dynamic acceleration structures for interactive ray tracing. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000* (London, UK, 2000), Springer-Verlag, pp. 299–306.
- [RSH05] RESHETOV A., SOUPIKOV A., HURLEY J.: Multi-level ray tracing algorithm. *ACM Trans. Graph.* 24, 3 (2005), 1176–1185.
- [RW80] RUBIN S. M., WHITTET T.: A 3-dimensional representation for fast rendering of complex scenes. In *SIGGRAPH '80: Proceedings of the 7th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1980), ACM, pp. 110–116.
- [RWPD05] REINHARD E., WARD G., PATTANAIK S., DEBEVEC P.: Morgan Kaufmann Publishers, December 2005.
- [SBB*06] STEPHENS A., BOULOS S., BIGLER J., WALD I., PARKER S. G.: An Application of Scalable Massive Model Interaction using Shared Memory Systems. In *Proceedings of the 2006 Eurographics Symposium on Parallel Graphics and Visualization* (2006), pp. 19–26.
- [Sch06] SCHMITTLER J.: *SaarCOR - A Hardware-Architecture for Realtime Ray Tracing*. PhD thesis, Saarland University, 2006.
- [SCM04] SUNDSTEDT V., CHALMERS A., MARTINEZ P.: High fidelity reconstruction of the ancient egyptian temple of kalabsha. In *AFRIGRAPH 2004* (November 2004), ACM SIGGRAPH.
- [SFWG04] STOKES W. A., FERWERDA J. A., WALTER B., GREENBERG D. P.: Perceptual illumination components: a new approach to efficient, high quality global illumination rendering. *ACM Trans. Graph.* 23, 3 (2004), 742–749.

- [SGNS07] SLOAN P.-P., GOVINDARAJU N. K., NOWROUZEZAHRAI D., SNYDER J.: Image-based proxy accumulation for real-time soft global illumination. In *PG '07: Proceedings of the 15th Pacific Conference on Computer Graphics and Applications* (Washington, DC, USA, 2007), IEEE Computer Society, pp. 97–105.
- [SHHS03] SLOAN P.-P., HALL J., HART J., SNYDER J.: Clustered principal components for precomputed radiance transfer. *ACM Trans. Graph.* 22, 3 (2003), 382–391.
- [Shi90] SHIRLEY P.: A ray tracing method for illumination calculation in diffuse-specular scenes. In *Proceedings of Graphics Interface '90* (Toronto, Ontario, 1990), Canadian Information Processing Society, pp. 205–12.
- [SHSS00] STAMMINGER M., HABER J., SCHIRMACHER H., SEIDEL H.-P.: Walkthroughs with corrective texturing. In *Rendering Techniques 2000 (Proceedings of the Eleventh Eurographics Workshop on Rendering)* (New York, NY, 2000), Peroche B., Rushmeier H., (Eds.), Springer Wien, pp. 377–388.
- [SIMP06a] SEGOVIA B., IEHL J.-C., MITANCHEY R., PÉROCHE B.: Bidirectional instant radiosity. In *Proceedings of the 17th Eurographics Workshop on Rendering, to appear* (2006).
- [SIMP06b] SEGOVIA B., IEHL J.-C., MITANCHEY R., PÉROCHE B.: Non-interleaved deferred shading of interleaved sample patterns. In *Proceedings of SIGGRAPH/Eurographics Workshop on Graphics Hardware 2006, to appear* (2006).
- [SIP07] SEGOVIA B., IEHL J.-C., PÉROCHE B.: Metropolis instant radiosity. In *Proceedings of Eurographics 2007, to appear* (2007).
- [SKDM05] SMYK M., KINUWAKI S., DURIKOVIC R., MYSKOWSKI K.: Temporally coherent irradiance caching for high quality animation rendering. *Computer Graphics Forum* 24, 3 (2005), 401–412.
- [SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting

- environments. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), ACM Press, pp. 527–536.
- [SM03] SHIRLEY P., MORLEY R. K.: *Realistic Ray Tracing*. A. K. Peters, Ltd., Natick, MA, USA, 2003.
- [SP89] SILLION F. X., PUECH C.: A general two-pass method integrating specular and diffuse reflection. In *SIGGRAPH '89* (1989), ACM Press, pp. 335–344.
- [SP94] SILLION F. X., PUECH C.: *Radiosity and Global Illumination*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994.
- [SSH*98] SLUSALLEK P., STAMMINGER M., HEIDRICH W., POPP J.-C., SEIDEL H.-P.: Composite lighting simulations with lighting network. *IEEE Computer Graphics and Applications* 18, 2 (/1998), 22–31.
- [SSK07] SHEVTSOV M., SOUPIKOV A., KAPUSTIN A.: Highly parallel fast kd-tree construction for interactive ray tracing of dynamic scenes. *ACM Trans. Graph.* 26, 3 (2007).
- [SSW*06] SHIRLEY P., SLUSSALLEK P., WALD I., MARK W., STOLL G., MINOCHA D., STEPHENS A.: Interactive ray tracing. In *SIGGRAPH '06: Proceedings of the conference on SIGGRAPH 2006 course notes* (New York, NY, USA, 2006), ACM Press.
- [TL04] TABELLION E., LAMORLETTE A.: An approximate global illumination system for computer generated films. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM, pp. 469–476.
- [TM98] TOMASI C., MANDUCHI R.: Bilateral filtering for gray and color images. *iccv 00* (1998), 839.
- [TMS04] TAWARA T., MYSZKOWSKI K., SEIDEL H.-P.: Exploiting temporal coherence in final gathering for dynamic scenes. *cgi 00* (2004), 110–119.

- [TPWG02] TOLE P., PELLACINI F., WALTER B., GREENBERG D.: Interactive Global Illumination in Dynamic Scenes. In *SIGGRAPH'02* (2002), ACM Press.
- [Tsa09] TSAKOK J. A.: Faster incoherent rays: Multi-bvh ray stream tracing. In *HPG '09: Proceedings of the Conference on High Performance Graphics 2009* (New York, NY, USA, 2009), ACM, pp. 151–158.
- [TSr05] THRANE N., SIMONSEN L. O., RBK A. P.: *A comparison of acceleration structures for GPU assisted ray tracing*. Master's thesis, 2005.
- [VALBW06] VELAZQUEZ-ARMENDIZ E., LEE E., BALA K., WALTER B.: Implementing the render cache and the edge-and-point image on graphics hardware. In *GI '06: Proceedings of the 2006 conference on Graphics interface* (Toronto, Ont., Canada, Canada, 2006), Canadian Information Processing Society, pp. 211–217.
- [VG94] VEACH E., GUIBAS L. J.: Bidirectional Estimators for Light Transport. In *Fifth Eurographics Workshop on Rendering* (1994).
- [VG97] VEACH E., GUIBAS L. J.: Metropolis light transport. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 65–76.
- [WABG06] WALTER B., ARBREE A., BALA K., GREENBERG D. P.: Multi-dimensional lightcuts. *ACM Trans. Graph.* 25, 3 (2006), 1081–1088.
- [Wal04] WALD I.: Realtime Ray Tracing and Interactive Global Illumination. *PhD thesis, Saarland University* (2004).
- [Wal07] WALD I.: On fast Construction of SAH based Bounding Volume Hierarchies. In *SIGGRAPH '07* (2007).
- [War91a] WARD G.: Real pixels. *Graphics Gems 2* (1991), 15–31.
- [War91b] WARD G. J.: Adaptive shadow testing for ray tracing. In *2nd Annual Eurographics Workshop on Rendering* (1991), pp. 11–20.

