N$^o$ d'ordre: 3407

# THÈSE

Présentée devant

**devant l'Université de Rennes 1**

pour obtenir

le grade de : Docteur de l'Université de Rennes 1
Mention Informatique

par

Pascal Gautron

Équipe d'accueil : SIAMES - IRISA
École Doctorale : Matisse
Composante universitaire : IFSIC

Titre de la thèse :
*Cache de luminance et cartes graphiques : une approche pour la simulation d'éclairage temps réel dans des scènes animées*

soutenue le 19 septembre 2006 devant la commission d'examen

| | | | |
|---|---|---|---|
| M. : | Bruno | Arnaldi | Président |
| MM. : | Philip | Dutré | Rapporteurs |
| | François | Sillion | |
| MM. : | Christian | Bouville | Examinateurs |
| | Karol | Myszkowski | |
| | Kadi | Bouatouch | |

## Acknowledgements

There are many people I would like to thank for their support during my Ph.D. Above all, Kadi Bouatouch has been a permanent source of encouragement and advice. He also gave me the occasion of meeting and collaborating with Sumanta Pattanaik, at the University of Central Florida. This collaboration turned out to be very stimulating for my research work.

I particularly want to thank Daniel Meneveaux: besides his technical and moral support, he introduced me to Kadi Bouatouch. I cannot thank him enough for giving me the starting point of my Ph.D. studies.

During the first two years of my Ph.D., I collaborated closely with Jaroslav “walking encyclopedia” Křivánek. His scientific talents and huge knowledge, along with his kindness and sincerity, made this collaboration a very fruitful experience. I would also like to thank my friends/office-mates/former students, Guillaume “ padawan” François, Jonathan Brouillat, and Yoann Marion. Their company has been very enjoyable and scientifically stimulating.

I would like to thank François Sillion and Philip Dutré for reviewing my manuscript, and giving me many ideas for future work.

Many thanks to all the other people who supported me either morally or technically during this period. Particularly, I would like to thank Georges Wyckaërt for providing me with a necessary counterpart of computer science.

I would like to thank my family for their unconditional support during all my studies. Particularly, I want to thank my wife Myriam for her love, energy, sincerity, and her inalterable solidarity and support.

# Contents

# Chapter 1

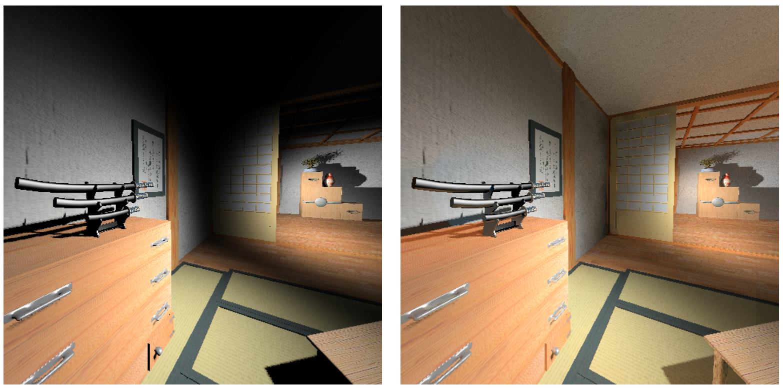
# Introduction

## 1.1 Realistic Rendering: Motivations

The goal of realistic rendering is to create images that are indistinguishable from reality. Many domains are particularly interested in realistic rendering, such as architectural design, cinema and video games. However, rendering realistic images requires an accurate representation and simulation of the interactions between light and matter. In this thesis, our main interest is the fast and accurate simulation of the light transport within dynamic scenes. More precisely, we aim at simulating the multiple bounces of light on the objects of a considered scene. The calculation of these multiple bounces is the computation of global illumination.

## 1.2 Global Illumination

Global illumination has been an intensive research area over the last decades. The goal of this computation is to simulate how light is scattered and reflected off objects made up of various materials. The computation of global illumination can be divided into two parts: direct lighting and indirect lighting. Direct lighting considers only one bounce of light between the light source and the observer (Figure 1.1(a)). Consequently, many aspects of the lighting cannot be rendered by direct lighting only. Indirect lighting accounts for the multiple bounces of light in the scene (Figure 1.1(b)). This allows for the computation of complex lighting effects, such as indirect light sources, color bleeding and caustics. However, the computation of indirect lighting is very complex in the general case and requires several hours even on high performance computers. In the context of interactive applications such as video games, approximations are required for fast rendering. In this case, programmable graphics hardware computes approximate, visually pleasing lighting effects in real-time.



(a) Direct lighting only (b) Global Illumination

Figure 1.1: Direct illumination (a) only accounts for once bounce of light, while indirect illumination (b) adds more realism to the rendering by accounting for several bounces of light.

## 1.3 Real-Time Rendering and GPUs

The recent development of programmable graphics hardware has drastically improved the image quality of real-time applications. Using the recent graphics processing units (GPUs), the user can write complex programs dedicated to vertex and pixel processing. Once compiled on graphics hardware, they provide real-time rendering of advanced lighting effects, even in complex scenes. However, the computational power of the GPUs is based on parallelism, which constraints the usable programs and data structures.

Many approaches have been proposed for fast or real-time computation of global illumination using GPUs. However, due to the inherent limitations of graphics hardware, those methods usually rely on strong assumptions, yielding physically-incorrect solutions in the general case.

## 1.4 Contributions: Steps Towards Real-Time Global Illumination

In this thesis, we investigate the field of real-time, high quality global illumination. To this end, we intensively use the graphics hardware to accelerate the rendering process. We first define *hemispherical harmonics* for global illumination computation in glossy environments using the Radiance Caching algorithm [KGPB05]. Second, we propose a solution to overcome the limitations of graphics processors to produce high quality images using Radiance Caching. Our last contribution is an extension of Radiance Caching to the temporal domain, relying on the coherency of the indirect lighting both in space and time.

### 1.4.1 Hemispherical Harmonics

Building on the well-known spherical harmonics, defined on the whole sphere, the hemispherical harmonics are basis functions restricted to the hemisphere. They can be straightforwardly applied to the computation of global illumination, in which the incoming radiance function is defined over the upper hemisphere only. We demonstrate the representation power of our basis in the context of real-time environment mapping and global illumination using Radiance Caching.

### 1.4.2 Radiance Cache Splatting

The Radiance Caching algorithm is designed for offline rendering using the central processing unit (CPU). This algorithm uses ray tracing for computing the radiance impinging at a given point. Furthermore, this method relies on hierarchical data structures such as octrees. The Radiance Cache Splatting method reformulates the Radiance Caching algorithm to meet the constraints of graphics hardware. This method can be used either for interactive previewing in which global illumination is computed on-the-fly, or for offline fast high-quality rendering.

### 1.4.3 Temporal Radiance Caching

The Radiance Caching algorithm generates high quality images and animations of static scenes. However, the inherent structure of Radiance Caching yields disturbing flickering artifacts while used in dynamic scenes. We propose an extension of Radiance Caching interpolation scheme to the temporal domain. Hence our method computes high quality global illumination in dynamic scenes, in which objects and light sources can move. This method can drastically reduce the computational cost, while significantly improving the rendering quality of the animation.

## 1.5 Thesis Overview

This thesis is divided into six chapters. After this introduction, Chapter 2 presents the most significant previous works in the field of global illumination in dynamic scenes and an overview of programmable graphics hardware. Chapter 3 presents our first contribution: the hemispherical harmonics for accurate representation of hemispherical functions. In Chapter 4, we define the Radiance Cache Splatting method for fast, high quality computation of global illumination using graphics hardware. The extension of the Radiance Caching algorithm to dynamic scenes is described in Chapter 5. Chapter 7 concludes this thesis and proposes directions for future work.

# Chapter 2

# Related Work

In this chapter, we first present the fundamentals of global illumination, and detail significant previous contributions. As our work relies on graphics hardware, we detail the GPU structure and briefly describe methods for GPU-based global illumination computation. Finally, we present an overview of our contributions compared to previous approaches.

## 2.1 Global Illumination

Global illumination accounts for the multiple light/matter interactions within a scene. This field being an intensively researched area, we assume the reader has a basic knowledge of the topic. The book by Pharr and Humphreys [PH04] provides both theoretical and practical details on rendering and global illumination. In [Gla95], Glassner details the physics and algorithms needed for accurate, physically-based rendering. Advanced topics on global illumination can be found in [DBB03].

The aim of global illumination is the resolution of the rendering equation [Kaj86]. This integral equation expresses the outgoing radiance $L_o$ at a given point $\mathbf{p}$ as a function of the self outgoing radiance $L_e$ and the radiance $L_i$ incoming from locations visible from $\mathbf{p}$:

$$L_o(\mathbf{p}, \omega_o) = L_e(\mathbf{p}, \omega_o) + \int_\Omega f_r(\mathbf{p}, \omega_o, \omega_i) L_i(\mathbf{p}, \omega_i) \cos\theta_i \mathrm{d}\omega_i \tag{2.1}$$

where $f_r$ is the Bidirectional Reflectance Distribution Function (BRDF), representing how the incoming lighting is reflected by the surface a point $\mathbf{p}$. Unfortunately, this equation cannot be solved analytically in the general case. Therefore, many numerical methods have been proposed. In the next section we present the most significant approaches. A more thorough state of the art of global illumination for dynamic environments can be found in [DDM02].

## 2.2 Lighting Simulation Techniques

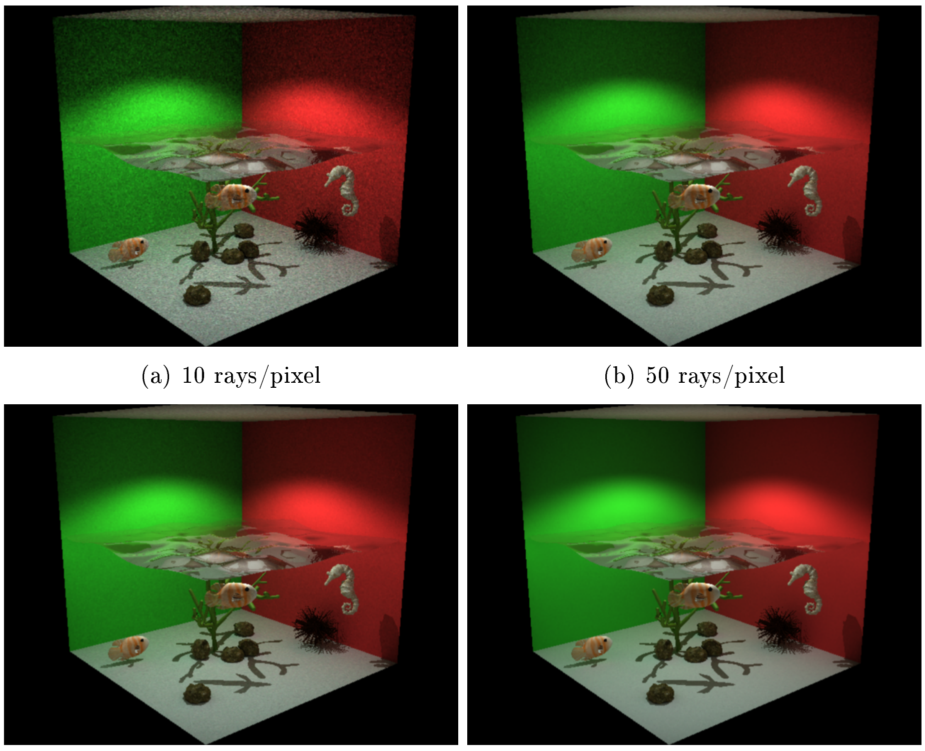
### 2.2.1 Radiosity

The radiosity method was originally introduced by Goral *et al.* [GTGB84]. This approach aims at computing diffuse interreflections by using a finite elements method: each surface of the considered scene (including light sources) is subdivided into patches, for which the lighting is assumed to be constant. The radiosity method consists in calculating the light transfer between patches. Even though this method provides high quality results, it requires finely subdivided surfaces, hence high computation time. Moreover, the use of finite elements can lead to meshing artifacts. Many methods have been proposed for adaptive subdivision, yielding finely tesselated surfaces only in the regions containing discontinuities (discontinuity meshing) thus reducing the rendering time at the cost of high memory consumption. However, high quality results can only be obtained through the use of costly final gathering [SSS01, SSS02]. Once the global illumination solution is computed, the result can be stored in light maps and displayed in real-time using graphics hardware. Many computer games rely on this method to render globally illuminated scenes at the cost of simple texturing.

Even though several methods have been proposed for efficient radiosity in animated scenes [DSH01, Dam01, MPT03, DS97, GD04], computing accurate radiosity solutions for animations remains an expensive task in terms of implementation, rendering time and memory consumption. Therefore, more flexible methods such as Monte-Carlo path tracing are generally preferred due to their low memory usage, high accuracy and ease of implementation.

### 2.2.2 Monte-Carlo Path Tracing

Monte-Carlo integration [PTVF88] is one of the most commonly used method for solving the rendering equation. This method can simulate various global illumination effects, such as diffuse and glossy interreflections. Monte-Carlo path tracing follows the reverse path of light: rays are traced from to observer to the scene elements. For each visible point, the rendering equation is solved numerically by Monte-Carlo integration: the lighting at a given point is obtained by tracing random rays from this point towards the scene. Note that for two distinct visible points the computation of the lighting is completely independent, making this method easily parallelizable and inexpensive in terms of memory. However, the incoming radiances of adjacent pixels of the final image may be uncorrelated, yielding noise artifacts (Figure 2.1). Due to the slow convergence of Monte-Carlo methods, high quality rendering requires to trace many rays, yielding a significant computational cost while only reducing the noise [VG95]. Even though importance sampling increases the rendering quality by tracing the rays in the most important directions, the cost of Monte-Carlo path tracing remains high. Furthermore, this method cannot account for caustics, which require tracing rays from the light sources to the objects. This is the purpose of the photon mapping method.


(a) 10 rays/pixel (b) 50 rays/pixel

(c) 100 rays/pixel (d) 500 rays/pixel

Figure 2.1: Monte Carlo path tracing: the accuracy of the method comes at the expense of computational cost.

### 2.2.3 Photon Mapping

Photon mapping [Jen96, Jen01] is based on the computation of the real path of the light, that is from light sources to the objects of the scene. The flux of each light source is split in a number of outgoing rays, carrying *photons*. When a ray intersects an object, the energy and incoming direction of the photon are stored in a kD-Tree [Ben75], called *photon map*. Then, the photon is reflected in a random direction depending on the surface BRDF. Once the photons are stored in the photon map, the indirect illumination at a given point **p** can be computed by gathering the photon hits in a close neighborhood around **p**. Since the computed light path corresponds to the actual light path, this method proves accounts for complex lighting effects such as caustics and participating media. Many approaches attempt to optimize the construction and rendering of the photon map, such as [SW00, Chr99, FMH05, GWS04, GMSJ03, HP02, KW00, MM02, Sch03, WGS04]. Also, even though the photon mapping algorithm is not originally designed for dynamic scenes, approaches such as [DBMS02, CJ02, JMP05, LC04, WMM$^+$04] exploit temporal coherence for fast rendering of animations.

Using a direct visualization of the photon map, the reconstructed indirect illumination is generally very noisy due to the limited number of photons available around the visible points (Figure 2.2(a)). An additional computation step called *final gathering* is often used for high-quality rendering (Figure 2.2(b)).

### 2.2.4 Final Gathering

The final gathering step consists in computing the outgoing radiance of each visible point $\mathbf{p}$ by solving the equation:

$$L_o(\mathbf{p}, \omega_o) = \int_\Omega f_r(\mathbf{p}, \omega_o, \omega_i) L_i(\mathbf{p}, \omega_i) \cos\theta_i \mathrm{d}\omega_i \tag{2.2}$$

This equation represents the outgoing radiance at point $\mathbf{p}$ and direction $\omega_o$ in terms of the radiance incoming from all the directions of the hemisphere above $\mathbf{p}$. Those incoming radiances can be provided by a rough estimate of the global illumination solution at points visible from point $\mathbf{p}$. Namely, the final gathering gathers the incoming radiances to reconstruct the indirect lighting at the points visible from the camera. The outgoing radiance (Equation 2.2) is computed by Monte Carlo integration for each pixel of the final image. Moreover, high quality results require tracing many rays for each visible point, yielding a prohibitive computational cost. Nevertheless, this costly additional step provides high quality images even with a very coarse global illumination solution (Figure 2.2). Several approaches have been proposed to speed up the final gathering,

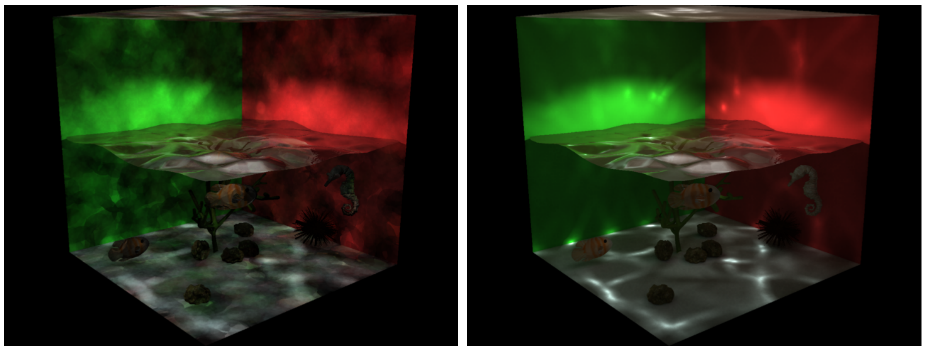

(a) Direct visualization (b) Final gathering

Figure 2.2: Photon map renderings using direct visualization of the photon map (a) and final gathering (b).

such as [HHS05]. One of the most efficient methods for fast and accurate final gathering is the irradiance caching algorithm.

### 2.2.5 Irradiance Caching

This method is the core of the Radiance imaging system [War94]. The principle underlying irradiance caching is the fact that "indirect illuminance changes slowly over a surface" [WRC88]. The irradiance caching algorithm is based on sparse sampling and spatial interpolation of indirect lighting: the indirect diffuse lighting is computed accurately at a small number of visible points using classical Monte Carlo integration. These values, called *irradiance records*, are stored in the *irradiance cache.* Indirect lighting at other visible points is then inexpensively and accurately computed by extrapolating and interpolating the values of the records. In our work, we mainly build on the irradiance caching algorithm, which is therefore precisely detailed in this section.

The irradiance caching algorithm aims at computing the indirect *diffuse* lighting. In this case, the BRDF $f_r(\mathbf{p}, \omega_o, \omega_i)$ can be replaced by the diffuse reflectance of the surface $\rho_d(\mathbf{p})$ in Equation 2.2:

$$\begin{aligned} L_o(\mathbf{p}, \omega_o) &= \rho_d(\mathbf{p}) \int_\Omega L_i(\mathbf{p}, \omega_i) \cos\theta_i \mathrm{d}\omega_i & (2.3) \\ &= \rho_d(\mathbf{p}) E_i(\mathbf{p}) & (2.4) \end{aligned}$$

where $E_i(\mathbf{p})$ is the irradiance at point $\mathbf{p}$. As shown in Equation 4, the radiance outgoing from point $\mathbf{p}$ in direction $\omega_o$ does not depend on $\omega_o$. Therefore, the irradiance $E_i$ is sufficient for representing indirect diffuse lighting.

As described above, the irradiance caching algorithm is based on sparse sampling and interpolation of indirect lighting. However, even though the indirect lighting changes slowly, the change rate of the lighting at a given point is highly dependent of the surrounding geometry. Ward *et al.* [WRC88] propose an adaptive method for deciding whether a record contributes to the indirect lighting of a neighboring point. The authors calculate an upper bound of the change rate of incoming radiance with respect to rotation and translation. For a record $k$ located at point $\mathbf{p_k}$ with normal $\mathbf{n_k}$, the weight at point $\mathbf{p}$ with normal $\mathbf{n}$ is the inverse of this change rate:

$$w_k(\mathbf{p}) = \left( \frac{\|\mathbf{p} - \mathbf{p_k}\|}{R_k} + \sqrt{1 - \mathbf{n} \cdot \mathbf{n_k}} \right)^{-1} \tag{2.5}$$

where $R_k$ is the harmonic mean distance of objects visible from point $\mathbf{p_k}$. The set of records contributing to the indirect lighting at point $\mathbf{p}$ is:

$$S(\mathbf{p}) = \{\text{record } k \ \ / \ \ w_k(\mathbf{p}) \geq 1/a\} \tag{2.6}$$

where $a$ is a user-defined accuracy criterion. On a flat surface, each record contributes to the indirect lighting of points within a circle (Figure 2.3). The size of the influence zone of a record is constrained by $R_k$: the closer the surrounding objects, the smaller the zone. Even though the change of indirect lighting within the influence zone of a record is small, this change cannot be completely neglected. Consequently, Ward *et al.* [WH92] propose *irradiance gradients*, which represent how the indirect lighting changes around the location of the records. The indirect lighting at a given point $\mathbf{p}$ within the contribution zone of record $k$ is extrapolated as follows:

$$E_k(\mathbf{p}) = E_k(1 + (\mathbf{n_k} \times \mathbf{n})\nabla_r + (\mathbf{p} - \mathbf{p_k})\nabla_t) \tag{2.7}$$

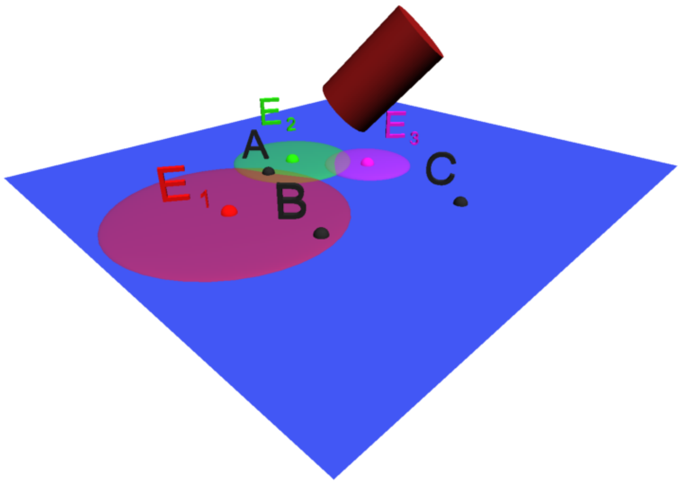


Figure 2.3: The indirect lighting of points near the irradiance records $E_1$, $E_2$, $E_3$ can be extrapolated and interpolated from the values computed at the location of the records. During the rendering process, three situations can happen: point A is in the zone of influence of records $E_1$ and $E_2$. In this case, the indirect lighting is interpolated from the values obtained from those records. Point B is located in the zone of influence of $E_1$ only, and hence its indirect lighting is extrapolated from $E_1$. Since no records can contribute to the indirect lighting at point C, a new record will be generated at C. Note that the size of the contribution zone of each record is inversely proportional to the distance of the surrounding cylinder.

where $\nabla_r$ and $\nabla_t$ are respectively the rotational and translational gradients, accounting for the change of lighting after rotation and translation.

Since several records may contribute to the indirect lighting of a single point $\mathbf{p}$, Ward *et al.* [WRC88] propose the following interpolation formula:

$$E(\mathbf{p}) = \frac{\sum_{k \in S(\mathbf{p})} E_k(\mathbf{p}) w_k(\mathbf{p})}{\sum_{k \in S(\mathbf{p})} w_k(\mathbf{p})} \tag{2.8}$$

The overall algorithm for irradiance caching is described in Algorithm 1. For each visible point $\mathbf{p}$, the irradiance cache is queried. The sum of the weights of the neighboring records is checked against the user-defined accuracy value $a$ (Equation 2.9). If the contributions of existing records is not sufficient (or if no records are present), an irradiance record is created at the point of interest, and stored in the irradiance cache (represented by an octree). Otherwise, existing records are used to obtain an estimate of the indirect lighting by evaluating Equation 2.8.

$$\sum_{k \in S(\mathbf{p})} w_k(\mathbf{p}) \begin{cases} < a & \text{New record is required} \\ \geq a & \text{Interpolation can provide a satisfying result} \end{cases} \tag{2.9}$$

**Algorithm 1** Irradiance Caching

```
for all visible point p do
    Query the irradiance cache
    if enough contributing records are found then
        Estimate indirect lighting by interpolation
    else
        Compute a record at point p
        Store the record in the cache
        Use the record to estimate the indirect lighting at p
    end if
end for
```

In dynamic scenes, a basic approach consists in computing a new cache from scratch for each frame. However, irradiance caching is based on the extrapolation of irradiance around the location of the records. As any extrapolation method, the estimated value cannot be completely accurate, yielding flickering artifacts due to the change of location of the records (see Chapter 5 for more details). Several methods have been proposed for extending irradiance caching to dynamic scenes and to avoid such flickering artifacts.

The approach of Tawara *et al.* in [TMS02] aims at localizing the points at which final gathering is performed while rendering an animation sequence. The indirect lighting is split into two parts. On the one hand, the static part is an irradiance cache representing the light transfer between static objects. This part remains unchanged for the whole animation segment. The dynamic part accounts for dynamic objects: both a separate irradiance cache and a photon map are created. This part is used for updating the static lighting, which can be modified due to the displacement of dynamic objects. This method shows significant speedup, but is restricted to animations during which "lighting conditions do not change significantly" [TMS02]. Otherwise, the static irradiance cache has to be recomputed from scratch, leading to the temporal artifacts inherent in this method.

Tawara *et al.* [TMS04] propose to detect the incoming radiance changes related to the displacement of objects. The rays traced for computing irradiance records are stored and updated using either a user-defined update rate or an adaptive rate based on the number of rays hitting a dynamic object. This method requires an high amount of memory to store the rays, making it difficult to use in complex scenes where many records have to be computed.

The method described in [SKDM05] is based on a data structure called *anchor*. Using this structure, changes in visibility and incoming radiance of the records can be efficiently detected and accounted for during an animation sequence. While this method provides high-quality results, it requires an explicit storage of the rays used to compute each irradiance record, and hence is highly memory demanding.

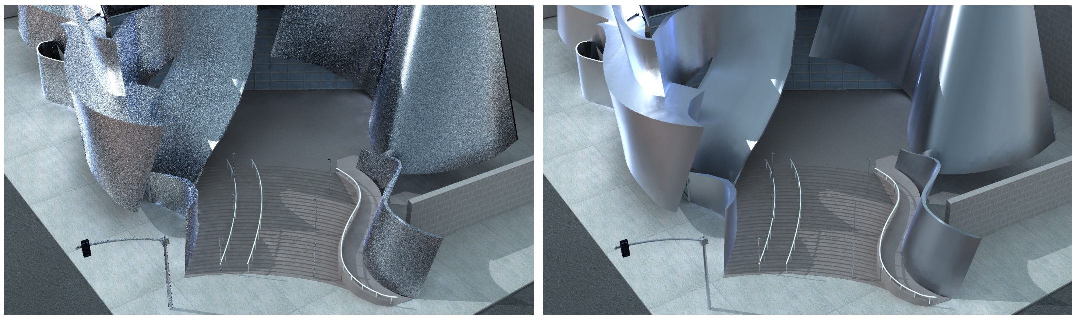
### 2.2.6 Radiance Caching

As described above, the irradiance caching algorithm computes and stores the indirect diffuse lighting. Since the directional information of the incoming radiance is not accounted for (Equation 4), irradiance caching-based renderers resort to classical Monte Carlo path tracing to render glossy surfaces. Křivánek *et al.* [KGPB05, Kři05] introduced Radiance Caching for efficient computation of indirect glossy lighting. The rendering of glossy indirect lighting requires the knowledge of the directional information of the incoming radiance: instead of storing and interpolating irradiance, this algorithm projects the incoming radiance function into the spherical or hemispherical harmonics basis [GKPB04] (Chapter 3). Using this projection, the incoming radiance function is represented compactly by a small set of coefficients (typically 100). One of the advantages of this technique is the orthogonality of the basis functions: if both the BRDF and incoming radiance function are represented with hemispherical harmonics, Equation 2.2 reduces to:

$$\begin{aligned} L_o(\mathbf{p},\omega_o) &= \int_\Omega f_r(\mathbf{p},\omega_o,\omega_i)L_i(\mathbf{p},\omega_i)\cos\theta_i \mathrm{d}\omega_i && (2.10)\\ &= \sum_{k=0}^{n-1} L_i^k f_r^k && (2.11)\end{aligned}$$

where $L_i^k$ and $f_r^k$ are respectively the $k^{\text{th}}$ projection coefficients of the incoming radiance function and of the BRDF, and $n$ is the number of coefficients used for the projection. Křivánek *et al.* propose to reuse the weighting and interpolation scheme proposed by Ward *et al.* [WRC88] described above. As in [WH92], Křivánek *et al.* propose radiance gradients for accurate extrapolation of the incoming radiance function [KGPB05, KGBP05]. Such gradients represent how each projection coefficient of the incoming radiance function changes with displacement, that is the translational gradient. In the context of radiance caching, the rotational gradient becomes useless: the frame change between the record location and the point of interest is accounted for using rotation matrices included in the hemispherical harmonics framework. Instead of using a rotational gradient, this method calculates the rotated incoming radiance function accurately. Details on hemispherical harmonics rotation are presented in Chapter 3. As shown in Figure 2.4, this method provides significant quality improvement compared to classical Monte Carlo path tracing. However, as any basis function set, the hemispherical harmonics are better for representing low frequency functions. Moreover, the high spatial coherency required to perform sparse sampling and interpolation of indirect lighting is only present with diffuse and moderately glossy surfaces. Therefore, radiance caching can only be used to compute indirect lighting on moderately glossy surfaces. For highly specular surfaces, classical Monte Carlo path tracing with importance sampling is more efficient and accurate.

The methods presented in this section aim at fast computation of global illumination using the Central Processing Unit (CPU). As GPUs get more and more powerful, expensive computations such as global illumination can be usefully reformulated and



(a) Monte Carlo path tracing (b) Radiance Caching

Figure 2.4: A model of the Walt Disney Hall in Los Angeles rendered using Monte Carlo path tracing (a) and Radiance Caching (b) in the same amount of time. Radiance Caching increases the rendering quality by removing the per-pixel noise of Monte Carlo path tracing. ( Images courtesy of Jaroslav Křivánek )

implemented to fit the constraints of graphics hardware. Next section presents the fundamentals of graphics processors, and briefly introduces the programming of the latest graphics hardware.

## 2.3 Graphics Processing Unit

The Graphics Processing Units, or GPUs, are designed to reduce the load of the CPU by performing the operations related to the computation and display of synthetic images. In this section we first present the classical structure of GPUs. Then, we introduce the concept of GPU programming, which we use intensively in the rest of this thesis.

The GPUs rely on the rasterization algorithm to generate images of polygonal scenes in realtime, . This algorithm, illustrated in Figure 2.6 and Algorithm 2, is divided in 4 consecutive parts. First, each polygon of the scene is sent to the vertex processor. This processor projects each vertex onto the image plane. This projected polygon is the input of the rasterizer. For each pixel covered by the polygon, this component uses Gouraud interpolation to interpolate the normal and texture coordinates bound to the vertex (Figure 2.6). The output of the rasterizer is a set of *fragments* corresponding to the pixels covered by the polygon. Then those fragments are sent to the fragment processor, which performs per-pixel operations such as texture mapping. Since the coverage of distinct polygons may overlap, several fragments may be mapped to a same pixel. To obtain the final image, the processed fragments may be discarded or blended with fragments mapped to the same pixel. This particularly accounts for occlusion between triangles using depth buffering (Z-Buffer): if several fragments are mapped to a same pixel, only the closest fragment is used in the final image. Other operations may be performed for each pixel, such as alpha blending (fragments are blended together according to their transparency coefficient).

**Algorithm 2** Rasterization

```
for all polygon of the scene do
    // Vertex processor
    for all vertex of the polygon do
        Project the vertex onto image plane
    end for
    // Rasterizer
    for all pixel covered by the polygon do
        Gouraud interpolation of position, normal, texture coordinates, ...
        Generate a fragment
    end for
    // Fragment processor
    for all fragment generated by the rasterizer do
        Perform per-pixel operations
    end for
    // Post processing of fragments
    for all fragments processed by the fragment processor do
        Alpha blending
        Depth test
        ...
    end for
end for
```

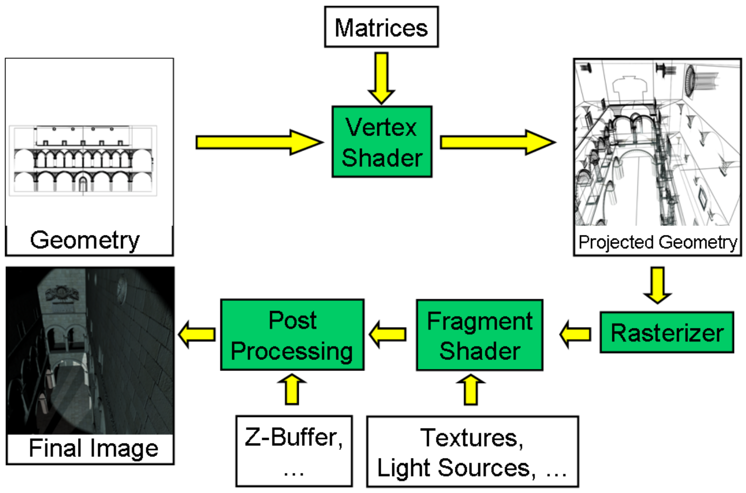
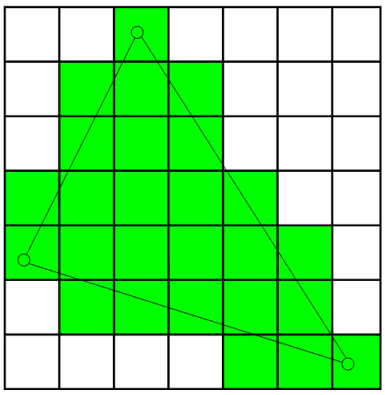
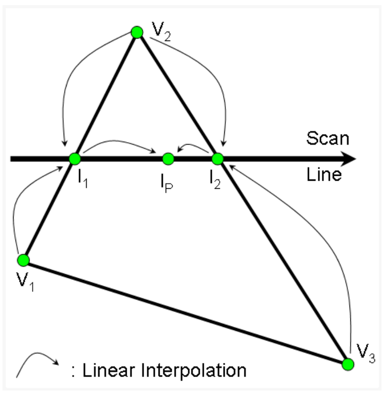


Figure 2.5: The Graphics Pipeline


(a) Rasterization



(b) Gouraud Interpolation

Figure 2.6: The rasterizer fills the pixels covered by the considered polygon (a). For each pixel, the properties of the vertices are interpolated using Gouraud interpolation (b).

As shown in Algorithm 2, each polygon or fragment is processed independently by a same program. Hence this algorithm can be easily parallelized using a SIMD (Single Instruction, Multiple Data) processor. Current graphics hardware typically contain 8 vertex processors and 24 to 48 pixel processors. Furthermore, in todays graphics hardware, the user can program the vertex and fragment processors to perform custom computations in parallel. Such programs can be written in C-like languages such as the OpenGL Shading Language (GLSL) [KBR04], nVidia Cg [nC05] and DirectX High

Level Shading Language [Lau05]. The user can either use those programs either to perform advanced shading of pixels, or for more general computations [Buc03, MT04]. Due to the high computational power and programmability of GPUs, graphics hardware is intensively employed in demanding applications such as global illumination.

## 2.4 Programmable GPUs for Global Illumination

Nowadays, many approaches for global illumination rely on programmable graphics hardware for fast computation. In this section, we present an overview of significant works on accurate GPU-based global illumination.

In the context of radiosity, [CHL04, CH05, CHH03] propose methods for fast computation on the GPU. The work from Coombe *et al.* relies on texturing and visibility testing, whereas Carr *et al.* use the GPU to process the radiosity matrix.

In [NPG04, NPG03], Nijasure *et al.* use the basic idea of the irradiance volume [GSHG98] to propose a method for non diffuse global illumination computation using graphics hardware. The incoming radiance function at a number of locations a priori selected 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. Although the authors demonstrate real-time performance, the main drawback of this method is the choice of sample points. In [NPG04, NPG03] these points are placed on a uniform grid within the volume of the scene, regardless of the lighting complexity.

In the Precomputed Radiance Transfer (PRT) approaches, the radiance transfer between surfaces of an object is precomputed offline and represented using spherical harmonics [SKS02, SHHS03, SLS05] or wavelets [LSSS04, WTL04]. Using this precomputed information, the global illumination solution can be computed and displayed at interactive rates using the GPU. Although the PRT approaches allow for real-time, high quality relighting, they rely on a costly precomputation.

Wand and Straßer [WS03] describe a GPU-based method for real-time caustics computation. This algorithm relies on the selection of sample points on glossy surfaces. Each sample point is considered as a pinhole camera from which the scene is rendered. The reflection of the light incoming at the sample point is computed for each point visible from the pinhole camera. Using projective texturing, the reflections computed at each sample point are projected and combined onto diffuse receiver surfaces. This method handles several dynamic light sources and objects at interactive rates, but speed drops quickly as quality improves.

Several attempts have been made to compute global illumination using GPU-based ray tracing. These methods rely on the versatility of programmable graphics hardware and use fragment shaders to perform ray-primitive intersections [CHH02, PBMH02]. The work described in [PBMH02] has been extended to the computation and rendering of photon maps [PDC$^+$03]. In addition, another photon map rendering method is presented in [MM02]. Those approaches suffer from the same drawback: the GPU architecture does cannot handle complex data structures such as trees, which are commonly used in ray tracing optimization and photon map storage. The photon map is stored either in a uniform grid [PDC$^+$03], or in a costly hash table [MM02]. The related nearest-neighbors queries have been simplified to meet the structure and constraints of the GPUs, yielding quality or performance drop. Three other approaches for GPU-accelerated photon map rendering have been proposed. Larsen *et al.* [LC04] use graphics hardware to perform the costly final gathering: the photon map is built on the Central Processing Unit (CPU) using the classical method defined in [Jen01]. For each surface, an "approximate illumination map" is built using the data contained in the photon map. The GPU is used to perform final gathering and caustics filtering. The approaches presented in [SB97] and [LP03] use the GPU for irradiance reconstruction: each photon is rendered as a textured quadrilateral. The corresponding texture represents the kernel function for the photon. Although those methods show encouraging results, they are limited by the large number of photons required to render a high quality image.

Besides hardware acceleration, every global illumination method relies on computations of functions defined on the hemisphere. More precisely, in the context of radiance caching [KGPB05], we propose a set of hemispherical basis functions for compact and accurate representation of incoming radiance functions and BRDFs.

## 2.5 Representation of Hemispherical Functions

Accurate representation of hemispherical functions is crucial for many lighting methods such as precomputed radiance transfer [SKS02] or radiance caching [KGPB05]. In this section we describe common methods for representing such functions: spherical harmonics, Makhotkin hemispherical harmonics, Zernike polynomials and spherical wavelets.

Spherical harmonics are basis functions relying on Legendre polynomials [Sze75], and representing functions defined on the sphere. They are widely used for representation of BRDFs and environment maps [RH02, SKS02, WAT92, SAWG91a, CMS87]. When using spherical harmonics for both BRDFs and environment maps, the orthogonality of those basis functions (Chapter 3), the lighting integral reduces to a simple vector dot product. Moreover spherical harmonics are used to store precomputed shadowing and inter-reflection data [SKS02]. The support of simple rotation matrices [IR96, IR98, BFB97, CIGR99] makes those basis functions very practical in the context of global illumination, where frame changes are very frequent. However, BRDFs, shadowing and inter-reflections at a point are all functions defined on a hemisphere, whereas the spherical harmonics basis functions are defined on the whole sphere. Hence an unnecessary large number of spherical harmonics coefficients is required for accurate representation. Sloan *et al.* [SHHS03] propose a solution to the domain mismatch by padding a hemispherical function with a mirrored copy of the function itself ("even reflection"). This process significantly improves the accuracy of the representation of a hemispherical function. However, with this approach, the spherical harmonics coefficients no longer represent the original function. Problems appear when combining with a spherical environment map, since the nonzero values in the lower hemisphere yield erroneous dot product result. To distinguish this process of hemispherical function representation from the standard spherical harmonics representation we will refer to it as “even reflection spherical harmonics representation”, or ESH. In [SHHS03] Sloan *et al.* also propose a least squares optimal spherical harmonics (LSOSH) projection for hemispherical functions. Although this method improves the representation accuracy in the upper hemisphere, any hemispherical function projected using this method would contain nonzero, random values in the lower hemisphere. Consequently, this method is only suitable for evaluation purposes, since the added nonzero values yield erroneous results when computing dot product with other LSOSH-projected functions.

Makhotkin hemispherical harmonics [Mak96, Sme98] are hemispherical basis functions relying on shifted adjoint Jacobi polynomials [Sze75]. Those basis functions have been defined in the context of radiative transfer. Fast evaluation of the basis functions can be achieved by simple recurrence relations. However, this representation lacks rotation matrices, which are crucial in our case (Chapter 3).

Zernike polynomials [WC92] are basis functions defined on the disc. Those functions have been adapted by Koenderink *et al.* [KvDS96] to build a hemispherical basis. The CUReT BRDF database [DvGNK99] uses these basis functions for representing measured BRDFs. Despite an accurate and compact representation of hemispherical functions, Zernike polynomials have a high evaluation cost (up to 10 times more than the associated Legendre polynomials used for spherical harmonics). Furthermore, to the best of our knowledge, no rotation matrices have been proposed yet for functions projected into the Zernike polynomials.

The methods described above rely on basis functions defined on the whole sphere or hemisphere, achieving simple projection and storage of the projection coefficients. However, due to the global scope of the basis functions, those methods cannot represent accurately the high frequency components of the functions. In the wavelet projection method, multiresolution representation is achieved by using localized basis functions. Wavelets are typically used for efficient representation of functions defined on a plane such as images [ISO00]. Schröder and Sweldens [SS95] extended wavelets to the spherical domain for efficient representation of spherical functions. They demonstrated the usefulness of spherical wavelets to accurately represent hemispherical functions, particularly BRDFs. Wavelets are advantageous especially for functions of complex shape. Representing smooth functions using a small set of wavelet coefficients might lead to aliasing (see Figure 13 in [SS95]). Aliasing affects the visual quality of the generated images more than an incorrect, although smooth, BRDF shape obtained by spherical harmonics. Moreover the hierarchical data structure proposed in [SS95] is not well suited to current graphics hardware.

## 2.6 Our Approach for Fast Global Illumination using GPUs

In this thesis, we focus on fast, GPU-based computation of high quality global illumination in nondiffuse, dynamic environments. In the context of radiance caching, we first propose a set of hemispherical basis functions for efficient global illumination in nondiffuse scenes (Chapter 3). We build on the existing spherical harmonics to propose hemispherical harmonics, defined on the upper hemisphere only. Such basis functions represent hemispherical functions more compactly and accurately than spherical harmonics. Spherical and hemispherical harmonics being very similar, they share two key advantages: fast evaluation using recurrence formulæ, and simple rotation matrices. Those advantages make our hemispherical harmonics well-suited for complex computations involving hemispherical functions such as global illumination.

As irradiance caching, the radiance caching algorithm relies on ray tracing for record computation and on a hierarchical data structure for record storage. Those constraints prevent from the direct use of GPUs to speedup the computation. We propose the radiance cache splatting algorithm (Chapter 4) for fast global illumination computation on the GPU. Unlike methods such as [PBMH02] we do not aim at directly porting the global illumination computation on the GPU. Instead, we reformulate the irradiance and radiance algorithms to use the basic features of graphics hardware: rasterization, alpha blending, and per-pixel shading of visible points. Without introducing any quality loss, this method achieves significant speedups compared to the Radiance system [War94], and provides interactive frame rates in simple, static scenes.

The radiance and irradiance caching algorithms aim at the computation of global illumination in static scenes. We extend the irradiance caching interpolation scheme to the temporal domain by defining a temporal weighting function and temporal gradients for smooth temporal interpolation (Chapter 5). Without using any particular data structure, our method reuses records in several frames adaptively: the lifespan of each record is chosen according to the change rate of the local incoming radiance. Furthermore, we do not divide the computation into static and dynamic lightings, achieving fast high quality rendering of complex lighting changes such as moving light sources.

Our last contribution is a method for efficient rendering of global illumination solutions obtained by photon mapping. Whereas previous approaches are generally based on final gathering and irradiance caching, we propose to use only the photon map to generate irradiance records without tracing additional rays. This approach significantly reduces the computational cost of photon map rendering while preserving image quality. This work was carried out by Jonathan Brouillat for his masters degree [Bro06].

# Chapter 3

# Hemispherical Harmonics

## 3.1 Motivations

The process of computing global illumination requires an accurate representation of the interactions between light and matter. Such interactions involve functions defined over the hemisphere: the radiance impinging at a given point can be represented as an hemispherical function. The BRDFs of the materials are also defined on the cartesian product of two hemispheres: the hemispheres of incoming and outgoing directions. The outgoing radiance at a given point is defined as the dot product of the incoming radiance function and the BRDF. Therefore, an accurate and compact representation of hemispherical functions is crucial for fast computation of global illumination. As described in the previous chapter, current research particularly focused on the adaptation of the well-known spherical harmonics to the hemispherical domain. However, representing an hemispherical function using spherical harmonics introduces a discontinuity on the equator of the sphere, therefore requiring a high number of coefficients for accurate representation.

In this chapter, we first introduce the core of spherical and hemispherical harmonics: the orthogonal polynomials and the generalized Fourier transform. Then, we propose a simple derivation of the spherical harmonics yielding a new set of basis functions defined over the upper hemisphere only [GKPB04]. Since computing a global illumination solution often involves the definition and combination of hemispherical functions defined in distinct frames, we describe simple methods for rotating functions projected into hemispherical harmonics. As spherical harmonics, our basis is orthogonal: the value of the dot product of two projected functions can be obtained through a simple dot product of the respective vectors of projection coefficients. We use our basis to represent BRDFs and hemispherical radiance functions, and demonstrate realtime performance for environment map rendering. Our basis is also applied for representing incoming radiance functions and BRDFs in the context of global illumination computation using radiance caching [Kři05].

## 3.2 Orthogonal Polynomials

### 3.2.1 Overview

A set of polynomials $\{p_l(x)\}_{l\geq 0}$ are said to be orthogonal over an interval $[a,b]$ if $p_l(x)$ is a polynomial of degree $l$, and for $n,m \geq 0$

$$\int_a^b w(x)p_n(x)p_m(x)\mathrm{d}x = \delta_{nm}c_n \tag{3.1}$$

where $\delta_{nm}$ is the Knonecker delta (0 or 1 as $n \neq m$ or $n = m$), and $w(x)$ is a non negative weighting function. If $c_n = 1$ then the set of polynomials form an orthonormal basis [Sze75]. Any integrable function $f(x)$ can be expanded in generalized Fourier series with respect to this set of polynomials:

$$f(x) \approx \sum_{n=0}^{\infty} f_n p_n(x) \qquad \text{where} \qquad f_n = \int_a^b f(x)p_n(x)w(x)\mathrm{d}x \tag{3.2}$$

Therefore, any integrable function can be represented by a set of coefficients and an orthonormal basis. For implementation purposes, a finite number of coefficients is used to represent a function. However, unlike polynomials, the function $f$ can be discontinuous, hence the expansion of $f$ may not exactly represent $f$. In this case, the expansion of $f$ into generalized Fourier series leads to the Gibbs phenomenon [Wei06a], also known as *ringing*: the Fourier series exhibit large oscillations near the discontinuity (see Figure 3.1). Those oscillations can be reduced by increasing the number of coefficients, but they cannot be completely removed. In Section 3.5, we examine the consequences of this phenomenon in the context of function representation for global illumination.

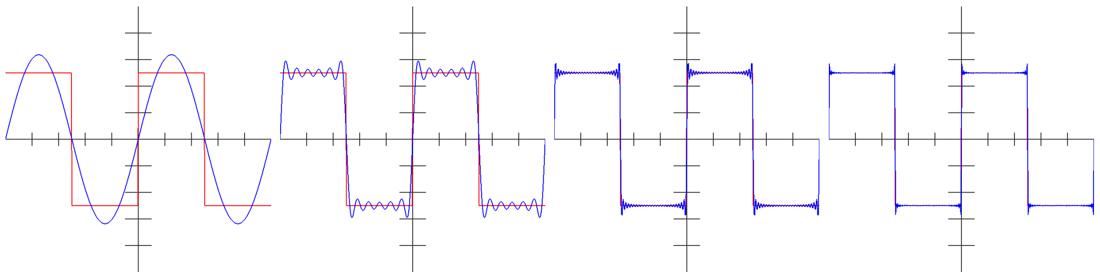

(a) Order 1 (b) Order 10 (c) Order 50 (d) Order 150

Figure 3.1: Representation of a square signal (red) using Fourier series (blue). The oscillations near the discontinuities (Gibbs phenomenon) are reduced by increasing the order of the representation, but cannot be completely removed.

In the remainder of this chapter, we focus on Legendre polynomials which form the base of both spherical harmonics and our hemispherical basis.

### 3.2.2 Associated Legendre Polynomials

Associated Legendre polynomials $\{P_l^m(x)\}$ with $l \geq 0$ and $-l \leq m \leq l$ are orthogonal polynomials of degree $l$ defined on the interval $[-1, 1]$ with weighting function $w(x) = 1 \ \ \forall x \in [-1, 1]$ [Wei06b]. Associated Legendre polynomials are orthogonal with respect to $l$:

$$\int_{-1}^{1} P_l^m(x) P_{l'}^m(x) \mathrm{d}x = \frac{2(m+l)!}{(2l+1)(l-m)!} \delta_{ll'}. \tag{3.3}$$

Those polynomials can be efficiently evaluated using simple recurrence relations:

$$(l-m)P_l^m(x) = x(2l-1)P_{l-1}^m(x) - (l+m-1)P_{l-2}^m(x) \tag{3.4}$$
$$P_m^m(x) = (-1)^m (2m-1)!!(1-x^2)^{m/2} \tag{3.5}$$
$$P_{m+1}^m = x(2m+1)P_m^m(x) \tag{3.6}$$

where !! denotes the double factorial operator.

As the associated Legendre polynomials $P_l^m(x)$ are defined on the interval $[-1, 1]$, setting $x = \cos\theta$ yields a set of basis functions $P_l^m(\cos\theta)$ defined on the angular interval $[0, \pi]$. This variable change is the first step towards the definition of spherical harmonics.

### 3.2.3 Spherical Harmonics (SH)

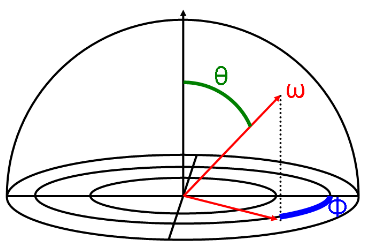


Figure 3.2: Spherical coordinates: any direction $\omega$ can be represented by a polar angle $\theta$ and an azimuthal angle $\phi$.

Spherical harmonics extend the definition domain of the associated Legendre polynomials by performing an additional transform on the azimuthal angle $\phi$ (Figure 3.2). Hence real-valued spherical harmonics are defined as:

$$Y_l^m(\theta, \phi) = \begin{cases} \sqrt{2} K_l^m \cos(m\phi) P_l^m(\cos\theta) & \text{if } m > 0 \\ \sqrt{2} K_l^m \sin(-m\phi) P_l^{-m}(\cos\theta) & \text{if } m < 0 \\ K_l^0 P_l^0(\cos\theta) & \text{if } m = 0 \end{cases} \tag{3.7}$$

where $K_l^m$ is the SH normalization value:

$$K_l^m = \sqrt{\frac{(2l+1)(l-|m|)!}{4\pi(l+|m|)!}} \tag{3.8}$$

Figure 3.4 represents the first spherical harmonics.

Any integrable function $f$ defined on the hemisphere $\Omega$ can be projected into the spherical harmonics basis:

$$f(\omega) \approx \sum_{l=0}^{\infty}\sum_{m=-l}^{l} f_l^m Y_l^m(\omega) \quad \text{where} \quad f_l^m = \int_{\Omega} f(\omega)Y_l^m(\omega)\mathrm{d}\omega \tag{3.9}$$

One of the most interesting properties of spherical harmonics is their orthogonality. Let $f$ and $g$ be two functions defined on the sphere, and $f_l^m$ and $g_l^m$ be their respective projection coefficients. The functional dot product of $f$ and $g$ is:

$$< f, g > = \int_{\Omega} f(\omega)g(\omega)\mathrm{d}\omega \tag{3.10}$$

$$\approx \int_{\Omega}\left(\sum_{l=0}^{\infty}\sum_{m=-l}^{l} f_l^m Y_l^m(\omega)\right)\left(\sum_{l'=0}^{\infty}\sum_{m'=-l'}^{l'} g_{l'}^{m'} Y_{l'}^{m'}(\omega)\right)\mathrm{d}\omega \tag{3.11}$$

The orthogonality property of spherical harmonics is:

$$\int_{\Omega} Y_l^m(\omega)Y_{l'}^{m'}(\omega)\mathrm{d}\omega = \delta_{ll'}\delta_{mm'} \tag{3.12}$$

Therefore, the terms containing $Y_l^m(\omega)Y_{l'}^{m'}(\omega)$ with $m \neq m'$ or $l \neq l'$ cancel out:

$$< f, g > \approx \sum_{l=0}^{\infty}\sum_{m=-l}^{l}\left(\int_{\Omega} f_l^m Y_l^m(\omega)\sum_{l'=0}^{\infty}\sum_{m'=-l'}^{l'} g_{l'}^{m'} Y_{l'}^{m'}(\omega)\right)\mathrm{d}\omega \tag{3.13}$$

$$\approx \sum_{l=0}^{\infty}\sum_{m=-l}^{l} f_l^m g_l^m \tag{3.14}$$

Hence, the dot product of two functions projected into spherical harmonics basis reduces to the dot product of the vectors of projection coefficients. This property is very useful in the context of global illumination: if both the incoming radiance and the cosine weighted BRDF are represented using spherical harmonics, Equation 2.2 reduces to a simple vector dot product, which can be computed in realtime.

Another important property of spherical harmonics is the ability to rotate a projected function by applying a simple rotation matrix to the vector of coefficients. This matrix is obtained using recursion, starting from a regular $3 \times 3$ rotation matrix [IR96, IR98, CIGR99, BFB97]. In the context of radiance caching, the estimate of the incoming radiance at a point is obtained by interpolating the projection coefficients of nearby records. However, such records are generally defined in different coordinate systems. To compute the interpolated incoming radiance at a point $\mathbf{p}$, the incoming radiance of each record must be rotated to match the local frame at $\mathbf{p}$ (Figure 3.3).

However, spherical harmonics are not well-suited for representing functions defined on a single hemisphere such as the incoming radiance functions or the BRDFs. In particular, hemispherical functions introduce a discontinuity at the equator of the sphere,

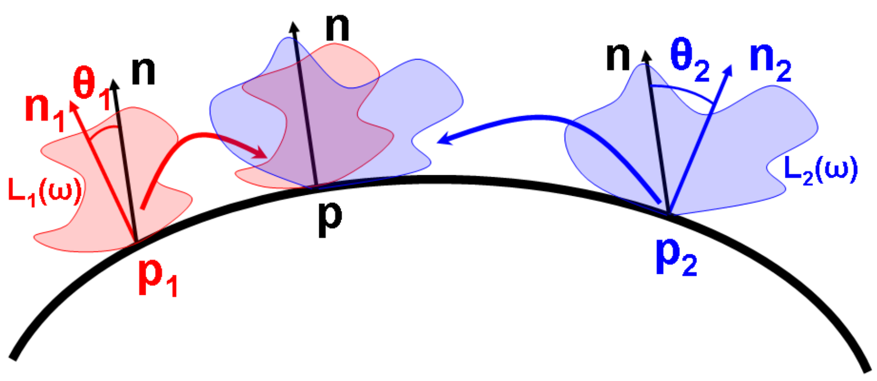


Figure 3.3: In the radiance caching algorithm, the incoming radiance functions $L_1(\omega)$ and $L_2(\omega)$ are defined in the local frames at points $p_1$ and $p_2$. To perform the interpolation of those records at point $\mathbf{p}$, $L_1(\omega)$ and $L_2(\omega)$ must be expressed in terms of the local coordinate system at $\mathbf{p}$. This change of coordinate system can be carried out by a rotation of angle $\theta_1$ for $L_1(\omega)$ and of angle $\theta_2$ for $L_2(\omega)$.

yielding ringing artifacts. Furthermore, the representation is not efficient, since only half of the representation domain of spherical harmonics is used. We propose a set of basis functions defined on the hemisphere rather than on the sphere. Our basis offers the advantages of spherical harmonics, such as the simplicity of the dot product, and rotations.

## 3.3 A Novel Hemispherical Basis

As spherical harmonics are defined on the sphere, the basic idea of our basis is to shift the definition domain of spherical harmonics so that it fits the hemisphere. Therefore, instead of using the regular associated Legendre polynomials, we use a shifted version of these polynomials.

### 3.3.1 Shifted Polynomials

Shifting is a linear mapping that maps $x$ to $k_1x + k_2$, where $k_1 \neq 0$. If the polynomials $\{p_l(x)\}_{l\geq 0}$ are orthogonal over an interval $[a, b]$ with a weighting function $w(x)$, then the polynomials $\{p_l(k_1x + k_2)\}_{l\geq 0}$ are orthogonal over the interval $[\frac{a-k_2}{k_1}, \frac{b-k_2}{k_1}]$ with the weighting function $w(k_1x + k_2)$ [Sze75]. If $\{p_l(x)\}_{l\geq 0}$ are orthonormal then $\{(\text{sign } k_1)^l \sqrt{|k_1|} p_l(k_1x + k_2)\}_{l\geq 0}$ are orthonormal. The polynomials $\{p_l(k_1x + k_2)\}_{l\geq 0}$ are the shifted versions of $\{p_l(x)\}_{l\geq 0}$.

Using the linear mapping of $x$ to $2x - 1$ we obtain shifted associated Legendre polynomials [Wei06b] over the interval $x \in [0, 1]$:

$$\widetilde{P}_l^m(x) \quad = \quad P_l^m(2x - 1) \tag{3.15}$$

The shifted polynomials remain orthogonal, but the normalization changes according to the definition given above. The following equation shows the orthogonal relation with respect to $l$. Note that $w(x) = 1$ for Legendre polynomials. We get:

$$\begin{aligned} \int_0^1 \widetilde{P}_l^m(x)\widetilde{P}_{l'}^m(x)dx &= \int_0^1 P_l^m(2x-1)P_{l'}^m(2x-1)\mathrm{d}x \\ &= \frac{(m+l)!}{(2l+1)(l-m)!}\delta_{ll'} \end{aligned} \tag{3.16}$$

The rationale behind this shifting is that by replacing the argument $x$ with $\cos\theta$, we get functions $\{\widetilde{P}_l^m(\cos\theta)\}$. Therefore, those polynomials are defined in the interval $\theta \in [0, \frac{\pi}{2}]$, which is the polar angle range of the hemisphere:

$$\widetilde{P}_l^m(\cos\theta) \quad = \quad P_l^m(2\cos\theta - 1) \text{ with } \theta \in [0, \frac{\pi}{2}]. \tag{3.17}$$

### 3.3.2 Hemispherical Harmonics

As spherical harmonics functions are constructed from associated Legendre polynomials, we construct our hemispherical basis functions $\{H_l^m(\theta,\phi)\}$ from shifted associated Legendre polynomials (Figure 3.4).

$$H_l^m(\theta,\phi) = \begin{cases} \sqrt{2}\widetilde{K}_l^m \cos(m\phi)\widetilde{P}_l^m(\cos\theta) & \text{if } m > 0 \\ \sqrt{2}\widetilde{K}_l^m \sin(-m\phi)\widetilde{P}_l^{-m}(\cos\theta) & \text{if } m < 0 \\ \widetilde{K}_l^0\widetilde{P}_l^0(\cos\theta) & \text{if } m = 0 \end{cases} \tag{3.18}$$

with the following normalization value:

$$\widetilde{K}_l^m = \sqrt{\frac{(2l+1)(l-|m|)!}{2\pi(l+|m|)!}} \tag{3.19}$$

By analogy with spherical harmonics, we name the $H_l^m(\theta,\phi)$ basis functions "hemispherical harmonics", or HSH. HSH are orthogonal over $[0, \frac{\pi}{2}] \times [0, 2\pi)$ with respect to both $l$ and $m$.

In methods such as radiance caching [Kři05] and precomputed radiance transfer [SKS02], the projected hemispherical functions have to be rotated for combining functions defined in different frames. In the following section, we describe our rotation methods for hemispherical harmonics.

## 3.4 Hemispherical Harmonics Rotation

In the context of realistic rendering using HSH, one might have to rotate HSH-projected functions efficiently. We propose three methods dealing with such rotations: by converting hemispherical to spherical harmonics, using spherical harmonics rotation matrices, and precomputing rotation matrices.

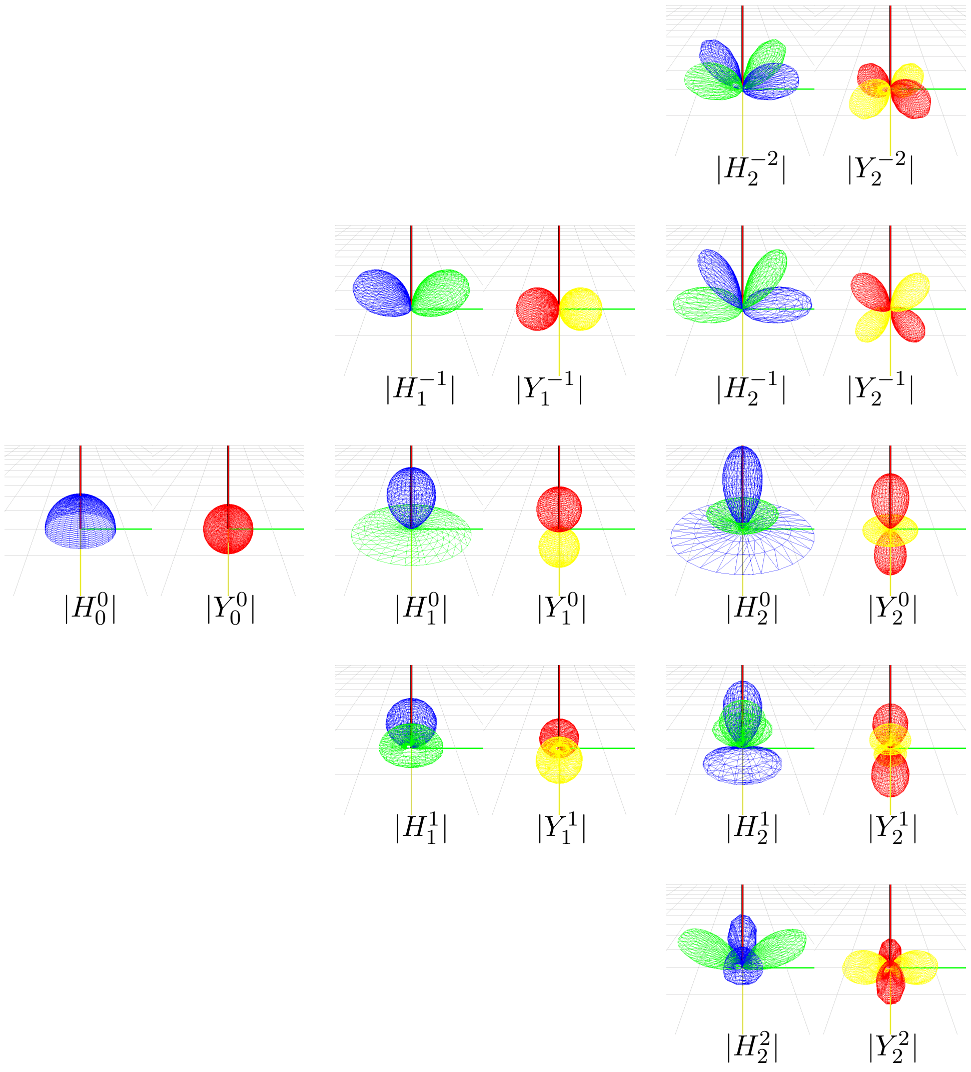


Figure 3.4: Hemispherical and spherical harmonics for $l$ from 0 to 2.

### 3.4.1 Conversion Matrix

This rotation method relies on the use of the rotation matrices of the spherical harmonics. To this end, we propose a conversion matrix for converting hemispherical harmonics coefficients into spherical harmonics and vice-versa. We define a basis change matrix $\mathbf{C}$, which expresses each hemispherical harmonic in terms of spherical harmonics:

$$\mathbf{C}_{l,l'}^{m,m'} = \int_0^{2\pi} \int_0^{\pi/2} H_l^m(\theta,\phi) Y_{l'}^{m'}(\theta,\phi) \sin\theta \mathrm{d}\theta \mathrm{d}\phi \tag{3.20}$$

Even though shifted and nonshifted Legendre polynomials are not mutually orthogonal, the $\phi$-dependent parts of SH and HSH are identical. Therefore, we can prove that the spherical and hemispherical harmonics are orthogonal with respect to $m$:

$$\begin{aligned}
\mathbf{C}_{l,l}^{m,m'} &= \textstyle\int_0^{2\pi}\int_0^{\pi/2} H_l^m(\theta,\phi)Y_l^{m'}(\theta,\phi)\sin\theta \mathrm{d}\theta\mathrm{d}\phi && (3.21)\\
&= \textstyle\int_0^{2\pi}\int_0^{\pi/2} \widetilde{K}_l^m K_l^{m'}\Phi(m\phi)\Phi(m'\phi)\widetilde{P}_l^m(\cos\theta)P_l^m(\cos\theta)\sin\theta \mathrm{d}\theta\mathrm{d}\phi && (3.22)\\
&= \widetilde{K}_l^m K_l^{m'} \textstyle\int_0^{2\pi}\Phi(m\phi)\Phi(m'\phi)\int_0^{\pi/2}\widetilde{P}_l^m(\cos\theta)P_l^m(\cos\theta)\sin\theta \mathrm{d}\theta\mathrm{d}\phi && (3.23)
\end{aligned}$$

Since $\int_0^{\pi/2}\widetilde{P}_l^m(\cos\theta)P_l^m(\cos\theta)\sin\theta \mathrm{d}\theta$ does not depend on $\phi$, we obtain:

$$\mathbf{C}_{l,l}^{m,m'} = \widetilde{K}_l^m K_l^{m'}\int_0^{\pi/2}\widetilde{P}_l^m(\cos\theta)P_l^m(\cos\theta)\sin\theta \mathrm{d}\theta\int_0^{2\pi}\Phi(m\phi)\Phi(m'\phi)\mathrm{d}\phi \qquad (3.24)$$

Where

$$\Phi(m\phi) = \begin{cases} \cos(m\phi) & \text{if } m > 0 \\ \sin(-m\phi) & \text{if } m < 0 \\ 1 & \text{if } m = 0 \end{cases} \qquad (3.25)$$

However,

$$\int_0^{2\pi}\cos(m\phi)\cos(m'\phi)\mathrm{d}\phi = \int_0^{2\pi}\sin(-m\phi)\sin(-m(\phi)\mathrm{d}\phi = 0 \text{ if } m \neq m' \qquad (3.26)$$

Therefore, the conversion matrix is very sparse as nonzero values occur only for $m = m'$. This significantly reduces the cost of the conversion from SH to HSH representation. For $l < 3$ and $l' < 3$ the conversion matrix $\mathbf{C}_{3,3}$ is:

$$\begin{pmatrix}
0.71 & 0 & 0.41 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0.60 & 0 & 0 & 0 & 0.19 & 0 & 0 & 0\\
0.82 & 0 & 0.71 & 0 & 0 & 0 & 0.18 & 0 & 0\\
0 & 0 & 0 & 0.60 & 0 & 0 & 0 & 0.19 & 0\\
0 & 0 & 0 & 0 & 0.53 & 0 & 0 & 0 & 0\\
0 & 0.77 & 0 & 0 & 0 & 0.51 & 0 & 0 & 0\\
0.40 & 0 & 0.78 & 0 & 0 & 0 & 0.53 & 0 & 0\\
0 & 0 & 0 & 0.77 & 0 & 0 & 0 & 0.51 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.53
\end{pmatrix}$$

The HSH rotation can be carried out by converting the HSH coefficients to SH using $\mathbf{C}$, then applying a SH rotation matrix, and finally convert the SH coefficients back to HSH using $\mathbf{C}^T$. Since the SH representation is less accurate than HSH, we propose to use a non-square conversion matrix: the intermediate SH representation uses more coefficients than HSH. It must be noted that this method automatically removes the data that disappears underneath the horizon during rotation (Figure 3.6(c)).

### 3.4.2 Digon Deletion

The next two methods rely on Euler's rotation theorem: any rotation may be described using three angles. We represent our rotations by the $ZYZ$ convention, where $Z$ is the vertical axis of a right handed orthogonal coordinate system.

The rotation of HSH functions around the $Z$ axis is the same as that of SH functions [SAWG91b]. The rotation around the $Y$ axis is lossy for all hemispherical functions: after such a rotation, a part of the function must be set to zero (Figure 3.6). Hence we carry out the $Y$ axis rotation in two steps.

In the first step we delete the hemisphere digon $I$ (Figure 3.5) that disappears after the rotation around axis $Y$ by angle $\beta$ (Figure 3.6(b)). We carry out this deletion by multiplying the coefficient vector by a matrix $\mathbf{M}(\beta)$. This matrix expresses each hemispherical harmonic deprived of the digon $I$ in terms of the regular hemispherical harmonics. The elements of $\mathbf{M}(\beta)$ are defined as:

$$\mathbf{M}_{l,l'}^{m,m'}(\beta) = \int_{H-I} H_l^m(\theta,\phi) H_{l'}^{m'}(\theta,\phi) \sin\theta \mathrm{d}\theta \mathrm{d}\phi \tag{3.27}$$

where $H - I$ is the hemisphere after the removal of the digon. By abruptly setting the function to 0 in the digon $I$, the digon deletion adds high frequencies in the signal. Consequently, the matrix $\mathbf{M}(\beta)$ is dense. For practical purposes, we precompute matrices for a set of rotation angles, then linearly interpolate them for any other angle at runtime. Note that the removal of the digon introduce high frequencies, and hence the shape of the rotated function may not be completely identical to the original function (Figure 3.7). However, compared to a rotation without the digon deletion, the change of shape is smaller.

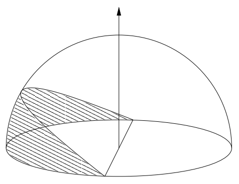


Figure 3.5: Hemisphere and a digon [Dut03]. The digon is the shaded part of the hemisphere.

Once this "deletion" transformation is applied, we compute and apply a rotation matrix for rotation around $Y$. Since HSH definition is close to SH, we first chose to use SH rotation matrices [IR96, IR98] to derive HSH rotations.

### 3.4.3 Analytical Solution

Applying a SH rotation matrix of angle $\beta_{SH}$ around the $Y$ axis to a HSH vector of coefficients $\{f_i\}$ is equivalent to:

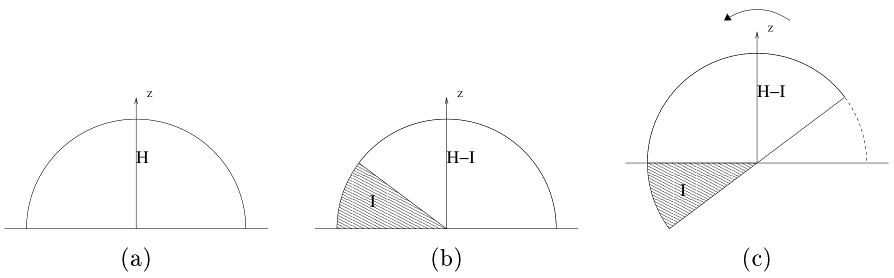


Figure 3.6: HSH rotation around $Y$ axis. Hemisphere before rotation (a), shaded digon that is set to zero (b), Hemisphere after rotation (c).

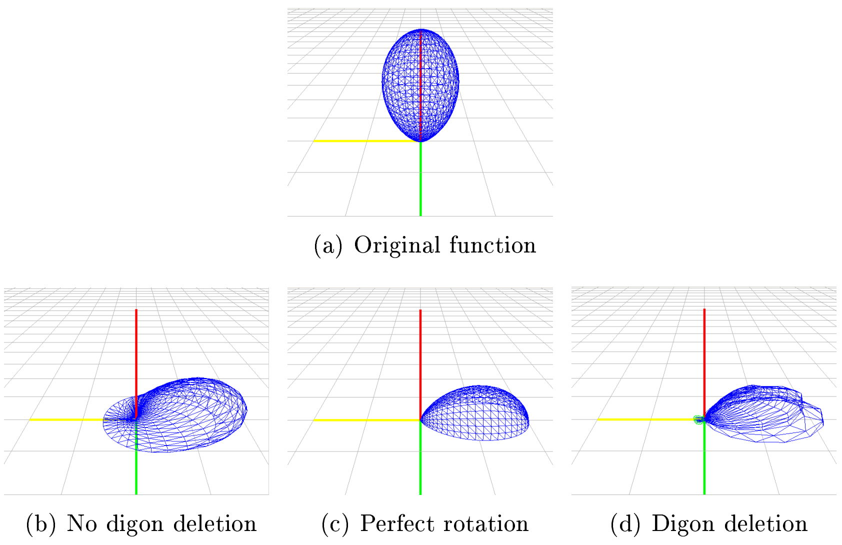


Figure 3.7: When rotating a function defined on the hemisphere (a), the function partially vanishes underneath the equator (b). If this vanishing part is not properly removed when rotating a HSH-projected function, the shape of the function is flattened on the equator, introducing artifacts (c). Using our digon deletion matrix, the vanishing parts are removed before performing the rotation, hence reducing the artifacts (d).

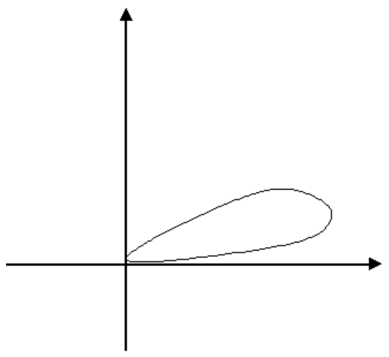
- shifting (i.e. mapping $x$ to $2x - 1$) the HSH-projected function $f_{HSH}$ to the whole sphere (Equations 3.28, 3.29), yielding $f_{SH}$ (Figure 3.8(b)).

$$f_{HSH}(\theta, \phi) = \sum_i f_i H_i(\theta, \phi) \quad [0, \frac{\pi}{2}] \times [0, 2\pi] \to \mathbf{R} \tag{3.28}$$

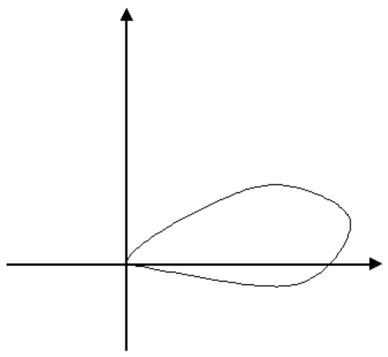
$$f_{SH}(\theta, \phi) = \sum_i f_i Y_i(\theta, \phi) \quad [0, \pi] \times [0, 2\pi] \to \mathbf{R} \tag{3.29}$$

- applying a SH rotation matrix $R^{SH}_{\beta_{SH}}$ to $f_{SH}$ (Figure 3.8(c))

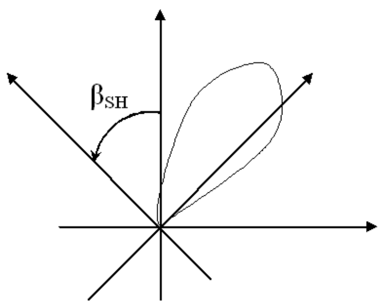
- shifting $R^{SH}_{\beta_{SH}}(f_{SH})$ to the upper hemisphere to obtain $R^{HSH}_{\beta_{HSH}}(f_{HSH})$ (Figure 3.8(d))



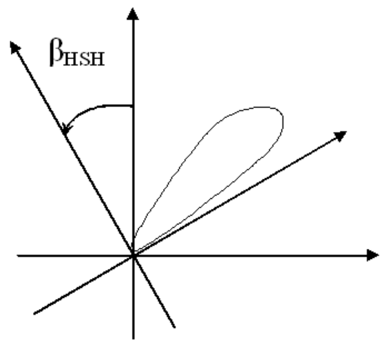
(a) Original function $f_{HSH}$



(b) $f_{SH}$ obtained by shifting $f_{HSH}$ to the entire sphere (i.e. using SH instead of HSH basis functions)



(c) Spherical harmonics rotation applied to $f_{SH}$



(d) The rotated hemispherical function is obtained by shifting $R^{SH}_{\beta_{SH}}(f_{SH})$ to the upper hemisphere

Figure 3.8: HSH rotation process : the original hemispherical function is first shifted to the whole sphere, then rotated using SH rotation matrices. The last step shifts the rotated function back to the hemisphere.

Our aim is to determine the relation between $\beta_{SH}$ and $\beta_{HSH}$ : since the definition domains of SH and HSH are different, $\beta_{SH} \neq \beta_{HSH}$. Using Equation 3.17, a rotation of angle $\beta_{HSH}$ on the hemisphere is equivalent to a rotation of angle $\beta_{HSH}$ (Equation 3.30).

$$\beta_{SH} = \arccos(2\cos\beta_{HSH} - 1) \tag{3.30}$$

This equation shows that the SH rotation matrices can be applied to HSH for a rotation around $Y$ axis provided that no part of the original function moves into the lower hemisphere after rotation (this condition is fulfilled by removing the considered digon (Equation 34)).

### 3.4.4 Precomputed Rotation

The last rotation method consists of the inclusion of the rotation in the precomputed matrices in Equation 34. The matrix element values are defined by

$$\mathbf{M_R}_{l,l'}^{m,m'}(\beta) = \int_{H-I} H_l^m(\theta,\phi) H_{l'}^{m'}(R_\beta(\theta,\phi)) \sin\theta \mathrm{d}\theta \mathrm{d}\phi \tag{3.31}$$

where $H - I$ is the hemisphere after removing the digon, and $R_\beta(\theta,\phi)$ represents the direction $(\theta,\phi)$ rotated around $Y$ axis by angle $\beta$.

The accuracy of this method comes at the expense of memory: an accurate rotation requires a large number of precomputed matrices.

The next section first presents the results obtained for the representation of hemispherical functions using hemispherical harmonics, followed by the applications of HSH in the context of image synthesis.

## 3.5 Applications and Results

### 3.5.1 Representation of Hemispherical Functions

The definition domain of HSH basis functions makes them ideal for representation of hemispherical functions such as BRDFs and radiance functions around surface points. In this section we show a comparison between HSH representation and previous methods in term of accuracy.

We first show the approximation of a Phong lobe [Pho75] $(\cos\alpha)^5$, where $\alpha$ is the angle between the specular reflection direction and the viewing direction. Figure 3.9 plots the approximation accuracy as function of the number of coefficients in HSH, SH, ESH, Zernike and Makhotkin's representation (see Chapter 2 for details on these representations). We estimate this accuracy of the representation of a function $f(\theta,\phi)$ by computing the fraction of the total energy captured for a given number of coefficients [RH02]:

$$\text{Accuracy} = \frac{\displaystyle\sum_{l=0}^{n} \sum_{m=-l}^{l} |f_l^m|^2}{\displaystyle\int_0^{2\pi} \int_0^{\frac{\pi}{2}} f(\theta,\phi)^2 \sin\theta \mathrm{d}\theta \mathrm{d}\phi} \tag{3.32}$$

The observation in this figure and the results in later sections show that the HSH representation indeed provides a better representation than SH with fewer coefficients. Moreover, our basis performs comparably to Makhotkin's method, and exhibits a higher accuracy than both ESH and Zernike approaches in the Phong case.

### 3.5.2 BRDF Representation

BRDFs are functions defined over the cartesian product of two hemispheres, corresponding to incoming and outgoing directions. One can use either of the following

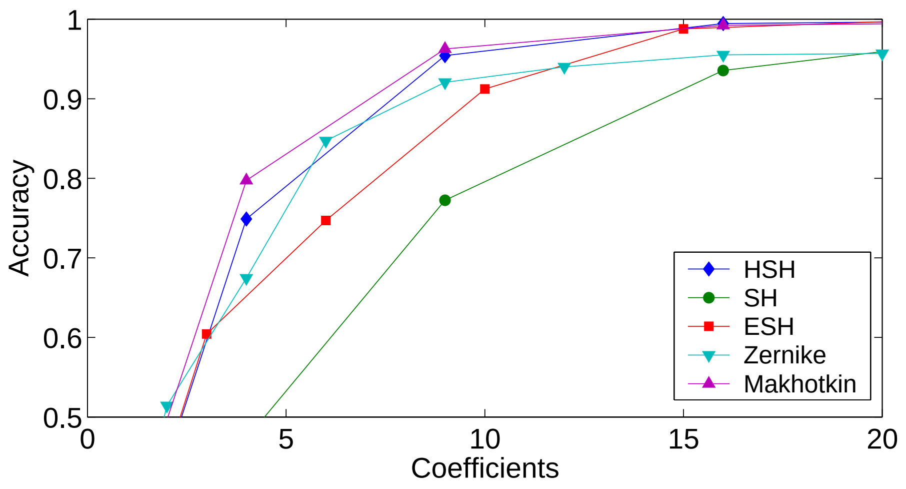


Figure 3.9: Accuracy against number of coefficients for approximation of a Phong lobe, $(\cos\alpha)^5$, using HSH, SH, ESH, Zernike and Makhotkin approaches.

two approaches to represent them using HSH basis. One approach is to use two successive HSH transformations, one for the incoming hemisphere and the other for the outgoing hemisphere (similar to the approach used by Westin et al. [WAT92] using SH basis). The other approach is to use HSH to represent the incoming hemisphere for each sampled outgoing direction (similar to the approach used by Kautz *et al.* [KSS02]).

We use the latter approach, in which the $n^{\text{th}}$ order representation of a BRDF $f_{(\theta_o,\phi_o)}$ for any given outgoing sample direction $(\theta_o,\phi_o)$ is

$$f_{(\theta_o,\phi_o)}(\theta_i,\phi_i) \approx \sum_{l=0}^{n}\sum_{m=-l}^{l} c_l^m(\theta_o,\phi_o) H_l^m(\theta_i,\phi_i) \ , \tag{3.33}$$

where

$$c_l^m(\theta_o,\phi_o) = \int_0^{2\pi}\int_0^{\pi/2} f(\theta_o,\phi_o,\theta_i,\phi_i) H_l^m(\theta_i,\phi_i) \sin\theta_i \mathrm{d}\theta_i \mathrm{d}\phi_i \ . \tag{3.34}$$

We sample the outgoing hemisphere using the paraboloid parametrization proposed by Heidrich and Seidel [HS99]. In this method, a direction $(x,y,z)$ is mapped onto a 2D plane $(u,v)$ as follows:

$$u = \frac{1}{2}(1+\frac{x}{z+1}) \tag{3.35}$$

$$v = \frac{1}{2}(1+\frac{y}{z+1}) \tag{3.36}$$

Using this mapping technique, the projection coefficients of a BRDF can be easily stored on graphics hardware as a 3D texture with texture coordinates $(u,v,w)$, where $(u,v)$ is the outgoing direction, and $w$ is the coefficient index.

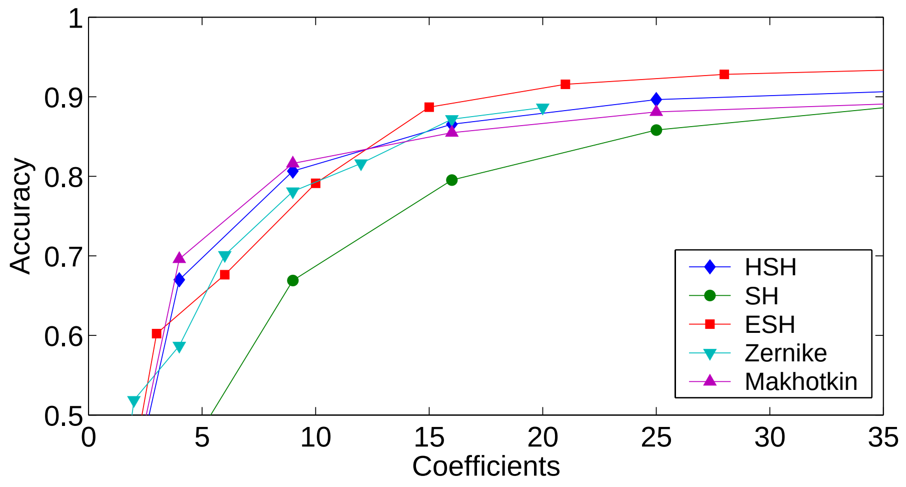


Figure 3.10: Accuracy against number of coefficients for HSH, SH, ESH, Zernike and Makhotkin representations. These values are obtained for an anisotropic Ward BRDF with $k_d = 0$, $k_s = 1$, $\alpha_x = 0.2$, $\alpha_y = 0.5$.

Figure 3.10 plots the approximation accuracy of the HSH, SH, ESH, Zernike and Makhotkin representation of an anisotropic Ward BRDF [War92] as a function of the number of coefficients. The plot shows that although the ESH provides better results in this case, the HSH representation is still more accurate than the SH representation when the same number of coefficients is used. As in the previous example, HSH, Zernike and Makhotkin's method perform comparably.

Figure 3.11 shows image frames from our GPU-based real-time renderer. The teapots shown in the images are lit by a single point light source. Those in the left column are assigned a Phong BRDF, and in the right column an anisotropic Ward BRDF. The renderings in the three rows from top to bottom correspond to analytic BRDF functions, and their HSH and SH representations respectively. $32 \times 32$ outgoing sample directions are used for the HSH and SH representations. The coefficients and the basis function values are stored into textures. For any incoming direction $(\theta_i, \phi_i)$ and outgoing direction $(\theta_o, \phi_o)$, $c_l^m(\theta_o, \phi_o)$ and $H_l^m(\theta_i, \phi_i)$ values are computed by bilinear texture interpolation. Equation 3.33 for $f_{(\theta_o,\phi_o)}(\theta_i, \phi_i)$ is evaluated using a multi-pass pixel shader program. The images using HSH representation (Figures 3.11(c) and (d)) are visually closer to the images obtained using analytical BRDFs. Note the ringing in the images obtained using SH representation of the BRDF (Figures 3.11(e) and (f)).

Figures 3.10 and 3.11 demonstrate that HSH representation of BRDF using a given number of coefficients outperforms SH representation both in terms of computed and visual accuracy.

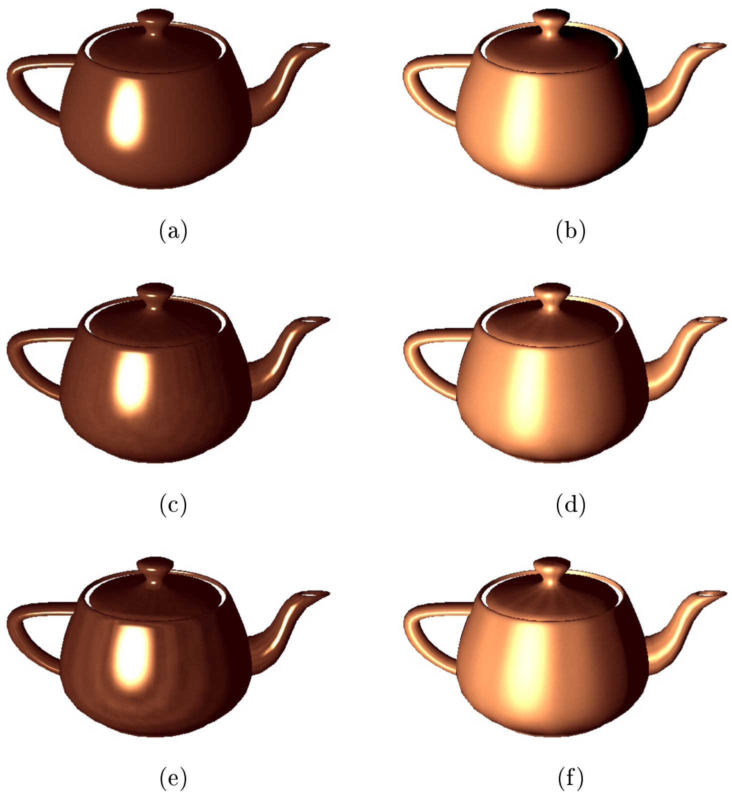


Figure 3.11: Images rendered using Phong BRDF (left column) and anisotropic Ward BRDFs (right column). Figures (a) and (b) use analytic BRDFs; Figures (c) and (d) use order 7 HSH representation and Figures (e) and (f) use order 7 SH representation.

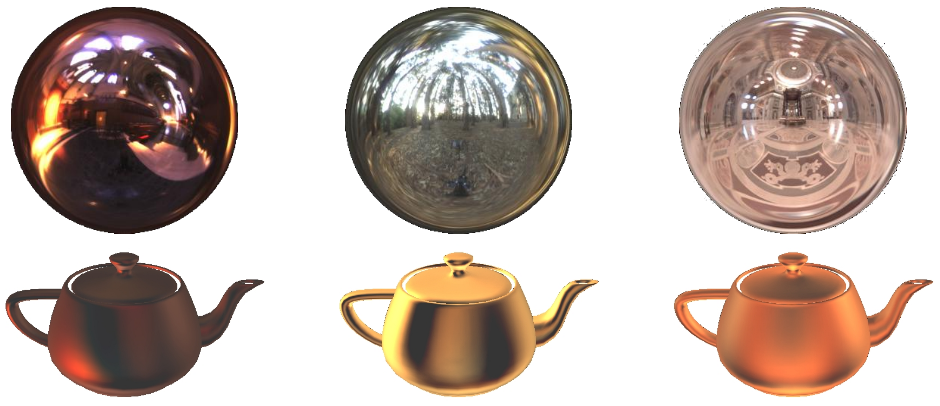
### 3.5.3 Environment Map Lighting

Hardware rendering of surfaces with arbitrary BRDFs illuminated by environment maps has received wide attention in the last few years. Environment maps are discrete representations of incoming radiance from directions defined on a whole sphere. Consequently they can be approximated using a spherical basis such as SH. In this section, we show that a SH representation of environment maps can be easily combined with a HSH representation of BRDFs. Our GPU-based renderer implements interactive hardware rendering using HSH representation for BRDFs and SH representation for environment maps.

Since SH and HSH bases are not mutually orthogonal, we cannot directly compute an integral containing a SH representation of an environment map and a HSH representation of a BRDF. However, incoming light from the environment at any surface point is only defined on a hemisphere. Consequently, we use the conversion matrix defined in Section 3.4.1 to convert the environment map coefficients from SH to HSH after rotating them from the global to the local frame.

Rendering with environment maps proceeds as follows:

- Compute SH representation of the environment map.
- Compute HSH representation of the surface BRDF for every surface in the scene
- For every rendered surface point:
  - Rotate the coefficients of the environment map to fit the local frame associated with the surface point.
  - Transform the coefficients from SH to HSH using the precomputed basis change matrix **C**.
  - Compute the dot product between the HSH coefficients of the environment map and HSH coefficients of the surface BRDF for the required outgoing direction.



(a) Grace Cathedral (b) Eucalyptus Grove (c) St Peter's Basilica

Figure 3.12: Environment map rendering using HSH representation of the BRDF (order 10). The glossy teapot is lit by the environment maps shown in upper row (Environment maps courtesy of Paul Debevec).

Figure 3.12 shows renderings of a teapot using three different environment maps. Table 3.1 compares frame rates achieved by the HSH approximated BRDF against the SH approximated BRDF. Our method achieves frame rates similar to these obtained with the SH approach for the same number of coefficients since the conversion step does not require much additional computation. Note that the ESH representation cannot be used in this context because of the extra non-zero BRDF information present in the lower hemisphere: the dot product between the environment map and the BRDF would account for both upper and lower hemispheres, leading to erroneous results.

| Order of representation ($n$) | 2 | 3 | 5 | 8 | 10 |
|---|---|---|---|---|---|
| HSH (fps) | 220 | 138 | 26.2 | 3.1 | 1.6 |
| SH (fps) | 223 | 160 | 29 | 3.28 | 1.73 |

Table 3.1: Timing results (in frames per second) for environment maps rendering using HSH and SH representation of BRDFs. The timing was obtained on a 2.4GHz Intel Xeon with ATI Radeon 9800 Pro card for an image size 700x600. The teapot model used has 1873 vertices.

### 3.5.4 Radiance Caching

The hemispherical harmonics basis can also be used in the context of the radiance caching algorithm [KGPB05, KGBP05, Kři05]. As described in Chapter 2, the radiance caching algorithm relies on the storage and interpolation of incoming radiance functions. Therefore, HSH can be used directly for representing both BRDF and incoming radiance functions. Theoretical and implementation details on radiance caching can be found in [Kři05].

## 3.6 Conclusion

We have devised a new basis for hemispherical data representation. This basis is ideal for the representation of low-frequency hemispherical functions. Being very close to spherical harmonics, our basis offers the same benefits, such as the implementation simplicity and the low evaluation cost of the basis functions. Our tests show that the hemispherical harmonics are more accurate than previous methods. Compared to the spherical harmonics basis, our basis is able to represent functions of higher frequency without introducing noticeable artifacts due to the Gibbs phenomenon ("ringing").

Since our method relies on a simple derivation from spherical harmonics, we proposed a simple conversion scheme between hemispherical and spherical harmonics basis. The conversion is performed by applying a basis change matrix to the vector of projection coefficients. Due to a partial orthogonality between hemispherical and spherical harmonics, the conversion matrix is very sparse, hence improving computational efficiency of the conversion.

The similarity between SH and HSH also allows the definition of rotation methods operating directly on the vector of projection coefficients. When rotating hemispherical functions, part of the function may disappear under the equator. To overcome this problem, we first propose to convert the vector of HSH projection coefficients into SH using our conversion matrix, and to apply the desired rotation on the latter. The rotated SH vector is then converted back to HSH. Since HSH prove more accurate than SH, we propose to use more SH coefficients than HSH (that is, using a non-square conversion matrix) to avoid a loss of quality during the rotation. Using this method, the parts of the function that disappear under the equator are implicitly discarded by the conversion from SH to HSH. An other approach to the rotation requires the application of a "digon deletion" matrix. When applied before performing the rotation, this matrix explicitly sets to zero the parts of the function that will disappear under the equator, enhancing the accuracy of the rotation. Then, the rotation can be carried out by using modified SH rotation matrices. Since the "digon deletion" matrices are dense and must be precomputed, our last rotation method simply consists in including the rotation within the those precomputed matrices.

Accurate representation of hemispherical functions is often required in computer graphics. Our hemispherical harmonics can be used to enhance the accuracy and compacity of the representation BRDFs and incoming radiance functions for real-time environment map rendering of non-diffuse surfaces and efficient global illumination computation using radiance caching.

Chapter 4

# Radiance Cache Splatting

## 4.1 Motivations

High quality global illumination solutions are generally computed offline using ray tracing and Monte Carlo sampling [War94, PH04]. Even though several approaches have been proposed to render globally illuminated scenes in real-time [WBS03, GWS04, TPWG02, WS99], interactive methods based on ray tracing usually rely on parallel processing using computers clusters and multiprocessor architectures to maintain a reasonable frame rate. Efficient algorithms such as irradiance [WRC88] and radiance caching [KGPB05] can be used for speedup purposes. However, the computation of global illumination still requires hours when using a single off-the-shelf computer.

We propose a new method for irradiance and radiance caching which leverages graphics hardware and computes global illumination an order of magnitude faster than currently available caching-based methods. Our approach takes advantage of two observations made by Tabellion *et al.* [TL04] in the context of production rendering. First, one bounce of indirect lighting accounts for most of the light transfers in a scene, providing visually pleasing results. Second, Tabellion *et al.* [TL04] show that a coarsely tessellated scene is sufficient to compute an accurate indirect component of the global illumination solution. Therefore, we focus on the efficient computation of first bounce global illumination in moderately complex scenes. Our method provides a fast and simple way of computing indirect lighting in simplified geometry, while the direct lighting can be computed independently either by an offline renderer using a detailed geometry or by graphics hardware for interactive performance.

In this chapter, we reformulate the irradiance and radiance caching algorithms by defining a fast image-space method based on splatting. Our approach aims at reducing the CPU workload by performing most of the computation on graphics hardware. The GPUs being SIMD (Single Instruction Multiple Data) processors, they cannot handle efficiently complex data structures and algorithms such as the octrees and nearest-neighbors queries used in irradiance caching. Our method avoids the need for such algorithms and structures by introducing *radiance cache splatting* [GKBP05], an image-space technique determining the contribution of each record to the indirect lighting of visible objects. The radiance cache splatting and the whole rendering algorithm are described hereafter. For the sake of clarity, our algorithm is presented in the case of irradiance caching only. If necessary, specific details about the extension to radiance caching are given at the end of each subsection.

## 4.2 Radiance Cache Splatting

As described in Chapter 2, the irradiance caching algorithm relies on the computation and the interpolation of irradiance records. For a point visible through a pixel, the irradiance caching determines which records contribute to the indirect lighting of this point. The radiance cache splatting uses the reverse approach: for a given record, our algorithm determines which visible points it contributes to by splatting the record onto the image plane. The result of radiance cache splatting is stored in the *radiance splat buffer*, which has the same size as the frame buffer. Each pixel $SPLATBUFF(x, y)$ of the radiance splat buffer is a pair $(L_o, w)$, where $L_o$ is the sum of the weighted contributions of the records, and $w$ is the cumulated weight.

The radiance cache splatting (Algorithm 3) is designed for computing the contribution of an irradiance record to the indirect lighting of visible points. Our approach is based on the equation used in the irradiance caching interpolation scheme (Equation 2.8), recalled below:

$$E(\mathbf{p}) = \frac{\sum_{k \in S(\mathbf{p})} E_k(\mathbf{p}) w_k(\mathbf{p})}{\sum_{k \in S(\mathbf{p})} w_k(\mathbf{p})} \tag{4.1}$$

The estimate $E(\mathbf{p})$ of the indirect lighting at point $\mathbf{p}$ is the weighted average of the contributions of nearby irradiance records evaluated at point $\mathbf{p}$. The weight allocated to a record $k$ at point $\mathbf{p}$ with normal $\mathbf{n}$ is defined in [WRC88]:

$$w_k(\mathbf{p}) = \frac{1}{\frac{\|\mathbf{p}-\mathbf{p}_k\|}{R_k} + \sqrt{1 - \mathbf{n} \cdot \mathbf{n}_k}} \tag{4.2}$$

where $\mathbf{p}_k$, $\mathbf{n}_k$ and $R_k$ are respectively the location of record $k$, its normal and the harmonic mean distance to the objects visible from $\mathbf{p}_k$. The user-defined value $a$ represents the accuracy of the computation. This value is used to threshold the contribution area of the records: a record $k$ contributes to the estimate of the outgoing radiance at point $\mathbf{p}$ if and only if

$$w_k(\mathbf{p}) \geq \frac{1}{a} \tag{4.3}$$

Substituting Equation 4.2 into Equation 4.3 and assuming $\mathbf{n} = \mathbf{n}_k$, one can see that record $k$ can contribute to the estimate of the outgoing radiance at point $\mathbf{p}$ only if:

$$\|\mathbf{p} - \mathbf{p}_k\| \leq aR_k \tag{4.4}$$

Therefore, Equation 4.4 guarantees that a record $k$ cannot contribute to the outgoing radiance of a point outside a sphere $I_k$ centered at $\mathbf{p}_k$, with radius $r_k = aR_k$.

**Algorithm 3** Radiance cache splatting

```
Let $k = \{\mathbf{p}_k, \mathbf{n}_k, E_k, R_k\}$ be the considered record
Let $I_k$ be the sphere centered at $\mathbf{p}_k$ with radius $aR_k$
Determine the bounding box of $I_k$ on the image plane
for all pixel $P(x, y) = \{\mathbf{p}, \mathbf{n}, \rho_d\}$ in the bounding box do
  // Evaluate weighting function at $\mathbf{p}$
  $w = \frac{1}{\frac{\|\mathbf{p}-\mathbf{p}_k\|}{R_k} + \sqrt{(1-\mathbf{n}\cdot\mathbf{n}_k)}}$
  if $w \geq \frac{1}{a}$ then
    //Compute the contribution of record $k$ at point $\mathbf{p}$
    $E'_k = E_k(1 + \mathbf{n}_k \times \mathbf{n} \cdot \nabla_r + (\mathbf{p} - \mathbf{p}_k) \cdot \nabla_t)$
    // Compute the outgoing radiance
    $L_o = \rho_d E'_k$
    // Accumulate into the radiance splat buffer
    $SPLATBUFF(x, y).L_o + = wL_o$
    $SPLATBUFF(x, y).w + = w$
  end if
end for
```

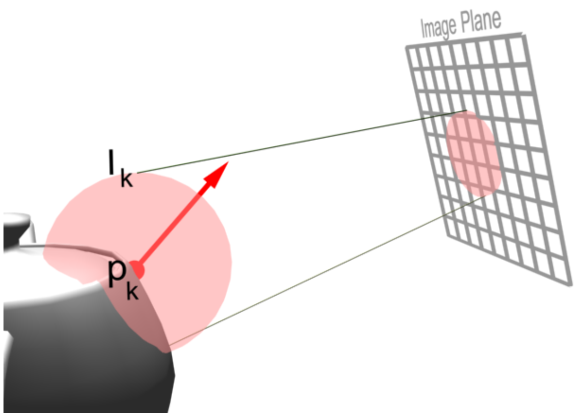


Figure 4.1: The sphere $I_k$ around the position $\mathbf{p}_k$ of the record $k$ is splatted onto the image plane. For each point within the sphere splat, the contribution of record $k$ is accumulated into the radiance splat buffer.