- [War94] WARD G. J.: The radiance lighting simulation and rendering system. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1994), ACM Press, pp. 459–472.
- [Wat93] WATT A.: *3d Computer Graphics*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1993.
- [WBB08] WALD I., BENTHIN C., BOULOS S.: Getting rid of packets - efficient simd single-ray traversal using multi-branching bvhs -. In *Interactive Ray Tracing, 2008. RT 2008. IEEE Symposium on* (Aug. 2008), pp. 49–57.
- [WBS02] WALD I., BENTHIN C., SLUSALLEK P.: *A Simple and Practical Method for Interactive Ray Tracing of Dynamic Scenes*. Tech. rep., Saarland University, Germany, 2002.
- [WBS03] WALD I., BENTHIN C., SLUSALLEK P.: Interactive global illumination in complex and highly occluded environments. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 74–81.
- [WBS07] WALD I., BOULOS S., SHIRLEY P.: Ray tracing deformable scenes using dynamic bounding volume hierarchies. *ACM Trans. Graph.* 26, 1 (2007), 6.
- [WCG87] WALLACE J. R., COHEN M. F., GREENBERG D. P.: A two-pass solution to the rendering equation: A synthesis of ray tracing and radiosity methods. In *SIGGRAPH '87* (1987), ACM Press, pp. 311–320.
- [WDG02] WALTER B., DETTRAKIS G., GREENBERG D. P.: Enhancing and Optimizing the Render Cache. In *Thirteenth Eurographics Workshop on Rendering* (2002).
- [WDP99] WALTER B., DRETTAKIS G., PARKER S.: Interactive rendering using the render cache. In *Rendering techniques '99* (June 1999), Lischinski D., Larson G., (Eds.), vol. 10, pp. 235–246.

- [WFA*05] WALTER B., FERNANDEZ S., ARBREE A., BALA K., DONIKIAN M., GREENBERG D. P.: Lightcuts: a scalable approach to illumination. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers* (2005), ACM Press, pp. 1098–1107.
- [WGBK07] WALD I., GRIBBLE C. P., BOULOS S., KENSLER A.: *SIMD Ray Stream Tracing - SIMD Ray Traversal with Generalized Ray Packets and On-the-fly Re-Ordering*. Tech. Rep. UUSCI-2007-012, SCI Institute, University of Utah, 2007.
- [WH92] WARD G., HECKBERT P.: Irradiance Gradients. In *3rd Annual Eurographics Workshop on Rendering* (Bristol, UK, 1992).
- [Whi80] WHITTED T.: An improved illumination model for shaded display. In *SIGGRAPH '80* (1980), ACM Press, p. 14.
- [WIK*06] WALD I., IZE T., KENSLER A., KNOLL A., PARKER S. G.: Ray Tracing Animated Scenes using Coherent Grid Traversal. *ACM Transactions on Graphics* (2006), 485–493. (Proceedings of ACM SIGGRAPH 2006).
- [Wil78] WILLIAMS L.: Casting curved shadows on curved surfaces. In *SIGGRAPH '78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1978), ACM Press, pp. 270–274.
- [WIP08] WALD I., IZE T., PARKER S. G.: Fast, parallel, and asynchronous construction of bvhs for ray tracing animated scenes. *Computers & Graphics* 32, 1 (Feb. 2008), 3–13.
- [WK06] WÄCHTER C., KELLER A.: Instant Ray Tracing: The Bounding Interval Hierarchy. In *Rendering Techniques 2006 (Proc. of 17th Eurographics Symposium on Rendering)* (2006), Akenine-Möller T., Heidrich W., (Eds.), pp. 139–149.
- [WKB*02] WALD I., KOLLIG T., BENTHIN C., KELLER A., SLUSALLEK P.: Interactive global illumination using fast ray tracing. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering* (Aire-la-Ville, Switzerland, Switzerland, 2002), Eurographics Association, pp. 15–24.

- [WLWD03] WOOLLEY C., LUEBKE D., WATSON B., DAYAL A.: Interruptible rendering. In *SI3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics* (New York, NY, USA, 2003), ACM Press, pp. 143–151.
- [WRC88] WARD G. J., RUBINSTEIN F. M., CLEAR R. D.: A ray tracing solution for diffuse interreflection. In *SIGGRAPH '88* (1988), ACM Press, pp. 85–92.
- [WS99] WARD G., SIMMONS M.: The holodeck ray cache: an interactive rendering system for global illumination in nondiffuse environments. *ACM Trans. Graph.* 18, 4 (1999), 361–368.
- [WSB01] WALD I., SLUSALLEK P., BENTHIN C.: Interactive Distributed Ray Tracing of Highly Complex Models. In *12th EUROGRAPHICS Workshop on Rendering* (London, United Kingdom, June 2001), pp. 274–285.
- [WSBW01] WALD I., SLUSALLEK P., BENTHIN C., WAGNER M.: Interactive Rendering With Coherent Raytracing. In *EUROGRAPHICS 2001* (Manchester, United Kingdom, September 2001), pp. 153–164.
- [WTL06] WANG R., TRAN J., LUEBKE D.: All-frequency relighting of glossy objects. *ACM Trans. Graph.* 25, 2 (2006), 293–318.
- [Wym05] WYMAN C.: An approximate image-space approach for interactive refraction. *ACM Trans. Graph.* 24, 3 (2005), 1050–1053.
- [Wym08] WYMAN C.: Hierarchical caustic maps. In *I3D '08: Proceedings of the 2008 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2008), ACM, pp. 163–171.
- [WZPB09] WANG R., ZHOU K., PAN M., BAO H.: An efficient gpu-based approach for interactive global illumination. In *SIGGRAPH '09* (2009).
- [YCM07] YOON S.-E., CURTIS S., MANOCHA D.: Ray tracing dynamic scenes using selective restructuring. In *SIGGRAPH '07: ACM SIGGRAPH 2007 sketches* (New York, NY, USA, 2007), ACM, p. 55.

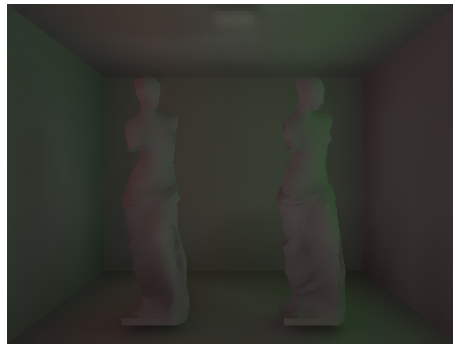
- [YPG01] YEE H., PATTANAIAK S., GREENBERG D. P.: Spatiotemporal sensitivity and visual attention for efficient rendering of dynamic environments. *ACM Trans. Graph.* 20, 1 (2001), 39–65.
- [YWC*10] YAO C., WANG B., CHAN B., YONG J., PAUL J.-C.: Multi-image based photon tracing for interactive global illumination of dynamic scenes. *Computer Graphics Forum* 29 (June 2010), 1315–1324(10).
- [YYM05] YU J., YANG J., McMILLAN L.: Real-time reflection mapping with parallax. In *I3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2005), ACM, pp. 133–138.
- [ZHL*05] ZHOU K., HU Y., LIN S., GUO B., SHUM H.-Y.: Precomputed shadow fields for dynamic scenes. *ACM Trans. Graph.* 24, 3 (2005), 1196–1201.
- [ZHWG08] ZHOU K., HOU Q., WANG R., GUO B.: Real-time kd-tree construction on graphics hardware. *ACM Trans. Graph.* 27, 5 (2008), 1–11.