Given a camera, the *radiance cache splatting* splats the sphere $I_k$ onto the image plane (Figure 4.1). The weighting function (Equation 4.2) is evaluated for each point visible through pixels covered by $I_k$. Then, the weight is checked against the accuracy value (Equation 4.3). For each pixel passing this test, our algorithm computes the contribution of record $k$ to the outgoing radiance estimate.

The outgoing radiance contribution $L_o$ at a point $\mathbf{p}$ as seen through a pixel is obtained by evaluating the rendering equation:

$$L_o(\mathbf{p}, \omega_o) = \int_H L_i(\mathbf{p}, \omega_i) f(\omega_o, \omega_i) \cos(\theta_i) \mathrm{d}\omega_i \tag{4.5}$$

where $\omega_i$ and $\omega_o$ are respectively the incoming and outgoing directions. $L_i(\mathbf{p}, \omega_i)$ is the radiance incoming at $\mathbf{p}$ from direction $\omega_i$. $f(\omega_o, \omega_i)$ is the surface BRDF evaluated for the directions $\omega_i$ and $\omega_o$. In the case of irradiance caching, we only consider diffuse interreflections. Therefore, Equation 4.5 simplifies to:

$$L_o(\mathbf{p}) = \rho_d \int_H L_i(\mathbf{p}, \omega_i) \cos(\theta_i) \mathrm{d}\omega_i = \rho_d E(\mathbf{p}) \tag{4.6}$$

where $\rho_d$ is the diffuse surface reflectance, and $E(\mathbf{p})$ is the irradiance at point $\mathbf{p}$. Therefore, the contribution of record $k$ to the outgoing radiance at point $\mathbf{p}$ is

$$L_o = \rho_d E'_k(\mathbf{p}) \tag{4.7}$$

where $E'_k(\mathbf{p})$ is the irradiance estimate of record $k$ at point $\mathbf{p}$, obtained through the use of irradiance gradients [WH92, KGBP05].

Note that our splatting approach can be used with any weighting function containing a distance criterion. In this work we focused on the weighting function defined in [WRC88], although the function proposed in [TL04] could be employed as well.

**Extension to Radiance Caching** The BRDFs of glossy surfaces are view-dependent. Therefore, Equation 4.6 cannot be used in this case. As described in [KGPB05], both the incoming radiance contribution and the cosine-weighted BRDF are represented using hemispherical harmonics. The contribution of a record to the outgoing radiance at a point is the dot product of the vector of projection coefficients of the incoming radiance and that of the BRDF (see Chapter 3 for details on the dot product and BRDF storage using hemispherical harmonics).

Using the radiance cache splatting, the contribution of a record to the outgoing radiance estimate at points visible from the current viewpoint is computed independently of other records. The outgoing radiance contribution of each cache record is accumulated into the radiance splat buffer. While avoiding the need of recursive data structures and algorithms, this method also simplifies the detection of points needing the calculation of new records (cache miss detection).

## 4.3 Cache Miss Detection

The rendering process of irradiance caching requires the generation of a number of irradiance records to obtain a reasonable estimate of the indirect lighting for each point visible from the camera. Therefore, even though the values of the records are view-independent, their locations are determined according to the current viewpoint.

Usually, the irradiance cache records are stored in an octree for fast lookups. For a given pixel of the image, a ray is traced in the scene to find the point $\mathbf{p}$ visible through this pixel. At this point, the octree is queried to determine whether a new record is required (Equation 2.9). If yes, the actual indirect lighting is computed at **p**: a record is created and stored in the cache. Otherwise the cache is used to compute a satisfying estimate if the indirect lighting using interpolation. Since each visible point is processed only once, a newly created record cannot contribute to the indirect lighting of points for which the indirect lighting has already been computed or estimated. Therefore, unless an appropriate image traversal algorithm is employed, artifacts may be visible (Figure 4.2(a)). The traversal typically relies on hierarchical subdivision of the image to reduce and spread the errors. Artifact-free images can only be obtained in two passes: in the first pass, the records required for the current viewpoint are generated. In the second pass, the scene is rendered using the contents of the cache.

On the contrary, the radiance cache splatting algorithm is based on the independent splatting of the records. The contribution of a record is splatted onto each visible point within its zone of influence, even though the points have been previously checked for cache miss (Figure 4.2(b)). Therefore, our method avoids the need of a particular image traversal algorithm without harming the rendering quality. For the sake of efficiency and memory coherence, we chose to perform the cache miss detection using a linear traversal of the image. However, any other image traversal algorithm could be used.

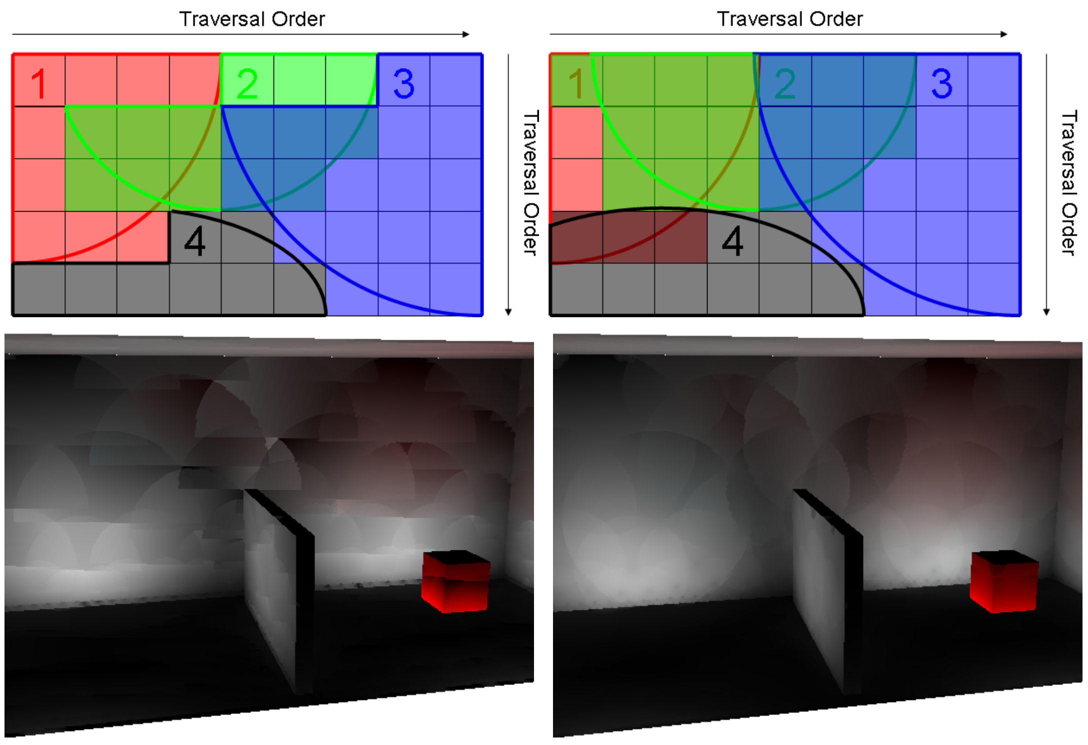
Once the algorithm has determined where a new record has to be generated, the irradiance value and gradients must be computed by sampling the hemisphere above the new record position. In the following section, we propose a simple and accurate method for hemispherical sampling using graphics hardware.

## 4.4 Record Computation

In our method, the incoming irradiance associated with a record is generated using both the GPU and CPU. First, the GPU performs the actual hemisphere sampling. Then, the CPU computes the corresponding irradiance or incoming radiance coefficients.

### 4.4.1 Hemisphere Sampling on the GPU

In the context of global illumination, the visibility tests are generally the bottleneck of the algorithms. We propose to leverage the power of graphics hardware to speed up the computation. In the classical hemi-cube method [CG85], 5 rendering passes are necessary for full hemisphere sampling. For the sake of efficiency, we preferred a 1-pass method similar to [SP89, LC04]: using a virtual camera placed at the record location, the rasterization engine samples the hemisphere above the record to compute the radiances incoming from each direction. However, as shown in Figure 4.3, using a single rasterization plane cannot account for the radiances incoming from grazing angles. To compensate for the incomplete hemisphere coverage, Larsen *et al.* [LC04] divide the obtained irradiance by the plane coverage ratio. We propose a more accurate method, in which border pixels are virtually stretched to fill the remaining solid angle (Figure 4.3). In our test scenes, this method yields more accurate results than the approach of Larsen *et al.* (Table 4.1). Furthermore, a key aspect of this method is



(a) Octree-based 1 pass (b) Octree-based 2-pass/Splatting 1-pass

Figure 4.2: Upper row: the irradiance records 1, 2, 3, 4 and their respective zones of influence. We consider the use of a sequential traversal of the image for determining the pixels for which a new record is needed. In the classical, octree-based method, a newly created record cannot contribute to pixels which have already been examined. Therefore, discontinuities appear in the contribution zone of the records, yielding artifacts. Those artifacts are illustrated in the upper row: record 1 is created at the point visible through the first pixel, and used for estimating the indirect lighting in the next 3 pixels. When record 2 is created at the point visible through the fifth pixel, its contribution is not propagated to the previous pixels, hence creating a discontinuity. A second rendering pass is needed to generate an image without such discontinuities. Using our radiance cache splatting method, each record contributes to all possible pixels, avoiding the need of a second rendering pass. Lower row: the indirect lighting obtained using these methods.

that the directional information of the incoming radiance remains plausible. Therefore, unlike in [LC04], the indirect glossy lighting can also be rendered correctly.

The GPU-based hemisphere sampling requires a proper shading of the visible points, accounting for direct lighting and shadowing. This computation is also carried out by the graphics hardware using per-pixel shading and shadow mapping. We use uniform shadow maps [Wil78]: unlike perspective shadow maps [SD02] or shadow volumes

| | Plane sampling | [LC04] | Our method |
|---|---|---|---|
| RMS Error | 18.1% | 10.4% | 5.8% |

Table 4.1: RMS error of 10000 irradiance values computed in the Sibenik Cathedral scene (Figure 4.8). Our method yields more accurate results than previous approaches, while preserving the directional information of the incoming radiance. The reference values are computed by full hemisphere sampling using Monte Carlo integration.

[Hei91], uniform shadow maps are view-independent: in static scenes the shadow map is computed once and reused for rendering the scene from each viewpoint. Therefore, the algorithm reuses the same shadow map to compute all the records, hence reducing the shading cost. Furthermore, the incoming radiances undergo a summation to compute either the irradiance or the projection into hemispherical harmonics. In both cases, this summation is equivalent to a low-pass filter which blurs out the aliasing artifacts of the shadow maps, hence providing inexpensive, high quality irradiance values and radiance coefficients.

Once the hemisphere has been sampled, the algorithm computes the actual (ir)radiance record.

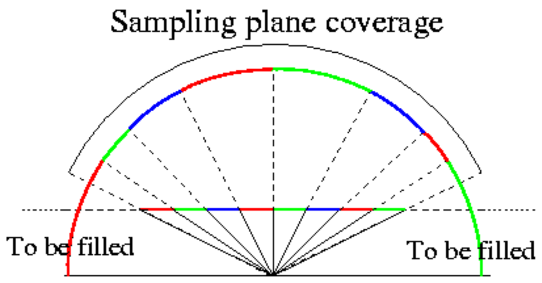


Figure 4.3: The hemisphere sampling is reduced to the rasterization of the scene geometry onto a single sampling plane. Since this plane does not cover the whole hemisphere, we use a border compensation method to account for the missing directions. Border pixels are virtually stretched to avoid zero lighting coming from grazing angles, yielding more plausible results.

## 4.4.2 Irradiance and Radiance Computation

As shown in [LC04] the irradiance is defined as a weighted sum of the pixels of the sampling plane. This sum can be calculated either using the GPU and automatic mipmap generation [LC04], or using frame buffer readback and CPU-based summation (Figure 4.4). In the case of irradiance caching, a record contains the irradiance, the rotational and translational gradients, and the harmonic mean distance to the objects visible from the record location. In our implementation, such values are stored within 22 floating-point values. Hence the method proposed by Larsen *et al.* would require the

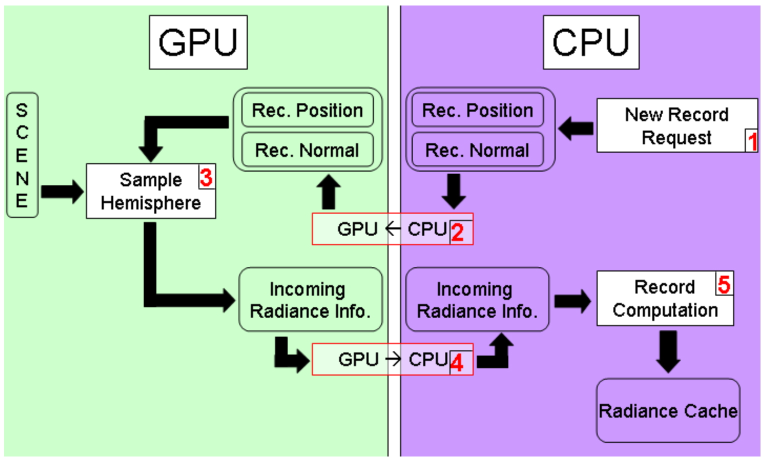


Figure 4.4: New record computation process. The numbers show the order in which the tasks are processed.

computation and storage of the mipmap levels of at least 6 RGBA textures. Even with the latest graphics hardware, this solution turns out to be slower than a readback and CPU-based summation. Furthermore, using a PCI-Express-based graphics card and the asynchronous readbacks provided by OpenGL Pixel Buffer Objects, the pipeline stalls introduced by the readback tend to be very small. Those stalls could be reduced in the future by considering multithreading and scheduling for enhanced performance.

The same approach is used to compute the incoming radiance function for radiance cache records. Instead of computing the irradiance, we project the incoming radiances for each pixel into the hemispherical harmonics basis using the CPU. In this case, using the CPU for the computation of the projection coefficients is definitely faster than using the GPU: each coefficient would require a separate mipmap generation pass. For a transform with order 10, 900 floating point values are required to store projection coefficients for the incoming radiance and gradients. In this case, mipmaps levels of typically 300 textures would have to be computed and stored, introducing a prohibitive overhead.

Once the records required to render global illumination from a given viewpoint have been computed, our method renders the final global illumination solution.

## 4.5 Global Illumination Rendering

The final image is generated in five main steps (Algorithm 4). Given a camera, the first step consists in obtaining per-pixel information about viewed objects: their position, local coordinate frame and BRDF (Figure 4.5).

In the second and third steps, the rendering process determines where new irradiance records are necessary to achieve the user-defined accuracy of indirect illumination computation. In Step 2, each existing record (possibly computed for previous frames) is splatted onto the splat buffer using the procedure described in 4.2. Step 3 consists

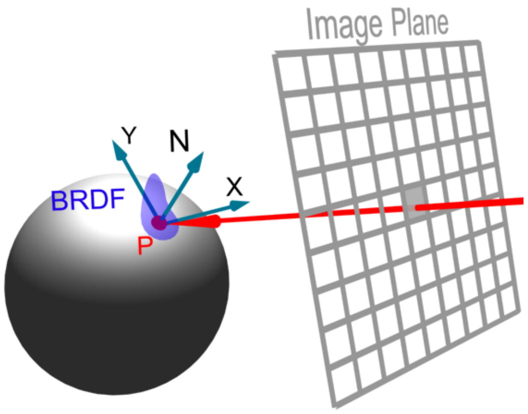


Figure 4.5: The radiance cache splatting requires per-pixel information about geometry and materials: the hit point, local coordinate frame, and BRDF.

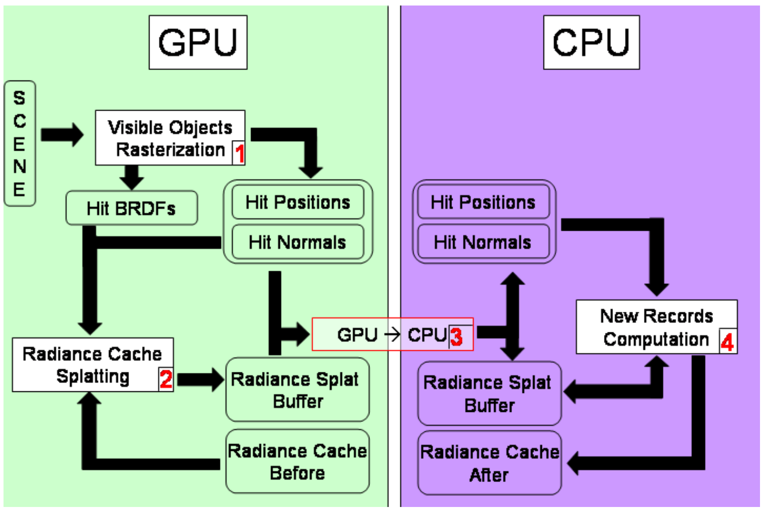


Figure 4.6: The irradiance cache filling process. The numbers show the steps defined in Algorithm 4. During this process, the irradiance cache stored on the CPU is updated whereas the copy on the GPU remains untouched.

**Algorithm 4** Global illumination rendering

```
// Step 1
Generate geometric and reflectance data of objects viewed through pixels (GPU)
Clear the splat buffer
// Step 2
for all cache records do
  // The radiance cache is empty for the first image,
  // and non empty for subsequent images
  Algorithm 3: splat the records onto the radiance splat buffer (GPU)
end for
// Step 3
Read back the radiance splat buffer from GPU to CPU
// Step 4
for all pixels $(x, y)$ in the radiance splat buffer do
  if $SPLATBUFF(x, y).w < a$ then
    Compute a new incoming radiance record at corresponding hit point
    (GPU/CPU)
    Apply Algorithm 3: splat the new record (CPU)
  end if
end for
// Step 5
for all cache records do
  Apply Algorithm 3: splat all the newly generated records (GPU)
end for
//Normalize the radiance splat buffer (GPU)
for all pixels $(x, y)$ in the radiance splat buffer do
  $SPLATBUFF(x, y).L_o / = SPLATBUFF(x, y).w$
end for
Combine the radiance splat buffer with direct lighting (GPU)
```

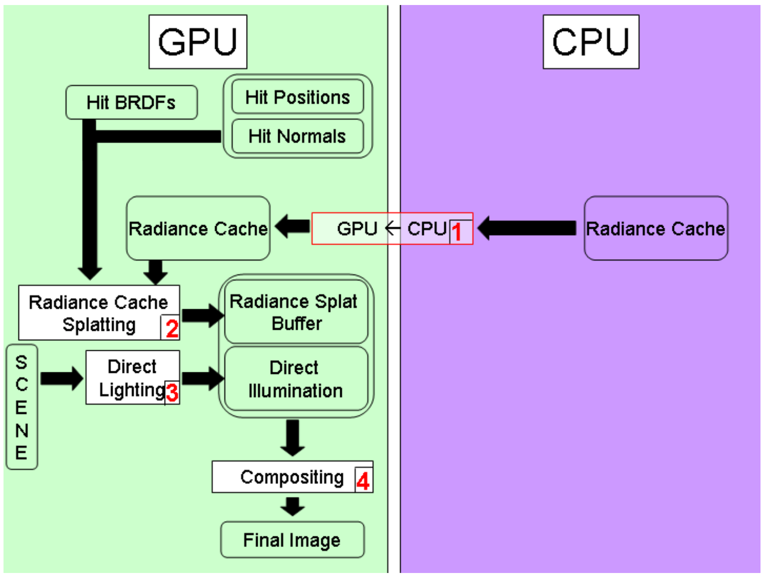


Figure 4.7: The final rendering task. The numbers show the processing order described in steps 4 and 5 of Algorithm 4.

in reading back the radiance splat buffer into the CPU memory. In Step 4, the algorithm traverses the radiance splat buffer to determine where new irradiance records are required to achieve the user-defined accuracy (Cache miss detection). For each pixel $(x, y)$ in the radiance splat buffer, the cumulated weight is checked against the accuracy value $a$:

$$SPLATBUFF(x, y).w \geq a \tag{4.8}$$

If a pixel $(x, y)$ fails this test, the existing cache records are insufficient to achieve the required accuracy. Therefore, a new record is generated at the location visible from pixel $(x, y)$, and is splatted by the CPU onto the radiance splat buffer (Figure 4.6).

Once $SPLATBUFF(x, y).w \geq a$ for every pixel, the data stored in the cache can be used to display the indirect illumination according to the accuracy constraint. At that time in the algorithm, the irradiance cache stored on the CPU memory differs from the cache stored on the GPU: the copy on the GPU represents the cache before the addition of the records described above, while the copy on the CPU is up-to-date.

The last rendering step is the generation of the final image using the cache contents (Figure 4.7). The irradiance cache on the GPU is updated, then the radiance cache splatting algorithm is applied on each newly generated cache record. Hence the radiance splat buffer contains the cumulated record weight and outgoing radiance contribution of all the irradiance records. Then, the cumulated contribution at each pixel is divided by cumulated weight. This process yields an image of the indirect lighting in the scene from the current point of view. Combined with direct lighting, this fast method generates a high quality global illumination solution.

As described above, our algorithm no longer relies on complex data structure and algorithm for the storage and retrieving of the records. The nearest-neighbors queries are replaced by simple sphere splatting and weighting function evaluation. These properties make the radiance cache splatting well-suited for GPU implementation.

## 4.6 GPU Implementation

Our algorithm has been implemented using OpenGL and the OpenGL Shading Language (GLSL). This section gives technical information about the implementation of radiance cache splatting. The rendering algorithm is based on deferred shading: the information on visible points is generated in a first pass by rasterizing the scene geometry. This information is used as the input of the radiance cache splatting. Using this method, the most costly operations are only performed once for each pixel of the final image.

As shown in Figures 4.6 and 4.7, our rendering algorithm makes an intensive use of the GPU computational power. In this implementation, we rely on the basic capabilities of graphics hardware: fast geometric primitive rasterization along with programmable vertex and fragment processors. The data required by our algorithm (Figure 4.5) is generated by rasterizing the scene geometry on the GPU using a dedicated shader. Using multiple render targets, this shader renders all the necessary information in a single rendering pass. Figure 4.5 shows that this step includes the storage of per-pixel BRDF. In the case of diffuse BRDFs, this information consists of the diffuse reflectance of the material (i.e. one RGB value). In the case of glossy BRDFs, the per-pixel information is only one identifier which will be used to fetch the corresponding BRDF from a 3D texture (see Chapter 3 for details on BRDF representation and storage on the GPU). During splatting, the material identifier is used to fetch the corresponding BRDF in the texture memory.

The radiance cache splatting can be performed either using the GPU or the CPU. Our implementation uses both, depending on the current context as explained hereafter. The radiance cache splatting on the GPU is performed by drawing a quadrilateral tightly bounding the splatted sphere on the image plane. The position and size of the quadrilateral are computed using a vertex shader. Then, each fragment is processed so that its value represents the weighted contribution of the record. The fragment is accumulated in the radiance splat buffer using the floating-point blending capabilities provided by graphics hardware. The final normalization (i.e. the division by the cumulated weight) is performed using a fragment shader in an additional pass for the final display. The GPU implementation is either used to splat the whole cache before adding new records, or to display the final image.

The cache miss detection is performed by traversing the radiance splat buffer sequentially using the CPU. Consequently, the GPU-based splatting cannot be used efficiently during the record addition step: this would require to read back the radiance splat buffer from the GPU memory after the computation and splatting of each new record. The CPU implementation is designed the same way as for the GPU. Figure 4.5 shows that the radiance cache splatting requires information about the visible objects. This information is computed on the GPU, then read back to the main memory once per frame. The overhead introduced by the data transfer from GPU to CPU turns out to be very small using PCI-Express hardware.

Once all necessary records are computed and splatted onto the radiance splat buffer, the final picture containing both direct and indirect lighting has to be generated. In our implementation, the direct lighting computation is carried out by the GPU: a fragment shader evaluates the BRDFs per-pixel, while shadow maps [Wil78, BP04] are used to simulate shadowing effects. To reduce the aliasing of shadow maps without harming the performance, we use high definition shadow maps (typically $1024 \times 1024$) with percentage-closer filtering [RSC87] using 16 samples. Hence the same shadow map can be used for both the record computation and the rendering of direct lighting in the final image. Note that higher quality images could be obtained using view-dependent shadowing methods such as shadow volumes for the visualization of the direct lighting in the final image.

The normalization of the radiance splat buffer and the combination with direct lighting are finally performed in a single fragment shader which displays the final image.

## 4.7 Results

This section discusses the results obtained using our implementation of radiance cache splatting. The images and timings have been generated using an nVidia Quadro FX 3400 PCI-E and a 3.6 GHz Pentium 4 CPU with 1 GB RAM.

### 4.7.1 High Quality Rendering

In this section, our aim is non-interactive high quality global illumination computation. First, we compare the results obtained with our GPU-based renderer with the well-known Radiance software [War94] in the context of irradiance caching. Second, we discuss the results of radiance caching in glossy environments using our renderer.

We have compared the rendering speed of the Radiance software and our renderer in diffuse environments: the *Sibenik Cathedral* and the *Sponza Atrium* (Figure 4.9). The images are rendered at a resolution of $1000 \times 1000$ and use a $64 \times 64$ resolution for hemisphere rasterization. The results are discussed hereafter, and summarized in Table 4.2.

*a)* *Sibenik Cathedral* This scene contains 80K triangles, and is lit by two light sources. The image is rendered with an accuracy parameter of 0.15. At the end of the rendering process, the irradiance cache contains 4076 irradiance records. The radiance cache splatting on the GPU is performed in 188 ms. The Radiance software rendered this scene in 7 min 5 s while our renderer took 14.3 s, yielding a speedup of about 30×.

*b)* *Sponza Atrium* This scene contains 66K triangles and two light sources. Using an accuracy of 0.1, this image is generated in 13.71 s using 4123 irradiance records. These records are splatted on the GPU in 242.5 ms. Using the Radiance software with the same parameters, a comparable image is obtained in 10 min 45 s. In this scene, our renderer proves about 47× faster than the Radiance software.

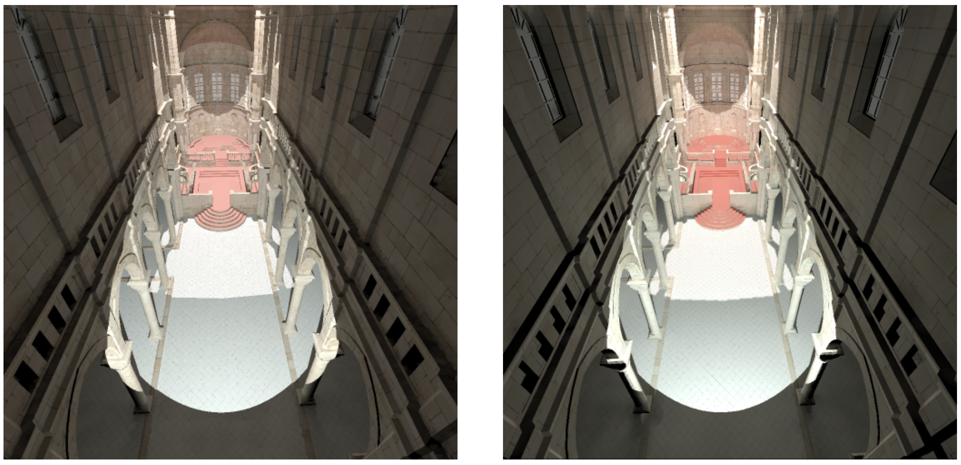

(a) Radiance (b) Our renderer

Figure 4.8: The Sibenik Cathedral scene (80K triangles). The images show first bounce global illumination computed with Radiance (a) and our renderer (b) (Model courtesy of Marko Dabrovic)

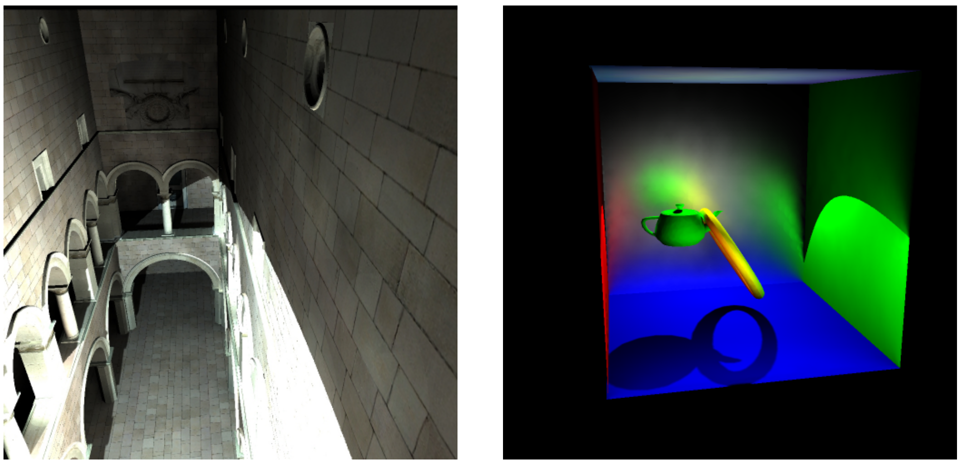

(a) Sponza Atrium (b) Cornell Box

Figure 4.9: Images obtained with our renderer. The Sponza Atrium (66K triangles) is only made of diffuse surfaces (Model courtesy of Marko Dabrovic). The Cornell Box (1K triangles) contains a glossy back wall.

| | Sibenik Cathedral | Sponza Atrium |
|---|---|---|
| Triangles | 80K | 66K |
| Accuracy | 0.15 | 0.1 |
| Radiance time (s) | 425 | 645 |
| Our renderer time (s) | 14.3 | 13.7 |
| Speedup | 29.7 | 47.1 |

Table 4.2: Rendering times obtained using Radiance and our renderer for high quality rendering of diffuse environments. Each image is rendered at resolution $1000 \times 1000$.

Our renderer also contains an implementation of radiance caching for the computation of global illumination in glossy environments. If the environment contains both diffuse and glossy surfaces, our renderer uses either radiance or irradiance caching depending on the materials.

The *Cornell Box* scene presented in Figure 4.9(b) contains a glossy back wall (Phong BRDF [Pho75], exponent 20), while the other objects are diffuse. The glossy BRDF and incoming radiance function are projected into the hemispherical harmonics basis using order 10 representation. The accuracy parameters are 0.25 for both radiance and irradiance caching. Figure 4.9(b) was rendered in 12.18 s using 3023 irradiance records and 869 radiance records. The GPU-based splatting for the irradiance cache is performed in 65.91 ms. The radiance cache is splatted in 935.5 ms. The higher rendering time of radiance caching is due to the number of texture lookups in the computation of the dot product of the BRDF and the incoming radiance. With order 10 representation (100 projection coefficients for each color channel), the splatting algorithms performs approximately 400 texture lookups for each fragment: 100 for the incoming radiance, 200 for the radiance gradients, and 100 for the BRDF.

Figure 4.10 shows an example of radiance cache splatting in a more complex glossy environment: the *Castle* scene contains about 57K triangles. In this scene, the glossiness of the roofs is obtained using a Phong BRDF [Pho75] with exponent 15. The BRDF and incoming radiance functions are represented by order 5 projection into the hemispherical harmonics basis. The accuracy parameter we used is 0.25 for irradiance caching and 0.2 for radiance caching. At the end of the rendering process, the irradiance and radiance cache respectively contain 3204 and 1233 records, computed in 10.1 s. The splatting on the GPU is performed in 58.6 ms for the irradiance cache. The radiance cache records are splatted in 493.7 ms.

The results presented above show that our algorithm is able to render fast high-quality glossy global illumination. However, the simplicity of our algorithm also allows for progressive rendering in interactive applications.

### 4.7.2 Interactive Global Illumination

An important aspect of the irradiance caching algorithm is that the values of the records are view-independent. In a static scene, records computed for a given viewpoint can be reused for other camera positions. Therefore, the radiance cache splatting approach

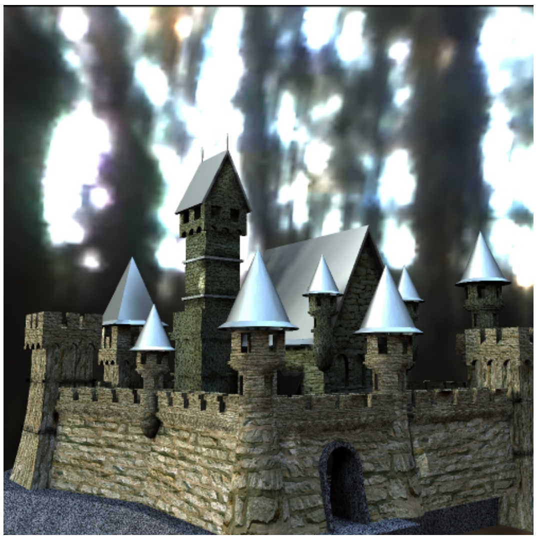


Figure 4.10: The Castle scene (58K triangles) illuminated by an environment map. Our renderer computes first-bounce glossy global illumination in 10.1 s at resolution $1000 \times 1000$.

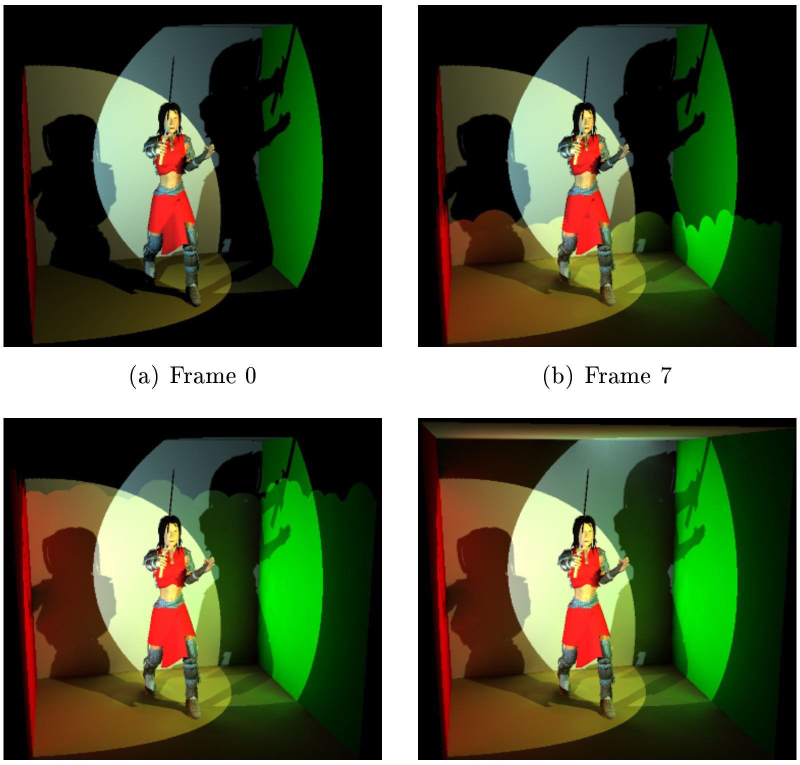
can also be used in the context of interactive computation of global illumination using progressive rendering. The direct lighting being computed independently, the user can walk through the environment while the irradiance and radiance caches are filled on the fly. Figure 4.11 shows sequential images of *Sam* scene (63K triangles) obtained during an interactive session with an accuracy parameter of 0.5 and resolution $512 \times 512$. The global illumination is computed progressively, by adding at most 100 new records per frame. Our renderer provides an interactive frame rate (between 5 and 32 fps) during this session, allowing the user to move even though the global illumination computation is not completed. This method has been used for interactive walkthroughs in diffuse and glossy environments such as the *Sibenik Cathedral* and the *Castle*.

In this context, the radiance cache splatting also proves useful for adjusting the value of the user-defined accuracy parameter $a$. In the classical irradiance caching method, the structure of the octree used to store the records is dependent on $a$. Therefore, the octree has to be regenerated for each new value of $a$. Using our approach the size of each splat is computed for each frame on the GPU, hence allowing the user to tune the value of $a$ interactively to obtain the desired visual quality (Figure 4.12).

### 4.7.3 Speedup Analysis

Previous subsections show that our method achieves a significant speedup compared to the Radiance software by using the GPU for both record computation and final rendering.

In the irradiance and radiance caching algorithms, the most expensive rendering task consists in sampling the hemisphere for computing the value of the records. We



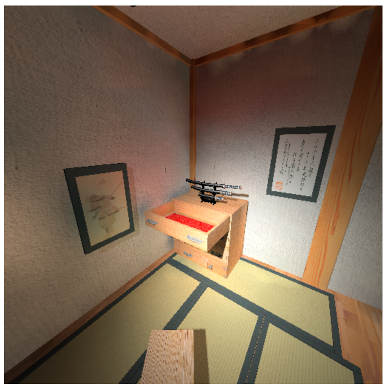
(a) Frame 0 (b) Frame 7

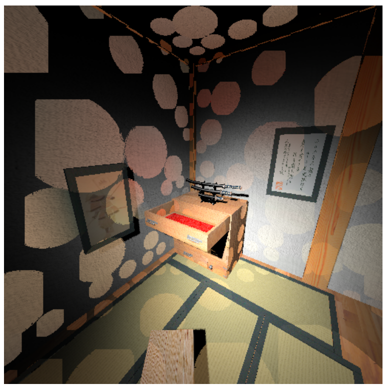
(c) Frame 11 (d) Frame 14

Figure 4.11: A progressive rendering session for interactive visualization of the *Sam* scene (63K triangles). Our renderer computes at most 100 new records per frame, hence maintaining an interactive frame rate (5 fps) during the global illumination computation. When the (ir)radiance cache is full, the global illumination solution is displayed at 32 fps.

propose to reduce this cost by using a sampling plane along with an accurate, novel compensation method. The incoming radiance values are computed accurately using fast GPU rasterization and uniform shadow maps. The possible aliasing of shadow maps is made unnoticeable due to the summation and/or projection of the radiances into a low-frequency basis.

Once all the needed records are computed, the final rendering in Radiance relies on ray tracing and nearest-neighbors queries to calculate the outgoing radiance for each pixel. Although the cost involved is not dominant compared to the records computation, the speedup due to the simplicity of radiance cache splatting is the bridge between offline and interactive rendering of global illumination.


(a) $a = 0.5$


(b) $a = 0.2$ using the records generated with $a = 0.5$

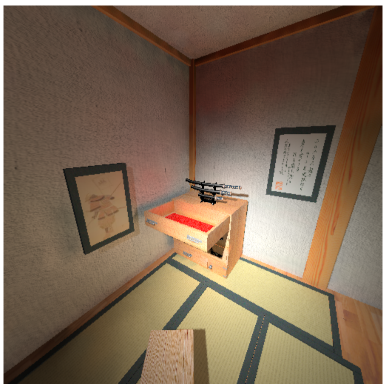

(c) $a = 0.2$

Figure 4.12: Our method allows for the dynamic modification of the user-defined accuracy value $a$, hence easing the parameter tuning process.

## 4.8 Conclusion

In this work, our aim was to use the computational power of the latest graphics hardware to speedup both the computation and the rendering of global illumination.

To this end, we reformulated the irradiance and radiance cache algorithms by introducing the radiance cache splatting. In this method, the sphere of influence of each record is splatted onto the image plane. For each pixel within the splatted sphere, a fragment shader computes the contribution and weight of the record to the indirect lighting of the corresponding point. The record weight and weighted contribution are accumulated into the radiance splat buffer using floating-point alpha blending. The final indirect lighting of visible points is obtained by dividing the weighted sum of the records by the cumulated weights in a fragment shader. Using this approach, each record contributes to all possible pixels, hence simplifying the cache miss detection algorithm. By avoiding the need of complex data structures and by extensively using the power of graphics hardware, the radiance cache splatting allows for displaying a global illumination solution in real-time.

Another speedup factor is the use of both GPU and CPU to compute the values of the irradiance and radiance records. While the GPU performs the costly hemisphere sampling, the CPU sums up the incoming radiances to compute the actual values and gradients of the records. This method requires many data transfers from the GPU to the CPU. Even though classical architectures such as AGP [Int02] would have introduced a prohibitive bottleneck [CHH02], the new PCI-Express 16$\times$ port [PCI03] provides transfer rates up to 8GB/s. Compared to AGP, the pipeline stalls are drastically reduced, making our method usable even in the case of on-the-fly computation of global illumination for interactive rendering.

We compared our implementation with the well-known Radiance software in moderately complex scenes. Our renderer shows a speedup of more than 29$\times$ compared to the Radiance software for the same rendering quality. We also demonstrated interactive performance for global illumination computation and visualization. To our knowledge, the radiance cache splatting is the first implementation of irradiance and radiance caching using programmable graphics hardware. We believe that our method could be integrated into film production renderers for fast and accurate computation of indirect illumination.

However, this method is not yet complete: an important improvement is the efficient handling of multiple light bounces, using recursive splatting or a second level cache accounting for further light bounces as described in Chapter 6. Also, the radiance caching method could be improved to account for higher frequency BRDFs. The improvement discussed in the next chapter is the extension of the irradiance and radiance caching interpolation schemes to the temporal domain for efficient global illumination computation in animated environments.

# Chapter 5

# Temporal Radiance Caching

## 5.1 Motivations

Generally, global illumination methods aim at simulating the light/matter interactions within static scenes. Accounting for the displacement of objects and light sources either require a complete recomputation of the global illumination solution for each frame of the animation, or involve complex data structures and algorithms for temporal optimization. Furthermore, the global illumination solutions commonly exhibit low temporal quality when used in dynamic scenes: flickering, popping, ... In the context of computer-assisted effects for movies, high quality global illumination is obtained through temporal filtering: a 30 fps animation is first rendered at 60 fps by recomputing the global illumination for each frame. Then, each frame of the 30 fps animation is generated by averaging two frames of the 60 fps animation, hence reducing the temporal artifacts at the cost of high computational cost. For interactive applications such as video games, the illumination must be computed interactively. In this case, approximate models are generally preferred, such as the precomputation of a static global illumination solution, and the update of direct lighting only at runtime.

We propose a simple and accurate method based on temporal caching for efficient computation of global illumination effects in animated environments [GBP06]. Our method provides rapid generation of image sequences for environments in which viewer, objects and light sources move. Our approach focuses on a temporal optimization for lighting computation based on the irradiance caching [WRC88] and radiance caching [KGPB05] techniques. As these algorithms leverage the spatial coherence of indirect lighting to reduce the cost of global illumination, we propose an extension of these methods for sparse temporal sampling and interpolation. In [WRC88], Ward *et al.* propose a reuse an irradiance value in the neighborhood of the actual computation point. While the weighting function and gradients of [WRC88] account for the spatial change of irradiance, our method considers the temporal change of the indirect lighting (Figure 5.2). In the spirit of the irradiance caching interpolation scheme, we define a temporal weighting function and temporal radiance gradients based on an estimate of the change of incoming radiance in the course of time. This estimate requires a basic knowledge of the incoming radiance at future time steps. We propose an inexpensive, GPU-based method based on reprojection to compute this information.

Unlike many previous approaches, our method does not introduce any new data structures and adds very little overhead to the existing memory requirements. Since a same record in the cache can be used in several frames, both the computational cost and the memory required to store the records are significantly reduced. Thus the records used to render an animation segment can be kept within a small memory space. Once the records are computed in the scene, our GPU-based renderer can display the dynamic globally illuminated scene in real-time.

This chapter is organized as follows: In Section 5.2, we highlight the problems related to using the irradiance and radiance caching algorithms for animation rendering, and present our main contributions: the temporal weighting function, the estimation of future incoming radiance by reprojection, and the temporal gradients. Section 5.3 contains implementation details for efficient computation using graphics hardware. Our results are presented in Section 5.4.

## 5.2 Temporal Irradiance Caching

### 5.2.1 Irradiance Caching in Dynamic Scenes

As described in Chapter 2, the irradiance caching algorithm leverages the spatial coherency of the indirect lighting, and thus reduces the computation time of global illumination. In dynamic scenes, a straightforward and commonly used method consists in computing the global illumination solution from scratch for every frame. Thus, the distributions of record location in frames $n$ and $n+1$ are likely to be different. Figure 5.1 illustrates the consequence of the change of record distribution: since the gradients extrapolate the change of incoming radiance, they are not completely accurate compared to the ground truth. Therefore, the accuracy of the lighting reconstructed by irradiance caching is not constant over a surface: the accuracy is maximum at the record location, and decreases as the extrapolation point gets away from the record. Changing the distribution of records also changes the distribution of the accuracy, yielding flickering artifacts. Thus, simple use of irradiance caching in dynamic scenes gives rise to poor animation quality. Additionally, as the record computation is performed from scratch for every frame, a significant amount of computational effort is wasted.

In this chapter, we propose a simple and unified framework for view-dependent global illumination in dynamic environments with predefined animation in which objects, light sources, and cameras can move. The irradiance and radiance caching algorithms being very similar, we only consider irradiance caching in the following discussions. If necessary, specific details on the application to radiance caching are given at the end of the sections.

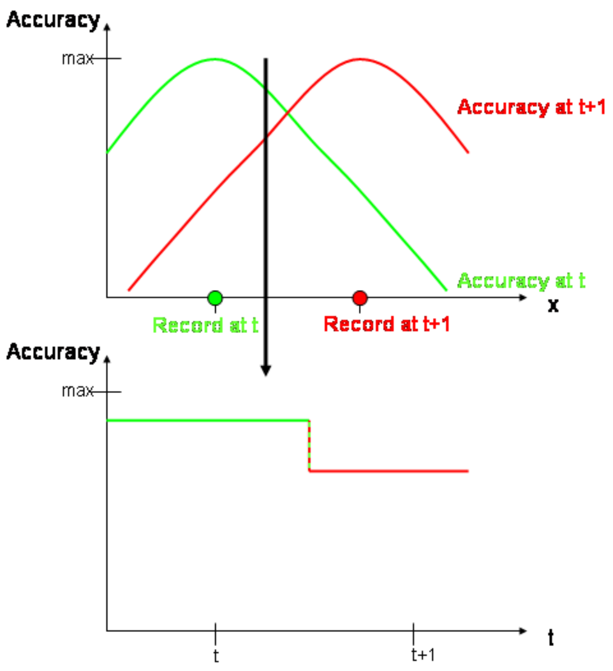
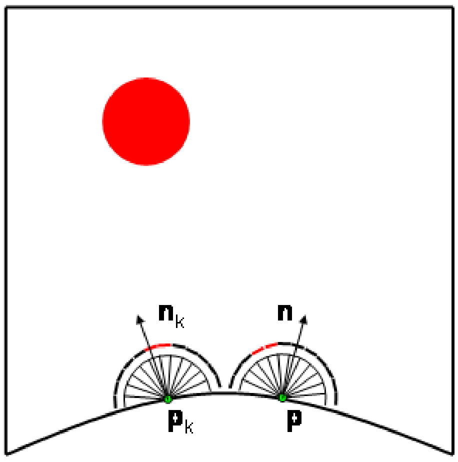


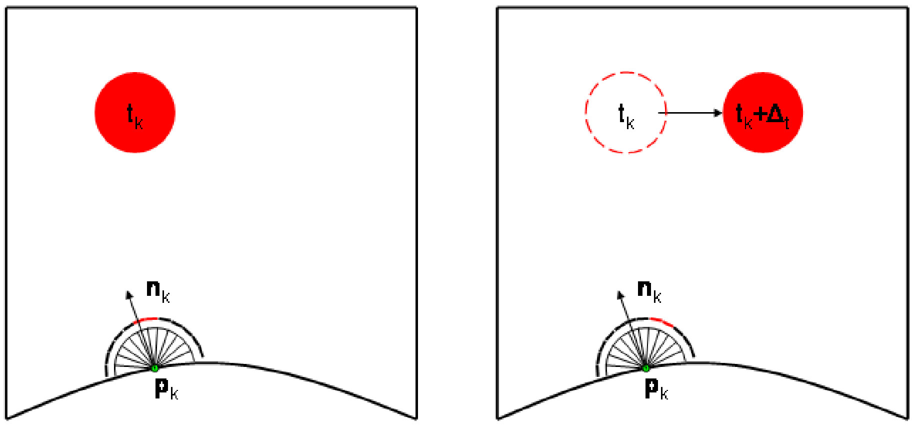
Figure 5.1: Changing the location of the records between successive frames can significantly modify the accuracy of the lighting at a given point (marked by the arrow). Therefore, the incoming radiance at this point can change noticeably between two frames, yielding flickering artifacts. Note that the maximum accuracy is obtained at the point and time of actual computation of the incoming radiance.

### 5.2.2 Overview of Temporal Radiance Caching

Following the spatial interpolation scheme of Ward *et al.* [WRC88, WH92], our aim is to reuse records across frames by performing a sparse temporal sampling and temporal interpolation of the irradiance (Algorithm 5). When a record $k$ is created at frame $n$, the future incoming lighting is estimated. This estimate is first used to compute the temporal change rate of the irradiance. In the spirit of [WRC88], we define our temporal weighting function $w_k^t$ as the inverse of the temporal change. Hence the number of frames in which $k$ can contribute is inversely proportional to the future change of incoming radiance. Since the actual computation of the future incoming lighting may be expensive, we use a simple reprojection technique for estimating the future lighting using the data sampled at the current frame. As the irradiance at a point is extrapolated using the irradiance and gradients of neighboring records, we propose temporal gradients for smooth temporal interpolation and extrapolation of the irradiance.



(a) Spatial change



(b) Temporal change

Figure 5.2: Change of incoming radiance with respect to space and time. The black-red zone above the hemisphere represents the incoming radiance function. Classical irradiance caching (a) estimate the change of incoming radiance with respect to displacement and normal divergence. In the case of temporal irradiance caching (b), the incoming radiance can change between two frames even though the record location remains constant.

### 5.2.3 Temporal Weighting Function

The temporal weighting function expresses the validity of a given record over time. Using a derivation similar to that of [WRC88], we define the temporal change $\epsilon^t$ of incoming radiance between time $t$ and $t_0$ as:

$$\epsilon^t = \frac{\partial E}{\partial t}(t_0) \quad (t - t_0) \tag{5.1}$$

**Algorithm 5** Temporal Radiance Caching

```
for all frames $n$ do
  for all existing records $k$ do
    if $w_k^t(n)$ is sufficient then
      Use $k$ in frame $n$
    end if
  end for
  for all points $\mathbf{p}$ where a new record is needed do
    Sample the hemisphere above $\mathbf{p}$
    Estimate the future incoming lighting (Section 5.2.4)
    Generate $w_k^t$ (Section 5.2.3)
    Compute the temporal gradients (Section 5.2.5)
    Store the record in the cache
  end for
end for
```

This derivative $\frac{\partial E}{\partial t}(t_0)$ can be approximated using estimates of incoming radiance at two successive times $t_0$ and $t_1$, denoted $E_0$ and $E_1$.

$$\frac{\partial E}{\partial t}(t_0) \approx \frac{E_1 - E_0}{t_1 - t_0} \tag{5.2}$$

$$= \frac{\tau E_0 - E_0}{t_1 - t_0} \quad \text{where } \tau = E_1/E_0 \tag{5.3}$$

$$= E_0 \frac{\tau - 1}{t_1 - t_0} \tag{5.4}$$

In our method, the time range of the animation is discretized into integer frame indices. Therefore, we always choose $t_1 - t_0 = 1$, i.e. $E_1$ and $E_0$ represent the estimated irradiance at two successive frames.

As in [WRC88], we define the temporal weighting function as the inverse of the change, excluding the term $E_0$:

$$w_k^t(t) = \frac{1}{(\tau - 1)(t - t_0)} \tag{5.5}$$

where $\tau = E_1/E_0$ is the *temporal irradiance change rate.*

This weighting function expresses the confidence of the incoming radiance value estimated at time $t_0$ in subsequent frames. This function can be evaluated and tested against a user-defined accuracy value $a^t$. A record $k$ created at $t_0$ is allowed to contribute to the image at $t$ if

$$w_k^t(t) \geq 1/a^t \tag{5.6}$$

The temporal weighting function is used to adjust the time segment during which a record is considered as valid. Since a given record can be reused in several frames, the computational cost can be significantly reduced. In the context of radiance caching, the directional information of the incoming radiance must be very accurate. Therefore, we define the temporal radiance change rate as:

$$\tau_{radiance} = max\{\lambda_1^i/\lambda_0^i, 0 \geq i < n\} \tag{5.7}$$

where $\lambda_0^i$ and $\lambda_1^i$ are respectively the $i^{\text{th}}$ projection coefficient of the current and future incoming lighting. By using the maximum rate of change, our method can account for directional change of incoming radiance without loss of accuracy.

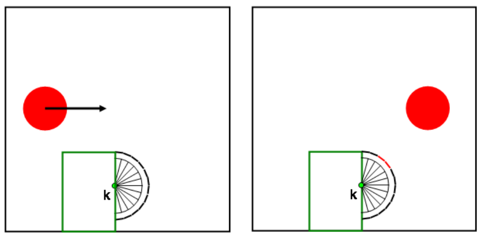
However, Equation 5.3 shows that if the environment remains static starting from frame $t_0$, we obtain $\tau = 1$. Therefore, Equation 5.6 shows that $w_k^t$ is infinite for any frame, and hence record $k$ is allowed to contribute at any time $t > t_0$. However, since part of the environment is dynamic, the inaccuracy becomes significant when $t - t_0$ gets high (see Figure 5.3). This is a limitation of our technique for estimating the temporal change of incoming radiance, which determines the lifespan of a record by only considering the change between $E_t$ and $E_{t+1}$. Therefore, we introduce a user-defined value $\delta t_{max}$ limiting the length of the validity time segment associated with each record. If Equation 5.8 does not hold, we decide that the record cannot be reasonably used.

$$t - t_0 < \delta_{t_{max}} \tag{5.8}$$

This reduces the risk of having records from being used while they are obsolete, hence controlling the artifacts due to residual global illumination effects also known as "ghosts", which commonly appear in interactive methods. However, as $a$ and $a^T$, this value must be user-defined by trial and error to obtain the best results. If $\delta_{t_{max}}$ is too low, many records may be recomputed unnecessarily. Setting $\delta_{t_{max}} = 1$ implies the recomputation of each record for each frame. In this case, the resulting performance would be similar to the classical per-frame computation. Nevertheless, this would completely avoid the ghosting artifacts, the indirect lighting being permanently computed from scratch. If $\delta_{t_{max}}$ is set too high, the same records might be reused in too many frames. Hence artifacts due to the residual global illumination effects are likely to appear in the vicinity of the moving objects, degrading the quality of the rendered frame. Hence such a high value significantly reduces the rendering time at the cost of accuracy.

However, the flickering problem described in Section 5.2.1 still appears when a record is suddenly discarded. As proposed in [TMS02], we avoid this problem by keeping track of the location of the records over time. Let us consider a record $k$ located at point $\mathbf{p_k}$. If $k$ was allowed to contribute to the previous frame and cannot be reused in current frame, a new record $l$ is created at the same location, i.e. $\mathbf{p_l} = \mathbf{p_k}$ (Figure 5.4(b)). Since the location of visible records remains constant in space, the distribution of accuracy has less temporal variations, which reduces flickering artifacts. Note that the location of records remain constant even though they lie on dynamic objects (Figure 5.5).

The temporal weighting function provides a simple and adaptive way of leveraging temporal coherence by introducing an aging method based on the change of incoming radiance. However, as shown in Equation 5.3, the determination of our temporal weighting function relies on the knowledge of the incoming radiance at the next time step, $E_{t_0+1}$. Since the explicit computation of $E_{t_0+1}$ would introduce a huge computational overhead, we propose a simple and accurate estimation method based on reprojection.



(a) Time $t_k$ (b) Time $t_k + n$

Figure 5.3: When record $k$ is created at time $t_k$, the surrounding environment is static ($\tau_k = 1$). However, the red sphere is visible from the $k$ $n$ frames later. The user-defined value $\delta_{t_{max}}$ prevents the record from contributing if $n > \delta_{t_{max}}$, reducing the risk of using obsolete records.

### 5.2.4 Estimating $E_{t_0+1}$

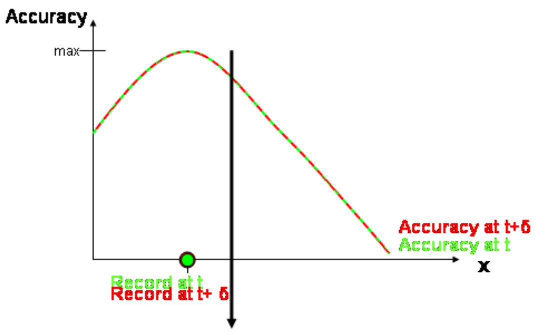
Our method follows the reprojection approach proposed in [WDP99, WDG02]. However, while Walter *et al.* use reprojection for interactive visualization using ray tracing, our aim is to provide a simple and reliable estimate of the incident radiance at a given point at time $t_0 + 1$ by using the data acquired at time $t_0$ only.

In the context of predefined animation, the changes in the scene are known and accessible at any time. When a record $k$ is created at time $t_0$, the hemisphere above $\mathbf{p_k}$ is sampled (Figure 5.6(a)) to compute the incoming radiance and gradients at this point. Since the changes between times $t_0$ and $t_0+1$ are known, it is possible to reproject the points visible at time $t_0$ to obtain an estimate of the visible points at time $t_0 + 1$ (Figure 5.6(b)). The outgoing radiance of reprojected visible points can be estimated by accounting for the rotation and displacement of both objects and light sources. In overlapping areas, a depth test accounts for the occlusion change (Figure 5.6(c)).

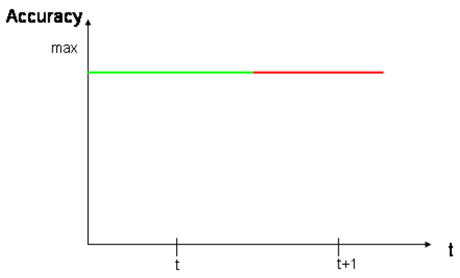
However, some parts of the estimated incoming radiance may be unknown (holes) due to displacement and overlapping of visible objects (Figure 5.6(d)). As proposed in [WDP99], we use a simple hole-filling method: each hole is filled using the background values, yielding a plausible estimate of the future indirect lighting.

Note that we use reprojection only to obtain an approximate of the incoming radiance at a given point at time $t_0 + 1$. As shown in Figure 5.7 and Table 5.1, the reprojection reduces the error in the estimate of the future incoming lighting. Those errors were measured by comparing the irradiances at time $t + 1$ with the irradiances estimated using records created at time $t$.

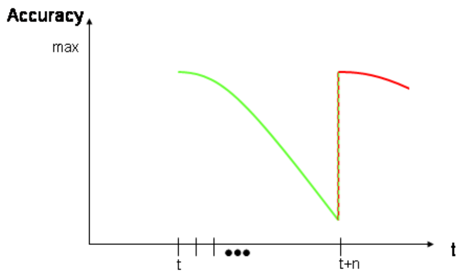
The temporal weighting function and record replacement scheme, along with an estimation of future incoming radiance, allow to improve both the computational efficiency and the animation quality. Nevertheless, Figure 5.4(c) shows that the accuracy of the computation is still not continuous in the course of time. Replacing obsolete records by new ones creates a discontinuity of accuracy, which causes the visible flickering artifacts. Therefore, we propose *temporal gradients* to generate a smooth and less noticeable transition between successive records.



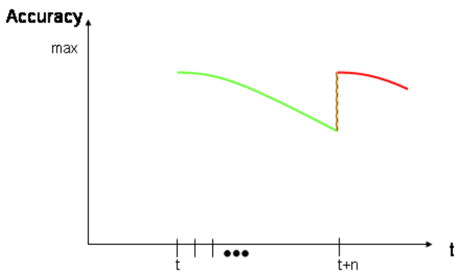
(a)



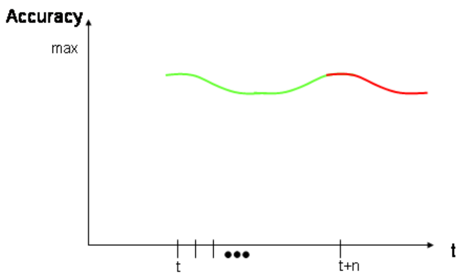
(b) New record after $\delta = 1$ frame



(c) New record after $\delta = n$ frames, no temporal gradient



(d) New record after $\delta = n$ frames, extrapolated temporal gradient

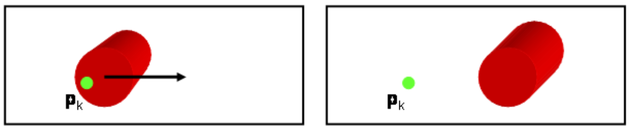


(e) New record after $\delta = n$ frames, interpolated temporal gradient

Figure 5.4: Empirical shape of the accuracy at a fixed time with respect to space (a), and at a fixed location with respect to time (b,c,d,e). If records are located at the same point between successive frames (a), the temporal accuracy is improved (b). However, flickering artifacts due to the temporal discontinuities of accuracy may appear when a record is reused in several frames, then recomputed (c). Extrapolated temporal gradients decrease the amplitude of the discontinuity in one pass, reducing flickering (d). Interpolated temporal gradients use two passes to eliminate the discontinuity by smoothing out the temporal changes (e).

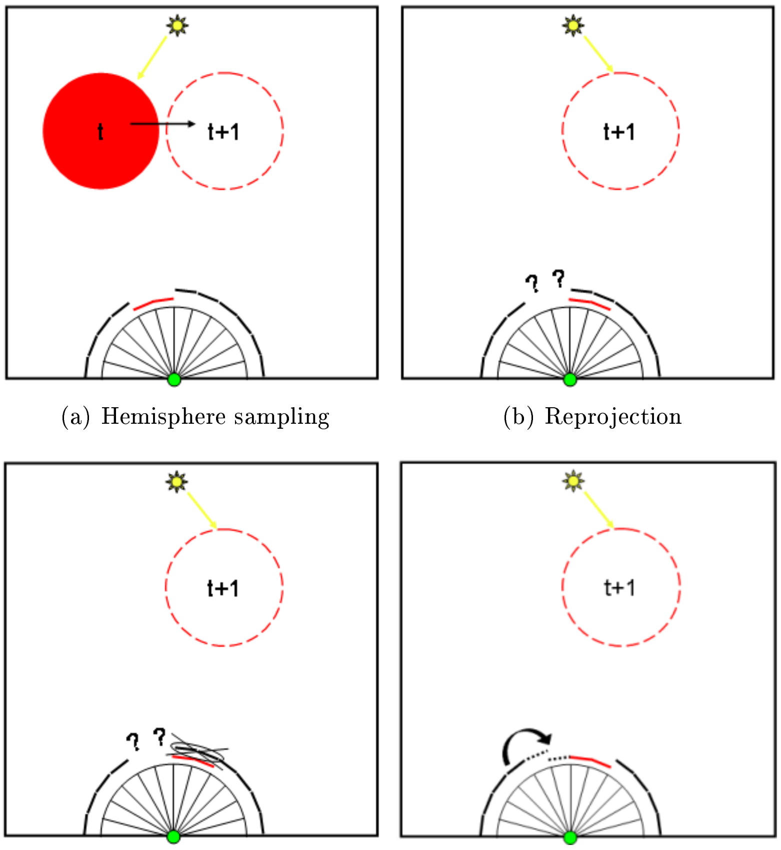
### 5.2.5 Temporal Gradients

Temporal gradients are conceptually equivalent to classical irradiance gradients. Instead of representing the incoming radiance change with respect to translation and rotation,



(a) Time $t_k$ (b) Time $t_k + n$

Figure 5.5: Record $k$ created at time $t_k$ remains at point $\mathbf{p_k}$ even though it lies on a dynamic object.



(a) Hemisphere sampling (b) Reprojection

(c) Depth test (d) Filtering

Figure 5.6: The hemisphere is sampled at time $t$ as in the classical irradiance caching process (a). For each ray, our method estimates where each visible point will be located at time $t + 1$ by reprojection (b). Distant overlapping points are removed using depth test (c), while resulting holes are filled using the farthest neighboring values (d).

those gradients represent how the incoming radiance gets altered over time.

In the context of irradiance caching, Equation 2.7 shows that the irradiance at point $\mathbf{p}$ is estimated using rotation and translation gradients. The temporal irradiance

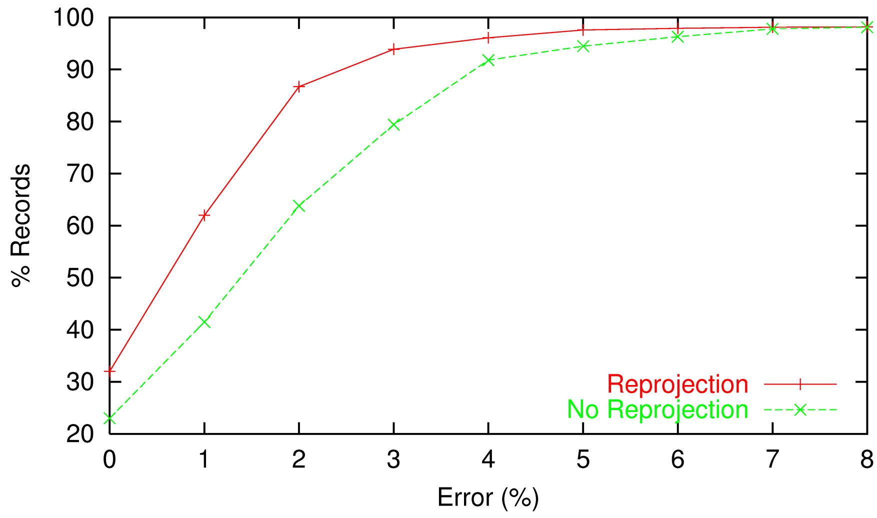


Figure 5.7: Percentage of records for which the estimate of the future incoming lighting is below a given RMS error level. The reprojection reduces the overall error in the estimate compared to a method without reprojection (i.e. the where the lighting is considered temporally constant). (Errors computed using 4523 values.)

| Error | Reprojection | No Reprojection |
|---|---|---|
| Min | 0% | 0% |
| Max | 30% | 32% |
| Mean | 2.9% | 3.7% |
| Median | 1.6% | 2.4% |

Table 5.1: RMS error of reprojection (Based on 4523 values).

gradient of record $k$ at a given point $\mathbf{p}$ with normal $\mathbf{n}$ is derived from Equation 2.7:

$$\nabla^{\mathbf{t}}(\mathbf{p}) = \frac{\partial}{\partial t}(E_k + (\mathbf{n_k} \times \mathbf{n}) \cdot \nabla_{\mathbf{r}} + (\mathbf{p} - \mathbf{p_k}) \cdot \nabla_{\mathbf{p}}) \tag{5.9}$$

where:

- $\nabla_{\mathbf{r}}$ and $\nabla_{\mathbf{p}}$ are the rotation and translation gradients
- $\mathbf{p_k}$ and $\mathbf{n_k}$ are the location and normal of record $k$

As described in section 5.2.3, we keep the location of all records constant over time. We choose any point of interest $\mathbf{p}$ with normal $\mathbf{n}$ which is also constant over time. Therefore, $\mathbf{n_k} \times \mathbf{n}$ and $\mathbf{p} - \mathbf{p_k}$ are constant with respect to time. The equation for temporal gradients becomes:

$$\nabla^{\mathbf{t}}(\mathbf{p}) = \nabla^{\mathbf{t}}_{\mathbf{E_k}} + (\mathbf{n_k} \times \mathbf{n}) \cdot \nabla^{\mathbf{t}}_{\nabla_{\mathbf{r}}} + (\mathbf{p} - \mathbf{p_k}) \cdot \nabla^{\mathbf{t}}_{\nabla_{\mathbf{p}}} \tag{5.10}$$

where:

- $\nabla^{\mathbf{t}}_{\mathbf{E_k}} = \frac{\partial E_k}{\partial t}$ is the *temporal gradient of irradiance*
- $\nabla^{\mathbf{t}}_{\nabla_\mathbf{r}} = \frac{\partial \nabla_\mathbf{r}}{\partial t}$ is the *temporal gradient of rotation gradient*
- $\nabla^{\mathbf{t}}_{\nabla_\mathbf{p}} = \frac{\partial \nabla_\mathbf{p}}{\partial t}$ is the *temporal gradient of translation gradient*

Using Equation 5.10, the contribution of record $k$ created at time $t_k$ to the incoming radiance at point $\mathbf{p}$ at time $t$ is estimated by:

$$\begin{aligned} E_k(\mathbf{p}, t) &= E_k + \nabla^{\mathbf{t}}_{\mathbf{E_k}}(t - t_k) + \\ &\quad (\mathbf{n_k} \times \mathbf{n}) \cdot (\nabla_\mathbf{r} + \nabla^{\mathbf{t}}_{\nabla_\mathbf{r}}(t - t_k)) + \\ &\quad (\mathbf{p_k} - \mathbf{p}) \cdot (\nabla_\mathbf{p} + \nabla^{\mathbf{t}}_{\nabla_\mathbf{p}}(t - t_k)) \end{aligned} \tag{5.11}$$

This formulation represents the temporal change of the incoming radiance around $p_k$ as 3 vectors. These vectors represent the change of the incoming radiance at point $\mathbf{p_k}$ and the change of the translation and rotation gradients over time. The values of these vectors can be easily computed using the information generated in Section 5.2.4. Since our method estimates the incoming radiance at time $t + 1$ using the information available at time $t$, we define *extrapolated* temporal gradients as:

$$\nabla^{\mathbf{t}}_{\mathbf{E_k}} \approx E(t_k + 1) - E(t_k) \tag{5.12}$$
$$\nabla^{\mathbf{t}}_{\nabla_\mathbf{r}} \approx \nabla_\mathbf{r}(t_k + 1) - \nabla_\mathbf{r}(t_k) \tag{5.13}$$
$$\nabla^{\mathbf{t}}_{\nabla_\mathbf{p}} \approx \nabla_\mathbf{p}(t_k + 1) - \nabla_\mathbf{p}(t_k) \tag{5.14}$$

However, as illustrated in Figure 5.4(d), these temporal gradients do not remove all the discontinuities in the animation. When a record $k$ is replaced by record $l$, the accuracy of the result exhibits a possible discontinuity, yielding some flickering artifacts. As explained in section 5.2.3, this problem can be avoided by keeping track of the history of the records: when record $k$ gets obsolete, a new record $l$ is created at the same location. Since $t_l > t_k$, we can use the value of incoming radiance stored in $l$ to compute *interpolated* temporal gradient for record $k$:

$$\nabla^{\mathbf{t}}_{\mathbf{E_k}} \approx (E_l - E_k)/(t_l - t_k) \tag{5.15}$$
$$\nabla^{\mathbf{t}}_{\nabla_\mathbf{r}} \approx (\nabla_\mathbf{r}(t_l) - \nabla_\mathbf{r}(t_k))/(t_l - t_k) \tag{5.16}$$
$$\nabla^{\mathbf{t}}_{\nabla_\mathbf{p}} \approx (\nabla_\mathbf{p}(t_l) - \nabla_\mathbf{p}(t_k))/(t_l - t_k) \tag{5.17}$$

In the context of radiance caching, the rotation gradient is not necessary [KGPB05]. The incoming radiance function and the translation gradient being represented using projection coefficients, we compute the temporal gradient of each coefficient independently. Thus the problem reduces to the computation of the above equations for each coefficient.

As illustrated in Figure 5.4(e), the temporal gradients enhance the continuity of the accuracy, hence removing the flickering artifacts. However, the gradients only account for the first derivative of the change of incoming lighting, hence temporally smoothing the changes. While this method proves accurate in scenes with smooth changes, it should be noted that the gradient-based temporal interpolation may introduce ghosting artifacts when used in scenes with very sharp changes of illumination. In this case, $a^t$ and $\delta_{t_{max}}$ must be reduced to obtain a sufficient update frequency.

This method provides a simple way of extending irradiance caching to dynamic objects and light sources, by introducing a temporal weighting function and temporal gradients. In the next section, we discuss the implementation of our method on a GPU for increased performance.

## 5.3 GPU Implementation

Our method has been implemented within a GPU-based renderer for irradiance and radiance caching. First, we detail the implementation of the incoming radiance estimate by reprojection (Section 5.2.4). Then, we describe how the GPU can be simply used in the context of radiance cache splatting (see Chapter 4) to discard useless records and avoid their replacement.

**Reprojection of Incoming Radiance** As shown in section 5.2.3, the computation of the temporal weighting function and temporal gradients for a given record $k$ requires an estimate of the radiance reaching point $\mathbf{p_k}$ at the next time step. This estimate is obtained through reprojection (section 5.2.4), provided that the position of the objects at next time step is known. Therefore, for a given vertex $v$ of the scene and a given time $t$, we assume that the transformation matrix corresponding to the position and normal of $v$ at time $t+1$ is known. We assume that such matrices are available for light sources as well. Using the hemisphere sampling method described in the previous chapter, a record $k$ can be generated by rasterizing the scene on a single plane above point $\mathbf{p_k}$. In a first pass, during the rasterization at time $t$, shaders can output an estimate of the position and incoming lighting at time $t+1$ of each point visible to $\mathbf{p_k}$ at time $t$. This output can be used to reconstruct an estimate of the incoming radiance function at time $t+1$.

This estimate is obtained in a second pass: each projected visible point generated in the first pass is considered as a vertex. This operation can be easily implemented using either the OpenGL Pixel Buffer Objects or OpenGL PBuffer rendering and per-vertex texture mapping. Each of those vertices is sent to the graphics pipeline as a pixel-sized point. The result of the rasterization process is an estimate of the incoming radiance function at time $t+1$. Since the size of the sampling plane is usually small (typically $64 \times 64$), this process is generally much faster than resampling the whole scene.

During the reprojection process, some fragments may overlap. Even though the occlusion can be simply solved by classical Z-Buffer, the resulting image may contain holes (Figure 5.6(d)). These holes are created at the location of dynamic objects. Since the time shift between two successive frames is very small, the holes are also small. As described in Section 5.2.4, we use a third pass to fill the holes using the local

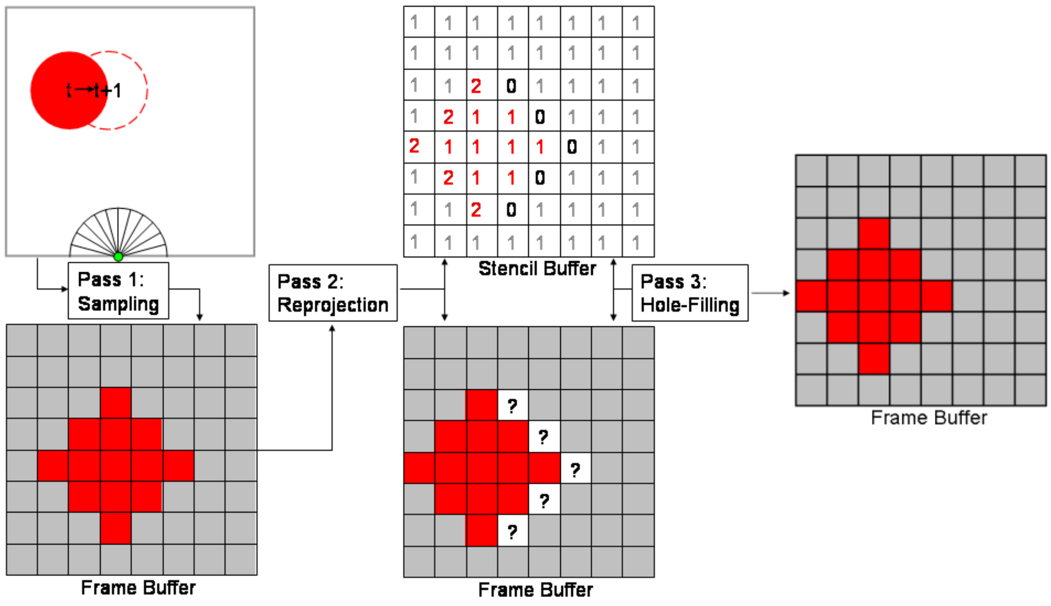
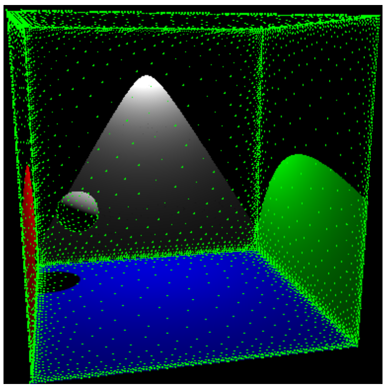


Figure 5.8: Reprojection using the GPU. The first pass samples the scene to gather the required information. Then, the visible points are reprojected to their estimated position at next time step. During this pass, each rendered fragment increments the stencil buffer. Finally, the holes (i.e. where the stencil value is 0) are filled using the deepest neighboring values.

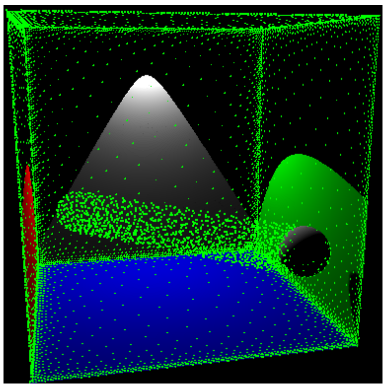
background (that is, the neighboring value with the highest depth). This computation can be performed efficiently on the GPU using the stencil buffer with an initial value of 0. During the reprojection process, each rasterized point increments the stencil buffer. Therefore, the hole-filling algorithm must be applied only on pixels where the stencil buffer is still 0. The final result of this algorithm is an estimate of the incoming radiance at time $t + 1$, generated entirely on the GPU (Figure 5.8). This estimate is used in the extrapolated temporal gradients and the temporal weighting function. As shown in Equation 5.6, this latter defines a maximum value of the lifespan of a given record, and triggers its recomputation. However, this recomputation is not always necessary.

**Replacement/Deletion Method** As described in the previous sections, the flickering artifacts of the lighting come from the temporal discontinuities of the accuracy. Therefore, if a record cannot contribute to the current image (i.e. out of the view frustum, or occluded), it can be simply deleted instead of being replaced by a novel, up-to-date record. This avoids the generation and update of a “trail” of records following dynamic objects (Figure 5.9), hence reducing the memory and computational costs. In the context of radiance cache splatting, this decision can be easily made using hardware occlusion queries: during the last frame of the lifespan of record $k$, an occlusion query is issued as the record is rasterized. In the next frame, valid records are first rendered.

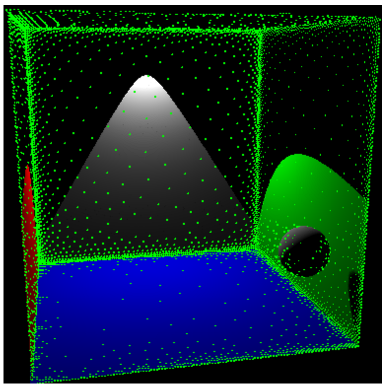
If a record $k$ is now obsolete, the result of the occlusion query is read from the GPU. If the number of covered pixels is 0, the record is discarded. Otherwise, a new record $l$ is computed at location $\mathbf{p_l} = \mathbf{p_k}$.



(a) Time 1



(b) Time 100, systematic update



(c) Time 100, our record removal method

Figure 5.9: The sphere moves from the left to the right of the Cornell Box. At time 1 (a), records (represented by green points) are generated to compute the global illumination solution. When the sphere moves, new records are created to evaluate the incoming radiance on the sphere. If every record is permanently kept up-to-date, a "trail" of records lies on the path of the dynamic sphere (b). Using our method, only useful records are updated (c).

The hardware occlusion queries are very useful, but they suffer from high latency. However, in our method, the result of a query is not needed immediately. Between the query issue and the reading of the record coverage, the renderer renders the other records, then switches the scene to next frame and renders valid records. In practice, the average latency appeared to be negligible (less than 0.1% of the overall computing time). Besides, in our test scenes, this method reduces the storage and computational costs by up to 25-30%.

## 5.4 Results

This section discusses the results obtained using our method and compares them with the classical approach in which a new cache is computed for each frame. This method is referred to as *per-frame computation* in the remainder of this section. The images, videos and timings have been generated using a 3.8GHz Pentium 4 with 2 GB RAM and an nVidia GeForce 7800 GTX 512MB. The scene details and timings are summarized in Table 5.2.

**Cube in a Box** This very simple, diffuse scene (Figure 5.12(a)) exhibits high flickering when no temporal gradients are used. Along with a significant speedup, our method reduces the flickering artifacts by using extrapolated temporal gradients. Such artifacts are unnoticeable with interpolated gradients. The animations are generated using a temporal accuracy parameter $a^t = 0.05$, and a maximum lifespan of 20 frames. The per-frame computation requires about 138 MB (772K records) to store all the irradiance records. In our method, the memory load is only 12.4 MB (50K records). Figure 5.10 shows the accuracy values obtained with and without temporal gradients. The remaining flickering of the extrapolated temporal gradients are due to the discontinuities of accuracy. Since our aim is high quality rendering, the following results focus on interpolated temporal gradients which avoid discontinuities.

**Moving Light** A similar scene (Figure 5.12(b)) illustrates the behavior of our algorithm in the context of dynamic light sources. The bottom of the box is tiled to highlight the changes of indirect lighting. Due to the highly dynamic indirect lighting, the lifespan of the records is generally very short, yielding frequent updates of irradiance values. Compared to per-frame computation, our method renders the animation with higher quality in a comparable time.

**Flying Kite** In a more complicated, textured scene (Figure 5.12(c)), our algorithm also provides a drastic quality improvement while significantly reducing the computation time. In the beginning of the animation the change of indirect lighting is small, and hence the records can be reused in several frames. However, when the kite gets down, its dynamic reflection on the ceiling and wall is clearly noticeable. Using our temporal weighting function, the global illumination solution of this zone is updated at a fast pace, avoiding ghosts in the final image (Figure 5.11).

**Japanese Interior** In this complex scene (Figure 5.12(d)), the glossy and diffuse objects are rendered using respectively the radiance and irradiance caching algorithms. The animation illustrates the features of our method: dynamic nondiffuse environment, and important changes of indirect lighting. In the beginning of the animation, the scene is lit by a single, dynamic light source. In this case, temporal gradients suppress the flickering artifacts present in per-frame computation, but do not provide a significant speedup (1.25×). In the remainder of the animation, most of the environment is static, even though some dynamic objects generate strong changes in the indirect illumination. Our temporal gradients take advantage of this situation by adaptively reusing records in several frames. The result is the elimination of flickering and a significant speedup (up to 9×) compared to per-frame computation. During the generation of this animation, the average latency introduced by occlusion queries is 0.001% of the overall rendering time.

**Spheres** This scene features complex animation with 66 diffuse and glossy bouncing spheres and a glossy back wall (Figure 5.12(e)). Our method eliminates the flickering while reducing the computational cost of a factor 4.24. We used a temporal accuracy value $a^t = 0.05$ and a maximum record lifespan $\delta_{t_{max}} = 5$.

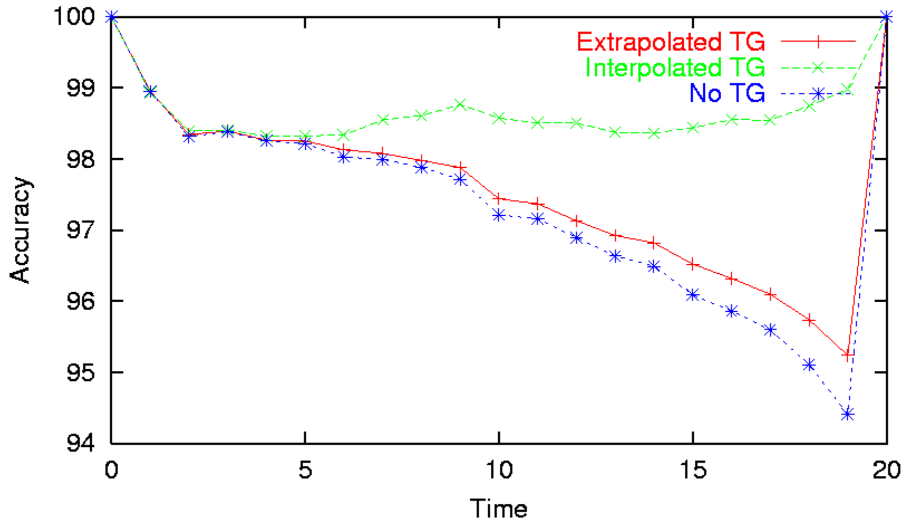
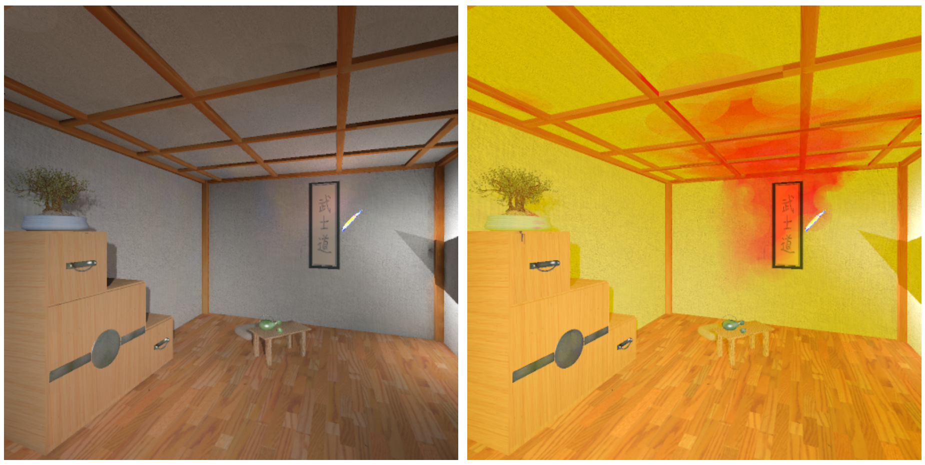


Figure 5.10: Temporal accuracy values obtained in scene Cube in a Box by creating records at time 0 and extrapolating their value until time 19. New records are re-computed at time 20. The temporal gradients (TG) provide a better approximation compared to the approach without those gradients. Using interpolated gradients, the accuracy is continuous and remains above 98%.

| Scene | Nb. Poly | Nb. Frames | Per-Frame Comp. | Our Method | Speedup |
|---|---|---|---|---|---|
| Cube in a Box | 24 | 400 | 2048s | 269s | 7.62 |
| Moving Light | 24 | 400 | 2518s | 2650s | 0.95 |
| Flying Kite | 28K | 300 | 5109s | 783s | 6.52 |
| Japanese Interior | 200K | 750 | 13737s | 7152s | 1.9 |
| Spheres | 64K | 200 | 3189s | 753s | 4.24 |

Table 5.2: Test scenes and timings



(a) Actual sampling frame (b) Records lifespan

Figure 5.11: When computing the global illumination solution for the current frame (a), our method estimates where the lighting changes. The lifespan of each generated record is computed by estimating the future change of lighting (b). Green and red colors respectively represent long and short lifespan.

**Computational Overhead of Reprojection** During the computation of a record, our method evaluates the value of the incoming lighting for both current and next time steps. As shown in Section 5.2.4, the estimation of the future incoming lighting is performed by simple reprojection. Therefore, the related computational overhead is independent of the scene geometry. In our tests, each record was computed at resolution $64 \times 64$. On our system, the reprojection is performed in approximately 0.46 ms. For comparison, the time required to compute the actual incoming lighting at a given point in our 200K polygons scene is 4.58 ms. In this case, the overhead due to the reprojection is only 10% of the cost of the actual hemisphere sampling. Even though this overhead is not negligible, our estimate enables us reduce the overall rendering time by reusing the records in several frames.

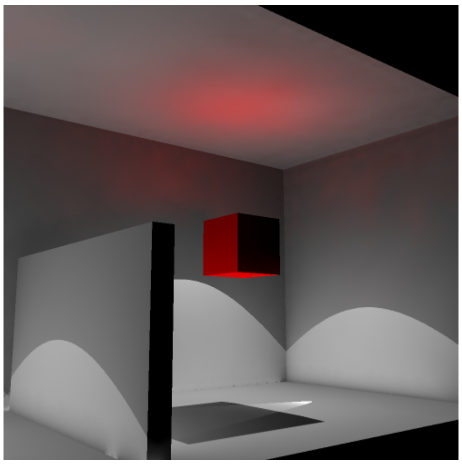
## 5.5 Conclusion

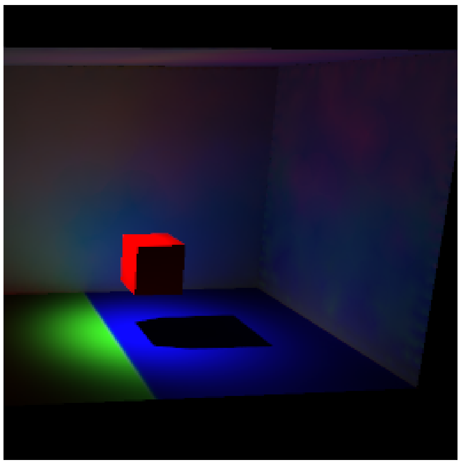
We presented a novel method for exploiting temporal coherence in the context of irradiance and radiance caching. When is used to render each frame of an animation, irradiance and radiance caching create the temporal discontinuities of the accuracy of the lighting, resulting in disturbing flickering artifacts. Our approach is based on sparse sampling and interpolation of the indirect lighting in the temporal domain. The irradiance and radiance records are adaptively reused in several frames using our temporal weighting function and temporal gradients. Our temporal weighting function determines the lifespan of each record depending on the change rate of the local incoming radiance: if strong changes of incoming radiance are detected, the corresponding lifespan is reduced to ensure a sufficient temporal sampling rate. At the end of the record lifespan, a new record is generated at the same location. This method reduces the flickering artifacts while allowing for the computation of interpolated temporal gradients for smooth temporal interpolation of the indirect lighting.

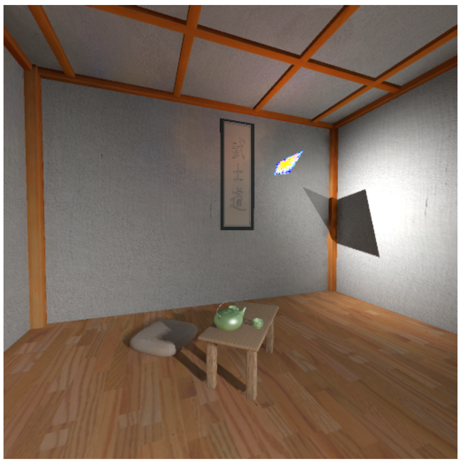
The evaluation of the temporal weighting function and of the extrapolated temporal gradients rely on an approximate knowledge of the future incoming radiance. We propose an inexpensive, GPU-accelerated reprojection method for fast computation of this estimate. The GPU is also used to determine which records must be updated at the end of their respective lifespans.

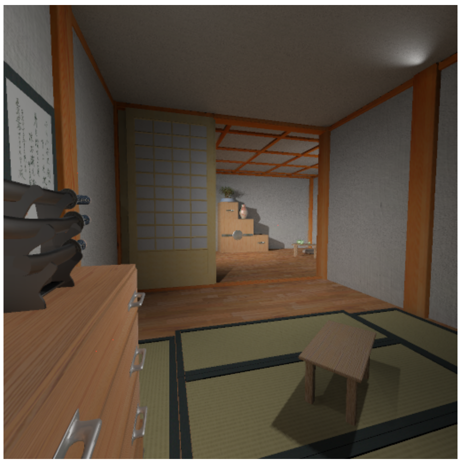
The results show both a significant speedup and an increased quality compared to per-frame computation. Due to our sparse temporal sampling, the incoming radiance values for the entire animation segment can be stored within main memory. Our method providing significant quality improvement and ease of implementation, we believe that our approach can be easily integrated in production renderers for efficient rendering of animated scenes.

Future work includes the design of a more accurate estimation method for extrapolated temporal gradients. Such a method will find use in on-the-fly computation of indirect lighting during interactive sessions. Another improvement would consist in designing an efficient method for faster aging of the records located near newly created records for which important changes have been detected. This would avoid the need for a user-defined maximum validity time, while guaranteeing the absence of global illumination ghosts.


(a) Cube in a Box


(b) Moving Light


(c) Flying Kite


(d) Japanese Interior

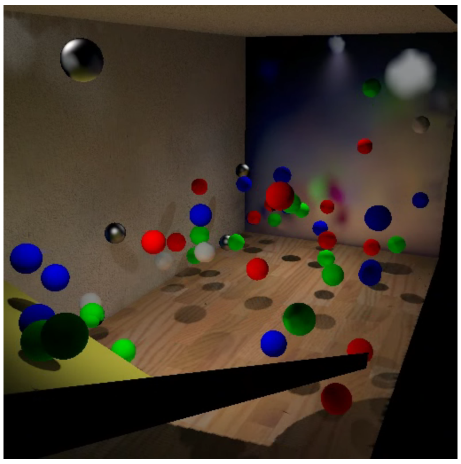

(e) Spheres

Figure 5.12: Images of scenes discussed in Section 5.4.

# Chapter 6

# Photon Mapped Irradiance Caching

## 6.1 Motivations

The photon mapping algorithm is one of the most popular global illumination method. By following the real path of light, this method can simulate any complex lighting effect such as caustics, lighting using light pipes, etc. However, as shown in Chapter 2, the photon map cannot be used directly to generate high quality images. For each viewpoint, a costly pass of final gathering must be performed by tracing rays from each visible point, yielding then a prohibitive computational overhead especially in the context of interactive applications.

The irradiance caching method leverages the spatial coherency of the indirect lighting by sparse sampling and interpolation. For a given viewpoint, irradiance cache records are computed to render the desired image. Even though the location of the records is defined by the current viewpoint, their value is view-independent. Therefore, if the scene remains static, images from other viewpoints can be generated quickly by reusing the previously created records instead of recomputing a novel global illumination solution from scratch. Still, this method cannot easily be used in the context of interactive applications due to the necessary computation of new irradiance records for each frame.

As we aim at combining the generality of the photon mapping and the computational efficiency of irradiance caching for interactive walkthroughs in globally illuminated scenes, we propose a fast method for generating irradiance cache records using only the information contained in the photon map. Once the records are computed in the whole scene, the radiance cache splatting algorithm (see Chapter 4) is used for rendering the global illumination solution interactively. We supervised this work performed by Jonathan Brouillat for his MSc degree. In this chapter we only present an overview of the method. Theoretical and implementation details can be found in [Bro06].

## 6.2 Photon Mapped Irradiance Caching

The algorithm developed in this work is divided into four parts. First, the photon map is generated by tracing photons from the light sources towards the scene. Second, the algorithm determines the locations at which irradiance records will be created. In the third part, the computation of the zone of influence, irradiance and gradients of each record is performed independently of the scene geometry using the contents of the photon map only. The estimation of these values is carried out using a precise density estimation technique. Fourth, our radiance cache splatting algorithm (Chapter 4) allows the user to walk through the globally illuminated scene in real-time.

### 6.2.1 Photon Map Generation

Our algorithm is based on a regular photon map. Each stored photon contains the corresponding position, flux, incoming direction, and distance to the last surface on which the photon has been reflected (Figure 6.1).

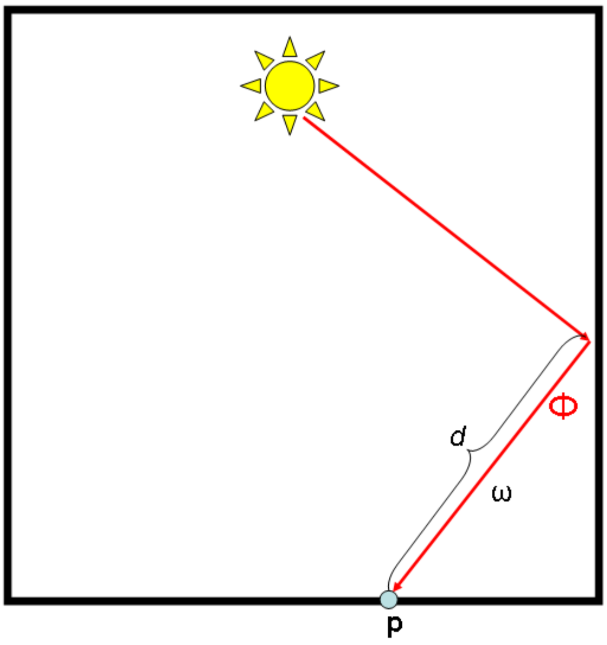


Figure 6.1: Each photon contains the corresponding position $\mathbf{p}$, incoming direction $\omega$, flux $\Phi$, and distance $d$ to the last surface on which the photon has been reflected.

### 6.2.2 Selection of Record Location

Our method for selecting the location of records is based on the following observation: the illuminated parts of the scene are covered by photons. Therefore, the irradiance records are created adaptively at the location of the photons (Algorithm 6). At the beginning of the algorithm, the irradiance cache is empty. An irradiance record is then created at the location of the first photon of the photon map, and stored in the irradiance cache. Then, the irradiance cache is queried at the location $\mathbf{p_i}$ of every other photon $i$ to determine whether a new record is required. This query identifies the set $S(\mathbf{p_i})$ of records contributing to the indirect lighting at $\mathbf{p_i}$ and computes the weights related to each contribution. Depending on the cumulated weight, a new record may be created or not at $\mathbf{p_i}$:

$$\sum_{k \in S(\mathbf{p_i})} w_k(\mathbf{p_i}) \begin{cases} < a & \text{New record is required at } \mathbf{p_i} \\ \geq a & \text{Interpolation can provide a satisfying result} \end{cases} \tag{6.1}$$

As in [Jen01], the memory layout of our photon map is a linear array. For the sake of efficiency and memory coherence, we chose to traverse this array sequentially: for each photon in the array, the irradiance cache is queried as described above. However, it must be noted that any used traversal algorithm would lead to similar results. When a new record is required our algorithm computes the corresponding irradiance and gradients using a density estimation on the nearby photons.

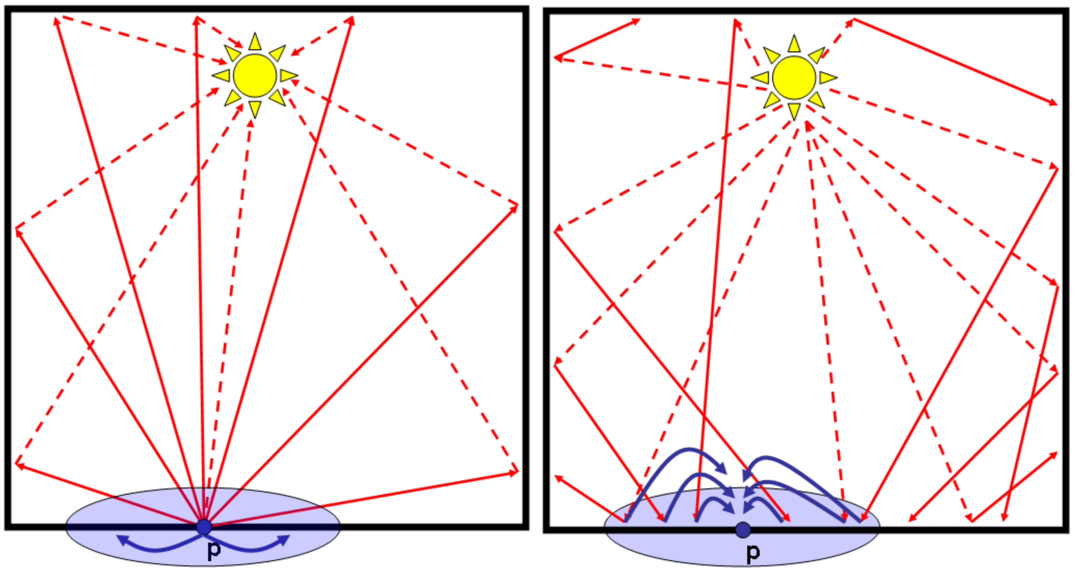
**Algorithm 6** Selection of record location

```
for all photon $P$ located at $\mathbf{p_i}$ in the photon map do
  Compute the cumulated contributions of nearby irradiance records by querying the
  irradiance cache
  if a new record is needed ((Equation 6.1)) then
    Create an irradiance record using the photons around $\mathbf{p_i}$
    Store the irradiance record in the irradiance cache
  end if
end for
```

### 6.2.3 Record Generation

In the irradiance caching algorithm, the irradiance computed at a given point is used to estimate the indirect lighting at the points within its zone of influence. Therefore, the irradiance of the record is considered as representative of the indirect lighting in this zone. In this work, we use the reverse approach: the irradiance at a point is estimated using the indirect lighting incoming in its zone of influence (Figure 6.3).