APPENDIX A

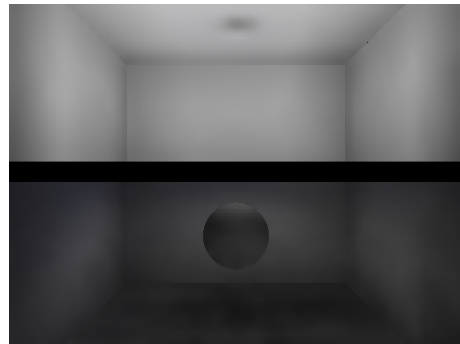
Adaptive Interleaved Sampling

This appendix contains the full result set for Chapter 5 and consists of the following figures:

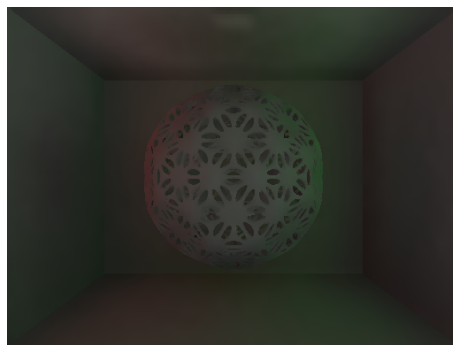
- **A.1** - Indirect diffuse (ID).
- **A.2** - Guidance ID.
- **A.3** - Soft shadows (SS).
- **A.4** - Guidance SS.
- **A.5** - Participating Media (PM).
- **A.6** - Guidance PM.
- **A.7** - Instant Global Illumination (IGI).
- **A.8** - AIS with maximum samples (A-MAX).
- **A.9** - Path-traced reference (PT).
- **A.10** - VDP results for AIS vs. PT.
- **A.11** - VDP results for A-MAX vs. PT.
- **A.12** - VDP results for IGI vs. PT.



(a) Cornell



(b) Subdivided



(c) Deskar



(d) Office



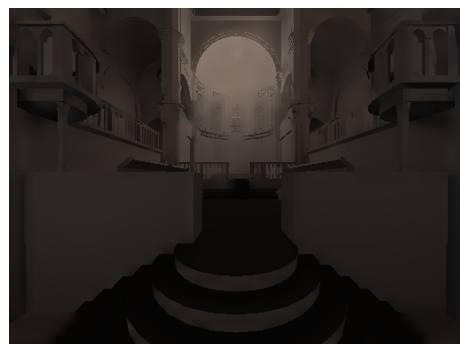
(e) Shirley6



(f) Desk



(g) Conference



(h) Sibenik

Figure A.1: Indirect diffuse (ID).

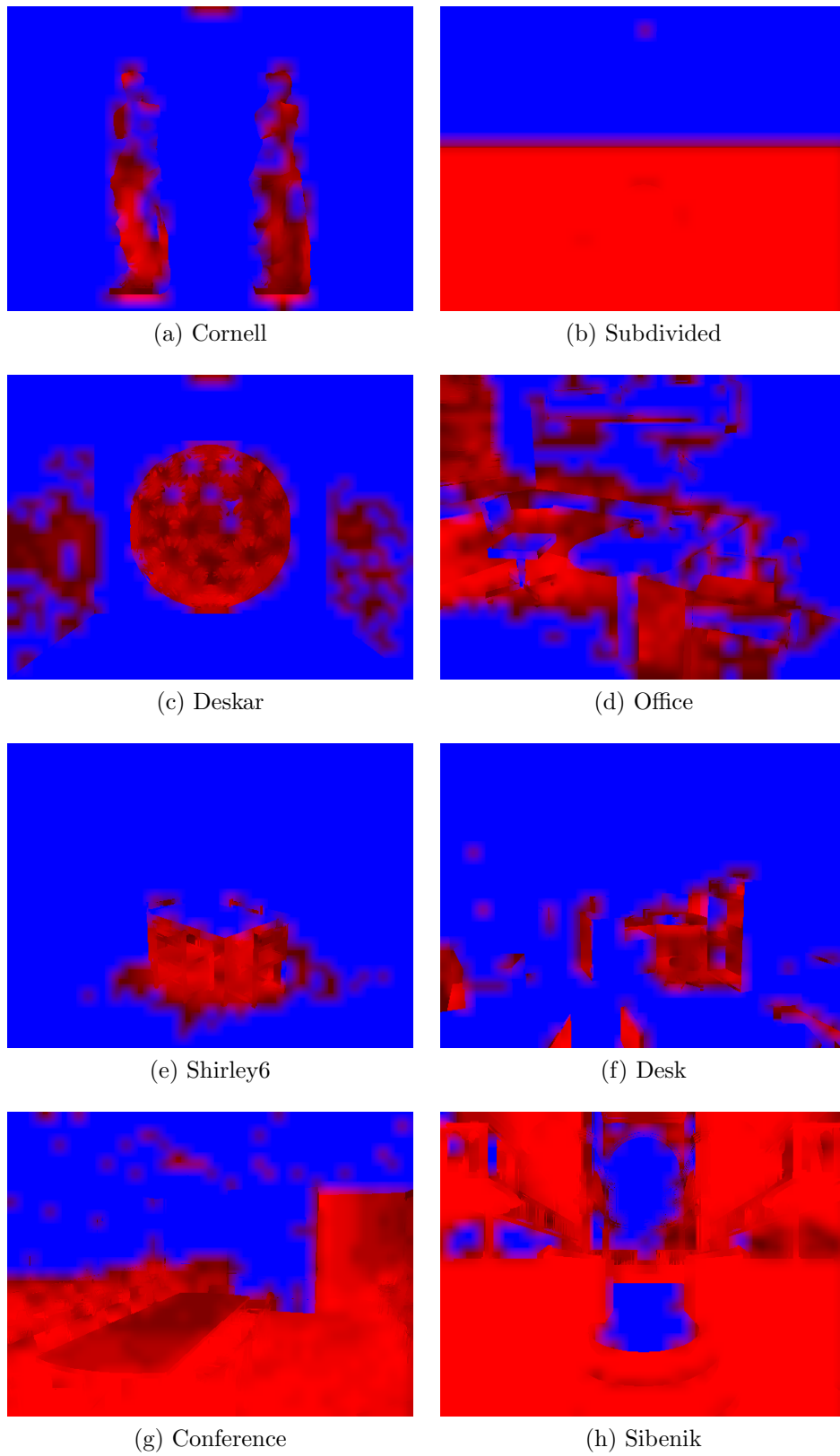
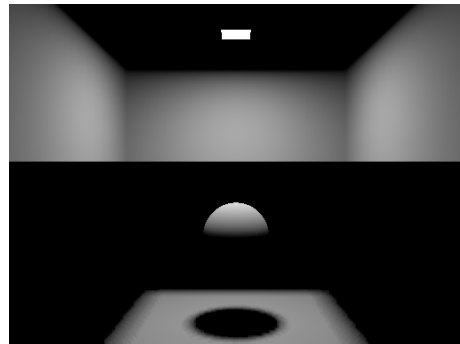


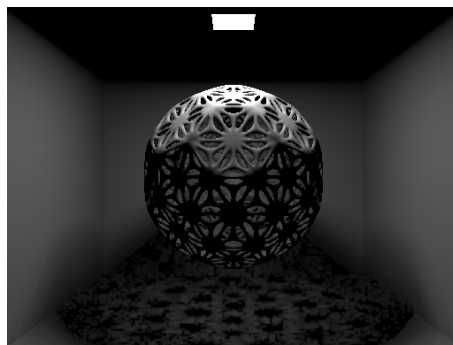
Figure A.2: Guidance ID.