Once the location $\mathbf{p}$ of the record is chosen according to Algorithm 6, the photon map is queried in a close neighborhood around $\mathbf{p}$ (Figure 6.3(a)). The last distance traversed by the photons within this neighborhood being known (Figure 6.1), those distances are averaged to estimate the harmonic mean distance to the surrounding objects. As described in Chapter 4, this mean distance determines the size of the zone of influence of the record. A second query centered at $\mathbf{p}$ is then issued in the photon map with the search radius set to the size of the zone of influence of the record (Figure 6.3(b)). Our



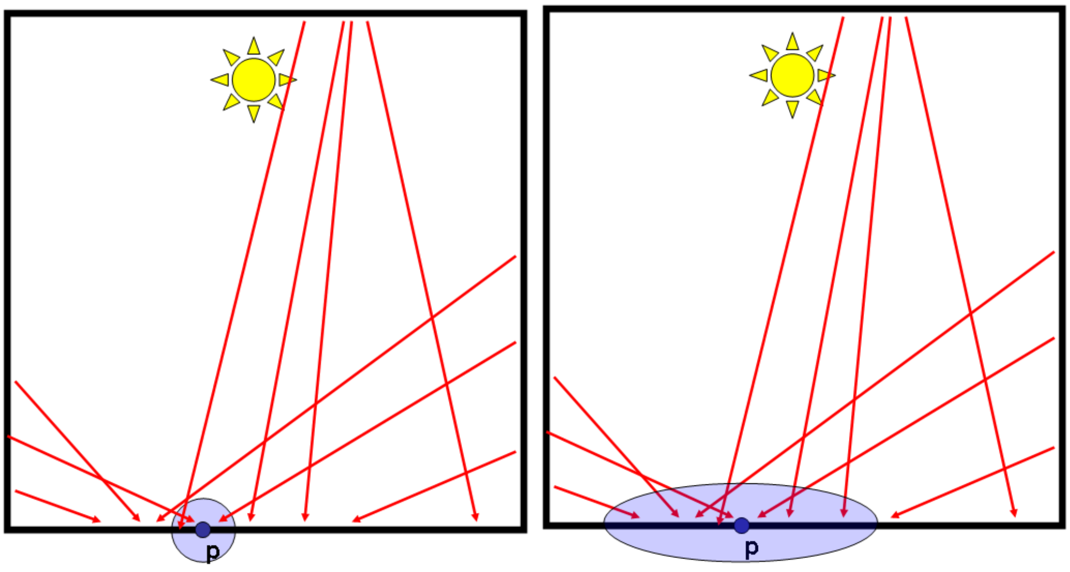
(a) Irradiance Caching (b) Photon Mapped Irradiance Caching

Figure 6.2: The irradiance caching algorithm (a) is based on the computation of the irradiance at a point **p**. This irradiance record is used to estimate the lighting at the points within its zone of influence. In our method (b), the irradiance of a record located at point **p** is computed using the flux of the photons within its zone of influence.

algorithm computes the irradiance and gradients using the photons found in this second query by density estimation.

Usually, the search radius used for density estimation in photon maps is chosen as a small disc to avoid an excessive blurring of the lighting details and to reduce the errors in density estimation at the borders of the geometry, known as *boundary bias* (Figure 6.4). Those errors are due to an incorrect estimation of the area spanned by the nearby photons (Figure 6.5(b)). However, the zone of influence of irradiance records is generally larger than the typical search radii used in the context of photon mapping. Therefore, the errors due to the estimation of the area at the borders of the geometry can introduce important errors in the estimation of the irradiance and gradients, yielding an incorrect global illumination solution. Therefore, instead of approximating the area spanned by the photons using a simple disc, we propose to use the convex hull of the photons within the zone of influence of the record (Figure 6.5). This approach avoids the boundary bias by estimating the area more accurately. Since this approach based on the computation of the convex-hull tends to underestimate the real area spanned by the photons (Figure 6.6), a statistical compensation method is detailed in [Bro06].

Using the incoming flux $\Phi_i$ for each photon $i$ and the area $A$ spanned by the photons,



(a) First query (b) Second query

Figure 6.3: The computation of an irradiance record is performed using two queries in the photon map. The first query (a) locates the photons (red arrows) within a small neighborhood around the record location **p**. This query is used to estimate the harmonic mean distance $h$ to the points visible from **p**. The radius of the zone of influence of the record is deduced from $h$, and is the search radius of a second query (b). The photons returned by this query are used to estimate the irradiance and gradients of the record.

the irradiance $E$ of the record is defined as in [Jen01]:

$$E = \frac{1}{A}\sum_{i=1}^{n}\Phi_i \tag{6.2}$$

where $n$ is the number of photons within the zone of influence of the record.

The density estimation method described above aims at computing the total incoming flux over the zone of influence of the records. However, the computation of irradiance gradients requires the knowledge of the directional distribution of the incoming radiance. Using a method similar to [WH92, KGBP05], we discretize the hemisphere above the record location into cells. Using the incoming directions of the photons, we compute the flux incoming through each cell (Figure 6.8). The density estimation described above aims at computing the irradiance at a point. However, the gradient computation described in [KGBP05] is based on incoming radiance values. The irradiance can be

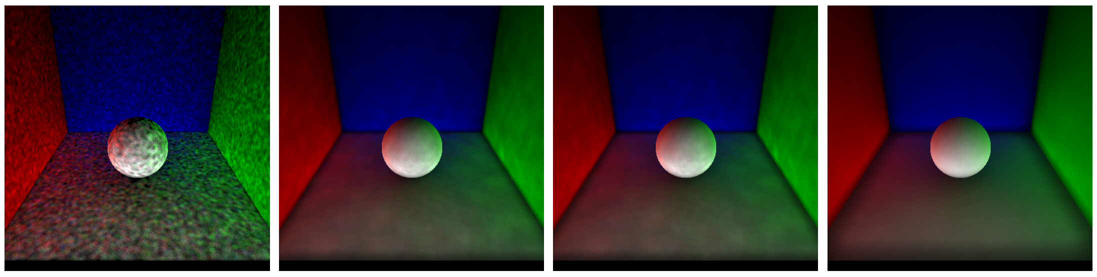

(a) Radius = 1 (b) Radius = 5 (c) Radius = 10 (d) Radius = 50

Figure 6.4: The size of the disc used for the density estimation in photon maps is a tradeoff between noise (a) and bias due to an excessive blurring (d). Note that using a large disc for the density estimation leads to visible artifacts on the borders of the geometry due to an incorrect estimation of the area spanned by the photons.

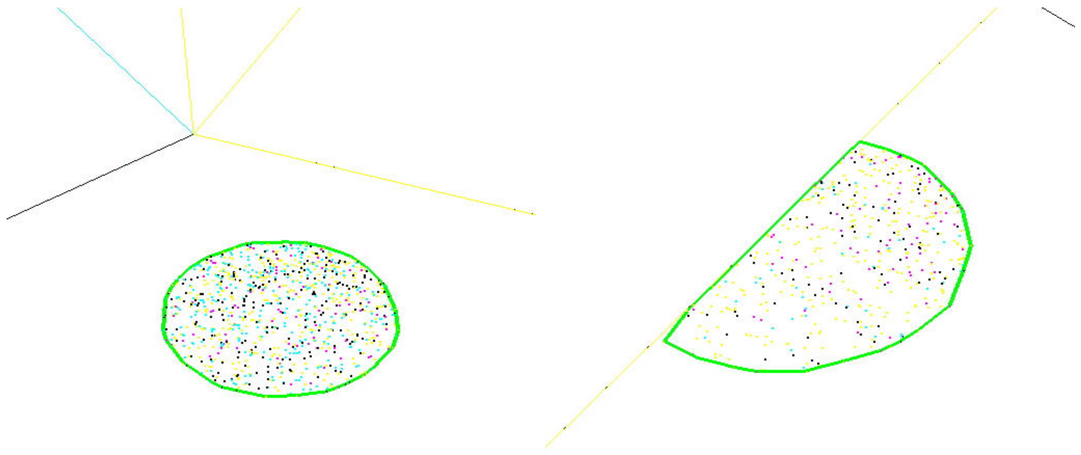

(a) (b)

Figure 6.5: Usually, the area spanned by the photons (shown as colored dots) within a circular zone around a given point is considered as a disc. While this assumption is true in the middle of a surface (a), the area spanned by the photons on the borders is not circular (b). We propose to replace the disc approximation by the computation of the convex hull of the photons (in green) for an accurate estimation of the area.

expressed in terms of incoming radiance or in terms of incoming flux:

$$E = \frac{1}{A}\sum_{i=1}^{n}\Phi_i \tag{6.3}$$

$$E = \int_{\Omega} L_i(\omega)\cos\theta \mathrm{d}\omega \tag{6.4}$$

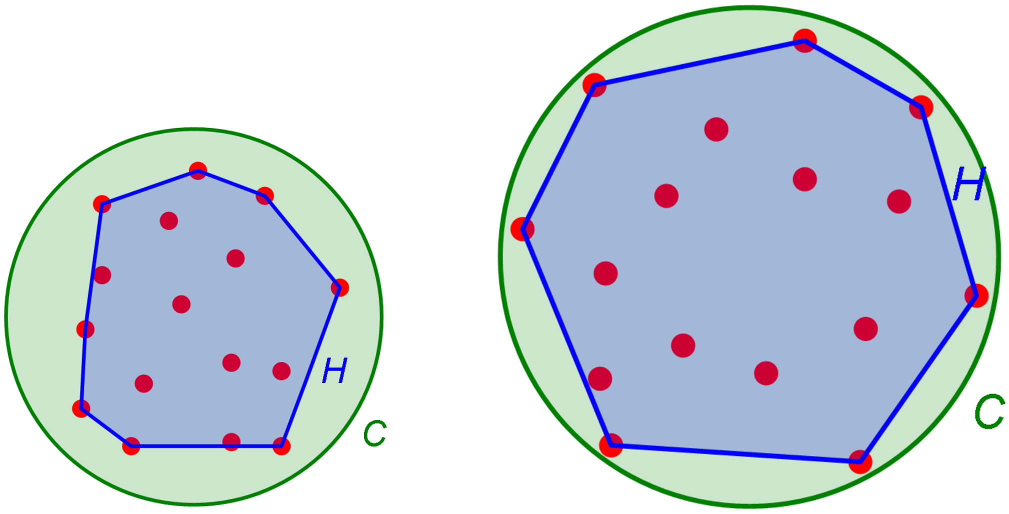


Figure 6.6: Convex hulls of photons distributed in the disc $C$. The area of the convex hull $H$ is systematically smaller than the area of the disc $C$, and depends on the distribution of the photons within the disc.

where $n$ is the number of neighboring photons.

Let us consider a cell $(i,j)$ defined by hemispherical coordinates $(\phi_-,\phi_+)$ and $(\theta_-,\theta_+)$ (Figure 6.7). The flux $\Phi_{i,j}$ incoming through the cell $(i,j)$ is:

$$\Phi_{i,j} = \sum_{k\in S} \Phi_k \quad , \quad S = \{\text{neighboring photon } k \mid \theta_- \leq k_\theta < \theta_+, \phi_- \leq k_\phi < \phi_+\} \tag{6.5}$$

where $(k_\theta, k_\phi)$ are the hemispherical coordinates of the incoming direction of photon $k$.

Considering only the radiance $L_{i,j}$ and the flux $\Phi_{i,j}$ incoming through this cell, the irradiance $E_{i,j}$ incoming through $(i,j)$ can be calculated using Equations 6.3 and 6.4:

$$E_{i,j} = \frac{\Phi_{i,j}}{A} \tag{6.6}$$

$$E_{i,j} = \int_{i,j} L_{i,j}(\omega)\cos\theta \mathrm{d}\omega \tag{6.7}$$

The incoming radiance $L_{i,j}$ being assumed constant over the cell $(i,j)$, we obtain:

$$E_{i,j} = L_{i,j}\int_{i,j} \cos\theta \mathrm{d}\omega \tag{6.8}$$

$$= L_{i,j}\int_{\phi_-}^{\phi_+}\int_{\theta_-}^{\theta_+} \cos\theta\sin\theta \mathrm{d}\theta \mathrm{d}\phi \tag{6.9}$$

$$= \frac{L_{i,j}}{2}\left(\cos^2\theta_- - \cos^2\theta_+\right)(\phi_+ - \phi_-) \tag{6.10}$$

Therefore, the radiance incoming through the cell $(i, j)$ can be expressed in terms of the flux incoming through this cell:

$$L_{i,j} = \frac{\Phi_{i,j}}{A} \frac{2}{(\cos^2 \theta_- - \cos^2 \theta_+)(\phi_+ - \phi_-)} \tag{6.11}$$

The irradiance gradients are then computed using the incoming radiances as described in [KGBP05].

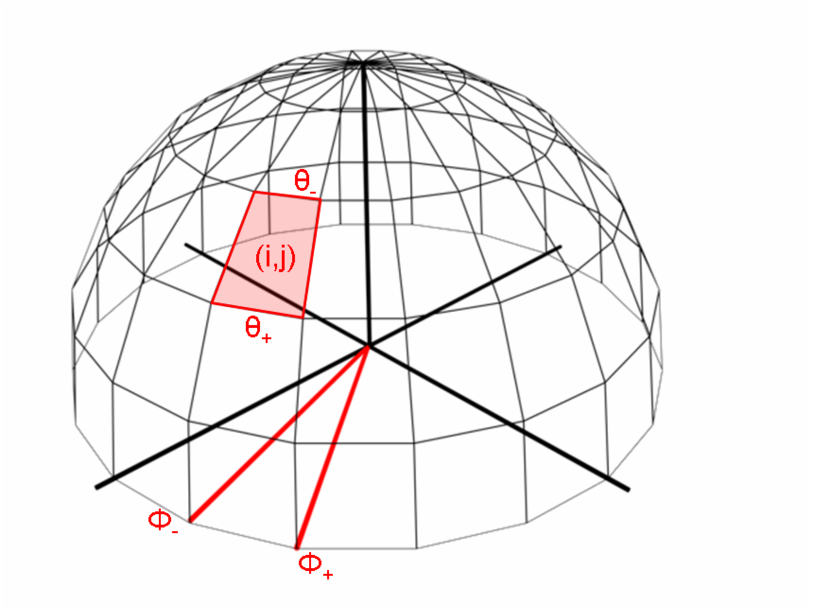


Figure 6.7: The hemisphere above the record location is discretized in cells $(i, j)$ defined by their boundaries $(\phi_-, \phi_+)$, $(\theta_-, \theta_+)$ in hemispherical coordinates.

This method allows the computation of irradiance records independently of the scene geometry by using the contents of the photon map. Note that unlike the hemisphere sampling proposed in Chapter 4, this method accounts for an arbitrary number of light bounces in the scene. Once the records are computed, the global illumination solution can be visualized interactively.

### 6.2.4 Cache Visualization

Once computed as described above, the contents of the irradiance cache can be directly visualized using the radiance cache splatting algorithm described in Chapter 4. Since the irradiance cache is considered sufficient for rendering the global illumination solution from any viewpoint, the cache miss detection and record computation steps are bypassed. The globally illuminated scene can then be visualized in real-time.

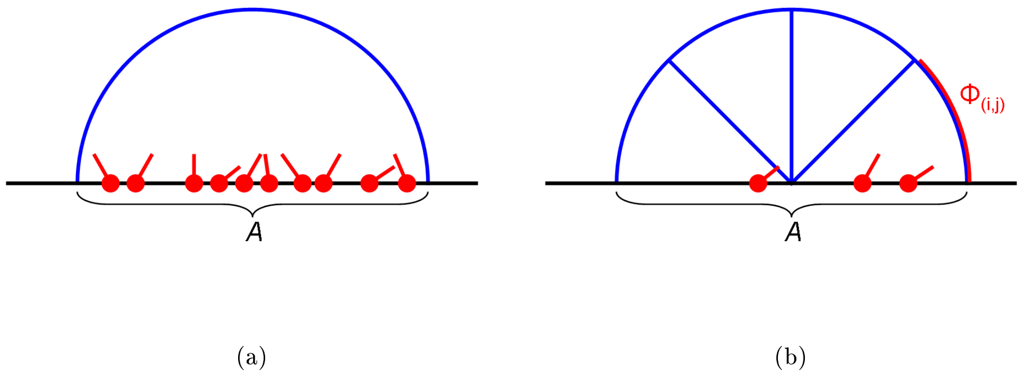


Figure 6.8: The photons gathered within a zone of size $A$ around the record location (a) are projected and accumulated onto each cell of a discretized hemisphere for computing the irradiance gradients.(b) represents the photons projected onto the rightmost cell.

## 6.3 Results

We compared the results obtained using direct photon map visualization, classical irradiance caching and our method (Figure 6.9). The images and timings presented in this section have been obtained using an 3.2GHz Pentium 4 with 2GB RAM and a nVidia 7800 GTX 512MB graphics card. In this simple test scene, our method provides results of higher quality compared to a direct visualization of the photon map. Furthermore, the boundary bias is removed by the use of our area approximation by convex hull. In this scene, the photon map contains 500K photons, traced in 3.49 s. The 2066 records were generated in 7.7 s, that is 3.71 ms/record. For comparison, the record generation using the hemisphere sampling presented in Chapter 4 took 2.6 ms/record. Even though in this simple scene the actual hemisphere sampling is faster than our method, it should be noted that the cost of our record generation is independent of the geometry, and that any number of light bounces can be simulated. However, holes may appear in zones containing no photons. Due to the efficiency of the record computation, such holes may be filled at runtime by computing a few additional records, without introducing a noticeable frame rate drop.

## 6.4 Conclusion

In this section we outlined the work performed by Jonathan Brouillat under our supervision. Note that a full description of this work can be found in [Bro06]. This method computes a full global illumination solution in two passes. A classical photon map is computed in the first pass. The second pass consists of an accurate density estimation for generating irradiance records using the contents of the photon map only. This den-

**Algorithm 7** Record generation at point $\mathbf{p}$

```
Query the photon map in a close neighborhood around $\mathbf{p}$
Using the distances stored in the photons, compute an estimate of $h$, the harmonic mean distance to the surrounding objects
Using $h$, compute the radius $r$ of the zone of influence of the record
Query the photon map to gather the photons within a radius $r$ around $\mathbf{p}$
Using the distances of those photons, update the harmonic mean distance $h$
Compute the convex hull of the photons
Compute the corresponding area $A$ using statistical correction
Compute the irradiance
Discretize the hemisphere above $\mathbf{p}$
Project each photon onto the hemisphere
for all cell of the hemisphere do
  Compute the incoming flux
  Use this flux and the area $A$ to compute the radiance incoming through the cell
end for
Compute irradiance gradients as in [KGBP05]
```

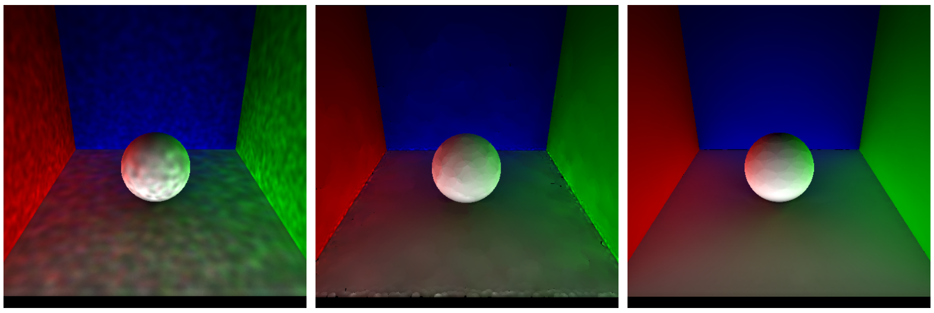

(a) Direct visualization of the photon map (b) Our method (c) Irradiance caching

Figure 6.9: Results obtained using a direct visualization of the photon map (a), irradiance caching (b) and our method (c).

sity estimation based on the computation of the convex hull of nearby photons allows us to compute reliable estimates of the irradiance records and gradients at any point in the scene. In this work we chose to generate irradiance records at the location of the photons stored in the photon map. Once the position of each photon has been covered by a sufficient number of irradiance records, the global illumination solution can be rendered in real-time using radiance cache splatting (Chapter 4).

Even though this method shows promising results, several remaining problems have to be solved. Among them, the selection of the record locations should be revised to ensure a complete coverage of the scene, hence avoiding the need of computing new records at runtime. An other open problem is the removal of noise in the estimation of the gradients: the flux incoming through a cell of the discretized hemisphere may be erroneously considered as zero due to the absence of photons traversing this cell. Such cells can introduce disturbing artifacts in the irradiance gradients. Finally, we would like to extend this method to the radiance caching algorithm for efficient photon map rendering on glossy surfaces.

# Chapter 7

# Conclusion

Our work focused on fast computation of global illumination and rendering using graphics hardware. In this chapter, we conclude our work by summarizing our contributions and proposing directions for future work.

## 7.1 Contributions

### 7.1.1 Hemispherical Harmonics

We first proposed the hemispherical harmonics basis for an accurate representation of hemispherical functions such as incoming radiance functions and BRDFs. While spherical harmonics are defined on the whole sphere, we restricted the definition domain of those basis functions to the upper hemisphere only. Compared to the spherical harmonics with a same number of coefficients, our basis represents hemispherical functions more accurately.

As global illumination methods often involve operations in different coordinate systems, we proposed three methods for efficiently rotating functions projected into the hemispherical harmonics basis. We also proposed a sparse matrix for converting spherical harmonics coefficients to hemispherical harmonics coefficients and *vice-versa*.

We applied our new basis functions in the context of GPU-based environment map lighting by representing the environment with spherical harmonics and the material BRDF with hemispherical harmonics. Our method achieves frame rates comparable to a method based on spherical harmonics only while improving the rendering quality. We also applied our basis to the radiance caching algorithm for efficient computation of global illumination on glossy surfaces. In this case the hemispherical harmonics rotation is particularly useful for combining incoming radiance functions defined in different coordinate systems.

### 7.1.2 Radiance Cache Splatting

Generally, the global illumination algorithms are designed for implementation on the CPU. However, as graphics hardware is getting more and more powerful and versatile, we used GPUs for the computation and rendering of global illumination solutions. In this work we reformulated the irradiance and radiance caching algorithms to make the implementation on graphics hardware easy and efficient. Using the weighting function of the irradiance caching interpolation scheme, we proposed the *radiance cache splatting* algorithm: the zone of influence of each record is splatted onto the image plane. For each pixel within the splat, the weight and weighted contribution of the record are accumulated into the *radiance splat buffer*. Once the contributions of each record have been accumulated, this buffer is used for rendering the global illumination solution. By processing each record independently and by avoiding the need for a particular cache data structure, our method is well-suited for GPU implementation: the splatting of the zone of influence of a record is performed using a vertex program. The contribution of each record to the indirect lighting of the points visible through each pixel is computed by a fragment program. The accumulation of the contributions is achieved using the floating-point alpha blending capability proposed in the last generations of graphics hardware.

The computation of irradiance and radiance records rely on the sampling of the hemisphere above the record location. Since the shading and visibility tests involved in this computation are the bottleneck of the irradiance and radiance caching algorithms, we propose a GPU-accelerated method for hemispherical sampling. We sample the incoming radiance above a point of interest by rasterizing the scene as seen from this point on a single plane. Since the radiances incoming from grazing angles cannot be accounted for, we propose a simple compensation method for an accurate estimation of the incoming radiance.

The results detailed in Chapter 4 show that our method achieves a significant speedup (up to 40×) compared to a CPU-based renderer, without degrading the rendering quality. Furthermore, our method allows interactive rendering for fast previewing of global illumination solutions.

### 7.1.3 Temporal Radiance Caching

The irradiance and radiance caching algorithms aim at the efficient computation of high quality global illumination in static scenes. However, the extrapolation and interpolation operations involved in these algorithms yield flickering artifacts when computing a new cache for each frame of an animation. We proposed to extend the spatial interpolation scheme proposed in [WRC88] to the temporal domain by reusing records over time. To this end, we introduced a temporal weighting function. This function determines the lifespan of each record independently based on the local change of incoming radiance over time. However, the evaluation of the weighting function requires the knowledge of the future incoming radiance. Therefore, we propose a simple estimation method based on reprojection: using a basic knowledge of the future displacement of objects and light sources, our algorithm estimates the future position and shading of the visible points. This method can be inexpensively implemented on graphics hardware for improved performance.

We also propose temporal gradients for an accurate temporal extrapolation of the incoming radiance. Those gradients can be either *extrapolated* (i.e. based on our estimate of the future incoming lighting) or *interpolated* using the actual computation of the incoming radiance.

We implemented our method within our GPU-based renderer for irradiance and radiance caching. By reusing the records over time, our method drastically reduces the rendering time. Furthermore, the rendering quality is significantly improved compared to the classical method in which a new cache is computed for each frame of the animation.

However, our method has some limitations, such as the need of a user-defined maximum lifespan of the records. This need is due to potential errors in the estimate of the future incoming lighting. Our method should be improved for providing a more reliable estimate of the change of incoming lighting. Furthermore, we only consider a linear temporal interpolation. While this method is valid in scenes with smooth temporal changes of indirect lighting, sharp changes cannot be accounted for accurately.

### 7.1.4 Photon Mapped Irradiance Caching

Our last contribution is an improved density estimation technique for photon map rendering. In this work, our aim was the computation of irradiance records using only the contents of a previously computed photon map. Relying on simple observations of the irradiance cache algorithm and of the distribution of the photons, this method, developed by Jonathan Brouillat, is able to generate a set of irradiance records without tracing additional rays.

At the location of each photon, the irradiance cache is queried. If a new record is needed, the photons within a close neighborhood around the point of interest are gathered from the photon map. The nearby photons are used to determine the size of the zone of influence of the record being created. A second query in the photon map is then issued to gather the photons located within the zone of influence of the record. The photons obtained in this second query are used to perform a density estimation. To avoid the well-known *boundary bias*, the area spanned by the photons is calculated accurately as the area of the convex hull of the photons. The irradiance of the record is obtained from the result of the density estimation. Finally, the irradiance gradients are computed by projecting the photon direction and accumulating the corresponding flux into the cells of a discretized hemisphere located above the point of interest. This discretized hemisphere is then used to compute the irradiance gradients as described in [KGBP05].

The preliminary results detailed in Chapter 6 showed that our method provides a significant quality improvement compared to classical photon mapping. Furthermore, the generated records can be visualized in realtime using the radiance cache splatting detailed in Chapter 4.

However, this method needs further developments. Future research will consider tests in more complex scenes, and the extension to global illumination on glossy surfaces using radiance caching.

## 7.2 Future Work

### 7.2.1 All-Frequency Representation of Hemispherical Functions

A challenging improvement of the methods described in this thesis is the accurate representation of high frequency hemispherical functions: one drawback of the radiance caching algorithm is its limitation to low-frequency BRDFs and incoming radiance functions. Using spherical or hemispherical harmonics to represent high frequencies lead to the disturbing *ringing* artifacts due to the Gibbs phenomenon. A multiresolution representation, such as the method proposed by Schröder *et al.* [SS95], should be investigated. However, the aliasing problems of this method have to be addressed. Furthermore, the use of such a representation in the context of the radiance caching algorithm requires the definition of translational gradients and of a method for combining functions defined in different coordinate systems. An efficient data structure suitable for implementation on GPUs should also be proposed.

### 7.2.2 GPU-Based Hemispherical Sampling

The most important speedup factor of the algorithm described in Chapter 4 is our GPU-based hemispherical sampling method for fast record computation. In our approach we extrapolate the radiances incoming from grazing angles using the data obtained by rasterizing the scene onto a single plane. While the errors introduced by our method are small, future work should consider a more accurate sampling method. An interesting direction is to use a paraboloid parametrization [HS99]. However, while the whole hemisphere can be sampled within a single rendering pass using this method, a finely tesselated scene is required for accurate rendering.

An other improvement of the radiance cache splatting algorithm is its extension to multiple bounces. A promising direction is recursive splatting: during the generation of a record using rasterization, a second-level radiance cache is splatted on the rasterization plane. Using a method similar to the cache miss detection detailed in Chapter 4, new records are generated if necessary. While this method would allow the computation and rendering of multiple-bounces global illumination, the costs of recursive splatting and cache miss detection should be investigated.

### 7.2.3 Improved Temporal Interpolation and Extrapolation

In Chapter 5 we described an temporal interpolation scheme similar to [WRC88]. However, our method relies on a simple linear interpolation in the temporal domain. As shown in the results, our method achieves a significant quality improvement compared to per-frame computation. However, the rendering quality is highly dependent on the changes of indirect lighting: sharp changes of indirect lighting cannot be accounted for properly due to our linear temporal interpolation. Therefore, future work should consider an accurate detection and modeling of the temporal changes of incoming radiance. This would improve the interpolation quality and avoid the need of a user-defined maximum lifespan of the records.

# Bibliography

[Ben75] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *ACM Student Award*, 18(9):509–517, 1975.

[BFB97] Miguel A. Blanco, M. Florez, and M. Bermejo. Evaluation of the rotation matrices in the basis of real spherical harmonics. *Journal Molecular Structure (Theochem)*, 419:19–27, 1997.

[BP04] Michael Bunnell and Fabio Pellacini. *GPU Gems: Shadow map antialiasing*, pages 185–192. Addison Wesley, 1 edition, 2004.

[Bro06] Jonathan Brouillat. Simulation d'éclairage à l'aide de cartes de photons et cache de luminance. Master's thesis, IRISA, 2006.

[Buc03] Ian Buck. *Brook Spec v0.2*. 2003.

[CG85] Michael Cohen and Donald P. Greenberg. The Hemi-Cube: A Radiosity Solution for Complex Environments. In *Proceedings of SIGGRAPH*, volume 19, pages 31–40, August 1985.

[CH05] Greg Coombe and Mark Harris. *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*, chapter Global Illumination Using Progressive Refinement Radiosity, pages 636–648. Addison-Wesley Professional, 2005.

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

[CHH03] Nathan A. Carr, Jesse D. Hall, and John C. Hart. GPU algorithms for radiosity and subsurface scattering. In *Proceedings of Graphics Hardware*, pages 51–59, 2003.

[CHL04] Greg Coombe, Mark J. Harris, and Anselmo Lastra. Radiosity on graphics hardware. In *Proceedings of Graphics Interface*, pages 161–168, 2004.

[Chr99] Per H. Christensen. Faster photon map global illumination. *Journal of Graphics Tools*, 4(3):1–10, 1999.

[CIGR99] Cheol Ho Choi, Joseph Ivanic, Mark S. Gordon, and Klaus Ruedenberg. Rapid and stable determination of rotation matrices between spherical harmonics by direct recursion. *Journal of Chemical Physics*, 111(19):8825–8831, 1999.

[CJ02] Mike Cammarano and Henrik Wann Jensen. Time dependent photon mapping. In *Proceedings of Eurographics Workshop on Rendering*, June 2002.

[CMS87] Brian Cabral, Nelson Max, and Rebecca Springmeyer. Bidirectional reflection functions from surface bump maps. In *Proceedings of SIGGRAPH*, pages 273–281, New York, NY, USA, 1987. ACM Press.

[Dam01] Cyrille Damez. *Simulation Globale de l'Eclairage Pour Des Sequences Animees Prenant En Compte la Coherence Temporelle*. PhD thesis, Universite Joseph Fourier, Grenoble, France, 2001.

[DBB03] Philip Dutré, Philippe Bekaert, and Kavita Bala. *Advanced Global Illumination*. A.K. Peters, 2003.

[DBMS02] Kirill Dmitriev, Stefan Brabec, Karol Myszkowski, and Hans-Peter Seidel. Interactive global illumination using selective photon tracing. In *Proceedings of Eurographics Workshop on Rendering*, June 2002.

[DDM02] Cyrille Damez, Kirill Dmitriev, and Karol Myszkowski. Global illumination for interactive applications and high-quality animations. In *Proceedings of Eurographics*, pages 55–77, September 2002.

[DS97] George Drettakis and Francois X. Sillion. Interactive update of global illumination using a line-space hierarchy. In *Proceedings of SIGGRAPH*, volume 31, pages 57–64, 1997.

[DSH01] Cyrille Damez, Francois X. Sillion, and Nicolas Holzschuch. Space-time hierarchical radiosity with clustering and higher-order wavelets. In *Proceedings of Eurographics*, pages 129–141, September 2001.

[Dut03] Philip Dutre. Global illumination compendium. http://www.cs.kuleuven.ac.be/ phil/GI/, 2003.

[DvGNK99] Kristin J. Dana, Bram van Ginneken, Shree K. Nayar, and Jan J. Koenderink. Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics*, 18(1):1–34, 1999.

[FMH05] David Fradin, Daniel Meneveaux, and Sebastian Horna. Out of core photon-mapping for large buildings. In *Proceedings of Eurographics Workshop on Rendering*, June 2005.

[GBP06] Pascal Gautron, Kadi Bouatouch, and Sumanta Pattanaik. Temporal radiance caching. Technical Report 1796, IRISA, Rennes, France, 2006.

[GD04] Xavier Granier and George Drettakis. A final reconstruction approach for a unified global illumination algorithm. *ACM Transactions on Graphics*, 23(2):163–189, 2004.

[GKBP05] Pascal Gautron, Jaroslav Křivánek, Kadi Bouatouch, and Sumanta Pattanaik. Radiance cache splatting: A GPU-friendly global illumination algorithm. In *Proceedings of Eurographics Symposium on Rendering*, June 2005.

[GKPB04] Pascal Gautron, Jaroslav Křivánek, Sumanta Pattanaik, and Kadi Bouatouch. A novel hemispherical basis for accurate and efficient rendering. In *Proceedings of Eurographics Symposium on Rendering*, pages 321–330, 2004.

[Gla95] Andrew S. Glassner. *Principles of Digital Image Synthesis.* Morgan Kaufmann, San Francisco, CA, 1995.

[GMSJ03] D. Gutierrez, A. Munoz, F. Seron, and E. Jimenez. Global illumination in inhomogeneous media based on curved photon mapping. In *Third IASTED International Conference on Visualization, Imaging and Image Processing*, 2003.

[GSHG98] Gene Greger, Peter Shirley, Philip M. Hubbard, and Donald P. Greenberg. The irradiance volume. *IEEE Computer Graphics and Applications*, 18(2):32–43, March/April 1998.

[GTGB84] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modelling the Interaction of Light Between Diffuse Surfaces. In *Proceedings of SIGGRAPH*, volume 18, pages 212–222, July 1984.

[GWS04] Johannes Guenther, Ingo Wald, and Phillip Slusallek. Realtime caustics using distributed photon mapping. In A. Keller and H. W. Jensen, editors, *Proceedings of Eurographics Symposium on Rendering*, pages 111–121, June 2004.

[Hei91] Tim Heidmann. Real shadows, real time. *Iris Universe*, 18:23–31, 1991.

[HHS05] V. Havran, R. Herzog, and H.-P. Seidel. Fast final gathering via reverse photon mapping. *Proceedings of Eurographics*, 24(3), September 2005. To appear.

[HP02] Heinrich Hey and Werner Purgathofer. Advanced radiance estimation for photon map global illumination. *Proceedings of Eurographics*, 21(3), September 2002.

[HS99] Wolfgang Heidrich and Hans-Peter Seidel. Realistic, hardware-accelerated shading and lighting. In *Proceedings of SIGGRAPH*, pages 171–178. ACM Press/Addison-Wesley Publishing Co., 1999.

[Int02] Intel Corporation. *AGP v3.0 Specification.* 2002.

[IR96] Joseph Ivanic and Klaus Ruedenberg. Rotation matrices for real spherical harmonics. direct determination by recursion. *Journal of Physical Chemistry*, 100(15):6342–6347, 1996.

[IR98] Joseph Ivanic and Klaus Ruedenberg. Additions and corrections : Rotation matrices for real spherical harmonics. *Journal of Physical Chemistry Part A*, 102(45):9099–9100, 1998.

[ISO00] ISO/IEC 15444-1:2000. Information technology - JPEG 2000 coding system - part 1: Core coding system. 2000.

[Jen96] Henrik Wann Jensen. Global Illumination Using Photon Maps. In *Proceedings of Eurographics Workshop on Rendering*, pages 21–30. Springer-Verlag/Wien, 1996.

[Jen01] Henrik Wann Jensen. *Realistic Image Synthesis Using Photon Mapping.* A. K. Peters, Natick, MA, 2001.

[JMP05] Juan-Roberto Jimenez, Karol Myszkowski, and Xavier Pueyo. Interactive global illumination in dynamic participating media using selective photon tracing. In *Proceedings of Spring Conference on Computer Graphics*, pages 211–218. ACM, 2005.

[Kaj86] James T. Kajiya. The Rendering Equation. In *Proceedings of SIGGRAPH*, volume 20, pages 143–150, August 1986.

[KBR04] John Kessenich, Dave Baldwin, and Randi Rost. *The OpenGL Shading Language.* 2004.

[KGBP05] Jaroslav Křivánek, Pascal Gautron, Kadi Bouatouch, and Sumanta Pattanaik. Improved radiance gradient computation. In *Proceedings of SCCG*, pages 149–153, 2005.

[KGPB05] Jaroslav Křivánek, Pascal Gautron, Sumanta Pattanaik, and Kadi Bouatouch. Radiance caching for efficient global illumination computation. *IEEE Transactions on Visualization and Computer Graphics*, 2005.

[Kři05] Jaroslav Křivánek. *Radiance Caching for Global Illumination Computation on Glossy Surfaces.* Ph.d. thesis, Université de Rennes 1 and Czech Technical University in Prague, December 2005.

[KSS02] Jan Kautz, Peter-Pike Sloan, and John Snyder. Fast, arbitrary BRDF shading for low-frequency lighting using spherical harmonics. In *Proceedings of Eurographics Workshop on Rendering*, pages 291–296, 2002.

[KvDS96] J.J. Koenderink, A.J. van Doorn, and M. Stavridi. Bidirectional reflection distribution function expressed in terms of surface scattering modes. *ECCV*, B:28–39, 1996.

[KW00] Alexander Keller and Ingo Wald. Efficient importance sampling techniques for the photon map. In *Proceedings of Vision, Modeling and Visualization*, pages 271–279, Saarbruecken, Germany, November 2000. IOS Press.

[Lau05] Sebastien St Laurent. *The Complete HLSL Reference.* Paradoxal Press, 2005.

[LC04] Bent Dalgaard Larsen and Niels Jorgen Christensen. Simulating photon mapping for real-time applications. In A. Keller and H. W. Jensen, editors, *Proceedings of Eurographics Symposium on Rendering*, pages 123–131, June 2004.

[LP03] Fabien Lavignotte and Mathias Paulin. Scalable photon splatting for global illumination. In *Proceedings of GRAPHITE.* ACM Press, 2003.

[LSSS04] Xinguo Liu, Peter-Pike Sloan, Heung-Yeung Shum, and John Snyder. All-frequency precomputed radiance transfer for glossy objects. In A. Keller and H. W. Jensen, editors, *Proceedings of Eurographics Symposium on Rendering*, pages 337–344, June 2004.

[Mak96] O.A. Makhotkin. Analysis of radiative transfer between surfaces by hemispherical harmonics. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 56(6):869–879, 1996.

[MM02] V. C. H. Ma and M. D. McCool. Low latency photon mapping using block hashing. In *SIGGRAPH/Eurographics Workshop on Graphics Hardware*, 2002.

[MPT03] Ignacio Martín, Xavier Pueyo, and Dani Tost. Frame-to-frame coherent animation with two-pass radiosity. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):70–84, 2003.

[MT04] Michael McCool and Stefanus Du Toit. *Metaprogramming GPUs with Sh.* A. K. Peters, 2004.

[nC05] nVidia Corporation. *Cg User Manual.* 2005.

[NPG03] Mangesh Nijasure, Sumanta Pattanaik, and Vineet Goel. Interactive global illumination in dynamic environments using commodity graphics hardware. In *Proceedings of Pacific Graphics*, 2003.

[NPG04] Mangesh Nijasure, Sumanta Pattanaik, and Vineet Goel. Real-time global illumination on the GPU. *Journal of Graphics Tools*, 2004. To appear.

[PBMH02] Timothy J. Purcell, Ian Buck, William R. Mark, and Pat Hanrahan. Ray tracing on programmable graphics hardware. *Proceedings of SIGGRAPH*, 21(3):703–712, 2002.

[PCI03] PCI-SIG. *PCI-Express v1.0e Specification.* 2003.

[PDC$^+$03] Timothy J. Purcell, Craig Donner, Mike Cammarano, Henrik Wann Jensen, and Pat Hanrahan. Photon mapping on programmable graphics hardware. In *Proceedings of Graphics Hardware*, pages 41–50, 2003.

[PH04] Matt Pharr and Greg Humphreys. *Physically Based Rendering.* Morgan Kaufmann, 2004.

[Pho75] Bui Tuong Phong. Illumination for computer generated pictures. *Commununications of the ACM*, 18(6):311–317, 1975.

[PTVF88] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C - THe Art of Scientific Computing.* Cambridge University Press, 1988.

[RH02] Ravi Ramamoorthi and Pat Hanrahan. Frequency space environment map rendering. In *Proceedings of SIGGRAPH*, pages 517–526, 2002.

[RSC87] W.T. Reeves, D.H. Salesin, and R.L. Cook. Rendering antialiased shadows with depth maps. In *Proceedings of SIGGRAPH*, volume 21, pages 283–291, August 1987.

[SAWG91a] Francois Sillion, James R. Arvo, Stephen H. Westin, and Donald P. Greenberg. A Global Illumination Solution for General Reflectance Distributions. In *Proceedings of SIGGRAPH*, volume 25, pages 187–196, July 1991.

[SAWG91b] Francois X. Sillion, James R. Arvo, Stephen H. Westin, and Donald P. Greenberg. A global illumination solution for general reflectance distributions. In *Proceedings of SIGGRAPH*, pages 187–196. ACM Press, 1991.

[SB97] Wolfgang Sturzlinger and Rui Bastos. Interactive rendering of globally illuminated glossy scenes. In Julie Dorsey and Philipp Slusallek, editors, *Proceedings of Eurographics Workshop on Rendering*, pages 93–102. Springer Wien, 1997. ISBN 3-211-83001-4.

[Sch03] Roland Schregle. Bias compensation for photon maps. *Computer Graphics Forum*, 22(4), 2003.

[SD02] Marc Stamminger and George Drettakis. Perspective shadow maps. In *Proceedings of SIGGRAPH*, July 2002.

[SHHS03] Peter-Pike Sloan, Jesse Hall, John Hart, and John Snyder. Clustered principal components analysis for precomputed radiance transfer. In *Proceedings of SIGGRAPH*, 2003.

[SKDM05] Miloslaw Smyk, Shin-ichi Kinuwaki, Roman Durikovic, and Karol Myszkowski. Temporally coherent irradiance caching for high quality animation rendering. In *Proceedings of Eurographics*, volume 24, pages 401–412, 2005.

[SKS02] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *Proceedings of SIGGRAPH*, 21(3):527–536, 2002.

[SLS05] Peter-Pike Sloan, Ben Luna, and John Snyder. Local, deformable precomputed radiance transfer. In *Proceedings of SIGGRAPH*, August 2005.

[Sme98] V.V. Smelov. On completeness of semispherical harmonics system. *Siberian journal of Mathematics*, 1(4):391–395, 1998. In Russian.

[SP89] Francois Sillion and Claude Puech. A General Two-Pass Method Integrating Specular and Diffuse Reflection. In *Proceedings of SIGGRAPH*, volume 23, pages 335–344, July 1989.

[SS95] Peter Schröder and Wim Sweldens. Spherical wavelets: efficiently representing functions on the sphere. In *Proceedings of SIGGRAPH*, pages 161–172, 1995.

[SSS01] Annette Scheel, Marc Stamminger, and Hans-Peter Seidel. Thrifty final gather for radiosity. In *Proceedings of Eurographics Workshop on Rendering*, June 2001.

[SSS02] Annette Scheel, Marc Stamminger, and Hans-Peter Seidel. Grid based final gather for radiosity on complex clustered scenes. In *Proceedings of Eurographics*, volume 21, September 2002.

[SW00] Frank Suykens and Yves D. Willems. Density control for photon maps. In B. Peroche and H. Rushmeier, editors, *Proceedings of Eurographics Workshop on Rendering*, pages 23–34, 2000.

[Sze75] Gabor Szegö. *Orthogonal polynomials.* American Mathematical Society, Providence, Rhode Island, 4 edition, 1975.

[TL04] Eric Tabellion and Arnauld Lamorlette. An approximate global illumination system for computer-generated films. In *Proceedings of SIGGRAPH*, August 2004.

[TMS02] Takehiro Tawara, Karol Myszkowski, and Hans-Peter Seidel. Localizing the final gathering for dynamic scenes using the photon map. In *VMV*, 2002.

[TMS04] T. Tawara, K. Myszkowski, and H.-P. Seidel. Exploiting temporal coherence in final gathering for dynamic scenes. In *Proceedings of Computer Graphics International*, pages 110–119, June 2004.

[TPWG02] Parag Tole, Fabio Pellacini, Bruce Walter, and Donald P. Greenberg. Interactive global illumination in dynamic scenes. *Proceedings of SIGGRAPH*, 21(3):537–546, 2002.

[VG95] Eric Veach and Leonidas J Guibas. Optimally combining sampling techniques for monte carlo rendering. In *Proceedings of SIGGRAPH*, pages 419–428, 1995.

[War92] Gregory J. Ward. Measuring and modeling anisotropic reflection. In *Proceedings of SIGGRAPH*, pages 265–272, 1992.

[War94] Gregory J. Ward. The RADIANCE Lighting Simulation and Rendering System. In *Computer Graphics Proceedings, Annual Conference Series, 1994 (Proceedings of SIGGRAPH*, pages 459–472, 1994.

[WAT92] Stephen H. Westin, James R. Arvo, and Kenneth E. Torrance. Predicting reflectance functions from complex surfaces. In *Proceedings of SIGGRAPH*, pages 255–264, 1992.

[WBS03] Ingo Wald, Carsten Benthin, and Philipp Slusallek. Interactive global illumination in complex and highly occluded environments. In *Proceedings of Eurographics Symposium on Rendering*, June 2003.

[WC92] James C. Wyant and Katherine Creath. Basic wavefront aberration theory for optical metrology. In *Applied optics and Optical Engineering, Vol XI*, pages 27–39. Academic Press, Inc., 1992.

[WDG02] Bruce Walter, George Drettakis, and Donald P. Greenberg. Enhancing and optimizing the render cache. In *Proceedings of Eurographics Workshop on Rendering*, pages 37–42, 2002.

[WDP99] Bruce Walter, George Drettakis, and Steven Parker. Interactive rendering using the render cache. In *Proceedings of Eurographics Workshop on Rendering*, pages 235–246, 1999.

[Wei06a] Eric Weisstein. *World of Mathematics: A Wolfram Web Resource.* http://mathworld.wolfram.com/
GibbsPhenomenon.html, 2006.

[Wei06b] Eric Weisstein. *World of Mathematics: A Wolfram Web Resource.* http://mathworld.wolfram.com/
LegendrePolynomial.html, 2006.

[WGS04] Ingo Wald, Johannes Gunter, and Phillip Slusallek. Balancing considered harmful - faster photon mapping using the voxel volume heuristic. *Proceedings of Eurographics*, 23(3), 2004.

[WH92] Gregory J. Ward and Paul Heckbert. Irradiance Gradients. In *Proceedings of Eurographics Workshop on Rendering*, pages 85–98, Bristol, UK, May 1992.

[Wil78] Lance Williams. Casting curved shadows on curved surfaces. In *Proceedings of SIGGRAPH*, pages 270–274, 1978.

[WMM$^+$04] Markus Weber, Marco Milch, Karol Myszkowski, Kirill Dmitriev, Przemyslaw Rokita, and Hans-Peter Seidel. Spatio-temporal photon density estimation using bilateral filtering. In *Proceedings of Computer Graphics International*, pages 120–127. IEEE Computer Society, June 2004.

[WRC88] Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A Ray Tracing Solution for Diffuse Interreflection. In *Proceedings of SIGGRAPH*, volume 22, pages 85–92, August 1988.

[WS99] Gregory Ward and Maryann Simmons. The holodeck ray cache: an interactive rendering system for global illumination in nondiffuse environments. *ACM Transactions on Graphics*, 18(4):361–368, 1999.

[WS03] Michael Wand and Wolfgang Straßer. Real-time caustics. *Proceedings of Eurographics*, 22(3), September 2003.

[WTL04] R. Wang, J. Tran, and D. Luebke. All-frequency relighting of non-diffuse objects using separable brdf approximation. In A. Keller and H. W. Jensen, editors, *Proceedings of Eurographics Symposium on Rendering*, June 2004.

# Résumé en Français

## Introduction

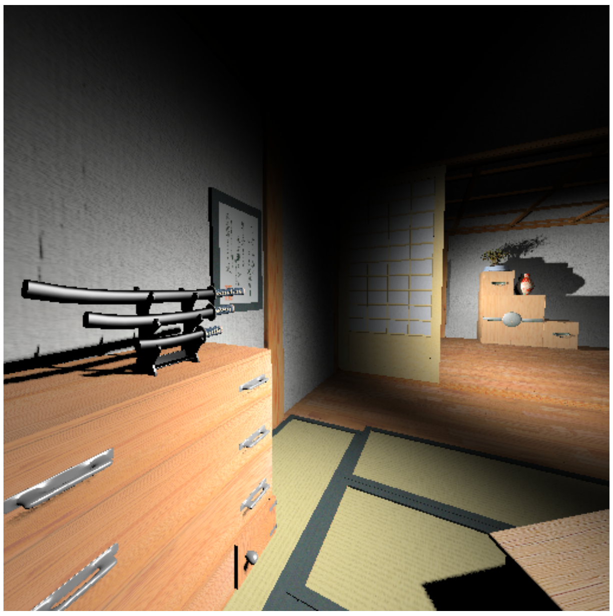
L'objectif du rendu réaliste est la génération d'images de synthèse représentant la réalité aussi fidèlement que possible. De nombreux domaines industriels sont particulièrement concernés, tels que l'architecture, le cinéma, ou encore les jeux vidéo. Toutefois, le calcul d'images réalistes nécessite une représentation et une simulation précise des interactions lumière/matière. Dans cette thèse, nous nous intéressons au calcul de simulation d'éclairage rapide dans des scènes animées. Plus précisément, notre objectif est la simulation des multiples interreflexions de la lumière sur les objets d'une scène. Ce calcul d'interreflexions est appelé calcul d'illumination globale.