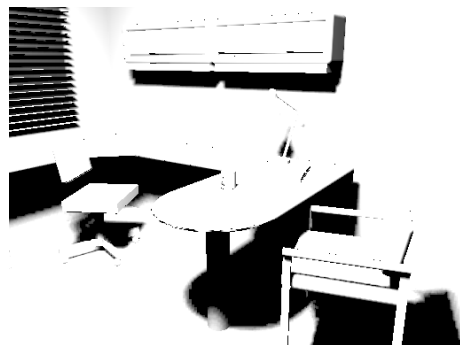
(a) Cornell



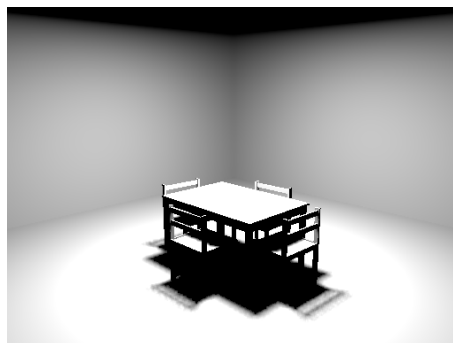
(b) Subdivided



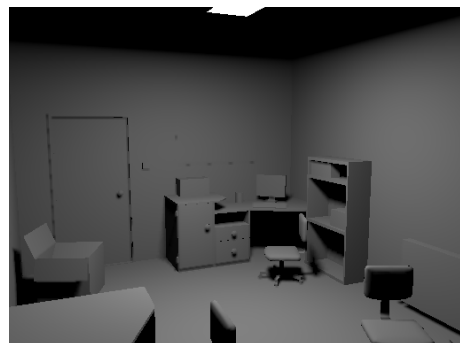
(c) Deskarnet



(d) Office



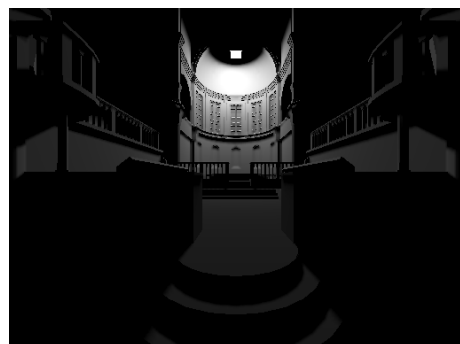
(e) Shirley6



(f) Desk



(g) Conference



(h) Sibenik

Figure A.3: Soft shadows (SS).

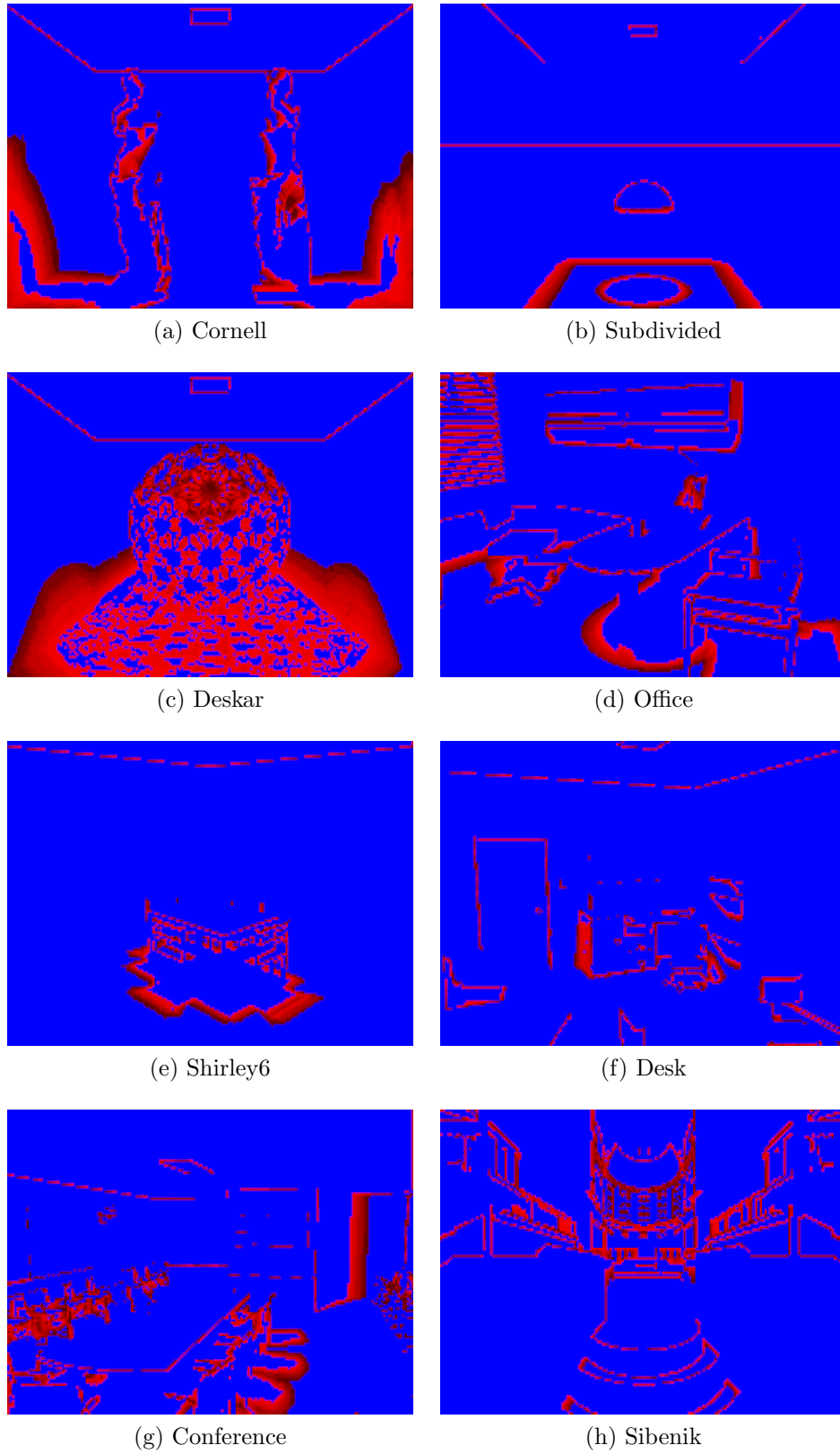
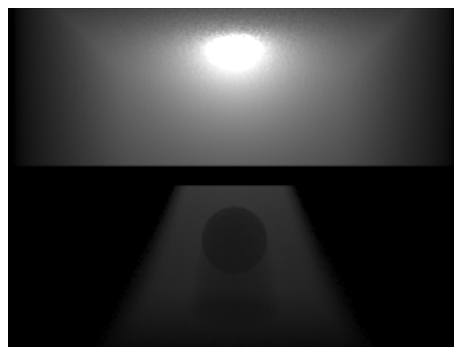


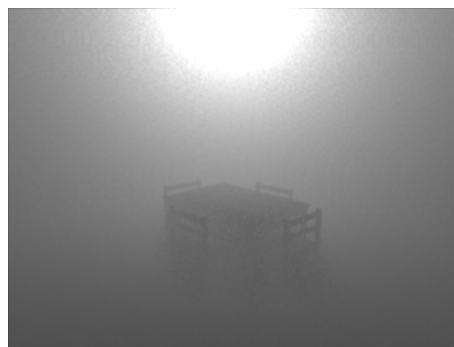
Figure A.4: Guidance SS.



(a) SubdividedPM



(b) OfficePM



(c) Shirley6PM

Figure A.5: Participating Media (PM).

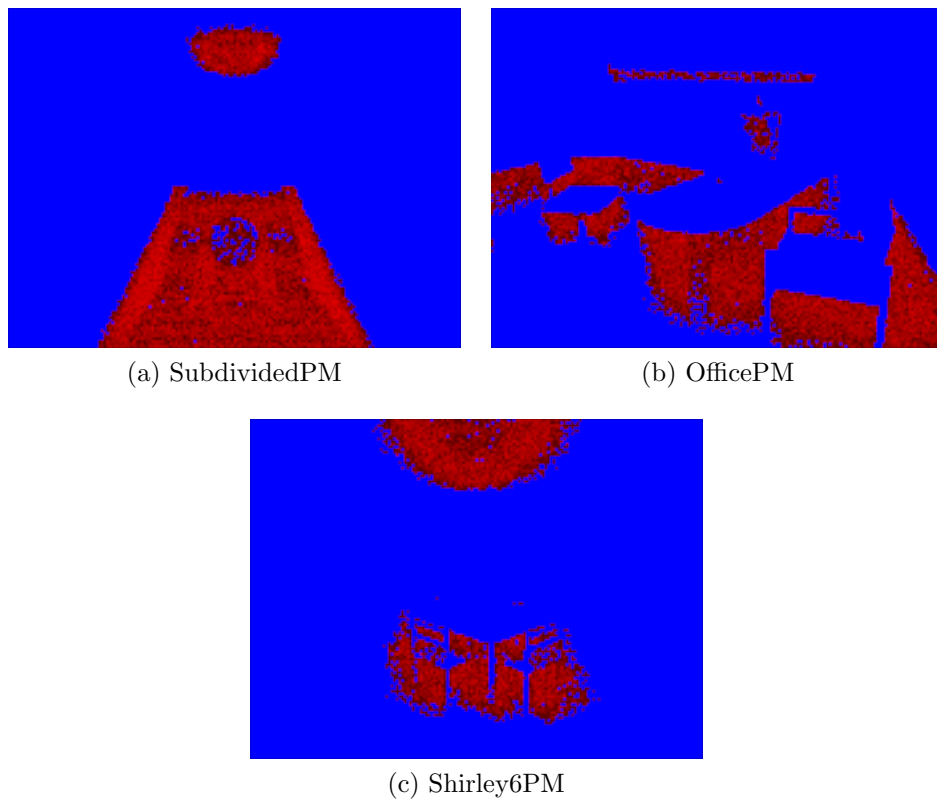
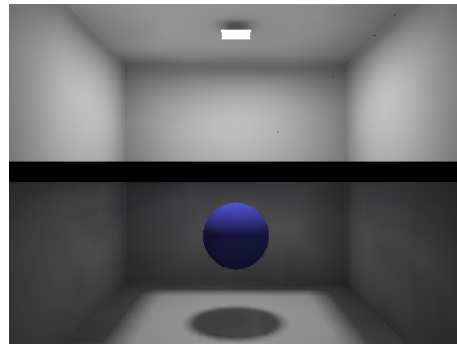


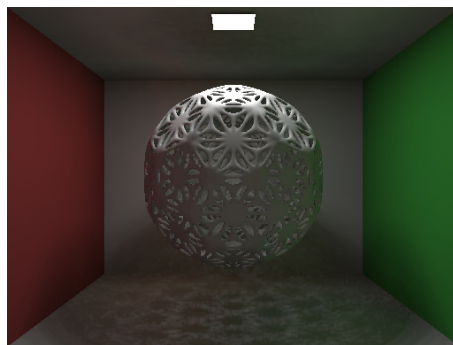
Figure A.6: Guidance PM.



(a) Cornell



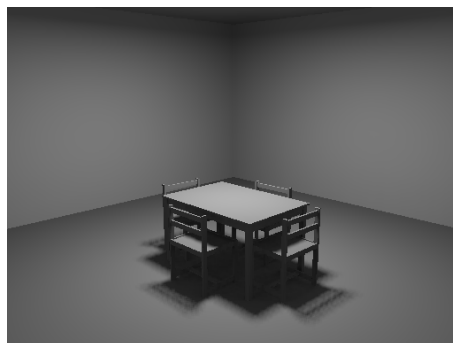
(b) Subdivided



(c) Deskarnet



(d) Office



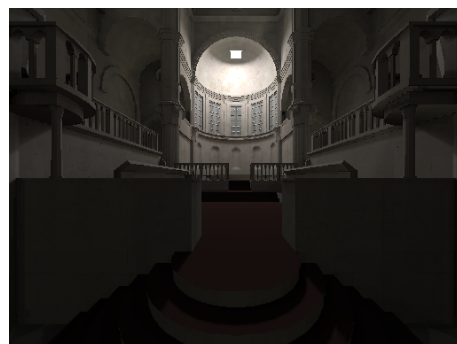
(e) Shirley6



(f) Desk

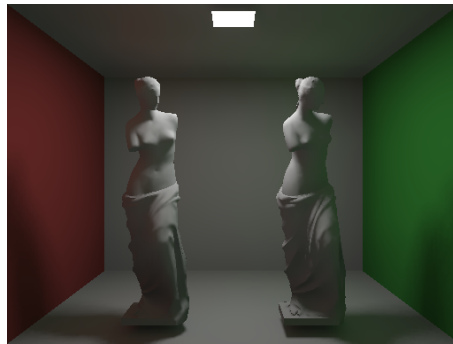


(g) Conference

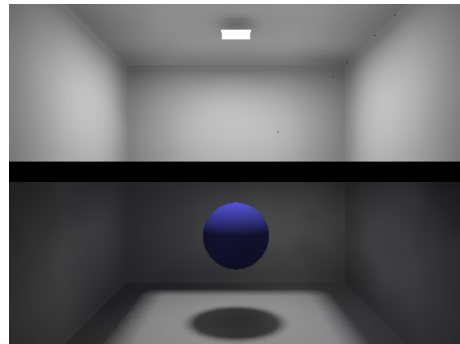


(h) Sibenik

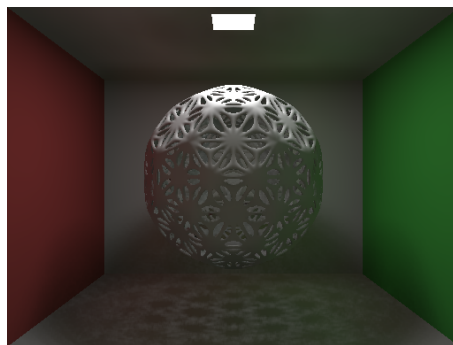
Figure A.7: Instant Global Illumination (IGI).



(a) Cornell



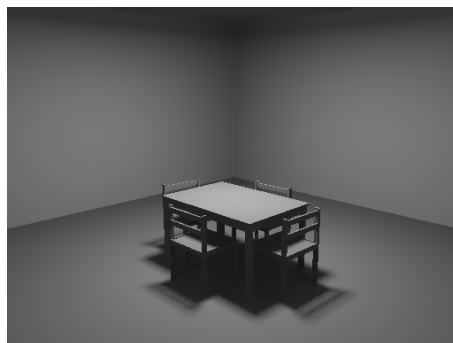
(b) Subdivided



(c) Deskarn



(d) Office



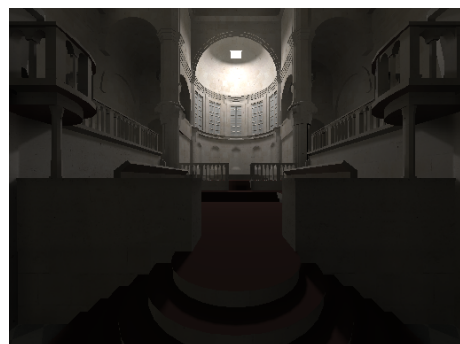
(e) Shirley6



(f) Desk

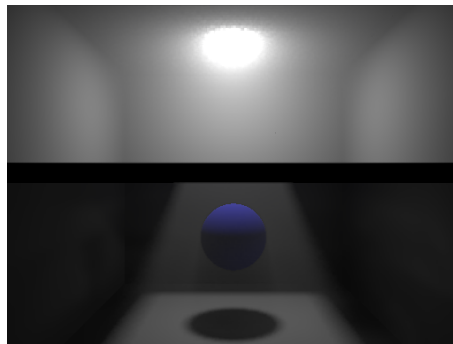


(g) Conference



(h) Sibenik

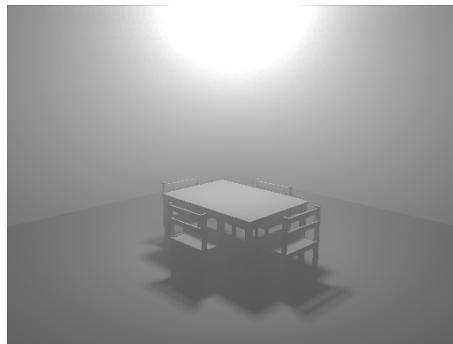
Figure A.8: AIS with maximum samples (A-MAX).