### Illumination Globale

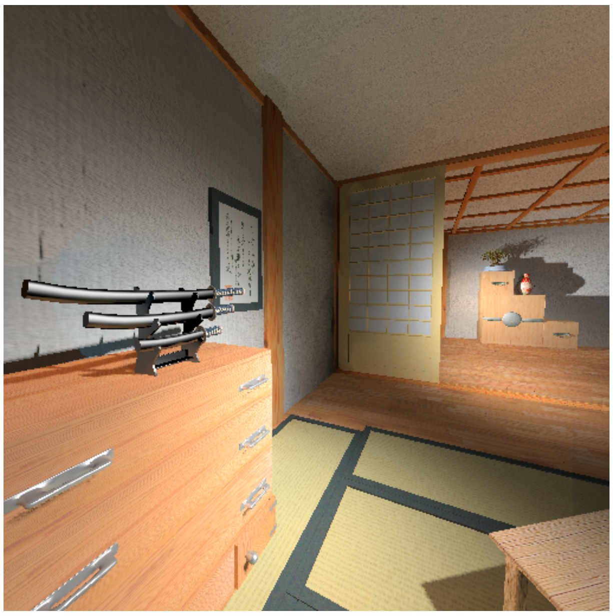
De nombreux travaux de recherche sur le calcul d'illumination globale ont été publiés durant les dernières décennies. L'objectif est une simulation précise du comportement de la lumière sur des objets faits de matériaux variés. Le calcul d'illumination globale peut être divisé en deux parties: l'éclairage direct et l'éclairage indirect. L'éclairage direct ne considère qu'une seule réflexion de la lumière entre la source lumineuse et l'observateur (Figure 1(a)). L'éclairage indirect prend en compte les multiples interreflexions de la lumière dans la scène (Figure 1(b)). L'éclairage indirect permet ainsi une simulation précise d'effets complexes tels que les caustiques et l'éclairage par sources de lumières indirectes. Toutefois, le calcul de l'éclairage indirect est généralement très long et complexe. C'est pourquoi de nombreuses approximations sont utilisées, notamment dans le contexte d'applications interactives telles que les jeux vidéo. Dans ce cas, les cartes graphiques programmables de dernière génération permettent de calculer des solutions approximatives et visuellement agréables en temps-réel.

### Rendu Temps-Réel et Cartes Graphiques

Les évolutions récentes des cartes graphiques programmables ont augmenté de manière importante la qualité des images calculables en temps-réel. Les derniers processeurs graphiques, ou GPUs (Graphics Processing Units) permettent à l'utilisateur d'écrire des programmes complexes exécutés pour chaque sommet (vertex shader) et pour chaque pixel (fragment shader). Une fois compilés et stockés dans la mémoire de la carte



(a) Éclairage direct uniquement



(b) Illumination globale

Figure 1: L'éclairage direct (a) ne considère qu'une seule réflexion de la lumière. L'éclairage indirect prend en compte les multiples interreflexions de la lumière, augmentant ainsi le réalisme de l'image obtenue.

graphique, ces programmes peuvent être exécutés en temps-réel pour simuler des effets d'éclairage complexes. Toutefois, les hautes performances des processeurs graphiques sont dues à la parallélisation des algorithmes. Par conséquent, les algorithmes et structures de données utilisables sont relativement limités.

De nombreuses méthodes de calcul d'illumination globale accéléré par cartes graphiques ont été proposées. Toutefois, les limitations dues à la structure parallèle des GPUs contraignent fortement les algorithmes, qui reposent donc sur des approximations incorrectes du point de vue physique.

## Méthodes Classiques de Calcul d'Illumination Globale

Les méthodes d'illumination globale étant très étudiées, nous supposons que le lecteur dispose de connaissances de base dans ce domaine. Le livre de Pharr et Humphreys [PH04] contient de nombreux détails théoriques et pratiques sur le rendu et le calcul d'illumination globale. Dans [Gla95], Glassner détaille la physique de la lumière ainsi que des algorithmes pour un calcul précis et physiquement réaliste des interactions lumière/matière. Des thèmes avancés concernant l'illumination globale sont détaillés dans [DBB03].

Le but des calculs d'illumination globale est la résolution de l'équation de rendu [Kaj86]. Cette équation intégrale exprime la luminance incidente $L_o$ en un point $\mathbf{p}$ en fonction de la luminance propre $L_e$ et de la luminance incidente $L_i$ provenant des éléments de la scène visibles depuis le point $\mathbf{p}$:

$$L_o(\mathbf{p},\omega_o) = L_e(\mathbf{p},\omega_o) + \int_\Omega f_r(\mathbf{p},\omega_o,\omega_i)L_i(\mathbf{p},\omega_i)\cos\theta_i \mathrm{d}\omega_i \tag{1}$$

Dans cette équation, $f_r$ est la fonction de réflectance bidirectionnelle du matériau, ou BRDF (Bidirectional Reflectance Distribution Function). Cette fonction représente comment l'éclairage incident est reflété par la surface en un point $\mathbf{p}$.

L'équation de rendu ne peut pas être résolue analytiquement dans le cas général. Toutefois, de nombreuses méthodes numériques ont été proposées. Dans les sections suivantes nous donnons un aperçu des méthodes existantes les plus utilisées. Un état de l'art complet a été publié en 2002 [DDM02].

### Radiosité

La méthode de radiosité a été introduite par Goral *et al.* [GTGB84]. Cette approche est basée sur un calcul par éléments finis: chaque surface de la scène (incluant les sources lumineuses) est divisée en éléments de surfaces, ou *patches*. La méthode de radiosité calcule les échanges d'énergie lumineuse entre patches. Bien que cette méthode permette d'obtenir des résultats de haute qualité, les surfaces doivent être subdivisées très finement afin d'éviter des effets de maillage. De nombreuses méthodes ont été proposées pour résoudre ce problème, notamment en créant un maillage s'adaptant aux discontinuités de l'éclairage. Toutefois, ces méthodes requièrent généralement des temps de calcul et des espaces de stockage importants.

### Suivi de Chemin par Méthode de Monte-Carlo

L'intégration par méthodes de Monte-Carlo [PTVF88] est l'une des méthodes les plus utilisées pour résoudre l'équation de rendu. Le suivi de chemins suit le trajet inverse de la lumière : des rayons sont tracés depuis le point de vue vers les objets de la scène. Pour chaque point visible, l'équation de rendu est résolue numériquement par méthode de Monte-Carlo : la luminance incidente en un point est évaluée en traçant des rayons depuis le point concerné dans des directions aléatoires. Un aspect important de cette méthode est que le calcul de la luminance en deux points distincts est totalement indépendant. Ceci permet d'implémenter facilement cet algorithme pour exécution en parallèle, et n'est pas coûteux en termes de mémoire. Toutefois, les valeurs de luminance calculées pour deux pixels voisins sont totalement décorrélées. Il s'ensuit donc un problème de bruit dû à l'utilisation de nombres aléatoires (Figure 2). De plus, le suivi de chemins ne permet pas de simuler certains effets lumineux complexes tels que les caustiques. Toutefois, ces effets peuvent être simulés à l'aide de cartes de photons présentées ci-après.

### Cartes de photons

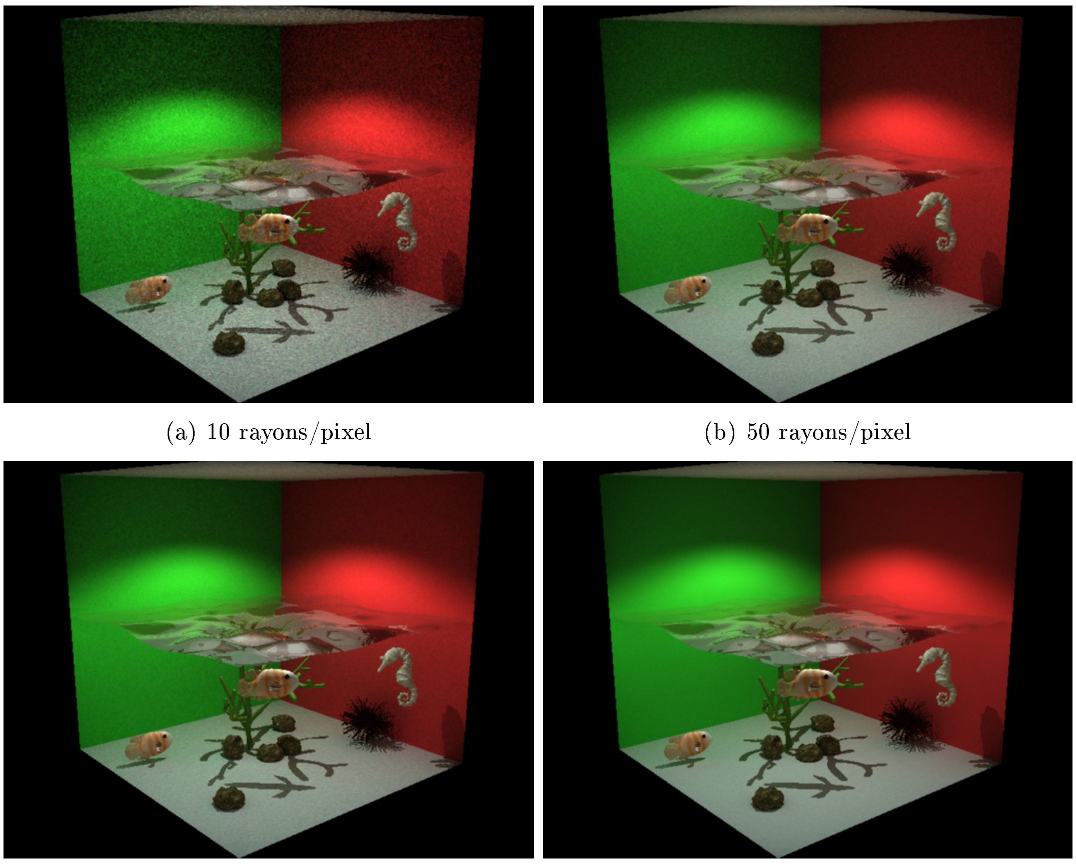
L'algorithme de calcul d'illumination globale par cartes de photons, ou *photon mapping* a été introduit par Jensen [Jen96, Jen01]. Cet algorithme calcule le chemin réel de la lumière, depuis les sources lumineuses vers les objets de la scène. Le flux émis par chaque source lumineuse est séparé en un certain nombre de rayons transportant des *photons*. Lors de l'impact d'un rayon sur un objet, le photon correspondant est stocké dans un kD-Tree [Ben75], appelé carte de photon ou *photon map*. Par la suite, un autre rayon est réémis depuis le point d'impact, transportant un photon réfléchi. Quand une quantité suffisante de photons est stockée dans la carte de photons, l'éclairement indirect en un point $\mathbf{p}$ peut être calculé en regroupant tous les photons se trouvant dans le voisinage de $\mathbf{p}$. Toutefois, l'utilisation directe de la carte de photons pour le rendu ne génère pas des résultats satisfaisants en général (Figure 3(a)). C'est pourquoi un calcul additionnel appelé *regroupement final* est le plus souvent utilisé pour obtenir des images de haute qualité (Figure 3(b)).

### Regroupement final

L'étape de regroupement final ou *final gathering* consiste à résoudre l'équation suivante pour tout point visible $\mathbf{p}$:

$$L_o(\mathbf{p}, \omega_o) = \int_{\Omega} f_r(\mathbf{p}, \omega_o, \omega_i) L_i(\mathbf{p}, \omega_i) \cos\theta_i \mathrm{d}\omega_i \tag{2}$$

Cette équation exprime la luminance réfléchie en un point $\mathbf{p}$ et dans une direction $\omega_o$ en fonction des luminances incidentes provenant de toutes les directions de l'hémisphère se trouvant au dessus de $\mathbf{p}$. Ces luminances incidentes peuvent être obtenues à l'aide d'une solution d'illumination globale grossière, typiquement calculée par carte de photons. Bien que cette méthode puisse générer des résultats de haute qualité (Figure 3), de


(a) 10 rayons/pixel (b) 50 rayons/pixel

(c) 100 rayons/pixel (d) 500 rayons/pixel

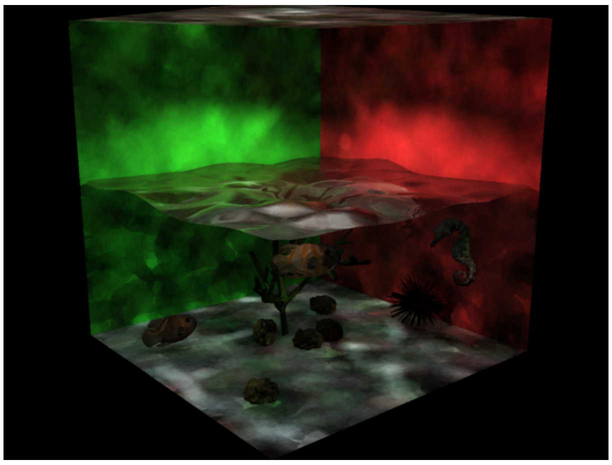
Figure 2: Suivi de chemins par méthode de Monte Carlo: la qualité du résultat est proportionnelle à la racine carrée du nombre de rayons envoyés.

nombreux rayons doivent être tracés pour regrouper les luminances incidentes pour chaque point visible, rendant le regroupement final très coûteux.

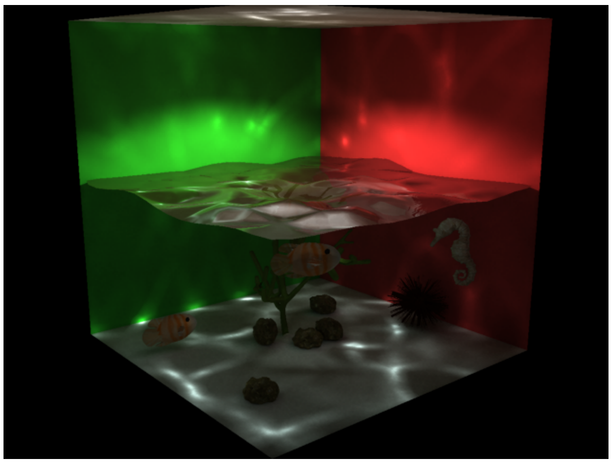
Plusieurs méthodes telles que [HHS05] ont été proposées pour accélérer cette phase de regroupement final. Toutefois, la méthode la plus efficace est l'algorithme de cache d'éclairement.

## Cache d'éclairement

L'algorithme de cache d'éclairement, ou *irradiance caching*, est la base du logiciel de simulation d'éclairage Radiance [War94]. Cet algorithme se base sur le principe suivant : l'éclairement indirect varie graduellement sur une surface donnée [WRC88]. Cet algorithme est donc basé sur un échantillonnage épars et une interpolation spatiale de l'éclairement indirect : l'éclairage indirect est calculé précisément en un certain nombre de points visibles par intégration de Monte-Carlo. Ces valeurs appelées *enregistrements*



(a) Visualisation directe de la carte de photons



(b) Regroupement final

Figure 3: Rendu de carte de photons par visualisation directe (a) et par regroupement final (b).

*d'éclairement* ou *irradiance records* sont stockées dans le *cache d'éclairement*, ou *irradiance cache*. Le calcul de l'éclairement indirect aux autres points visibles est calculé à moindre coût par extrapolation et interpolation des valeurs stockées dans les enregistrements. Notre travail étant basé sur cet algorithme, nous le présentons en détails dans les paragraphes suivants.

L'algorithme de cache d'éclairement a pour but le calcul de l'éclairement *diffus* indirect. Dans ce cas, la BRDF $f_r(\mathbf{p}, \omega_o, \omega_i)$ se limite à la réflectance diffuse de la surface $\rho_d(\mathbf{p})$ dans l'Equation 2:

$$\begin{aligned} L_o(\mathbf{p}, \omega_o) &= \rho_d(\mathbf{p}) \int_\Omega L_i(\mathbf{p}, \omega_i) \cos\theta_i \mathrm{d}\omega_i &(3)\\ &= \rho_d(\mathbf{p}) E_i(\mathbf{p}) &(4) \end{aligned}$$

où $E_i(\mathbf{p})$ est l'éclairement indirect au point $\mathbf{p}$. La luminance réfléchie ne dépendant plus de $\omega_o$, l'éclairement $E_i$ est suffisant pour représenter l'éclairement diffus indirect.

Comme indiqué précédemment, l'algorithme de cache d'éclairement est basé sur un échantillonnage épars et une interpolation de l'éclairement indirect. Toutefois, bien que l'éclairement indirect ne change que graduellement, le taux de changement de cet éclairement est dépendant de la géométrie environnante. C'est pourquoi Ward *et al.* [WRC88] proposent une fonction de pondération adaptative pour l'échantillonnage de l'éclairement indirect. La zone d'influence d'un enregistrement est calculée à l'aide de l'estimation d'un majorant du taux de changement de l'éclairement indirect. Pour un enregistrement $k$ situé en un point $\mathbf{p_k}$ de normale $\mathbf{n_k}$, le poids de $k$ en un point $\mathbf{p}$ de normale $\mathbf{n}$ est défini comme l'inverse de ce majorant du taux de changement:

$$w_k(\mathbf{p}) = \left( \frac{\|\mathbf{p} - \mathbf{p_k}\|}{R_k} + \sqrt{1 - \mathbf{n} \cdot \mathbf{n_k}} \right)^{-1} \tag{5}$$

où $R_k$ est la moyenne harmonique des distances des objets visibles depuis le point $\mathbf{p_k}$. L'ensemble des enregistrements contribuant à l'estimation de l'éclairement indirect au point $\mathbf{p}$ est:

$$S(\mathbf{p}) = \{\text{enregistrement } k \ \ / \ \ w_k(\mathbf{p}) \geq 1/a\} \tag{6}$$

où $a$ est une valeur de précision donnée par l'utilisateur. Sur une surface plane, chaque enregistrement $k$ contribue à l'éclairement indirect des points situés dans un cercle centré en $\mathbf{p_k}$ (Figure 4). La taille de la zone d'influence d'un enregistrement est directement liée à $R_k$: plus la géométrie environnante est proche, plus la zone sera petite.

Bien que le changement d'éclairement indirect soit faible à l'intérieur de la zone d'influence d'un enregistrement, ce changement ne peut pas être négligé. Ward *et al.* [WH92] proposent les gradients d'éclairement ou *irradiance gradients*. Ces gradients expriment comment l'éclairement diffus indirect change à l'intérieur de la zone d'influence. La contribution de l'enregistrement $k$ à l'éclairement diffus indirect du point $\mathbf{p}$ est donnée par:

$$E_k(\mathbf{p}) = E_k(1 + (\mathbf{n_k} \times \mathbf{n})\nabla_r + (\mathbf{p} - \mathbf{p_k})\nabla_t) \tag{7}$$

où $\nabla_r$ et $\nabla_t$ sont les gradients représentant le changement d'éclairement respectivement en fonction de la rotation et de la translation.

Dans le cas général, plusieurs enregistrements contribuent à l'estimation de l'éclairement diffus indirect en un point $\mathbf{p}$. Ward *et al.* [WRC88] définissent la formule d'interpolation des éclairements indirects comme suit:

$$E(\mathbf{p}) = \frac{\sum_{k \in S(\mathbf{p})} E_k(\mathbf{p}) w_k(\mathbf{p})}{\sum_{k \in S(\mathbf{p})} w_k(\mathbf{p})} \tag{8}$$

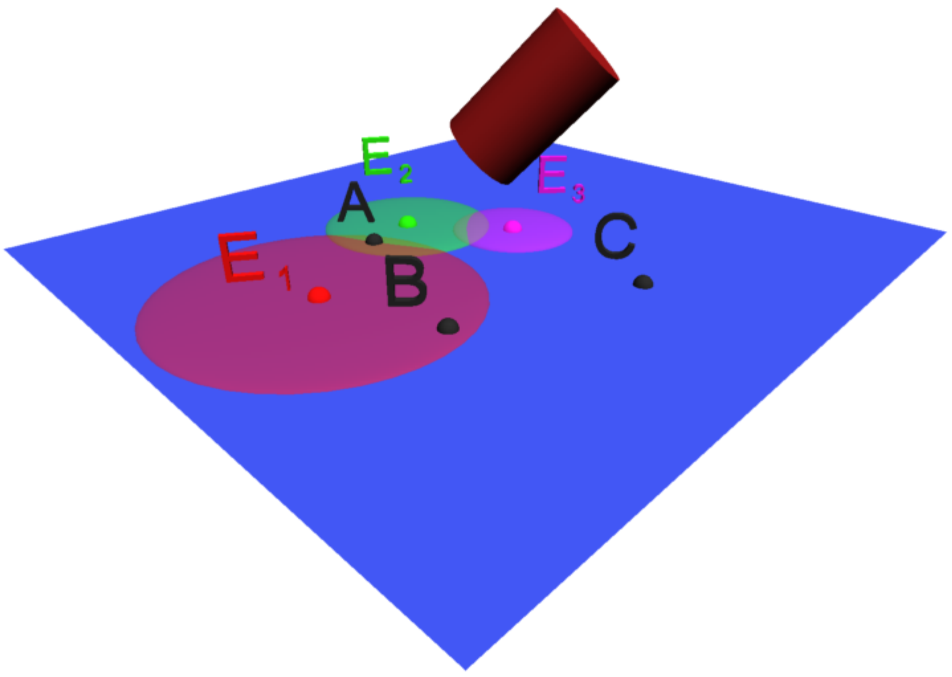


Figure 4: L'éclairement indirect dans le voisinage des enregistrements $E_1$, $E_2$, $E_3$ peut être extrapolé et interpolé en utilisant les valeurs des enregistrements. Dans l'algorithme de cache d'éclairement, trois situations peuvent se produire: le point A se trouve dans les zone d'influence de $E_1$ et $E_2$. Dans ce cas, l'éclairement indirect est interpolé en utilisant les valeurs de ces enregistrements. Le point B se trouve uniquement dans la zone d'influence de $E_1$; son éclairement indirect est uniquement extrapolé en utilisant la valeur de $E_1$. Aucun enregistrement ne peut contribuer à l'éclairement indirect du point C. Un nouvel enregistrement sera donc créé en C. A noter que la taille des zones d'influence de chaque enregistrement est inversement proportionnelle à la distance les séparant du cylindre voisin.

L'algorithme de cache d'éclairement est décrit dans l'Algorithme 8. Le cache d'éclairement est examiné pour chaque point visible $\mathbf{p}$. La somme des poids des enregistrements voisins est comparée à la valeur de précision $a$ donnée par l'utilisateur (Equation 9). Si les contributions des enregistrements existants ne sont pas suffisantes, un nouvel enregistrement est créé et stocké dans le cache. Dans le cas contraire, l'éclairement diffus indirect est obtenu par simple interpolation des valeurs des enregistrements voisins :

$$\sum_{k \in S(\mathbf{p})} w_k(\mathbf{p}) \begin{cases} < a & \text{Nouvel enregistrement requis} \\ \geq a & \text{Interpolation} \end{cases} \tag{9}$$

Dans des scènes dynamiques, l'approche de base consiste à calculer un nouveau cache pour chaque image d'une animation. Toutefois, l'algorithme de cache d'éclairement est

**Algorithm 8** Algorithme de cache d'éclairement

```
for all point visible p do
  Examen du cache d'éclairement
  if les contributions sont suffisantes then
    Estimation de l'éclairement indirect par interpolation
  else
    Calcul d'un enregistrement au point p
    Stockage dans le cache
    L'éclairement indirect au point p est la valeur de l'enregistrement calculé.
  end if
end for
```

basé sur l'extrapolation et l'interpolation de l'éclairement diffus indirect. Par conséquent, les valeurs obtenues ne sont pas exactes. Lors de la génération d'un nouveau cache pour chaque image, les erreurs sont rendues visibles sous forme de clignotements. Plusieurs méthodes décrites dans la version anglaise de cette thèse ont été proposées pour réduire ces clignotements, mais mettent généralement en œuvre des structures de données complexes et encombrantes en termes de mémoire.

## Cache de luminance

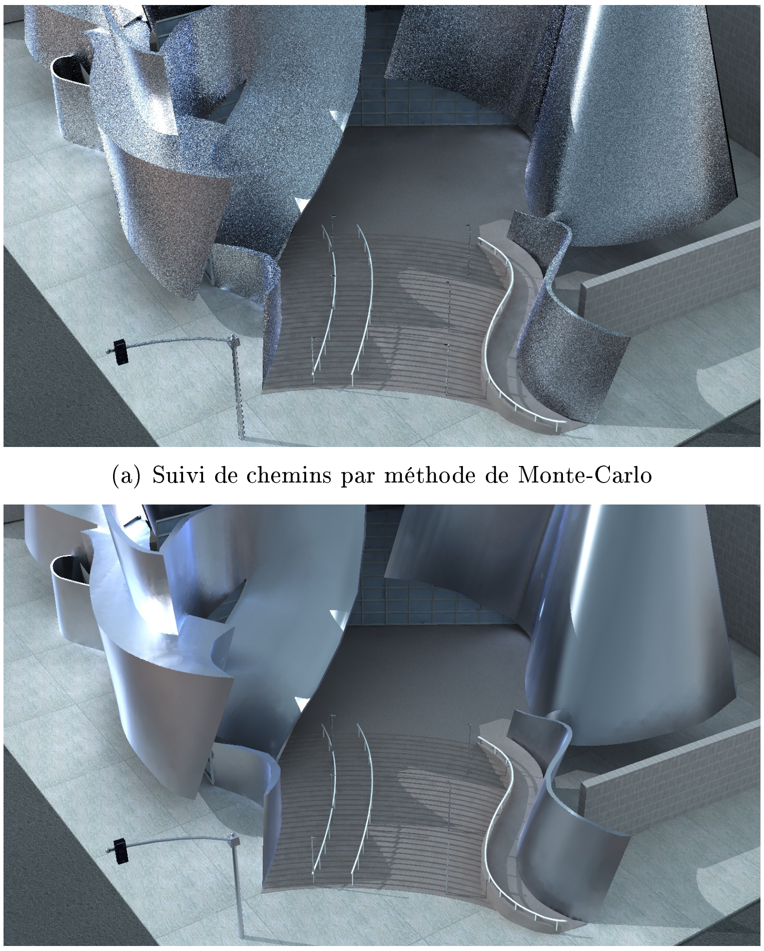
Comme décrit précédemment l'algorithme de cache d'éclairement a pour but le calcul de l'éclairement diffus indirect, et ne prend donc pas en compte la provenance des luminances incidentes. De manière générale, l'éclairement indirect non diffus est calculé à l'aide de suivi de chemins par méthode de Monte-Carlo. Křivánek *et al.* [KGPB05, Kři05] ont introduit l'algorithme de cache de luminance, ou *radiance caching*. L'évaluation de la BRDF de surfaces non diffuses requiert la connaissance de la provenance des luminances incidentes. C'est pourquoi l'algorithme de cache de luminance projette la fonction de luminance incidente dans une base de fonctions hémisphériques, appelée harmoniques hémisphériques [GKPB04] (Voir la contribution "harmoniques hémisphériques"). Cette projection permet de représenter la fonction de luminance incidente à l'aide d'un faible nombre de coefficients (typiquement 100). L'un des avantages de cette technique provient de l'orthogonalité des fonctions de base : si la BRDF et la fonction de luminance incidente sont représentées à l'aide d'harmoniques hémisphériques, l'Equation 2 se réduit à:

$$L_o(\mathbf{p},\omega_o) = \int_\Omega f_r(\mathbf{p},\omega_o,\omega_i)L_i(\mathbf{p},\omega_i)\cos\theta_i \mathrm{d}\omega_i \tag{10}$$

$$= \sum_{k=0}^{n-1} L_i^k f_r^k \tag{11}$$

où $L_i^k$ et $f_r^k$ sont respectivement le $k^{\text{ème}}$ coefficient de projection de la fonction de luminance incidente et de la BRDF, et $n$ est le nombre de coefficients utilisés pour la projection. Křivánek *et al.* proposent de réutiliser la fonction de pondération et la méthode d'interpolation utilisées dans [WRC88].

A l'instar de [WH92], Křivánek *et al.* proposent des gradients de luminance [KGPB05, KGBP05]. Ces gradients représentent comment chaque coefficient de projection change en fonction de la translation. Dans le contexte du cache de luminance, le gradient de rotation n'est plus utilisé : le changement de repère local entre le point d'interpolation et l'emplacement de l'enregistrement est calculé à l'aide de matrices de rotation directement appliquées aux coefficients de projection (voir "harmoniques hémisphériques"). Comme le montre la Figure 5, cette méthode permet d'augmenter significativement la


(a) Suivi de chemins par méthode de Monte-Carlo

(b) Cache de luminance

Figure 5: Un modèle du Walt Disney Hall de Los Angeles (Californie, USA) rendu à l'aide de suivi de chemins Monte-Carlo (a) et de l'algorithme de cache de luminance en des temps équivalents. Le cache de luminance augmente la qualité de rendu en éliminant le bruit par pixel du suivi de chemins (Images: Jaroslav Křivánek )

qualité de rendu par rapport au suivi de chemins classique. Toutefois, comme toute base de fonctions, la base des harmoniques hémisphériques est uniquement adaptée à la représentation de fonctions de basse fréquence. De plus, le principe de cohérence spatiale de l'éclairage indirect est uniquement valable dans le cas de surfaces diffuses ou modérément spéculaires. C'est pourquoi l'algorithme de cache de luminance ne peut être utilisé que dans le cas de surfaces modérément spéculaires, dites surfaces rugueuses ou *glossy*. Pour des surfaces hautement spéculaires telles que des miroirs, le suivi de chemins par méthode de Monte-Carlo avec échantillonnage d'importance est non seulement plus précis, mais également moins coûteux.

Les méthodes présentées ci-dessus ont pour but un calcul d'illumination globale efficace utilisant le processeur principal, ou *Central Processing Unit (CPU)*. Toutefois, les processeurs graphiques deviennent de plus en plus puissants en termes de vitesse de calcul, de mémoire et de programmabilité. La section suivante présente les principes fondamentaux et la programmation des processeurs graphiques.

## Processeurs Graphiques

Les processeurs graphiques, ou *GPUs* (Graphics Processing Units) sont conçus dans le but de décharger le processeur central des tâches liées au calcul et à l'affichage des images de synthèse. Nous présentons ici la structure et la programmation des GPUs, notions utilisées tout au long de cette thèse.

Le fonctionnement des GPUs est basé sur l'algorithme de remplissage de polygones, ou *rasterization* (Figure 7 et Algorithme 9). Cet algorithme est divisé en 4 parties. Lors de la première étape, chaque polygone de la scène est transmis au processeur de sommets (*vertex processor*). Ce processeur projette chaque sommet de chaque polygone de la scène dans le plan image. Le polygone projeté est alors envoyé au *rasterizer*, qui génère un certain nombre de *fragment* correspondant aux pixels couverts par le polygone. Pour chaque fragment, une interpolation de Gouraud (Figure 7) est appliquée aux attributs des sommets (normale, couleur, coordonnées de texture, ...).

Lors de la troisième étape de l'algorithme les fragments sont transmis au processeur de fragments (*fragment processor*). Ce processeur effectue des opération au niveau des pixels, telles que le placage de textures. Durant le processus de *rasterization*, des fragments de polygones distincts peuvent se recouvrir, i.e. correspondre au même pixel de l'image finale. La dernière partie de l'algorithme consiste à ne générer qu'une seule valeur finale par pixel. Cette valeur peut être obtenue en ne gardant que la valeur la plus proche (tampon de profondeur, ou *Z-Buffer*), ou encore en mélangeant les différentes valeurs pour simuler un effet de transparence (mélange alpha, ou *alpha-blending*).

Comme le montre l'Algorithme 9, chaque polygone ainsi chaque fragment est traité de manière indépendante. C'est pourquoi cet algorithme peut facilement être implémenté en parallèle. Les processeurs graphiques actuels incluent 8 processeurs de sommets et jusqu'à 48 processeurs de fragments. De plus, les processeurs graphiques actuels permettent à l'utilisateur de programmer les fonctionnalités des processeurs de sommets et de fragments. Ces programmes peuvent être écrits en langages similaires au langage

**Algorithm 9** *Rasterization*

```
for all polygone de la scene do
  // Processeur de sommets
  for all sommet du polygone do
    Projeter le sommet dans le plan image
  end for
  // Rasterizer
  for all pixel couvert par le polygone do
    Interpolation de Gouraud des attributs du sommet: position, normale, coordon-
    nées de texture ...
    Générer un fragment
  end for
  // Processeur de fragments
  for all fragment généré par le rasterizer do
    Effectuer les opérations par pixel
  end for
  // Mélange/Sélection des fragments
  for all fragment généré par le processeur de fragment do
    Mélange alpha
    Test de profondeur
    ...
  end for
end for
```

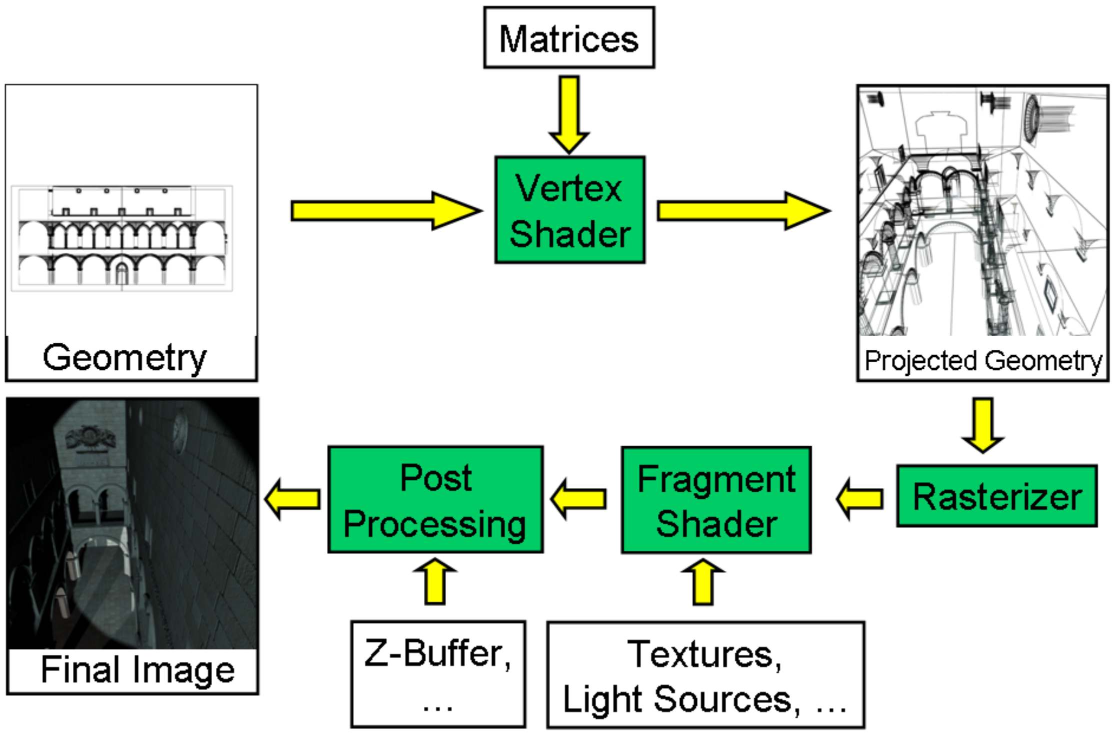


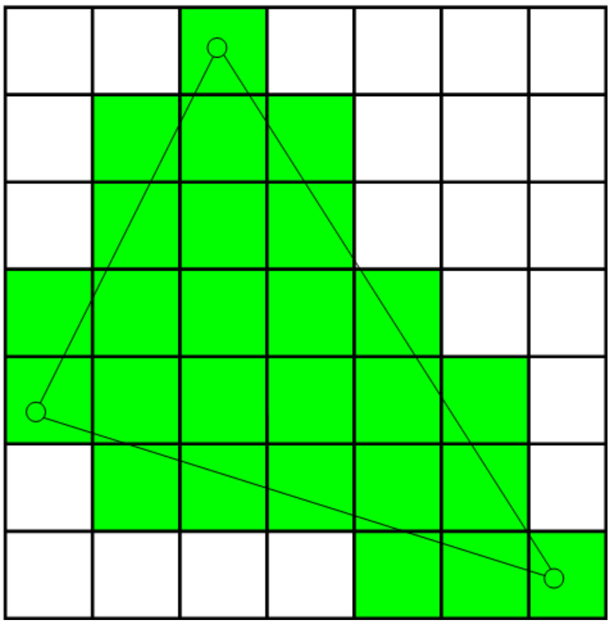
Figure 6: Le pipeline graphique

C, tels que *OpenGL Shading Language (GLSL)* [KBR04], *Cg* [nC05] et *DirectX High Level Shading Language* [Lau05]. Cette généricité permet de mettre la puissance de calcul des processeurs graphiques au service d'algorithmes complexes tels que les calculs d'illumination globale.

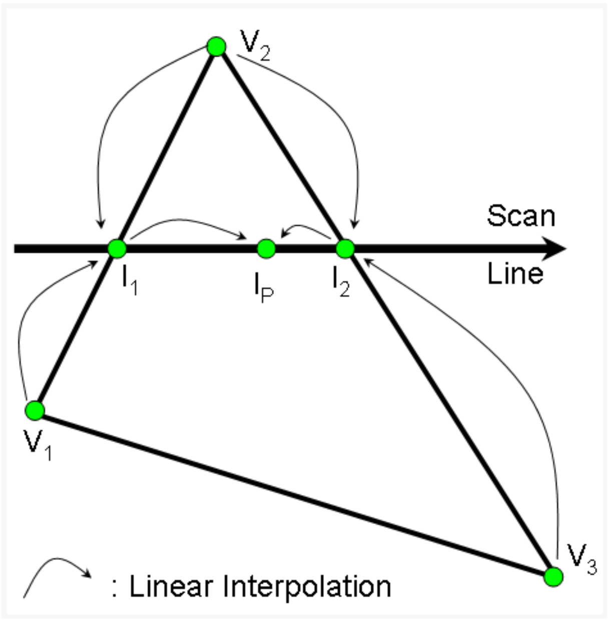
## Notre approche

Dans cette thèse, nous nous concentrons sur le calcul rapide de solutions d'illumination globale de haute qualité. Nous souhaitons prendre en charge des scènes dynamiques composées d'objets non diffus. De surcroît, nos utilisons les performances des cartes graphiques afin d'accélérer les calculs.

Dans le contexte de l'algorithme de cache d'éclairement, nous proposons tout d'abord un ensemble de fonctions de base dédiées à la représentation de fonctions hémisphériques ("harmoniques hémisphériques"). Prenant comme base les harmoniques sphériques, nos fonctions de base ne sont définies que sur l'hémisphère supérieure. Par conséquent, elle sont mieux adaptées au cache de luminance en termes de qualité et de compacité



(a) Rasterization



(b) Interpolation de Gouraud

Figure 7: Le *rasterizer* génère des fragments correspondant aux pixels couverts par le triangle (a). Pour chaque fragment, les attributs du sommet sont interpolés par interpolation de Gouraud (b).

de représentation. Les harmoniques sphériques et hémisphériques étant très similaires, ces bases partagent deux avantages : des formules de recurrence pour une évaluation rapide, et des matrices de rotation définies analytiquement. Ces deux avantages font des harmoniques hémisphériques des fonction très adaptées aux calculs sur l'hémisphère réalisés dans le cadre des algorithmes d'illumination globale.

Les algorithmes de cache d'éclairement et de luminance sont basés sur le lancer de rayons pour le calcul des enregistrements, et sur une structure de données hiérarchique pour le stockage du cache. Ces deux méthodes étant inadaptées à la programmation GPU, nous proposons l'algorithme de splatting de luminance (*radiance cache splatting*) pour un rendu rapide à l'aide du GPU. Nous reformulons les algorithmes de cache d'éclairement et de luminance en termes d'opérations facilement réalisables par les cartes graphiques actuelles : *rasterization*, mélange alpha, tampon de profondeur et calculs par pixel. Sans introduire de perte de qualité, notre méthode s'avère être beaucoup plus rapide que le logiciel de référence Radiance [War94]. Dans des scènes simples, notre méthode permet de calculer et d'afficher une solution d'illumination globale interactivement.

Les méthodes détaillées précédemment ont pour but le calcul d'illumination globale dans des scènes statiques. Nous étendons le principe d'interpolation proposé par Ward *et al.* [WRC88] au domaine temporel. Nous définissons une fonction de pondération temporelle ainsi que des gradients d'éclairement et de luminance temporels. Sans introduire de structure de données spécifique, notre méthode réutilise des enregistrements dans plusieurs images de manière adaptative : la durée de vie d'un enregistrement est déterminée par le taux de changement de la luminance incidente à sa position. De plus, nous ne séparons pas le calcul entre illuminations statique et dynamique. La généricité de notre méthode permet donc d'effectuer des calculs d'illumination globale de haute qualité quelle que soit la dynamique de l'éclairage.

Notre dernière contribution est une méthode pour le rendu efficace de cartes de photons. Alors que les travaux existants sont généralement basés sur le regroupement final pour un rendu de qualité, nous proposons de construire des enregistrements d'éclairement uniquement à l'aide des photons contenus dans la carte de photons (i.e. sans tracer de rayons supplémentaires). Cette approche réduit significativement les coûts de rendu par cartes de photons tout en préservant la qualité de l'image. Ce travail a été réalisé par Jonathan Brouillat durant son stage de Master [Bro06].

## Harmoniques Hémisphériques

### Motivations

Le calcul d'illumination globale nécessite une représentation précise des interactions lumière/matière. Ces interactions peuvent être représentées par des fonctions définies sur l'hémisphère : la fonction de luminance incidente en un point est une fonction hémisphérique. La BRDF d'un matériau est une fonction définie sur le produit cartésien de deux hémisphères : les hémisphères des directions incidentes et réfléchies. La luminance réfléchie en un point est le produit scalaire de la luminance incidente et de la BRDF.

Par conséquent, une représentation précise et compacte de ces fonctions est nécessaire pour un calcul d'illumination globale rapide et de qualité.

Bien que de nombreux travaux aient consisté en la représentation de fonction hémisphériques à l'aide d'harmoniques sphériques, nous proposons une nouvelle base de fonctions directement adaptée à l'hémisphère. Après avoir introduit les notions de polynômes orthogonaux et de transformée de Fourier généralisée nécessaires à l'approche de ce travail, nous présentons notre nouvelle base de fonctions : les harmoniques hémisphériques. Les changements de repère étant fréquemment utilisés en illumination globale, nous proposons des méthodes simples de rotation de fonctions projetées dans la base des harmoniques hémisphériques. De manière similaire aux harmoniques sphériques, notre base de fonctions est orthogonale. Par conséquent, le produit scalaire de deux fonctions projetées se réduit au produit scalaire des vecteur de coefficients de projection. Nous appliquons tout d'abord notre base à la représentation de BRDFs et de fonction de luminance incidente, et démontrons son utilisation dans le cadre du rendu temps-réel à l'aide de cartes d'environnement. Les harmoniques hémisphériques sont également appliquées à l'algorithme de cache de luminance [Křı05].

## Polynômes Orthogonaux

### Notions de base

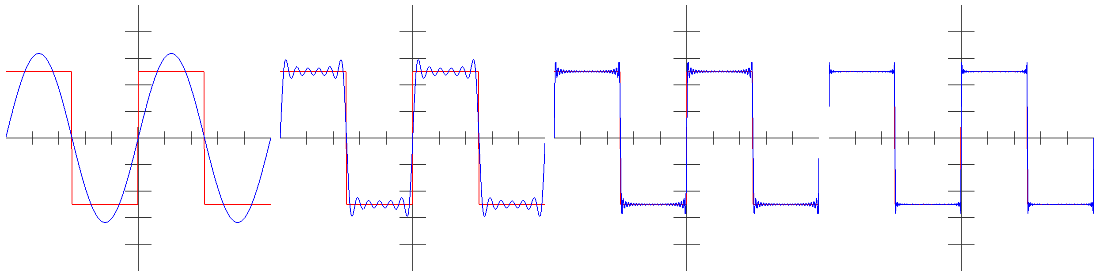
Un ensemble de polynômes $\{p_l(x)\}_{l\geq 0}$ est dit orthogonal sur un intervalle $[a, b]$ si $p_l(x)$ est un polynôme de degré $l$, et pour $n, m \geq 0$

$$\int_a^b w(x)p_n(x)p_m(x)\mathrm{d}x = \delta_{nm}c_n \tag{12}$$

où $\delta_{nm}$ est le symbole de Knonecker (0 ou 1 selon si $n \neq m$ ou $n = m$), et $w(x)$ est une fonction prenant des valeurs positives. Si $c_n = 1$ l'ensemble de polynômes forme une base orthonormale [Sze75]. Toute fonction integrable $f(x)$ peut alors être développée en séries de Fourier généralisées:

$$f(x) \approx \sum_{n=0}^{\infty} f_n p_n(x) \qquad \text{avec} \qquad f_n = \int_a^b f(x)p_n(x)w(x)\mathrm{d}x \tag{13}$$

Par conséquent, toute fonction intégrable peut être représentée par un ensemble de coefficients et une base orthonormale. En pratique, une fonction est représentée à l'aide d'un nombre fini de coefficients. Toutefois, à l'inverse des polynômes, la fonction $f$ peut contenir des discontinuités, rendant la représentation par séries de Fourier approximative : le phénomène de Gibbs peut alors apparaître sous forme d'oscillations autour de la discontinuité de la fonction (Figure 8). Bien que l'amplitude de ces oscillations réduise avec le nombre de coefficients, elles ne peuvent être totalement retirées. La présentation des résultats illustre ce phénomène dans le contexte des calculs d'illumination.



(a) Ordre 1 (b) Ordre 10 (c) Ordre 50 (d) Ordre 150

Figure 8: Représentation d'un signal carré (en rouge) à l'aide de séries de Fourier (en bleu). Les oscillations autour des discontinuités (phénomène de Gibbs) sont réduite en augmentant l'ordre de représentation (i.e. le nombre de coefficients), mais ne peuvent pas être supprimées.

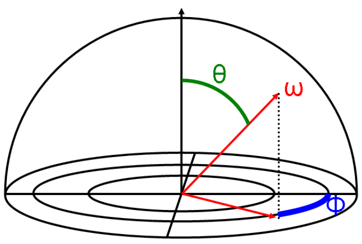


Figure 9: Coordonnées sphériques : toute direction $\omega$ peut être représentée à l'aide de ses angles polaire ($\theta$) et azimutal ($\phi$).

## Harmoniques Sphériques (SH)

Les harmoniques sphériques réelles sont des fonctions de base orthogonales définies sur la sphère (Figure 9). Ce fonctions sont définies comme suit:

$$Y_l^m(\theta,\phi) = \begin{cases} \sqrt{2}K_l^m \cos(m\phi)P_l^m(\cos\theta) & \text{si } m > 0 \\ \sqrt{2}K_l^m \sin(-m\phi)P_l^{-m}(\cos\theta) & \text{si } m < 0 \\ K_l^0 P_l^0(\cos\theta) & \text{si } m = 0 \end{cases} \tag{14}$$

où les $P_l^m$ sont les polynômes de Legendre associés, et $K_l^m$ est la valeur de normalisation des harmoniques sphériques :

$$K_l^m = \sqrt{\frac{(2l+1)(l-|m|)!}{4\pi(l+|m|)!}} \tag{15}$$

La Figure 10 représente les premières harmoniques sphériques.

Toute fonction $f$ intégrable sur la sphère peut être projetée dans la base des harmoniques sphériques:

$$f(\omega) \approx \sum_{l=0}^{\infty}\sum_{m=-l}^{l} f_l^m Y_l^m(\omega) \quad \text{avec} \quad f_l^m = \int_\Omega f(\omega) Y_l^m(\omega) \mathrm{d}\omega \tag{16}$$

Une propriété importante des harmoniques sphériques est leur orthogonalité. Soient $f$ et $g$ deux fonctions définies et intégrables sur la sphère, $f_l^m$ et $g_l^m$ étant leur coefficients de projection respectifs. Le produit scalaire de $f$ et $g$ est :

$$< f, g > = \int_\Omega f(\omega) g(\omega) \mathrm{d}\omega \tag{17}$$

$$\approx \int_\Omega \left( \sum_{l=0}^{\infty}\sum_{m=-l}^{l} f_l^m Y_l^m(\omega) \right) \left( \sum_{l'=0}^{\infty}\sum_{m'=-l'}^{l'} g_{l'}^{m'} Y_{l'}^{m'}(\omega) \right) \mathrm{d}\omega \tag{18}$$

La propriété d'orthogonalité des harmoniques sphériques est :

$$\int_\Omega Y_l^m(\omega) Y_{l'}^{m'}(\omega) \mathrm{d}\omega = \delta_{ll'} \delta_{mm'} \tag{19}$$

Par conséquent, les termes $Y_l^m(\omega)Y_{l'}^{m'}(\omega)$ avec $m \neq m'$ ou $l \neq l'$ s'annulent :

$$< f, g > \approx \sum_{l=0}^{\infty}\sum_{m=-l}^{l} \left( \int_\Omega f_l^m Y_l^m(\omega) \sum_{l'=0}^{\infty}\sum_{m'=-l'}^{l'} g_{l'}^{m'} Y_{l'}^{m'}(\omega) \right) \mathrm{d}\omega \tag{20}$$

$$\approx \sum_{l=0}^{\infty}\sum_{m=-l}^{l} f_l^m g_l^m \tag{21}$$

Le produit scalaire de deux fonctions projetées dans la base des harmoniques sphériques se réduit donc au produit scalaire des vecteurs de coefficients de projection. Cette propriété est directement utilisée en illumination globale : en représentant la fonction de luminance incidente et la BRDF à l'aide d'harmoniques sphériques, le calcul de la luminance réfléchie se réduit au produit scalaire des coefficients et peut donc être effectué efficacement.

Lors de calculs d'illumination par cartes d'environnement ou par cache de luminance, la fonction de luminance incidente est généralement définie dans un repère différent de celui du point auquel nous voulons calculer une luminance réfléchie. Dans ce cas il est nécessaire d'effectuer un changement de repère, i.e. une rotation. Pour une fonction projetée dans la base des harmoniques sphériques, le changement de repère peut être réalisé en appliquant une matrice de rotation au vecteur de coefficients de projection. Ces matrices de rotation peuvent être calculées par récursion à partir d'une matrice de rotation $3 \times 3$ classique [IR96, IR98, CIGR99, BFB97].

Toutefois, les harmoniques sphériques ne sont pas adaptées à la représentation de fonctions hémisphériques. En particulier, le phénomène de Gibbs peut apparaître au niveau de l'équateur dans la représentation en harmoniques sphériques. De plus, la capacité de représentation de ces fonctions de base n'est exploitée que sur la moitié du domaine de définition. C'est pourquoi nous proposons une nouvelle base de fonctions hémisphériques partageant les qualités des harmoniques sphériques tout en étant adaptées à l'hémisphère.

## Une nouvelle base de fonction hémisphériques

Les harmoniques sphériques étant définies sur la sphère, l'idée de base de notre méthode est de décaler le domaine de définition des harmoniques sphériques afin de se réduire à l'hémisphère. Par conséquent nous utilisons des polynômes de Legendre décalés.