(a) SubdividedPM

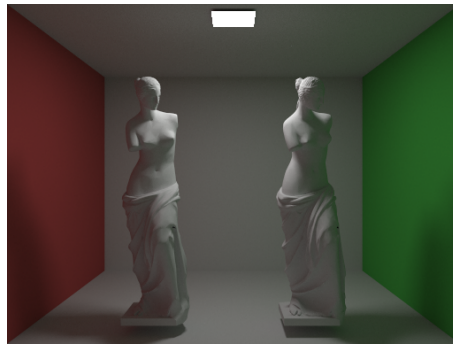


(b) OfficePM

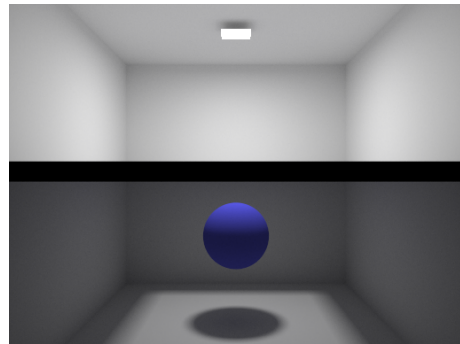


(c) Shirley6PM

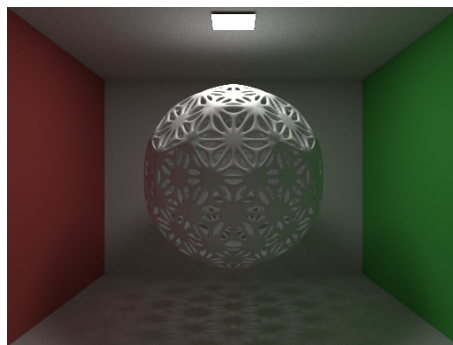
Figure A.8: AIS with maximum samples (A-MAX).



(a) Cornell



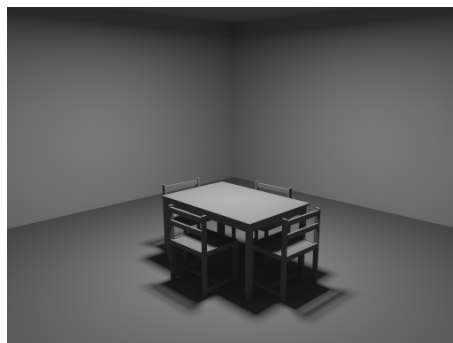
(b) Subdivided



(c) Deskar



(d) Office



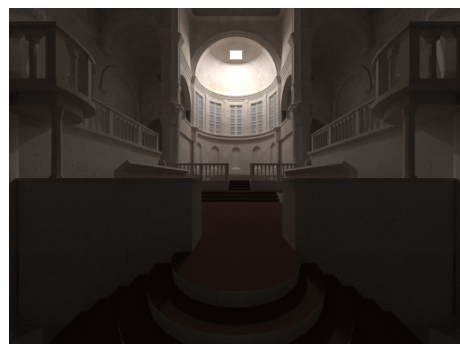
(e) Shirley6



(f) Desk

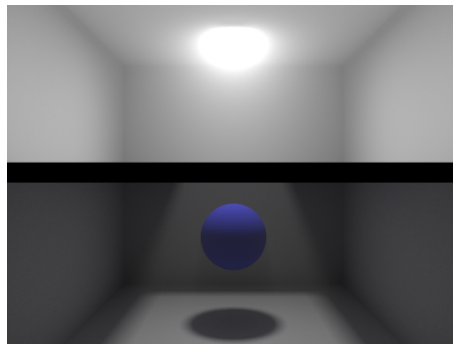


(g) Conference



(h) Sibenik

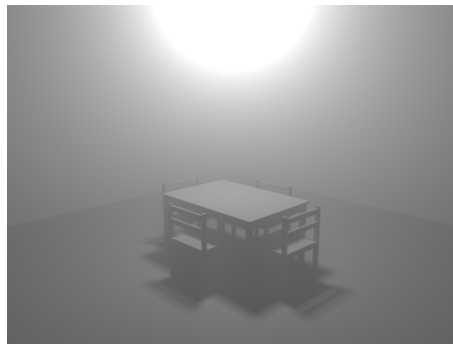
Figure A.9: Path-traced reference (PT).



(a) SubdividedPM



(b) OfficePM



(c) Shirley6PM

Figure A.9: Path-traced reference (PT).