### Polynômes de Legendre Associés Décalés

A l'aide une transformation linéaire de $x$ à $2x-1$, nous obtenons les polynômes de Legendre associés décalés sur l'intervalle $x \in [0,1]$ :

$$\widetilde{P}_l^m(x) \quad = \quad P_l^m(2x-1) \tag{22}$$

En remplaçant $x$ par $\cos\theta$, nous obtenons des polynômes $\{\widetilde{P}_l^m(\cos\theta)\}$ définis sur l'intervalle angulaire $\theta \in [0, \frac{\pi}{2}]$ correspondant aux valeurs prises par l'angle polaire sur l'hémisphère :

$$\widetilde{P}_l^m(\cos\theta) \quad = \quad P_l^m(2\cos\theta - 1) \text{ avec } \theta \in [0, \frac{\pi}{2}]. \tag{23}$$

Ces polynômes décalés restent orthogonaux, mais leur valeur de normalisation change :

$$\begin{aligned}\int_0^1 \widetilde{P}_l^m(x)\widetilde{P}_{l'}^m(x)dx \quad &= \quad \int_0^1 P_l^m(2x-1)P_{l'}^m(2x-1)\mathrm{d}x \\ &= \quad \frac{(m+l)!}{(2l+1)(l-m)!}\delta_{ll'} \end{aligned}\tag{24}$$

### Harmoniques Hémisphériques

Les harmoniques sphériques basées sur les polynômes de Legendre associés, nous définissons les harmoniques hémisphériques à l'aide des polynômes de Legendre associés décalés (Figure 10) :

$$H_l^m(\theta,\phi) = \begin{cases} \sqrt{2}\widetilde{K}_l^m \cos(m\phi)\widetilde{P}_l^m(\cos\theta) & \text{si } m > 0 \\ \sqrt{2}\widetilde{K}_l^m \sin(-m\phi)\widetilde{P}_l^{-m}(\cos\theta) & \text{si } m < 0 \\ \widetilde{K}_l^0\widetilde{P}_l^0(\cos\theta) & \text{si } m = 0 \end{cases} \tag{25}$$

avec la valeur de normalisation :

$$\widetilde{K}_l^m = \sqrt{\frac{(2l+1)(l-|m|)!}{2\pi(l+|m|)!}} \tag{26}$$

Les harmoniques hémisphériques sont orthogonale sur le domaine $[0, \frac{\pi}{2}] \times [0, 2\pi)$ par rapport à $l$ et $m$.

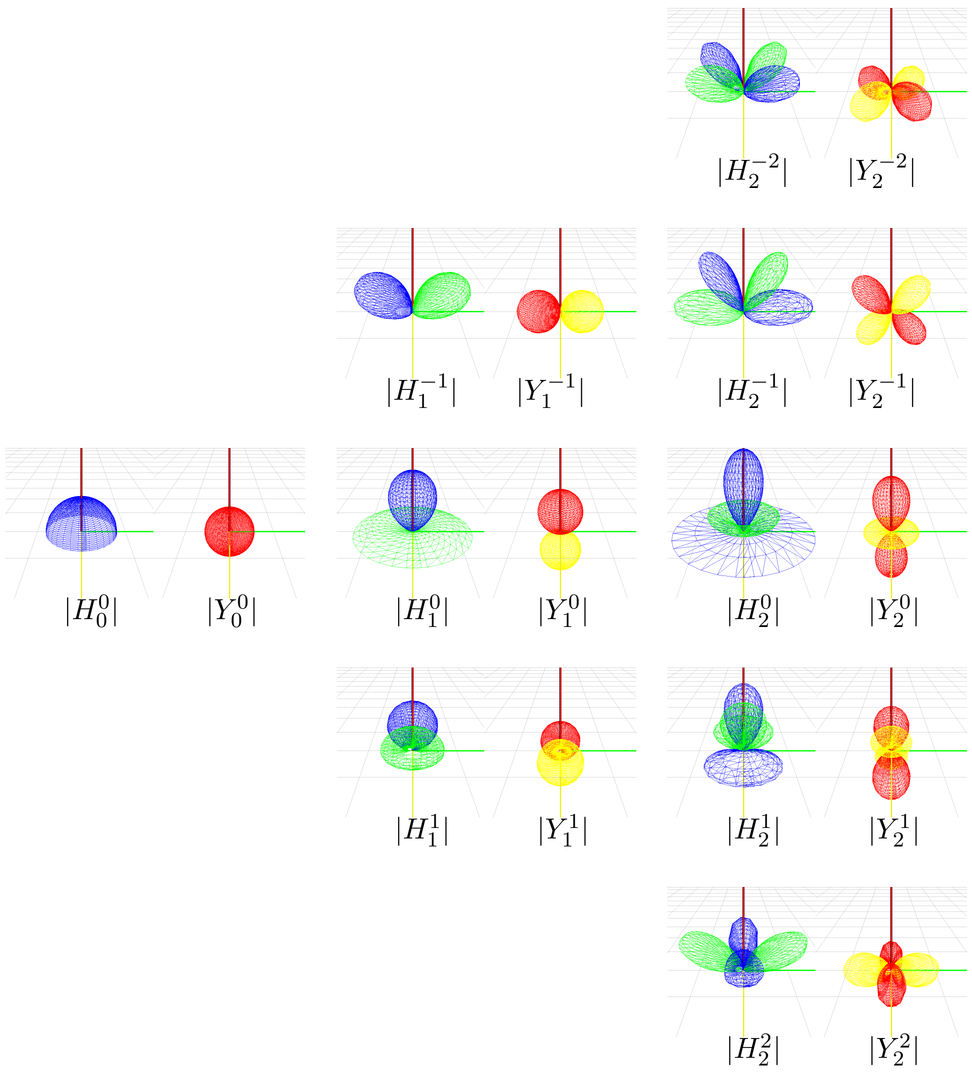


Figure 10: Harmoniques sphériques et hémisphériques de degré $l$ entre 0 et 2.

Comme indiqué précédemment, l'un des intérêts des harmoniques sphériques est l'existence de matrices de rotation opérant directement sur les vecteurs de coefficients. Ces rotations sont intensivement utilisées lors de calculs de changement de repère, typiquement dans l'algorithme de rendu par cache de luminance [Kři05].

## Rotation des Harmoniques Hémisphériques

Dans le contexte de rendu réaliste à l'aide d'harmoniques hémisphériques, il est fréquent de devoir pivoter une fonction de luminance incidente afin de l'exprimer dans un autre repère. Nous proposons ici trois méthodes de rotation de fonctions projetées en harmoniques hémisphériques : par conversion en harmoniques sphériques, en utilisant directement les matrices de rotation des harmoniques sphériques, et en précalculant les matrices de rotation.

### Matrice de conversion SH/HSH

Cette première méthode de rotation repose sur la conversion des coefficients d'harmoniques hémisphériques en harmoniques sphériques et vice-versa. Ainsi, la rotation peut être effectuée en convertissant les coefficients HSH en SH, puis en appliquant une matrice de rotation SH. Les coefficients de projection HSH de la fonction pivotée sont ensuite obtenus par conversion SH vers HSH. Nous définissons une matrice de changement de base $\mathbf{C}$ comme suit :

$$\mathbf{C}_{l,l'}^{m,m'} = \int_0^{2\pi}\int_0^{\pi/2} H_l^m(\theta,\phi) Y_{l'}^{m'}(\theta,\phi)\sin\theta \mathrm{d}\theta\mathrm{d}\phi \tag{27}$$

Les harmoniques sphériques et hémisphériques étant orthogonales par rapport à $m$, nous obtenons :

$$\begin{aligned}\mathbf{C}_{l,l}^{m,m'} &= \int_0^{2\pi}\int_0^{\pi/2} H_l^m(\theta,\phi)Y_l^{m'}(\theta,\phi)\sin\theta\mathrm{d}\theta\mathrm{d}\phi && (28)\\ &= \int_0^{2\pi}\int_0^{\pi/2}\widetilde{K}_l^m K_l^{m'}\Phi(m\phi)\Phi(m'\phi)\widetilde{P}_l^m(\cos\theta)P_l^m(\cos\theta)\sin\theta\mathrm{d}\theta\mathrm{d}\phi && (29)\\ &= \widetilde{K}_l^m K_l^{m'}\int_0^{2\pi}\Phi(m\phi)\Phi(m'\phi)\int_0^{\pi/2}\widetilde{P}_l^m(\cos\theta)P_l^m(\cos\theta)\sin\theta\mathrm{d}\theta\mathrm{d}\phi && (30)\end{aligned}$$

Étant donné que $\int_0^{\pi/2}\widetilde{P}_l^m(\cos\theta)P_l^m(\cos\theta)\sin\theta\mathrm{d}\theta$ ne dépend pas de $\phi$, les coefficients de $\mathbf{C}$ sont :

$$\mathbf{C}_{l,l}^{m,m'} = \widetilde{K}_l^m K_l^{m'}\int_0^{\pi/2}\widetilde{P}_l^m(\cos\theta)P_l^m(\cos\theta)\sin\theta\mathrm{d}\theta\int_0^{2\pi}\Phi(m\phi)\Phi(m'\phi)\mathrm{d}\phi \tag{31}$$

avec

$$\Phi(m\phi) = \begin{cases}\cos(m\phi) & \text{si } m > 0\\ \sin(-m\phi) & \text{si } m < 0\\ 1 & \text{si } m = 0\end{cases} \tag{32}$$

Toutefois,

$$\int_0^{2\pi}\cos(m\phi)\cos(m'\phi)\mathrm{d}\phi = \int_0^{2\pi}\sin(-m\phi)\sin(-m(\phi)\mathrm{d}\phi = 0 \text{ si } m \neq m' \tag{33}$$

Par conséquent, cette matrice ne contient des valeurs non nulles que pour $m = m'$, réduisant ainsi significativement de coût de transformation de SH vers HSH. Pour $l < 3$

et $l' < 3$ la matrice de conversion $\mathbf{C}_{3,3}$ est :

$$\begin{pmatrix}
0.71 & 0 & 0.41 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.60 & 0 & 0 & 0 & 0.19 & 0 & 0 & 0 \\
0.82 & 0 & 0.71 & 0 & 0 & 0 & 0.18 & 0 & 0 \\
0 & 0 & 0 & 0.60 & 0 & 0 & 0 & 0.19 & 0 \\
0 & 0 & 0 & 0 & 0.53 & 0 & 0 & 0 & 0 \\
0 & 0.77 & 0 & 0 & 0 & 0.51 & 0 & 0 & 0 \\
0.40 & 0 & 0.78 & 0 & 0 & 0 & 0.53 & 0 & 0 \\
0 & 0 & 0 & 0.77 & 0 & 0 & 0 & 0.51 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.53
\end{pmatrix}$$

La rotation d'une fonction projetée en harmoniques hémisphériques peut donc être réalisée en passant de HSH vers SH à l'aide $\mathbf{C}$, puis en appliquant une matrice de rotation SH, et enfin en repassant de SH vers HSH *via* $\mathbf{C}^T$. Afin de ne pas perdre en qualité durant la rotation, il est possible d'utiliser une matrice $\mathbf{C}$ non carrée, permettant d'utiliser plus de coefficients SH que de HSH. A noter que la disparition d'une partie de l'hémisphère sous l'horizon est automatiquement prise en compte par cette méthode (Figure 12(c)).

### Suppression de Digone

Les deux méthodes suivantes se basent sur le théorème de rotation d'Euler : toute rotation peut être décrite par de trois angles. Nous choisissons de représenter nos rotations à l'aide de la convention $ZYZ$, où $Z$ est l'axe vertical d'un repère orthogonal main droite.

La rotation autour de l'axe $Z$ est identique à celle des harmoniques sphériques [SAWG91b]. Toutefois, la rotation autour de $Y$ engendre une perte de données (Figure 12). Nous réalisons donc la rotation autour de $Y$ en deux étapes.

Dans la première étape nous supprimons le digone $I$ (Figure 11) qui disparaîtra sous l'équateur après la rotation d'angle $\beta$ autour de $Y$ (Figure 12(b)). Cette suppression est réalisée à l'aide d'une matrice $\mathbf{M}(\beta)$ définie comme suit :

$$\mathbf{M}_{l,l'}^{m,m'}(\beta) = \int_{H-I} H_l^m(\theta,\phi) H_{l'}^{m'}(\theta,\phi) \sin\theta \mathrm{d}\theta \mathrm{d}\phi \tag{34}$$

où $H - I$ est l'hémisphère privée du digone concerné. Cette matrice est dense dû à la mise à zéro brutale de la fonction dans le digone. Pour des raisons pratiques, nous précalculons des matrices pour un certain nombre d'angles de rotation, et les interpolons linéairement lors de l'exécution. Bien que cette méthode par suppression de digone entraîne une perte d'information, les résultats obtenus sont améliorés en comparaison d'une rotation sans suppression de digone (Figure 13).

Cette matrice de suppression de digone appliquée, nous pouvons calculer et appliquer la matrice de rotation autour de $Y$. Les harmoniques hémisphériques étant très similaires aux harmoniques sphériques, nous définissons tout d'abord les rotations HSH en fonction des matrices de rotations SH [IR96, IR98].

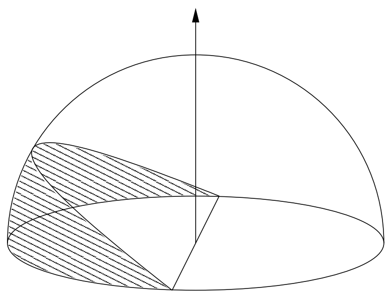


Figure 11: Hémisphère et un digone [Dut03]. Le digone est la partie hachurée de l'hémisphère.

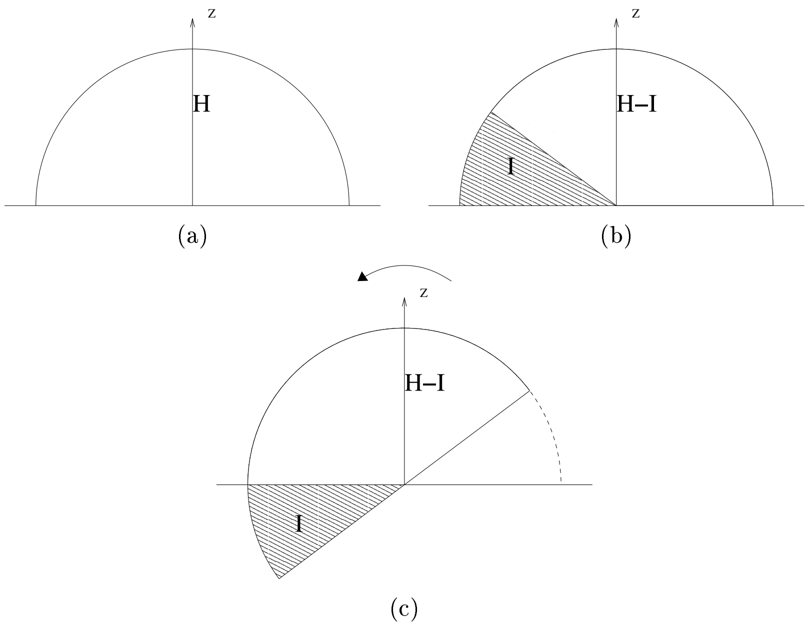


Figure 12: Rotation d'harmoniques hémisphériques autour de l'axe $Y$. Hémisphère avant rotation (a), digone supprimé (b), Hémisphère après rotation (c).

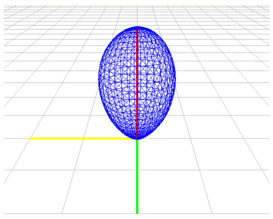
## Solution Analytique

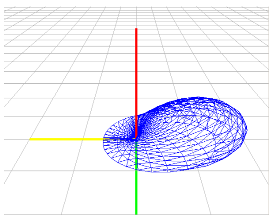
Appliquer une matrice de rotation SH d'un angle $\beta_{SH}$ autour de l'axe $Y$ à un vecteur de coefficients d'harmoniques sphériques $\{f_i\}$ est équivalent à:

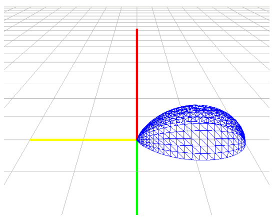
- étaler la fonction projetée $f_{HSH}$ sur la sphère entière (Equations 35, 36), ce qui nous donne $f_{SH}$ (Figure 14(b)).

$$f_{HSH}(\theta,\phi) = \sum_i f_i H_i(\theta,\phi) \quad [0,\frac{\pi}{2}] \times [0,2\pi] \to \mathbf{R} \tag{35}$$

$$f_{SH}(\theta,\phi) = \sum_i f_i Y_i(\theta,\phi) \quad [0,\pi] \times [0,2\pi] \to \mathbf{R} \tag{36}$$


(a) Fonction originelle


(b) Sans suppression de digone


(c) Rotation exacte

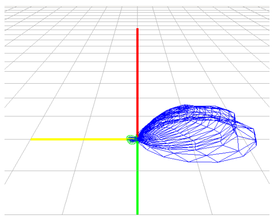

(d) Avec suppression de digone

Figure 13: Lors de la rotation d'une fonction hémisphérique (a), une partie de la fonction passe sous l'équateur (b). Si cette partie n'est pas retirée avant d'effectuer la rotation, la forme de la fonction est aplatie sur l'équateur (c). Au contraire, notre méthode de suppression de digone réduit les artefacts (d).

- appliquer une matrice de rotation SH $R^{SH}_{\beta_{SH}}$ à $f_{SH}$ (Figure 14(c))
- repasser $R^{SH}_{\beta_{SH}}(f_{SH})$ sur l'hémisphère supérieure seulement pour obtenir $R^{HSH}_{\beta_{HSH}}(f_{HSH})$ (Figure 14(d))

Notre but est de déterminer la relation entre $\beta_{SH}$ and $\beta_{HSH}$ : les domaines de définition des harmoniques sphériques et hémisphériques étant différents, $\beta_{SH} \neq \beta_{HSH}$. A l'aide de l'Equation 23, une rotation d'angle $\beta_{HSH}$ sur l'hémisphère est équivalente à une rotation d'angle $\beta_{HSH}$ (Equation 37):

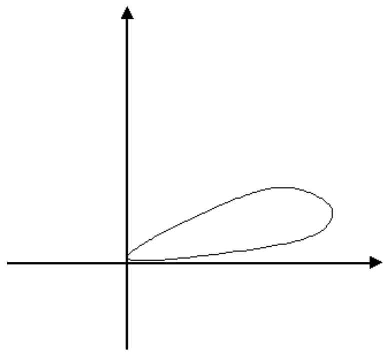
$$\beta_{SH} = \arccos(2\cos\beta_{HSH} - 1) \tag{37}$$

Cette équation démontre que les matrices de rotation SH peuvent être utilisées directement pour la rotation de fonctions HSH à la condition que la fonction soit nulle dans digone disparaissant sous l'équateur lors de la rotation.

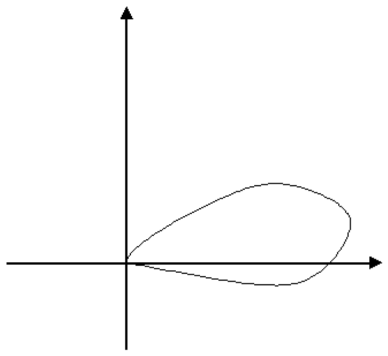
### Rotation Précalculée

La dernière méthode de rotation consiste à précalculer et insérer la rotation dans la matrice de suppression de digone. Les éléments de la matrice sont alors définis par :

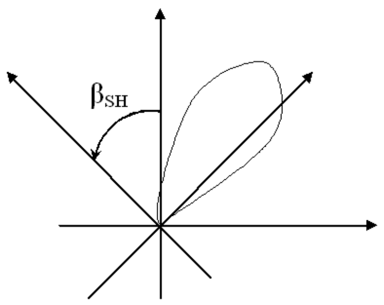
$$\mathbf{M_R}^{m,m'}_{l,l'}(\beta) = \int_{H-I} H^m_l(\theta,\phi) H^{m'}_{l'}(R_\beta(\theta,\phi)) \sin\theta \mathrm{d}\theta\mathrm{d}\phi \tag{38}$$



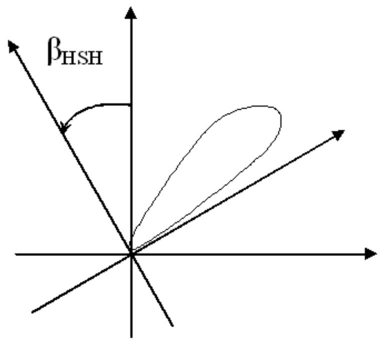
(a) Fonction originelle $f_{HSH}$



(b) $f_{SH}$ obtenue en étalant $f_{HSH}$ sur la sphère (i.e. en utilisant les SH au lieu des HSH)



(c) Application d'une rotation SH à $f_{SH}$



(d) La fonction hémisphérique pivotée est obtenue en repassant $R^{SH}_{\beta_{SH}}(f_{SH})$ sur l'hémisphère supérieure

Figure 14: Processus de rotation des harmoniques hémisphériques : la fonction originelle est tout d'abord étalée sur la sphère, puis pivotée à l'aide des matrices de rotation SH. Lors de la dernière étape, la fonction est repassée dans son domaine initial, l'hémisphère supérieure.

où $H - I$ est l'hémisphère privée du digone considéré, et où $R_\beta(\theta, \phi)$ représente la direction $(\theta, \phi)$ pivotée d'un angle $\beta$ autour de $Y$. Toutefois, cette méthode requiert de nombreuses matrices précalculées, et nécessite donc un espace mémoire important.

La section suivante présente les résultats obtenus à l'aide des harmonique hémisphériques pour la représentation de fonctions et pour le rendu réaliste.

## Applications et Résultats

### Représentation de Fonctions Hémisphériques

Nous avons tout d'abord testé notre base de fonctions pour la représentation d'un lobe de Phong [Pho75] $(\cos\alpha)^5$, où $\alpha$ est l'angle entre la direction de réflexion spéculaire et

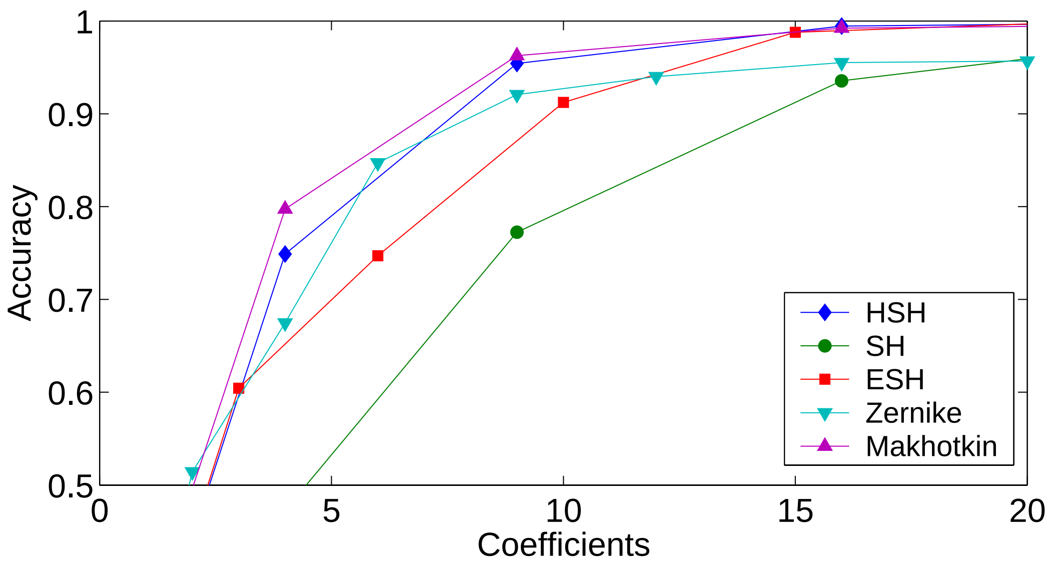


Figure 15: Qualité de représentation d'un lobe de Phong, $(\cos \alpha)^5$, en fonction du nombre de coefficients.

la direction de vue. La Figure 3.9 représente la qualité de l'approximation comparée à des méthodes existantes détaillées en Chapitre 2. Les harmoniques hémisphériques présentent une qualité supérieure ou comparable aux techniques existantes.

### Représentation de BRDFs

Les BRDFs sont définies sur le produit cartésien de deux hémisphères correspondant aux directions d'incidence et de réflexion. Nous avons choisi de discrétiser l'hémisphère des directions réfléchies et d'utiliser les harmoniques sphériques pour la représentation des valeurs de BRDF pour l'ensemble des directions incidentes. Cette méthode est également utilisée par Kautz *et al.* [KSS02]. Les directions de réflexion sont échantillonnées à l'aide du paramétrage parabolique proposé par Heidrich and Seidel [HS99]. Dans cette méthode, toute direction $(x, y, z)$ est plaquée sur un plan 2D $(u, v)$ :

$$u = \frac{1}{2}(1 + \frac{x}{z+1}) \tag{39}$$

$$v = \frac{1}{2}(1 + \frac{y}{z+1}) \tag{40}$$

Cette technique permet de stocker facilement les coefficients de projection de BRDFs sur carte graphique dans une texture 3D. Cette texture est indexée par des coordonnées $(u, v, w)$, où $(u, v)$ est la direction de réflexion, et $w$ est l'indice du coefficient considéré.

La Figure 16 résume les qualités de représentation des différentes méthodes testées pour une BRDF de Ward anisotrope [War92]. Bien que certaines méthodes soient

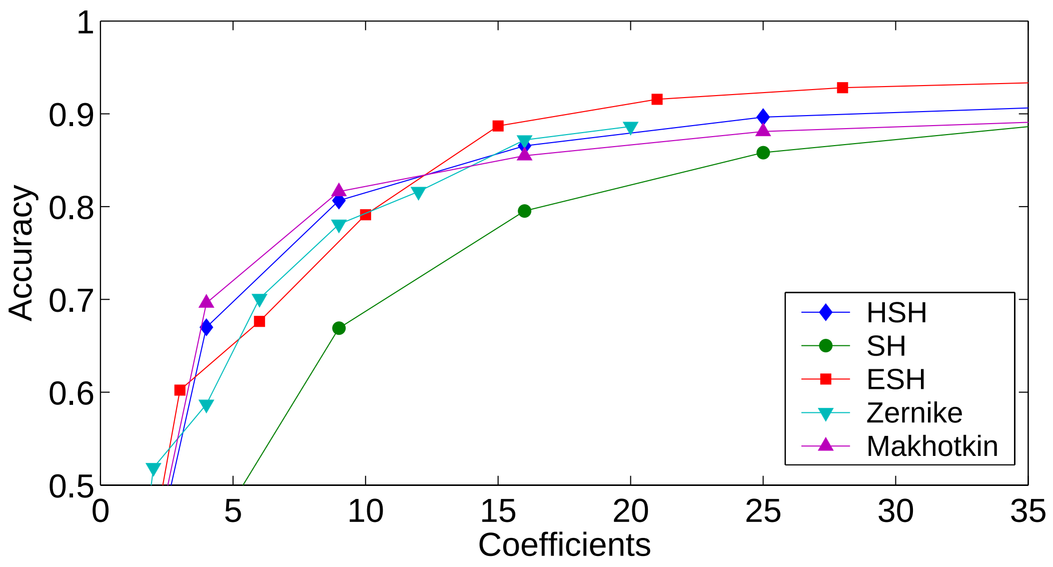


Figure 16: Qualité de représentation en fonction du nombre de coefficients. Ces valeurs ont été obtenus pour une BRDF de Ward anisotrope avec les valeurs $k_d = 0$, $k_s = 1$, $\alpha_x = 0.2$, $\alpha_y = 0.5$.

plus précises dans ce cas, les harmoniques hémisphériques restent plus précises que les harmoniques sphériques.

La Figure 17 montre des images obtenues en temps-réel à l'aide de notre logiciel de rendu basé GPU. Les objets sont éclairés par une source lumineuse ponctuelle. Une résolution $32 \times 32$ est utilisée pour représenter les direction de réflexion des BRDFs. Les images obtenues à l'aide d'harmoniques hémisphériques (Figures 3.11(c) et (d)) sont visuellement plus proches des images calculées à l'aide des BRDFs analytiques. A noter que le phénomène de Gibbs est très visible sur les images générées par BRDFs SH sous forme d'anneaux entourant le halo spéculaire (Figures 3.11(e) et (f)).

Les figures 16 and 17 montrent que la représentation HSH fournit des résultats de qualité supérieure aux SH pour un nombre de coefficients donné.

Les matrices de conversion entre SH et HSH sont utilisées dans le cadre du rendu par cartes d'environnement, dans lequel la luminance incidente est représentée par harmoniques sphériques et les BRDFs par harmoniques hémisphériques. La Figure 18 illustre les résultats obtenus. Le tableau 1 présente les résultats comparés entre les représentations SH et HSH des BRDFs. Notre méthode permet d'obtenir des vitesses de rendu similaires avec une qualité accrue.

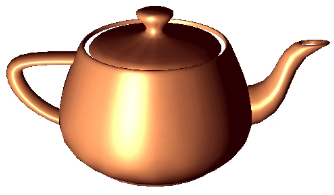
## Cache de Luminance

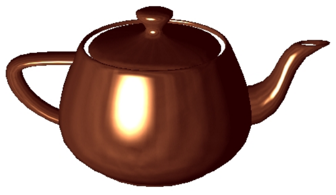
Les harmoniques hémisphériques peuvent également être utilisées dans le contexte de l'algorithme de cache de luminance introduit par Křivánek *et al.* . Les harmoniques

(a)

(b)

(c)


(d)


(e)

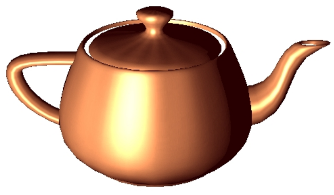

(f)

Figure 17: Images calculées avec les BRDFs de Phong (colonne de gauche) et de Ward anisotrope (colonne de droite). Les figures (a) et (b) sont générées avec les BRDFs analytiques; les figures (c) et (d) utilisent une représentation HSH d'ordre 7 (49 coefficients); les figures (e) et (f) utilisent une représentation SH d'ordre 7.

| Ordre de représentation ($n$) | 2 | 3 | 5 | 8 | 10 |
|---|---|---|---|---|---|
| HSH (ips) | 220 | 138 | 26.2 | 3.1 | 1.6 |
| SH (ips) | 223 | 160 | 29 | 3.28 | 1.73 |

Table 1: Temps de rendu (images/seconde, ou ips) pour le rendu par cartes d'environnement à l'aide de SH et HSH pour les BRDFs. Ces valeurs ont été obtenues avec un ordinateur équipé d'un processeur Intel Xeon 2.4GHz et d'une carte graphique ATI Radeon 9800 Pro. Les images ont été générées à résolution 700 $\times$ 600; le modèle de théière contient 1873 sommets.

hémisphériques sont utilisées pour la représentation des fonctions de luminance incidente et des BRDFs. De plus amples détails sur cet algorithmes peuvent être trouvés dans [Kři05] ainsi que dans le Chapitre 2.

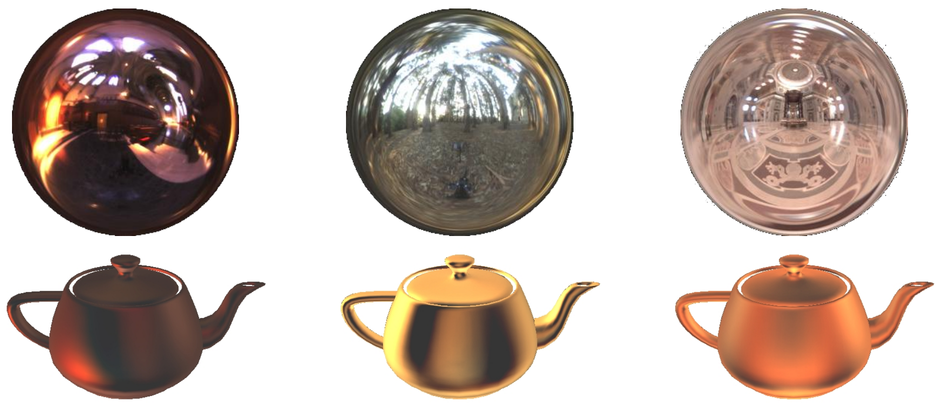

(a) Grace Cathedral (b) Eucalyptus Grove (c) St Peter's Basilica

Figure 18: Rendu par carte d'environnement SH à l'aide d'une représentation HSH des BRDFs (ordre 10). La théière non diffuse est éclairée par les cartes d'environnement de la ligne supérieure (Cartes d'environnement : Paul Debevec).

## Conclusion

Nous avons décrit une nouvelle méthode adaptée à la représentation de fonctions hémisphériques de basse fréquence. Les tests réalisés montrent que notre base de fonctions s'avère plus précise que les méthodes existantes, et est moins propice à l'apparition du phénomène de Gibbs.

Notre méthode étant très proche des harmoniques sphériques, nous avons défini une méthode de conversion permettant de passer d'une représentation à l'autre par simple produit matriciel. De plus, cette matrice de conversion s'avère être très éparse dû à l'orthogonalité partielle des harmoniques sphériques et hémisphériques.

La similitude entre harmoniques sphériques et hémisphériques permet également la définition de matrices de rotation s'appliquant directement aux coefficients de projection. Lors de la rotation de fonctions hémisphériques, une partie des valeurs passe sous l'équateur et doit donc être retirée. Nous avons proposé une méthode de suppression du digone considéré par l'application d'une matrice spécifique. Par la suite, nous avons démontré que les matrices de rotation SH peuvent être directement utilisées pour la rotation HSH à un coefficient près.

Cette nouvelle base ainsi que les méthodes de conversion et de rotation proposées font des harmoniques hémisphériques un outil idéal pour les calcul d'illumination tels que le rendu par cartes d'environnement ou le cache de luminance. De plus, la compacité de représentation permet le stockage des coefficients de projection dans la mémoire graphique pour le rendu temps-réel.

## Splatting de luminance

### Motivations

De manière générale, le calcul d'illumination globale est réalisé par lancer de rayons et échantillonnage de Monte-Carlo [War94, PH04]. Bien que de nombreuses approches aient été proposées pour le calcul et le rendu d'illumination globale en temps réel [WBS03, GWS04, TPWG02, WS99], ces méthodes basées sur le lancer de rayons sont généralement exécutées de manière massivement parallèle sur des grappes d'ordinateurs afin d'obtenir une certaine interactivité. Bien que des algorithmes tels que le cache d'éclairement [WRC88] et de luminance [KGPB05] permettent de réduire significativement les temps de rendu, la génération d'une solution d'illumination globale nécessite plusieurs heures de calcul sur un ordinateur standard.

Nous proposons ici une nouvelle méthode basée GPU pour le calcul et le rendu d'illumination globale à l'aide des algorithmes de cache d'éclairement et de luminance. Notre algorithme permet d'accélérer le calcul d'illumination globale d'un facteur 30 à 40. Nous nous appuyons sur deux observations exprimées par Tabellion *et al.* dans le contexte du rendu de production [TL04]. Tout d'abord, le calcul d'une seule interréflexion permet de prendre en charge la majeure partie des échanges lumineux entre les objets d'une scène, et fournit des résultats visuellement satisfaisants. Par ailleurs, Tabellion *et al.* [TL04] ont démontré qu'un maillage grossier de la scène d'origine est suffisant pour calculer précisément l'éclairage indirect. Par conséquent, nous nous concentrons sur le calcul efficace d'une interréflexion dans des scènes modérément complexes. L'éclairage direct peut quant à lui être calculé à part en utilisant la géométrie détaillée. Dans ce travail, l'éclairage direct est calculé à l'aide du processeur graphique afin de fournir des résultats interactivement.

Nous reformulons les algorithmes de cache d'éclairement et de luminance en définissant une méthode opérant dans l'espace image. Notre approche vise à réduire la charge de travail du CPU en effectuant les calculs les plus coûteux sur le GPU. Les processeurs graphiques étant basés sur une architecture parallèle, l'implémentation de structures de données requises par le lancer de rayons et les caches d'éclairement et de luminance ne peuvent pas être implémentées simplement, rendant leur utilisation inefficace et/ou imprécise. C'est pourquoi nous proposons le *splatting de luminance* [GKBP05], une reformulation de ces algorithmes adaptée au fonctionnement des cartes graphiques. Les algorithmes de cache d'éclairement et de luminance étant très similaires, les explications suivantes porteront sur le cache d'éclairement uniquement.

### Splatting de Luminance

Comme décrit précédemment, l'algorithme de cache d'éclairement est basé sur le calcul et l'interpolation d'enregistrements d'éclairement. Pour un point visible depuis le point de vue de l'utilisateur, l'algorithme détermine quels enregistrements contribuent à l'éclairement diffus indirect de ce point. Nous choisissons ici l'approche inverse : pour un enregistrement donné, notre méthode détermine les points visibles auquel l'enregistrement contribue en projetant la zone d'influence de l'enregistrement sur le plan image. Le résultat de cette projection, ou *splatting*, est stocké dans un tampon de la même taille que l'image finale appelé *tampon de splatting de luminance* ou *radiance splat buffer*. Chaque pixel $SPLATBUFF(x, y)$ de ce tampon contient une paire $(L_o, w)$, où $L_o$ est la somme pondérée des contributions des enregistrements à la luminance réfléchie vers ce pixel, et $w$ est la somme des pondérations correspondantes.

L'algorithme de splatting de luminance (Algorithme 10) est basé sur la formule d'interpolation de l'algorithme de cache d'éclairement:

$$E(\mathbf{p}) = \frac{\sum_{k \in S(\mathbf{p})} E_k(\mathbf{p}) w_k(\mathbf{p})}{\sum_{k \in S(\mathbf{p})} w_k(\mathbf{p})} \tag{41}$$

L'estimation de l'éclairement indirect $E_\mathbf{p}$ au point $\mathbf{p}$ est la moyenne pondérée des contributions au point $\mathbf{p}$ des enregistrements voisins. Le poids donné à un enregistrement $k$ en un point $\mathbf{p}$ avec une normale $\mathbf{n}$ est définie dans [WRC88] :

$$w_k(\mathbf{p}) = \frac{1}{\frac{\|\mathbf{p}-\mathbf{p}_k\|}{R_k} + \sqrt{1 - \mathbf{n} \cdot \mathbf{n}_k}} \tag{42}$$

où $\mathbf{p}_k$, $\mathbf{n}_k$ et $R_k$ sont respectivement l'emplacement de $k$, sa normale et la moyenne harmonique des distances des objets visibles depuis le point $\mathbf{p_k}$. La valeur $a$ donnée par l'utilisateur contrôle la qualité de la solution d'illumination globale calculée : un enregistrement $k$ contribue à l'estimation de l'éclairement indirect au point $\mathbf{p}$ si et seulement si :

$$w_k(\mathbf{p}) \geq \frac{1}{a} \tag{43}$$

En substituant l'Equation 43 dans l'Equation 42 et en choisissant $\mathbf{n} = \mathbf{n}_k$, nous constatons que l'enregistrement $k$ ne peut contribuer à l'estimation de l'éclairement indirect du point $\mathbf{p}$ que si :

$$\|\mathbf{p} - \mathbf{p}_k\| \leq aR_k \tag{44}$$

Par conséquent, l'Equation 44 garantit qu'un enregistrement $k$ ne peut pas contribuer à l'estimation de l'éclairement indirect d'un point hors d'une sphère $I_k$ de rayon $r_k = aR_k$ centrée en $\mathbf{p}_k$.

Étant donnée une caméra, le splatting de luminance projette la sphère $I_k$ sur le plan image (Figure 19). La fonction de pondération (Equation 42) est évaluée pour chaque point visible se trouvant dans la sphère $I_k$ projetée. Le poids résultant est comparé à la valeur de qualité $a$ donnée par l'utilisateur (Equation 43). Pour chaque pixel vérifiant cette condition, notre algorithme calcule la contribution de l'enregistrement $k$. Cette contribution est obtenue en utilisant les gradients d'éclairement définis dans [WH92, KGBP05].

Si aucun enregistrement ne contribue à l'éclairement indirect d'un point visible, un nouvel enregistrement doit être généré et projeté sur le plan image.

## Calcul d'un Enregistrement

Dans notre méthode, le calcul d'un enregistrement est réalisé en utilisant le CPU et le GPU. Tout d'abord, le GPU échantillonne l'hémisphère et effectue les calculs de

**Algorithm 10** Splatting de luminance

```
Soit $k = \{\mathbf{p}_k, \mathbf{n}_k, E_k, R_k\}$ l'enregistrement considéré.
Soit $I_k$ la sphère déterminant la zone d'influence de $k$.
Déterminer la boîte englobante de $I_k$ sur le plan image
for all pixel $P(x,y) = \{\mathbf{p}, \mathbf{n}, \rho_d\}$ dans la boîte englobante do
  // Évaluer la fonction de pondération en p
  $w = \frac{1}{\frac{\|\mathbf{p}-\mathbf{p}_k\|}{R_k} + \sqrt{(1-\mathbf{n}\cdot\mathbf{n}_k)}}$
  if $w \geq \frac{1}{a}$ then
    //Calculer la contribution de l'enregistrement k au point p
    $E'_k = E_k(1 + \mathbf{n}_k \times \mathbf{n} \cdot \nabla_r + (\mathbf{p} - \mathbf{p}_k) \cdot \nabla_t)$
    // Calculer la luminance réfléchie
    $L_o = \rho_d E'_k$
    // Accumuler le résultat dans le tampon de splatting de luminance
    $SPLATBUFF(x,y).L_o += wL_o$
    $SPLATBUFF(x,y).w += w$
  end if
end for
```

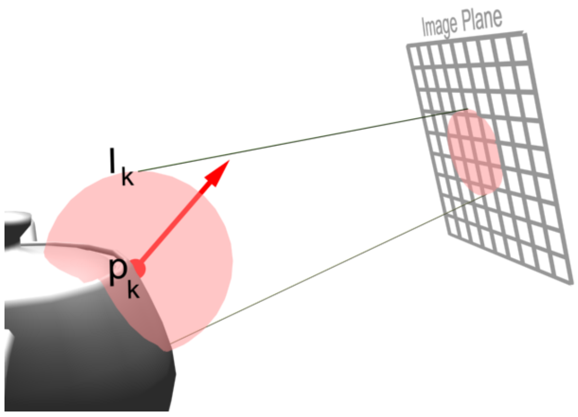


Figure 19: Projection de la sphère $I_k$ sur le plan image. La contribution de l'enregistrement $k$ est évaluée pour chaque point visible se trouvant dans cette sphère, et accumulée dans le tampon de splatting de luminance.

visibilité. Par la suite, le CPU utilise les valeurs de luminance incidente fournies par le GPU afin de générer l'enregistrement correspondant.

### Échantillonnage de l'Hémisphère sur GPU

De manière similaire à [SP89, LC04], nous proposons d'échantillonner l'hémisphère à l'aide d'un unique plan de projection : une caméra virtuelle est placée à l'emplacement de l'enregistrement à créer. Le processeur graphique effectue alors le rendu de la scène depuis ce point de vue. Toutefois, comme le montre la Figure 20, l'utilisation d'un unique plan de projection ne permet pas de prendre en compte les incidences rasantes. Nous proposons donc d'étendre virtuellement les pixels se trouvant au bord du plan de projection afin de combler la zone non échantillonnée (Figure 20). Un aspect important de cette méthode est que le caractère directionnel de la luminance incidente est conservé, tout en prenant en compte l'énergie totale provenant de toutes les directions de l'hémisphère. Par conséquent, cette méthode de compensation peut être utilisée également dans le cas du cache de luminance dans lequel l'information directionnelle est très importante.

L'échantillonnage de l'hémisphère par GPU nécessite le calcul de l'éclairage direct et de l'ombrage des points visibles. Ce calcul est réalisé par le GPU à l'aide du processeur de fragments programmable et de cartes d'ombres, ou *shadow maps* [Wil78]. A noter que le phénomène d'aliassage inhérent aux cartes d'ombres est gommé lors du calcul de l'éclairement par sommation des luminances incidentes, réalisée par le CPU. Cette étape nécessite le transfert de données depuis la mémoire graphique vers la mémoire centrale. Bien que ces transferts soient généralement très lents *via* le bus AGP, le bus PCI-Express permet de transférer les donnés très rapidement et réduit les pertes de temps dus à la synchronisation entre CPU et GPU. Le calcul d'un enregistrement est illustré dans la Figure 21.

Une fois les enregistrements nécessaires calculés, notre algorithme calcule l'image finale incluant l'éclairage direct et indirect.

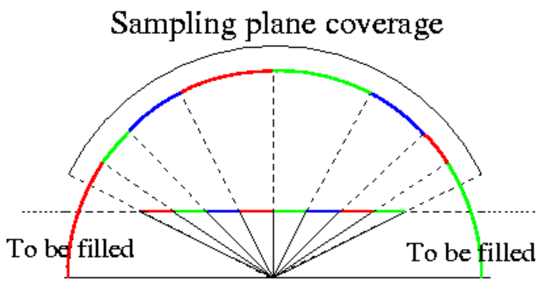


Figure 20: L'utilisation d'un unique plan de projection ne permet pas de prendre en compte les incidences rasantes. Nous définissons donc une méthode de compensation étendant virtuellement les pixels se trouvant au bord du plan de projection de manière à combler l'espace non échantillonné.

## Rendu Final

L'image finale est obtenue en cinq étapes (Algorithme 11). Pour une caméra donnée, la première étape consiste à obtenir des informations pour les objets visibles par chaque pixel : la position du point visible, le repère local et la BRDF (Figure 22).

Dans les étapes 2 et 3, le processus de rendu détermine les emplacements auxquels de nouveaux enregistrements sont requis en projetant les éventuels enregistrements ex-

**Algorithm 11** Calcul et rendu d'illumination globale

```
// Étape 1
Générer les informations sur les objets visibles pour chaque pixel (GPU)
Vider le tampon de splatting de luminance
// Étape 2
for all enregistrements du cache do
    // Le cache est vide pour la première image,
    // et contient des enregistrements dans les images suivantes.
    Algorithme 10: projection des enregistrements sur le tampon de splatting de lumi-
    nance (GPU)
end for
// Étape 3
Transférer le tampon de splatting de luminance du GPU vers le CPU
// Étape 4
for all pixels $(x, y)$ du tampon de splatting de luminance do
    if $SPLATBUFF(x, y).w < a$ then
        Calcul d'un nouvel enregistrement au point visible correspondant (GPU/CPU)
        Appliquer l'Algorithme 10: projeter le nouvel enregistrement (CPU)
    end if
end for
// Étape 5
for all enregistrements du cache do
    Appliquer l'Algorithme 10: projeter les nouveaux enregistrements (GPU)
end for
//Normaliser le tampon de splatting de luminance (GPU)
for all pixels $(x, y)$ du tampon de splatting de luminance do
    $SPLATBUFF(x, y).L_o / = SPLATBUFF(x, y).w$
end for
Combiner le tampon de splatting de luminance avec l'éclairage direct (GPU)
```

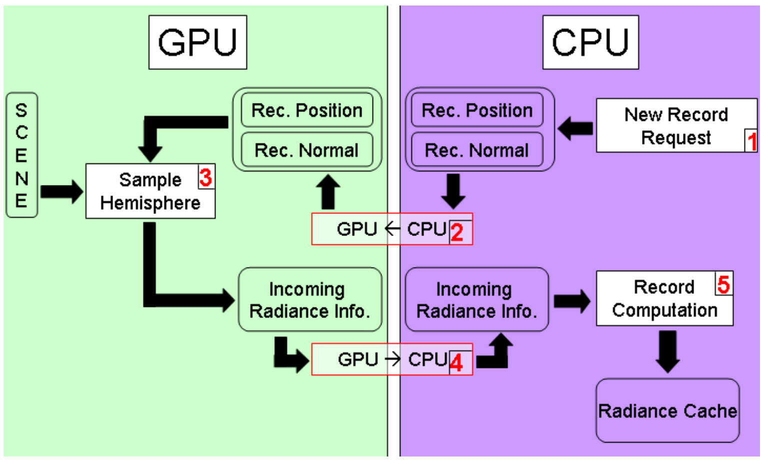


Figure 21: Processus de calcul d'un enregistrement. Les chiffres indiquent l'ordre de réalisation des tâches.

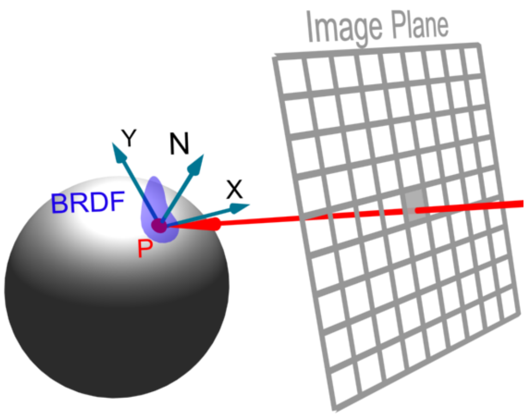


Figure 22: Information calculée pour chaque pixel : le point visible, le repère local et la BRDF.

istants et en transférant le contenu du tampon de splatting de luminance en mémoire centrale. L'étape 4 consiste à détecter les pixels $(x, y)$ du tampon pour lesquels les contributions ne sont pas suffisantes par le test suivant :

$$SPLATBUFF(x, y).w \geq a \tag{45}$$

Si le test n'est pas réussi, les enregistrements existants ne sont pas suffisants pour estimer correctement l'éclairage indirect au point correspondant. Un nouvel enregistrement est alors créé et projeté sur le tampon de splatting de luminance (Figure 23).

Lorsque $SPLATBUFF(x, y).w \geq a$ pour chaque pixel, le cache peut être utilisé pour le rendu de l'image finale. Les nouveaux enregistrements sont alors transférés au GPU pour l'affichage final dans lequel le tampon de splatting de luminance est ajouté à l'éclairage direct 24.

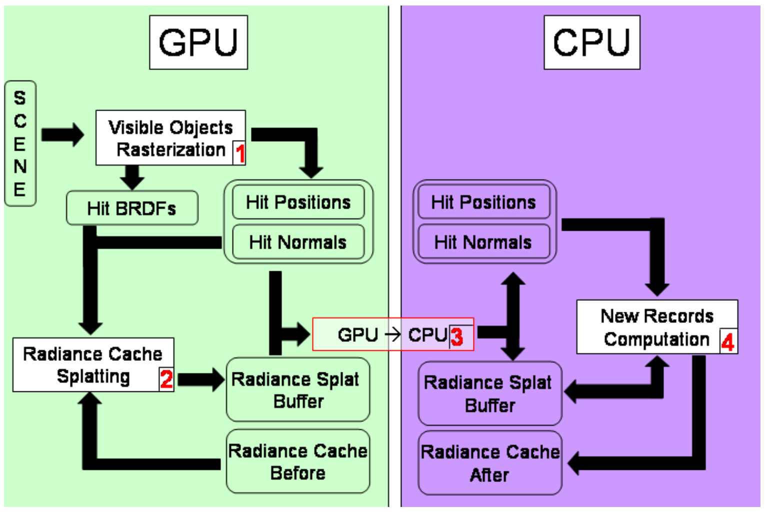


Figure 23: Processus de remplissage du cache d'éclairement. Les chiffres indiquent l'ordre de réalisation des tâches dans l'Algorithme 11. Durant ce processus le cache stocké sur le CPU est mis à jour à la volée alors que la version stockée sur le GPU n'est pas modifiée.