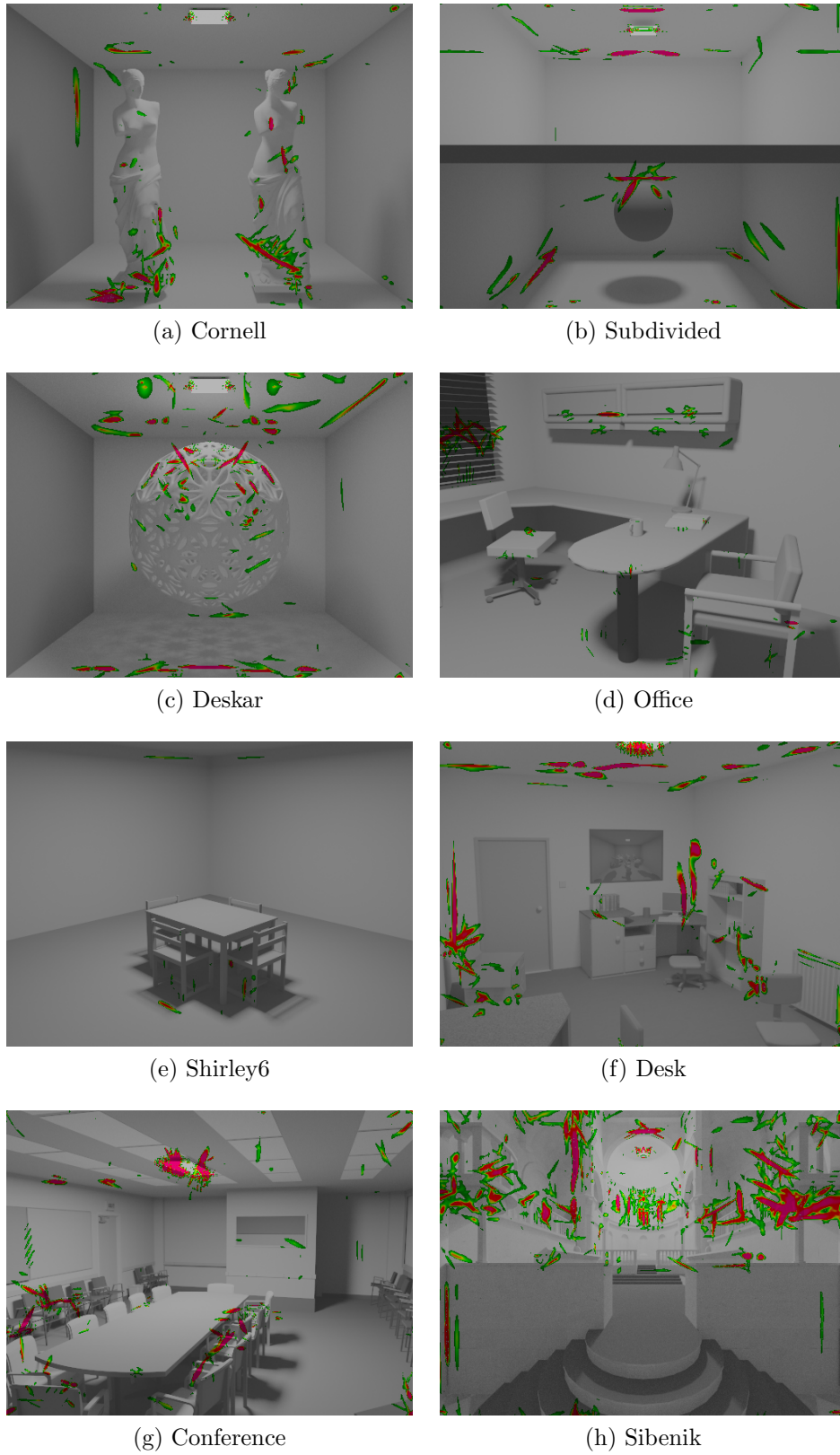


Figure A.10: VDP results for AIS vs. PT

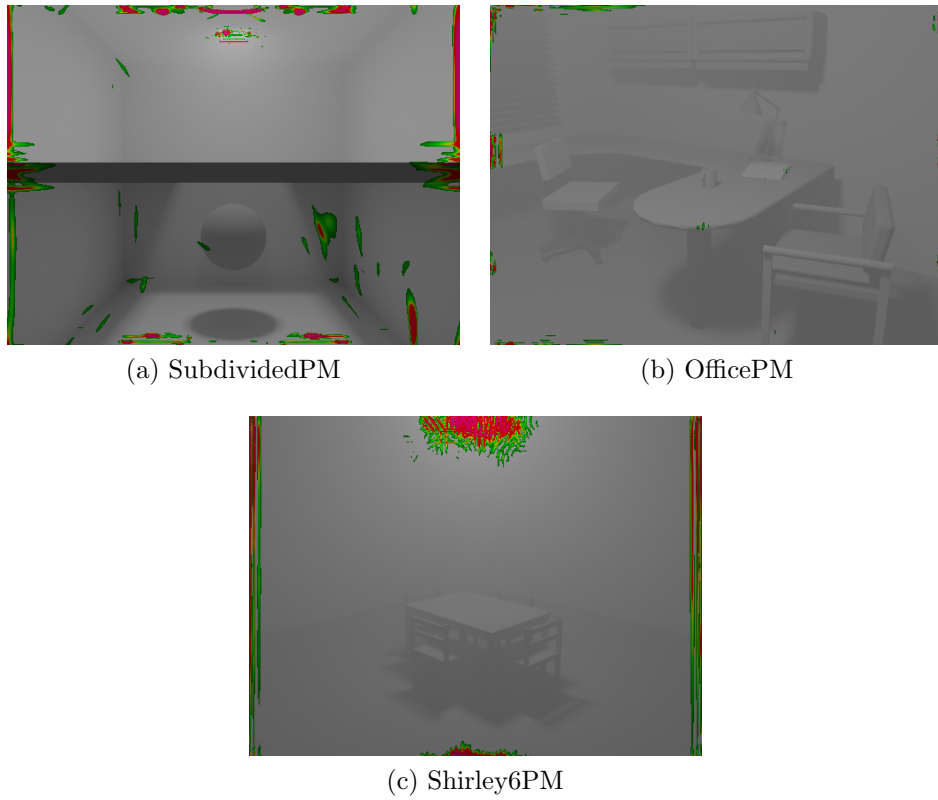


Figure A.10: VDP results for AIS vs. PT

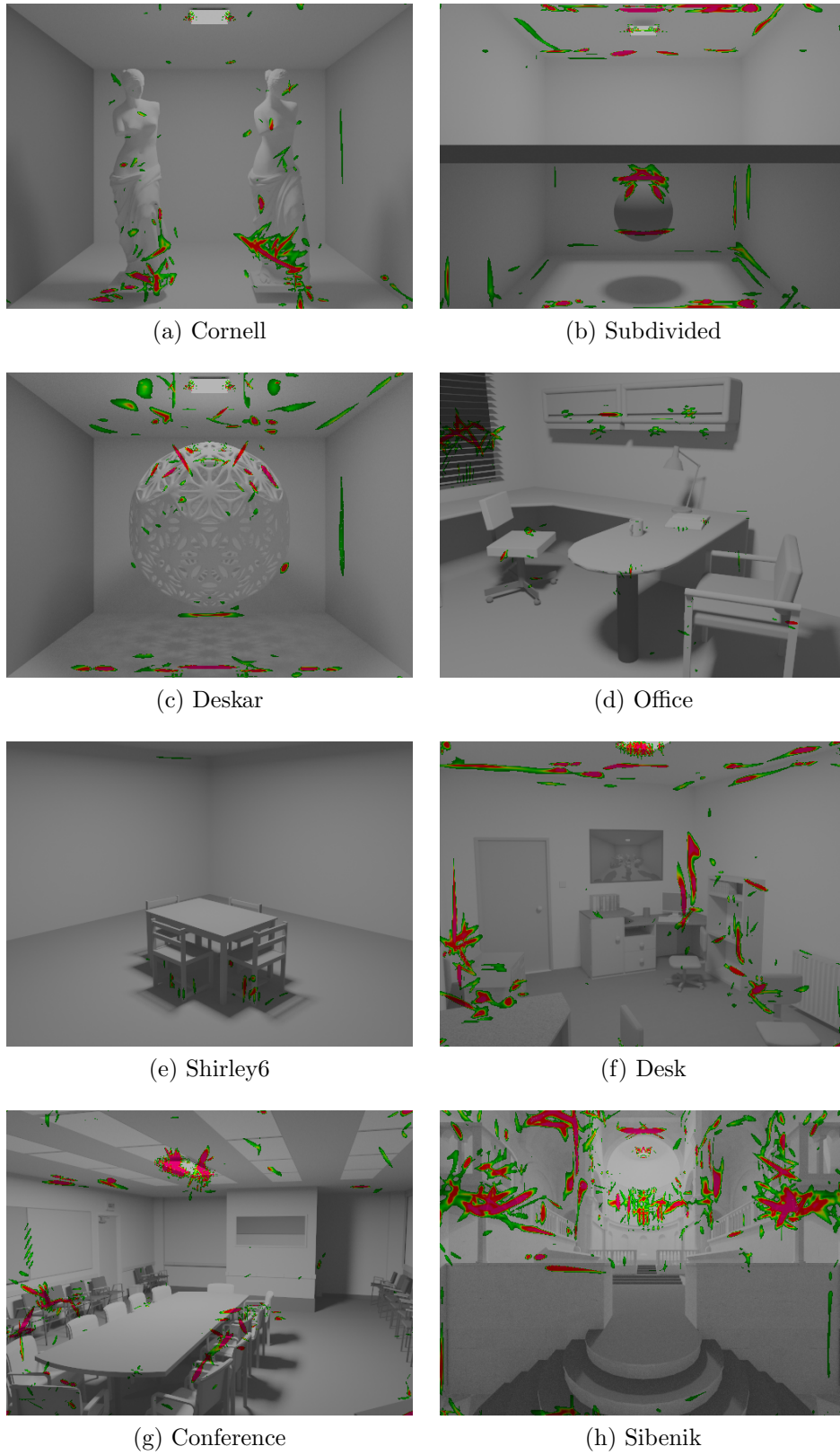


Figure A.11: VDP results for A-MAX vs. PT

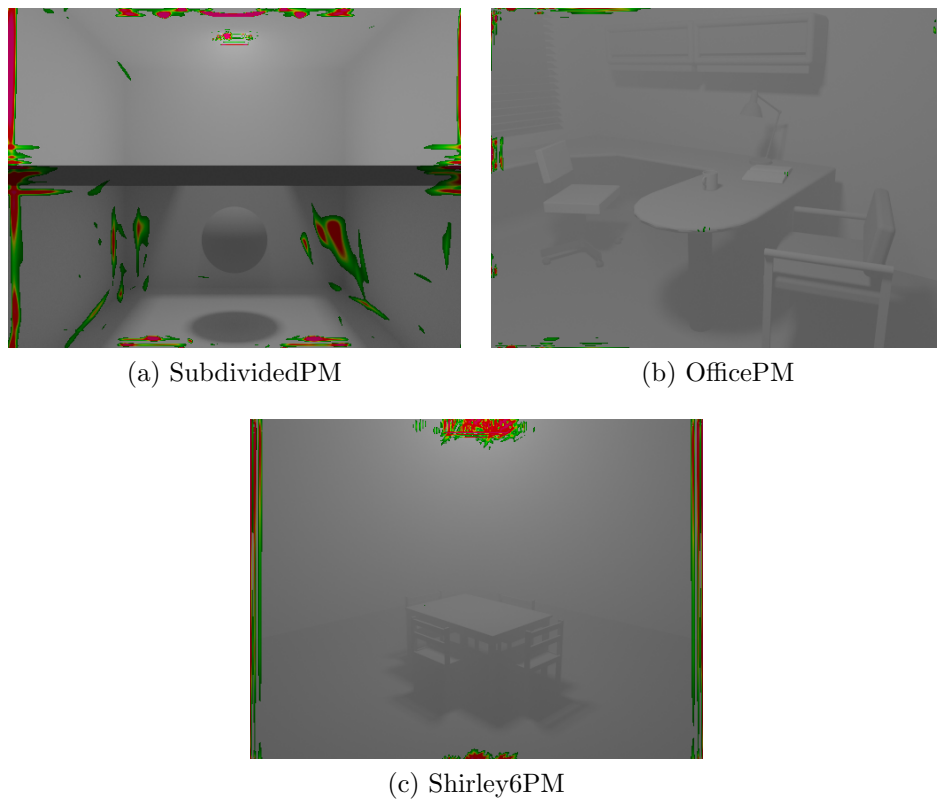
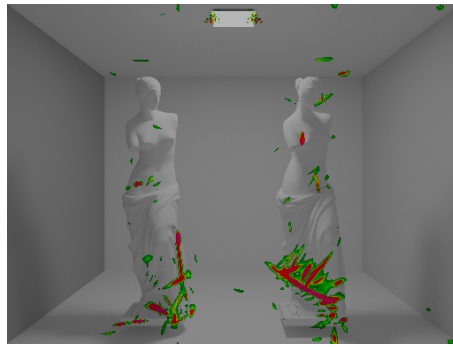
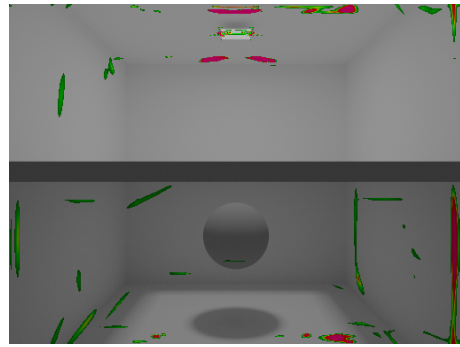


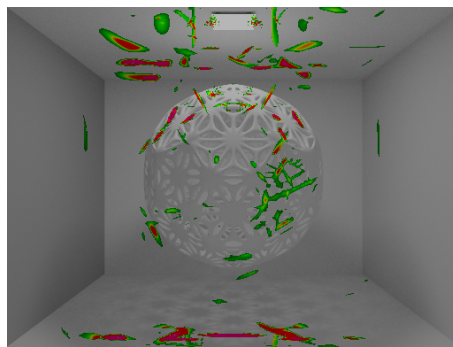
Figure A.11: VDP results for A-MAX vs. PT



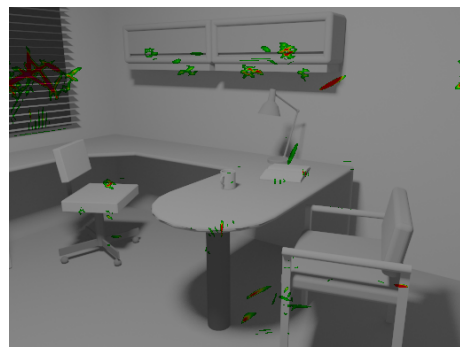
(a) Cornell



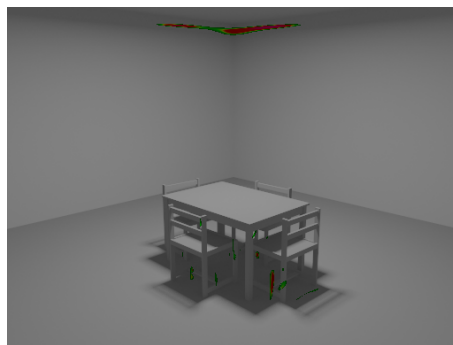
(b) Subdivided



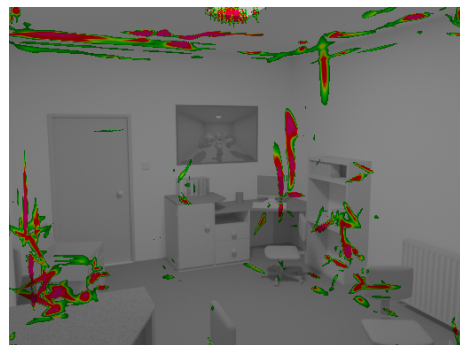
(c) Deskar



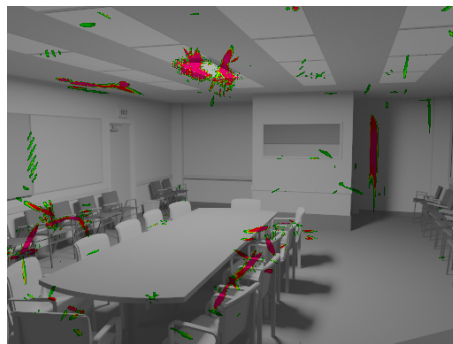
(d) Office



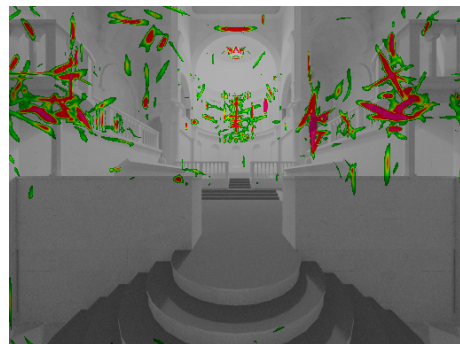
(e) Shirley6



(f) Desk



(g) Conference



(h) Sibenik

Figure A.12: VDP results for IGI vs. PT