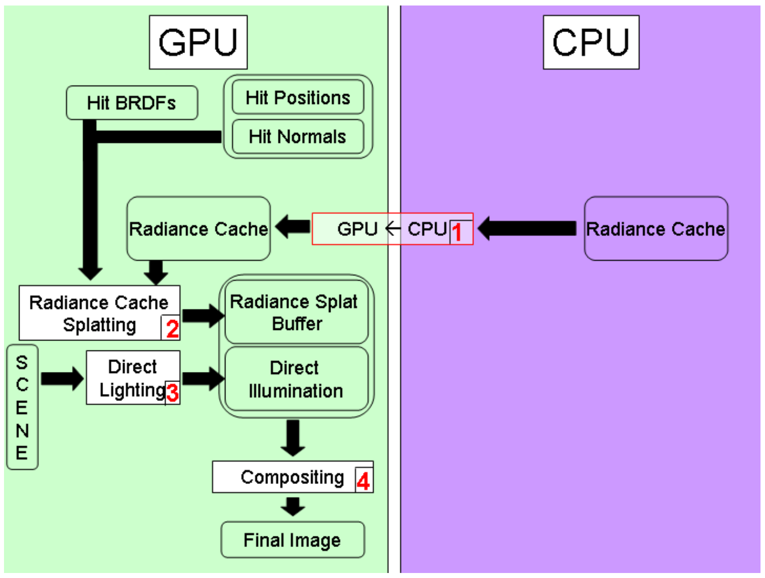


Figure 24: Rendu de l'image finale. Les chiffres indiquent l'ordre de réalisation des tâches décrites dans les étapes 4 et 5 de l'Algorithme 11.

Notre algorithme ne reposant plus sur des structures de données et algorithmes complexes, l'implémentation GPU peut être réalisée de manière directe. De plus, la version intégrale de cette thèse fournit les détails d'implémentation de notre méthode.

## Résultats

Nous présentons dans cette section les résultats obtenus à l'aide de notre implémentation du splatting de luminance. Les images et temps de calculs présentés ont été obtenus en utilisant un ordinateur doté d'un processeur Intel Pentium 4 3.6 GHz, d'1Go de RAM et d'une carte graphique nVidia Quadro FX 3400 PCI-Express.

### Rendu Haute Qualité

Dans cette partie, nous nous intéressons au rendu rapide de haute qualité. Nous comparons tout d'abord les performances de notre système de rendu avec celles du logiciel de référence Radiance [War94] dans le contexte de l'algorithme de cache d'éclairement. Puis, nous présentons les résultats obtenus pour le calcul d'illumination globale dans des scènes non diffuses par cache de luminance.

Nous avons comparé les temps de rendu de notre système avec le logiciel Radiance pour le calcul d'images dans deux scènes *Sibenik Cathedral* et *Sponza Atrium* (Figure 26). Ces images ont été calculées à résolution $1000 \times 1000$ en utilisant une résolution de $64 \times 64$ pour l'échantillonnage de l'hémisphère. Les résultats obtenus sont discutés ci-dessous, et résumés dans le tableau 2.

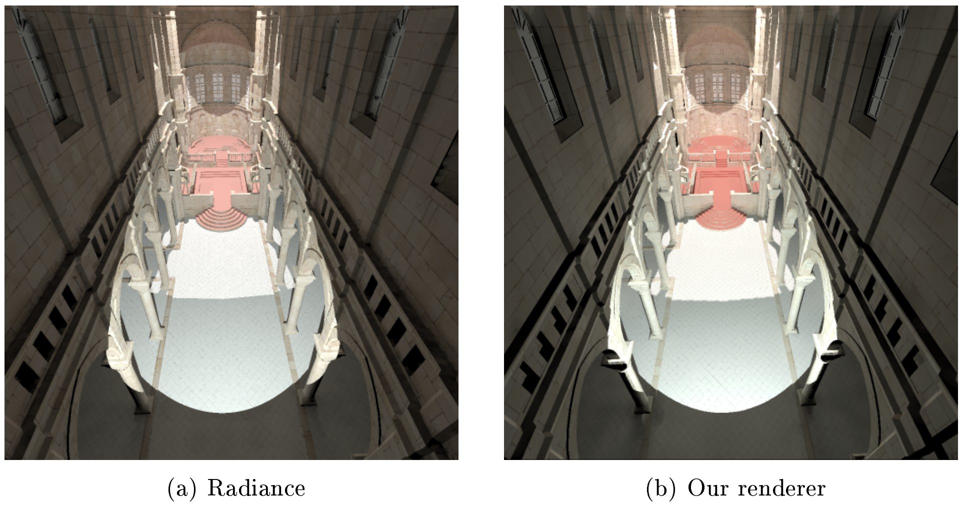
*a)* *Sibenik Cathedral* Cette scène contient 80000 triangles et est éclairée par deux sources ponctuelles. Les images sont rendues avec un paramètre de qualité $a = 0.15$. Le rendu de l'image nécessite le calcul de 4076 enregistrements, projetés à l'aide du GPU en 188 ms. Le logiciel Radiance calcule cette image en 7 min 5 s alors que le rendu en utilisant notre logiciel ne prend que 14.3 s (accélération : $\approx 30\times$)

*b)* *Sponza Atrium* L'image de cette scène contenant 66000 triangles et deux sources lumineuses a été générée en utilisant un paramètre de précision $a = 0.1$. Notre système calcule 4123 enregistrements en 13.71 s, et les projette en 242.5 ms. Avec les mêmes paramètres, le logiciel Radiance nécessite 10 min 45 s de calcul sur le CPU. Il en résulte un facteur d'accélération d'environ $47\times$.

| | Sibenik Cathedral | Sponza Atrium |
|---|---|---|
| Triangles | 80K | 66K |
| Précision | 0.15 | 0.1 |
| Radiance (s) | 425 | 645 |
| Notre système (s) | 14.3 | 13.7 |
| Accélération | 29.7 | 47.1 |

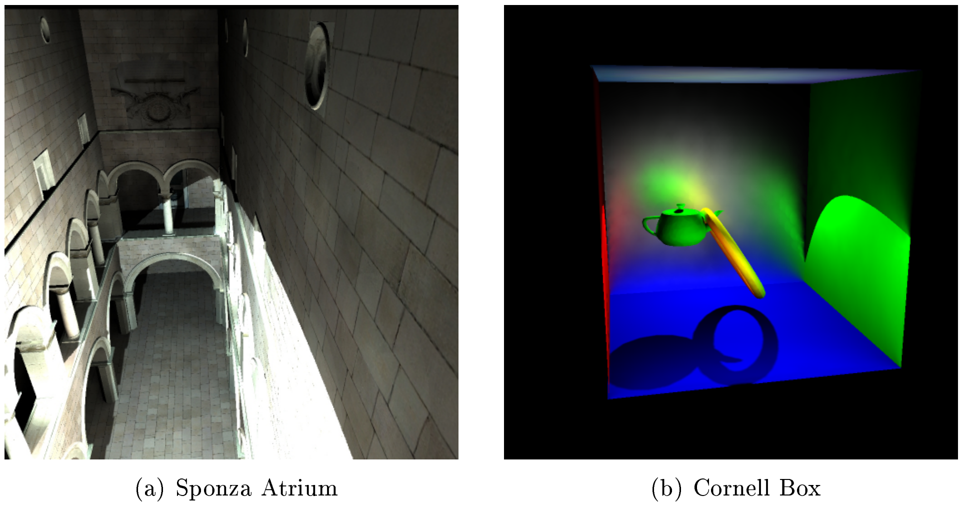
Table 2: Temps de rendu obtenus avec le logiciel Radiance et avec notre système de rendu pour des rendus de haute qualité d'environnements diffus. Les images ont été générées à résolution $1000 \times 1000$.

Notre système de rendu contient également une implémentation de l'algorithme de cache de luminance pour le calcul de l'éclairage indirect non diffus. La boîte de



(a) Radiance (b) Our renderer

Figure 25: Sibenik Cathedral (80K triangles). Illumination globale prenant en compte une seule interréflexion calculée avec Radiance (a) et notre système (b) (Modèle : Marko Dabrovic)



(a) Sponza Atrium (b) Cornell Box

Figure 26: Images obtenues avec notre système. L'atrium (66K triangles) est uniquement composé de surfaces diffuses (Modèle : Marko Dabrovic). Le mur du fond de la boîte de Cornell (1K triangles) est composé d'un matériau non diffus.

Cornell présentée en Figure 26(b) contient un mur non diffus (BRDF de Phong [Pho75], exposant 20), et des objets diffus. La BRDF non diffuse est projetée en harmoniques hémisphériques d'ordre 10 (i.e. 100 coefficients de projection), et la valeur de précision est $a = 0.25$. L'image a été générée en 12.18 s. Le cache contient 3023 enregistrements d'éclairement (splatting : 65.91 ms) et 869 enregistrements de luminance (splatting : 935.5 ms).

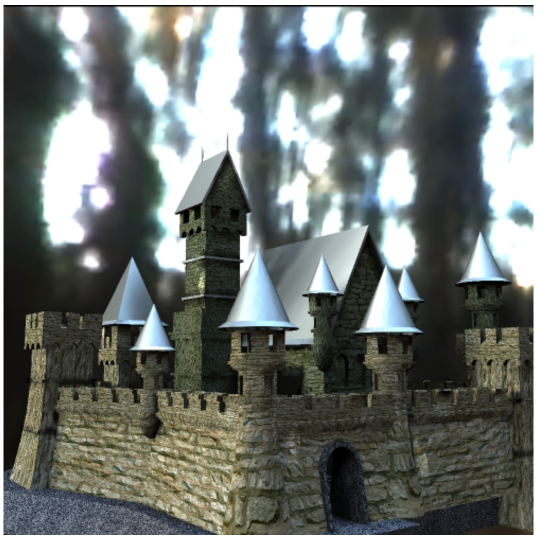


Figure 27: La scène du Château (58000 triangles) éclairée par une carte d'environnement. La solution d'illumination globale est calculée en 10.1 s par notre système.

La Figure 27 illustre l'utilisation du cache de luminance dans une scène plus complexe : la BRDF des toits du Château (57000 triangles) est une BRDF de Phong [Pho75] d'exposant 15. Ces images sont générées avec des valeurs de précision de 0.25 pour le cache d'éclairement et 0.2 pour le cache de luminance. La solution d'illumination globale est calculée en 10.1 s.

Les résultats présentés ci-dessus illustrent la capacité de notre algorithme à calculer des solutions d'illumination globale de haute qualité. Toutefois, la simplicité de notre algorithme permet également son utilisation dans le contexte du rendu progressif dans des applications interactives.

### Illumination Globale Interactive

Un aspect important des algorithmes de cache d'éclairement et de luminance est l'indépendance de la valeur des enregistrements par rapport au point de vue. Par conséquent, les enregistrements calculés pour un point de vue dans une scène statique peuvent être réutilisés pour d'autres points de vue. La vitesse de rendu du splatting de luminance permet de calculer une solution d'illumination globale au fur et à mesure des déplacements de l'utilisateur dans une scène. La Figure 28 illustre l'utilisation du splatting de luminance en montrant des images capturées lors d'une session interactive avec une valeur de précision de 0.5 et une résolution $512 \times 512$. L'interactivité est forcée en limitant à 100 le nombre maximal d'enregistrements calculés à chaque image.

Un autre aspect important du splatting de luminance est la possibilité de modifier la valeur de précision $a$ au cours d'une session de rendu interactif. Dans la méthode classique du cache d'éclairement, les enregistrements sont stockés dans un arbre octal dont la structure est étroitement liée à la valeur de $a$, qui doit donc rester fixe. En utilisant notre méthode, l'utilisateur peut régler la valeur de $a$ interactivement afin d'obtenir la qualité visuelle souhaitée (Figure 29).

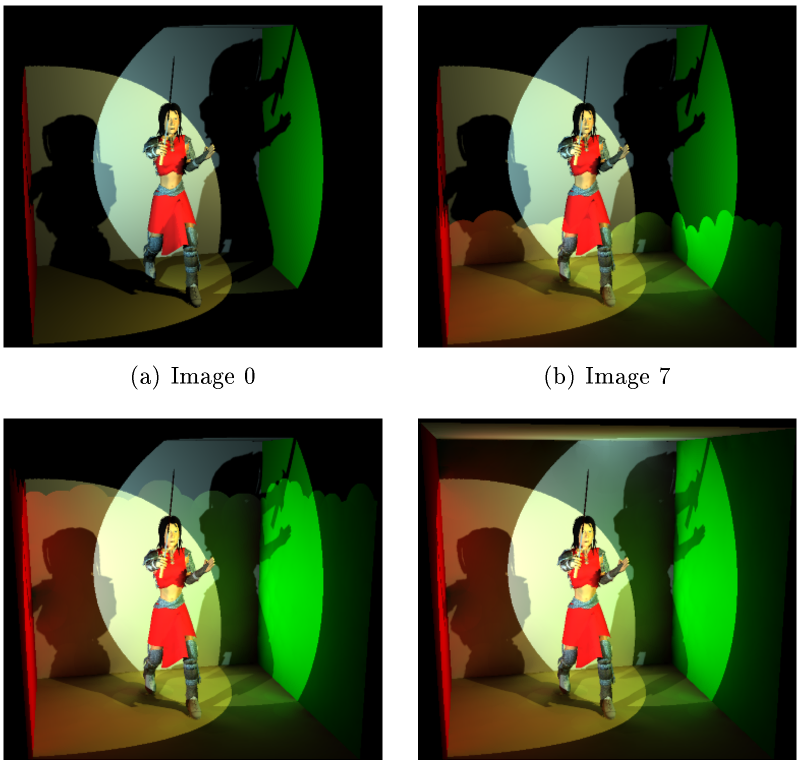
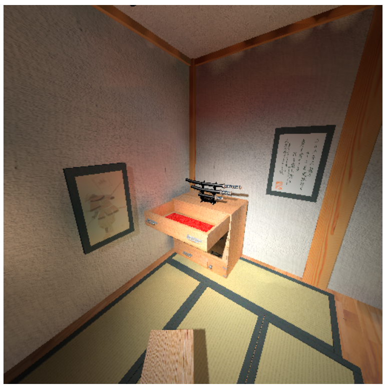

(a) Image 0
(b) Image 7
(c) Image 11
(d) Image 14

Figure 28: Une session de rendu interactif de la scène *Sam*(63000 triangles). Notre système de rendu calcule au plus 100 nouveaux enregistrements par image, maintenant donc un taux de rafraîchissement minimum de 5 images/seconde. Une fois le cache rempli, la solution d'illumination globale est affichée à 32 images/seconde.

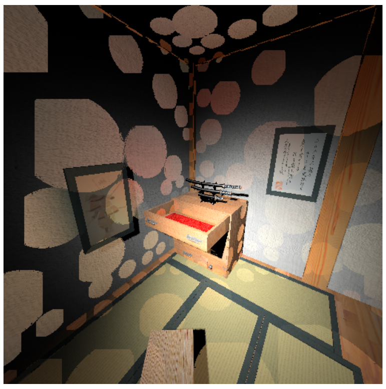
## Conclusion

Dans ce travail, notre but était d'utiliser la puissance de calcul des processeurs graphiques récents pour le calcul et le rendu d'illumination globale. Nous avons reformulé les algorithmes de cache d'éclairement et de luminance pour une implémentation simple sur GPU.

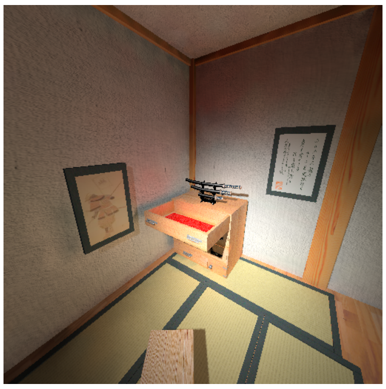
En se basant sur l'équation d'interpolation du cache d'éclairement, nous avons introduit la notion de splatting de luminance. Pour un enregistrement donné, sa contribution à l'éclairement indirect des autres points visibles est calculée rapidement dans le plan



(a) $a = 0.5$



(b) $a = 0.2$ using the records generated with $a = 0.5$



(c) $a = 0.2$

Figure 29: Notre méthode permet la modification à la volée de la valeur de précision $a$, facilitant ainsi le réglage de cette valeur par l'utilisateur.

image. Les contributions des enregistrements sont accumulées par le GPU dans le tampon de splatting de luminance à l'aide du mélange alpha. La visualisation finale du tampon est également effectuée par le GPU. Notre méthode permet l'affichage d'une solution d'illumination globale de haute qualité en temps-réel.

Afin d'accélérer le processus de rendu, nous avons également proposé une nouvelle méthode d'échantillonnage de l'hémisphère basée sur le GPU. Les luminances calculées par le GPU sont ensuite transférées rapidement en mémoire centrale par le bus PCI-Express pour calcul des valeurs des enregistrements par le CPU.

Nous avons comparé notre méthode avec le logiciel de référence Radiance. A qualité égale, notre système de rendu s'est avéré être au moins 30 fois plus rapide que le logiciel de référence.

Toutefois, cette méthode n'est pas encore complète. En particulier, nous souhaitons pouvoir gérer de multiples interréflexions, et également améliorer l'algorithme de cache de luminance pour la prise en compte de BRDFs et de fonctions de luminance incidentes de haute fréquence. L'amélioration discutée ci-après concerne l'extension de la méthode d'interpolation spatiale au domaine temporel pour un calcul d'illumination globale efficace et de haute qualité dans des scènes animées.

## Cache de Luminance Temporel

### Motivations

De manière générale, les méthodes de calcul d'illumination globale visent à simuler les interactions lumière/matière dans des scènes statiques. De plus, les approximations effectuées dans chaque méthode génèrent le plus souvent des clignotements lorsqu'une nouvelle solution d'illumination globale est calculée pour chaque image d'une animation. De nombreuses méthodes ont été proposées pour exploiter la cohérence temporelle de l'éclairage indirect, mais ces méthodes mettent en œuvre des structures de données et des algorithmes très complexes et encombrants en mémoire. C'est pourquoi, dans le contexte du rendu d'images de synthèse pour le cinema, les animations finales sont générées par filtrage temporel : une animation à 30 images/seconde est tout d'abord rendue à 60 images/seconde en recalculant une nouvelle simulation d'éclairage pour chaque image. Par la suite, chaque image de l'animation finale est obtenue en sommant deux images de l'animation à 60 images/seconde. Bien que les résultats soient probants, cette méthode requiert une puissance de calcul très importante. Dans le cadre du rendu interactif, l'illumination globale est typiquement précalculée en n'utilisant que les objets statiques. Pour les objets dynamiques, seul l'éclairage direct est remis à jour à la volée.

Nous proposons une méthode simple et précise exploitant la cohérence temporelle de l'éclairage indirect pour le rendu d'animation. En nous basant sur la méthode d'interpolation de Ward *et al.* [WRC88], nous étendons ce principe à l'échantillonnage épars et à l'interpolation dans le domaine temporel. A cette fin, nous définissons une fonction de pondération temporelle ainsi que des gradients temporels prenant en compte les changement d'éclairement locaux en termes spatiaux et temporels (Figure 31). Notre fonction de pondération et nos gradients temporels nécessitent une connaissance approximative de l'éclairage indirect futur. Nous proposons donc une méthode d'estimation de cet éclairage indirect facilement implémentable sur GPU.

Cette contribution est détaillée comme suit : après avoir introduit les problèmes liés à l'utilisation des caches d'éclairement et de luminance dans des scènes dynamiques, nous présentons notre fonction de pondération temporelle, l'estimation de l'éclairement futur par reprojection sur GPU, et les gradients temporels. Nous présentons ensuite les résultats obtenus. Les algorithmes de cache d'éclairement et de luminance étant similaires, nous ne détaillons ici que le cache d'éclairement. L'extension au cache de luminance ainsi que les détails d'implémentation GPU sont détaillés dans la version intégrale de cette thèse.

## Cache d'Éclairement Temporel

### Cache d'Éclairement dans des Scènes Dynamiques

L'algorithme de cache d'éclairement se base sur la haute cohérence spatiale de l'éclairement diffus indirect pour réduire les temps de calcul d'illumination globale. Dans des scènes dynamiques, l'approche la plus utilisée est un recalcul intégral de la solution d'illumination globale pour chaque image. Toutefois, les distributions des enregistrements d'éclairement des images $n$ et $n + 1$ peuvent être différentes. La Figure 30 illustre la conséquence de cette variation de distribution : les gradients d'éclairement n'étant pas précis à 100%, la qualité de l'éclairage indirect n'est pas constante dans l'espace. Par conséquent, un changement de la distribution des enregistrements se traduit par l'apparition de clignotements. De plus, le coût introduit par le recalcul systématique de la solution d'illumination globale pour chaque image est très important.

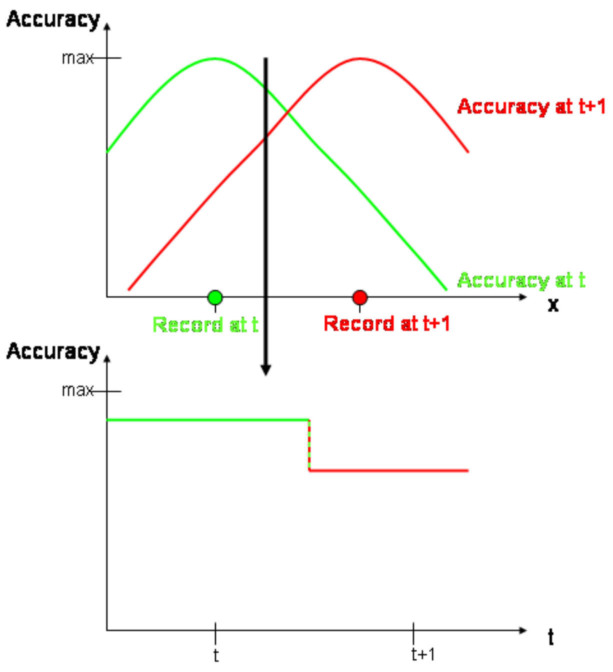
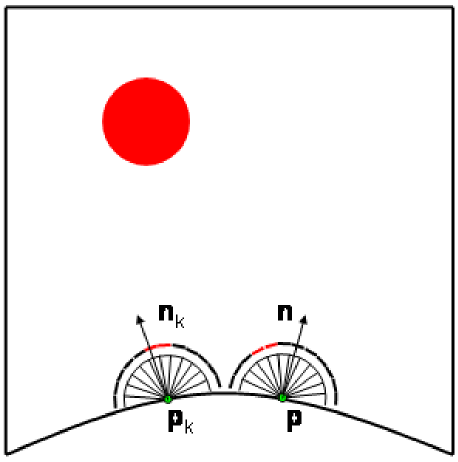


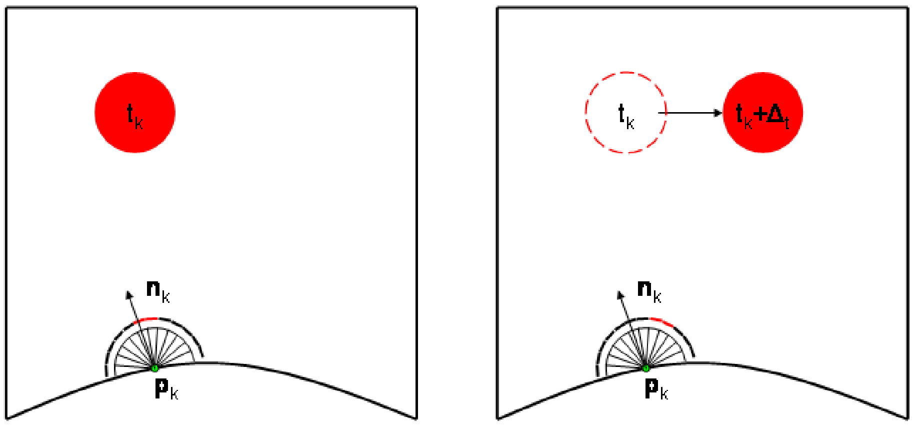
Figure 30: La modification de la distribution des enregistrements entre deux images successives modifie la qualité de l'éclairage indirect en un point donné (marqué par la flèche noire). Par conséquent l'éclairage en ce point peut varier notablement entre deux images, créant des clignotements. A noter que la qualité maximale est obtenue à l'emplacement et à l'instant exacts de création d'un enregistrement.

### Vue d'Ensemble du Cache d'Éclairement Temporel

Notre but est de tirer profit de la cohérence temporelle de l'éclairage indirect en réutilisant des enregistrements d'éclairement dans plusieurs images (Algorithme 12). Quand



(a) Changement spatial



(b) Changement temporel

Figure 31: Changement de luminance incidente dans l'espace et dans le temps. La zone noire/rouge au dessus de l'hémisphère illustre les valeurs de la fonction de luminance incidente. L'algorithme classique de cache d'éclairement (a) estime le changement potentiel d'éclairement en fonction de la rotation et de la translation. Le cache d'éclairement temporel (b) prend également en charge le changement temporel de l'éclairement.

un enregistrement $k$ est créé au point $\mathbf{p_k}$ à l'image $n$, nous estimons également l'éclairement futur en ce point. De manière similaire à Ward *et al.* [WRC88], nous définissons notre fonction de pondération temporelle $w_k^t$ comme l'inverse du changement temporel d'éclairement. Par conséquent, le nombre d'images successives dans lesquelles $k$ pourra contribuer est inversement proportionnel à ce changement. Un calcul précis de l'éclairement futur étant potentiellement coûteux en termes de temps de calcul, nous estimons cet éclairement par reprojection. Afin d'obtenir une animation dépourvue de clignotements, nous proposons également des gradients temporels pour une interpolation temporelle de qualité.

**Algorithm 12** Cache d'Éclairement Temporel

```
for all images $n$ do
  for all enregistrement existant $k$ do
    if $w_k^t(n)$ est suffisamment grande then
      Utiliser $k$ dans l'image $n$
    end if
  end for
  for all points $\mathbf{p}$ où un nouvel enregistrement est requis do
    Échantillonner l'hémisphère au-dessus de $\mathbf{p}$
    Estimer l'éclairement futur
    Générer $w_k^t$
    Calculer les gradients temporels
    Stocker le résultat dans le cache
  end for
end for
```

### Fonction de Pondération Temporelle

De manière similaire à [WRC88], nous définissons le changement temporel d'éclairement incident $\epsilon^t$ entre les temps $t$ et $t_0$ :

$$\epsilon^t = \frac{\partial E}{\partial t}(t_0) \;\; (t - t_0) \tag{46}$$

Cette dérivée peut être approchée en utilisant des valeurs d'éclairements $E_0$ et $E_1$ acquises en deux temps distincts $t_0$ et $t_1$ :

$$\begin{aligned} \frac{\partial E}{\partial t}(t_0) &\approx \frac{E_1 - E_0}{t_1 - t_0} && (47)\\ &= \frac{\tau E_0 - E_0}{t_1 - t_0} \quad \text{avec } \tau = E_1/E_0 && (48)\\ &= E_0 \frac{\tau - 1}{t_1 - t_0} && (49) \end{aligned}$$

Chaque pas de temps étant entier, nous choisissons $t_1 - t_0 = 1$, i.e. $E_1$ et $E_0$ représentent les éclairements estimés en deux pas de temps successifs.

Nous pouvons alors définir la fonction de pondération temporelle comme l'inverse du changement, en excluant $E_0$ :

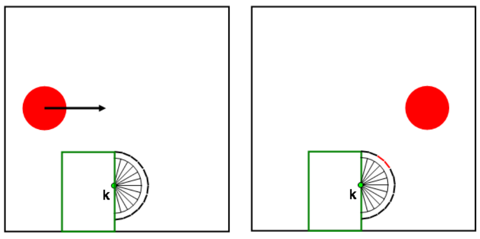
$$w_k^t(t) = \frac{1}{(\tau - 1)(t - t_0)} \tag{50}$$

où $\tau = E_1/E_0$ est le *taux de changement temporel d'éclairement.*

La fonction de pondération exprime la confiance en l'estimation d'éclairement en une image donnée. Cette fonction peut être évaluée et son résultat comparé à une valeur de précision $a^t$ donnée par l'utilisateur. Un enregistrement $k$ créé en $t_0$ ne peut contribuer à l'éclairage indirect en $t$ que si :

$$w_k^t(t) \geq 1/a^t \tag{51}$$

Toutefois, l'Equation 48 montre que si l'environnement est statique lors de la création de l'enregistrement, nous obtenons $\tau = 1$, ce qui implique une durée de vie infinie pour l'enregistrement considéré. Cette durée de vie peut alors introduire des erreurs si $t - t_0$ devient trop grand (Figure 32). Afin de pallier cette limitation, nous proposons d'utiliser une durée de vie maximale $\delta t_{max}$ spécifiée par l'utilisateur.



(a) Temps $t_k$ (b) Temps $t_k + n$

Figure 32: Lors de la création de l'enregistrement $k$ au temps $t_k$, l'environnement est statique ($\tau_k = 1$). Toutefois, la sphère rouge devient visible pour $k$ $n$ images plus tard. La valeur $\delta_{t_{max}}$ empêche l'utilisation de $k$ si $n > \delta_{t_{max}}$, réduisant ainsi le risque d'utilisation d'enregistrements obsolètes.

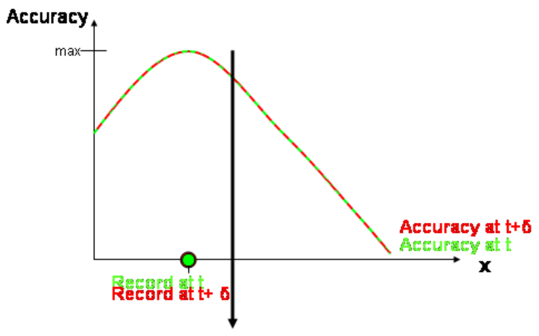
Toutefois, des clignotements peuvent apparaître lorsqu'un enregistrement est subitement supprimé. A l'instar de [TMS02], nous choisissons de supprimer ces clignotements en maintenant les positions des enregistrements constantes. Si un enregistrement $k$ devient obsolète, alors un nouvel enregistrement $l$ est créé exactement au même emplacement (Figure 33(b)).

Notre fonction de pondération temporelle permet de réutiliser de manière adaptative les enregistrements dans plusieurs images en fonction du changement temporel de l'éclairement local. Cette fonction étant basée sur une estimation de l'éclairement futur, nous introduisons une méthode de reprojection simple fournissant cette information.

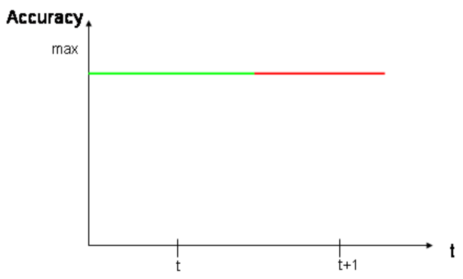
### Estimation de $E_{t_0+1}$

Notre méthode suit le principe de reprojection proposé dans [WDP99, WDG02]. Toutefois nous n'utilisons cette reprojection que pour estimer l'éclairement incident futur, non pour l'affichage de l'image finale.

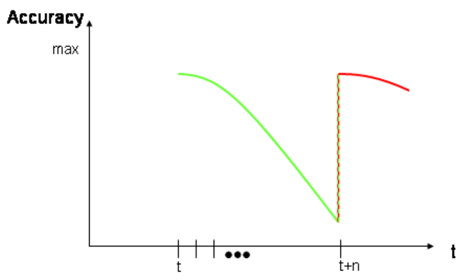
Dans le contexte d'une animation prédéfinie, tous les changements de la scène sont connus et accessibles. Lors de la création d'un enregistrement $k$ au temps $t_0$, l'hémisphère correspondante est échantillonnée (Figure 34(a)) pour calculer l'éclairement



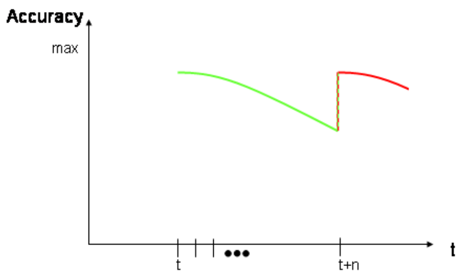
(a)



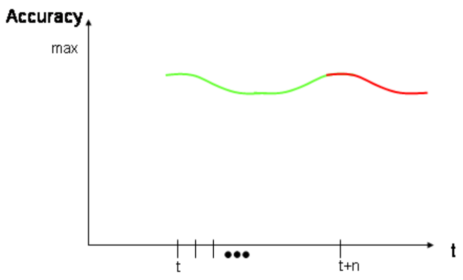
(b) Nouvel enregistrement après $\delta = 1$ image



(c) Nouvel enregistrement après $\delta = n$ images, sans gradient temporel



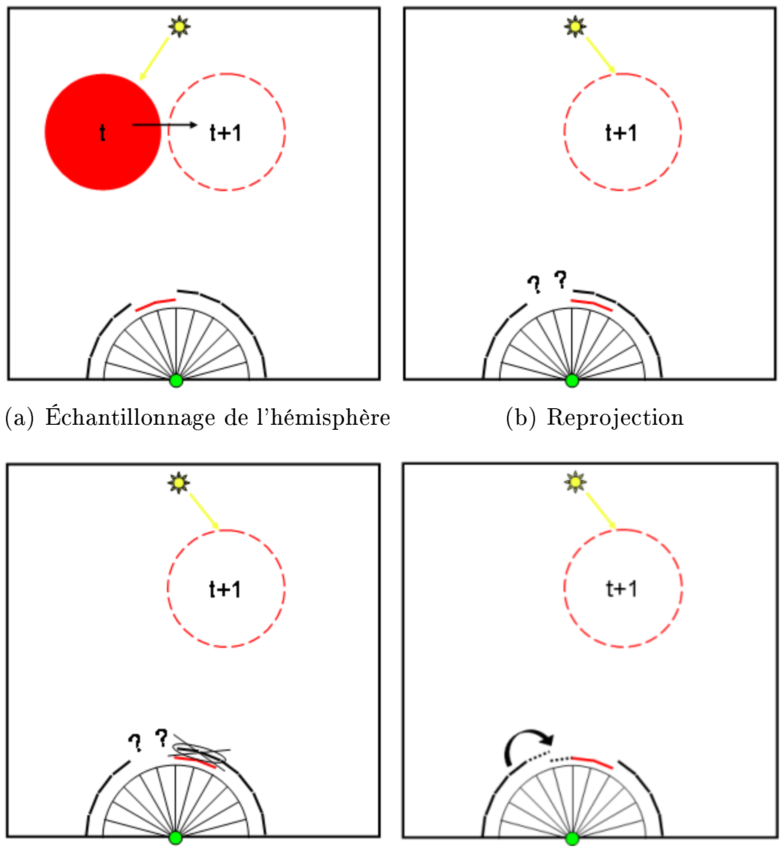
(d) Nouvel enregistrement après $\delta = n$ images, gradient temporel extrapolé



(e) Nouvel enregistrement après $\delta = n$ images, gradient temporel interpolé

Figure 33: Allure empirique de la précision du calcul en un instant fixé en fonction de la position (a) et en une position fixée en fonction du temps (b,c,d,e). Si les enregistrements sont situés en un même point dans des images successives, la qualité temporelle est augmentée (b). Toutefois, une discontinuité apparaît si un enregistrement est utilisé dans plusieurs images puis remplacé (c). Les gradients temporels extrapolés réduisent l'ampleur de cette discontinuité (d). Les gradients temporels interpolés lissent les changements temporels, éliminant ainsi les discontinuités (e).

et les gradients spatiaux en ce point. Les changements entre les temps $t_0$ et $t_0+1$ étant connus, il est possible de reprojeter les points visibles au temps $t_0$ à leur position au temps $t_0+1$ (Figure 34(b)). La luminance réfléchie aux points reprojetés peut être estimée en prenant en compte la rotation et le déplacement des objets et des sources lumineuses. Dans les zones de recouvrement, un test de profondeur permet de ne conserver que la valeur la plus proche (Figure 34(c)).



(a) Échantillonnage de l'hémisphère (b) Reprojection

(c) Test de profondeur (d) Filtrage

Figure 34: L'hémisphère est échantillonnée au temps $t$ (a). Pour chaque point visible, notre méthode estime l'emplacement de ce point au temps $t+1$ par reprojection (b). Le recouvrement est pris en charge par un test de profondeur (c), et les possibles trous sont remplis en utilisant les valeurs voisines ayant la plus grande profondeur (arrière-plan local) (d).

Certaines parties de la luminance incidente reprojetée ne peuvent pas être reconstituées par reprojection (Figure 34(d)). Comme proposé dans [WDP99], les cellules vides sont comblées en utilisant l'arrière plan local.

Notre fonction de pondération temporelle et notre remplacement des enregistrement, ainsi que l'estimation de l'éclairement futur, augmentent la qualité du rendu ainsi que l'efficacité du calcul. Toutefois, la Figure 33(c) montre que la qualité du calcul n'est toujours pas continue en fonction du temps. C'est pourquoi nous proposons les *gradients temporels* pour réduire voire éliminer ces discontinuités.

### Gradients Temporels

Conceptuellement, les gradients temporels sont très similaires aux gradients spatiaux : le changement d'éclairement est exprimé en fonction du temps au lieu de l'espace.

Dans le contexte du cache d'éclairement, l'éclairement en un point $\mathbf{p}$ avec une normale $\mathbf{n}$ est estimé à l'aide des gradients de translation et de rotation. Le gradient temporel d'un enregistrement $k$ est donné par :

$$\nabla^{\mathbf{t}}(\mathbf{p}) = \frac{\partial}{\partial t}(E_k + (\mathbf{n_k} \times \mathbf{n}) \cdot \nabla_{\mathbf{r}} + (\mathbf{p} - \mathbf{p_k}) \cdot \nabla_{\mathbf{p}}) \tag{52}$$

où:

- $\nabla_{\mathbf{r}}$ et $\nabla_{\mathbf{p}}$ sont respectivement les gradients de rotation et de translation
- $\mathbf{p_k}$ et $\mathbf{n_k}$ sont la position et la normale de l'enregistrement $k$

En gardant $\mathbf{n}$, $\mathbf{p}$, $\mathbf{n_k}$ et $\mathbf{p_k}$ constants dans le temps, le gradient temporel devient :

$$\nabla^{\mathbf{t}}(\mathbf{p}) = \nabla^{\mathbf{t}}_{\mathbf{E_k}} + (\mathbf{n_k} \times \mathbf{n}) \cdot \nabla^{\mathbf{t}}_{\nabla_{\mathbf{r}}} + (\mathbf{p} - \mathbf{p_k}) \cdot \nabla^{\mathbf{t}}_{\nabla_{\mathbf{p}}} \tag{53}$$

où:

- $\nabla^{\mathbf{t}}_{\mathbf{E_k}} = \frac{\partial E_k}{\partial t}$ est le *gradient temporel d'éclairement*
- $\nabla^{\mathbf{t}}_{\nabla_{\mathbf{r}}} = \frac{\partial \nabla_{\mathbf{r}}}{\partial t}$ est le *gradient temporel du gradient de rotation*
- $\nabla^{\mathbf{t}}_{\nabla_{\mathbf{p}}} = \frac{\partial \nabla_{\mathbf{p}}}{\partial t}$ est le *gradient temporel du gradient de translation*

Par conséquent, la contribution de l'enregistrement $k$ créé en $t_k$ à l'éclairement indirect du point $\mathbf{p}$ en $t$ est :

$$\begin{aligned} E_k(\mathbf{p}, t) &= E_k + \nabla^{\mathbf{t}}_{\mathbf{E_k}}(t - t_k) + \\ &\quad (\mathbf{n_k} \times \mathbf{n}) \cdot (\nabla_{\mathbf{r}} + \nabla^{\mathbf{t}}_{\nabla_{\mathbf{r}}}(t - t_k)) + \\ &\quad (\mathbf{p_k} - \mathbf{p}) \cdot (\nabla_{\mathbf{p}} + \nabla^{\mathbf{t}}_{\nabla_{\mathbf{p}}}(t - t_k)) \end{aligned} \tag{54}$$

En utilisant notre méthode d'estimation de l'éclairement au prochain pas de temps, nous définissons les *gradients temporels extrapolés* :

$$\nabla^{\mathbf{t}}_{\mathbf{E_k}} \approx E(t_k + 1) - E(t_k) \tag{55}$$

$$\nabla^{\mathbf{t}}_{\nabla_{\mathbf{r}}} \approx \nabla_{\mathbf{r}}(t_k + 1) - \nabla_{\mathbf{r}}(t_k) \tag{56}$$

$$\nabla^{\mathbf{t}}_{\nabla_{\mathbf{p}}} \approx \nabla_{\mathbf{p}}(t_k + 1) - \nabla_{\mathbf{p}}(t_k) \tag{57}$$

Ces gradients extrapolés réduisent les discontinuités Figure 5.4(d) mais ne les suppriment pas. Une solution pour supprimer les clignotements résiduels consiste à calculer des *gradients temporels interpolés* lors du remplacement d'un enregistrement $k$ par un enregistrement $l$ au même point :

$$\begin{aligned} \nabla^{\mathbf{t}}_{\mathbf{E_k}} &\approx (E_l - E_k)/(t_l - t_k) && (58) \\ \nabla^{\mathbf{t}}_{\nabla_\mathbf{r}} &\approx (\nabla_\mathbf{r}(t_l) - \nabla_\mathbf{r}(t_k))/(t_l - t_k) && (59) \\ \nabla^{\mathbf{t}}_{\nabla_\mathbf{p}} &\approx (\nabla_\mathbf{p}(t_l) - \nabla_\mathbf{p}(t_k))/(t_l - t_k) && (60) \end{aligned}$$

Ces gradients permettent de supprimer les discontinuités Figure 5.4(e) en lissant les changements temporels. Il est à noter que des changements d'éclairement très rapides peuvent ne pas être pris en compte en utilisant cette méthode. Dans ce cas, les valeurs $a^t$ et $\delta_{t_{max}}$ doivent être réduites afin d'augmenter la fréquence d'échantillonnage temporelle.

Cette méthode étend l'algorithme de cache d'éclairement aux scènes dynamiques à l'aide d'une fonction de pondération et des gradients temporels. Les détails d'implémentation de cette méthode sur carte graphique sont donnés dans la version intégrale de cette thèse.

## Résultats

Cette section présente les résultats obtenus à l'aide de notre méthode et les compare à l'approche classique dans laquelle un nouveau cache est calculé pour chaque image. Les images, vidéos and temps de calcul ont été générés sur un ordinateur doté d'un processeur Intel Pentium 3.8GHz avec 2 Go de RAM et une carte graphique nVidia GeForce 7800 GTX 512MB. Les scènes utilisées et les temps de calculs sont résumés dans le tableau 3.

**Cube in a Box** Cette scène diffuse très simple (Figure 37(a)) illustre les clignotements apparaissant en recalculant le cache pour chaque image. Avec une valeur de qualité temporelle $a^t = 0.05$ et une durée de vie maximale de 20 images, notre méthode s'avère environ 7.62 fois plus rapide que la méthode classique. La Figure 35 illustre les valeurs de qualité obtenues avec et sans gradients temporels. Notre travail se concentrant sur le rendu de haute qualité, les résultats suivants ne considèrent que le rendu par gradients temporels interpolés.

**Lumière Mobile** Cette scène (Figure 37(b)) illustre le comportement de notre algorithme dans le contexte de sources lumineuses dynamiques. Toutefois, les grandes variations d'éclairage indirect nécessitent un recalcul fréquent des enregistrements. Le temps de calcul n'est donc pas réduit par rapport à la méthode classique. Toutefois, la qualité est améliorée.

**Cerf-Volant** Dans cette scène plus complexe (Figure 37(c)), notre algorithme fournit une accélération significative. De plus, comme le montre la Figure 5.11, la durée de vie des enregistrements est réduite uniquement dans les zones de fort changement temporel.

**Intérieur Japonais** Cette scène plus complexe (Figure 5.12(d)), contient des objets diffus et métalliques. Selon la vitesse des changements d'éclairage indirect, les accélérations constatées sont de 1.25× à 9×.

**Sphères** Cette scène illustre l'utilisation de notre algorithme dans le cas d'animations complexes et rapides avec des surfaces diffuses et non diffuses (Figure 37(e)). En utilisant notre méthode avec une précision $a^t = 0.05$ et $\delta_{t_{max}} = 5$, le temps de calcul est réduit d'un facteur 4.24.

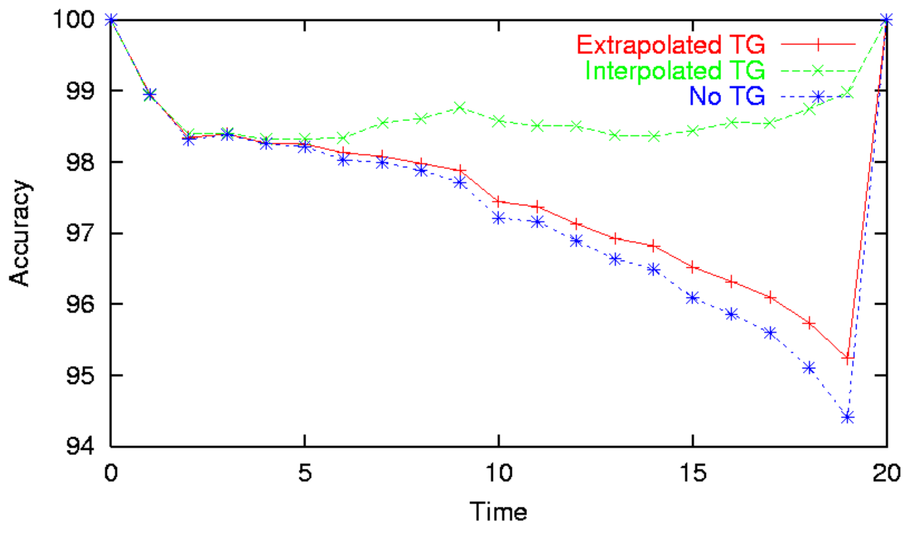


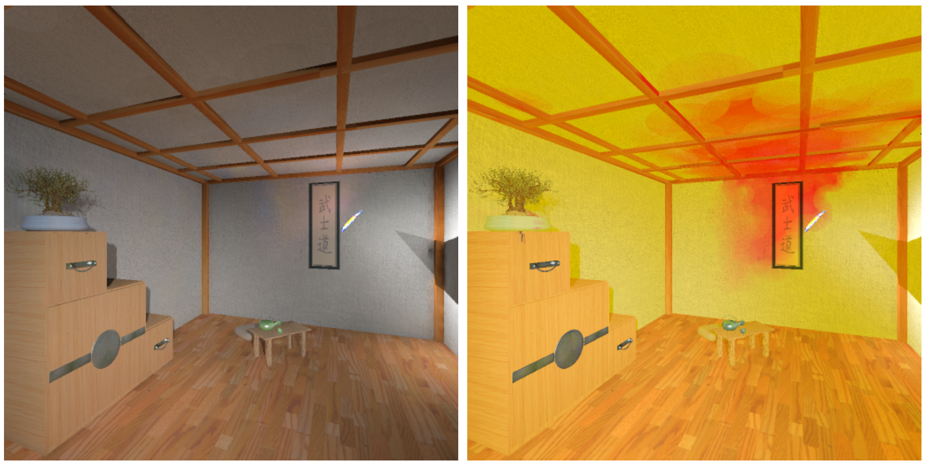
Figure 35: Précision temporelle obtenue dans la scène Cube in a Box en créant des enregistrements au temps 0 et en les extrapolant jusqu'au temps 19. De nouveaux enregistrements sont calculés au temps 20. Les gradients temporels (TG) fournissent une meilleure approximation comparée à une approche sans gradients. Les gradients interpolés rendent la précision continue dans le temps et constamment supérieure à 98%.

| Scene | Nb. Poly | Nb. Images | Calcul Classique | Notre Méthode | Accélération |
|---|---|---|---|---|---|
| Cube in a Box | 24 | 400 | 2048s | 269s | 7.62 |
| Lumière Mobile | 24 | 400 | 2518s | 2650s | 0.95 |
| Cerf-Volant | 28K | 300 | 5109s | 783s | 6.52 |
| Intérieur Japonais | 200K | 750 | 13737s | 7152s | 1.9 |
| Sphères | 64K | 200 | 3189s | 753s | 4.24 |

Table 3: Scènes de test et temps de calcul

## Conclusion

Nous avons présenté une méthode d'illumination globale exploitant la cohérence temporelle de l'éclairage indirect dans le contexte des algorithmes de cache d'éclairement et de luminance. Nous avons proposé de réutiliser les enregistrements d'éclairement et



(a) Image courante (b) Durée de vie des enregistrements

Figure 36: Lors du calcul d'illumination globale pour l'image courante (a), notre méthode estime le changement d'éclairement futur. Selon l'amplitude de ce changement, la durée de vie des enregistrements est adaptée (b). Le vert et le rouge représentent respectivement des durées de vie longues et courtes

de luminance dans plusieurs images afin de réduire les temps de calcul et d'augmenter la qualité de rendu d'animations.

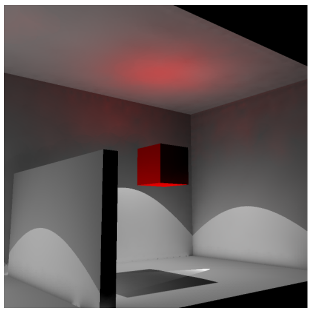
Pour cela, nous avons proposé une fonction de pondération temporelle. Cette fonction adapte la durée de vie des enregistrements en fonction du changement local d'illumination indirecte.

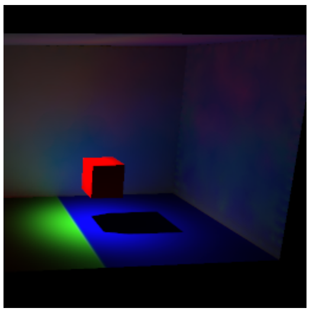
Afin d'estimer correctement la durée de vie des enregistrements, nous proposons une méthode simple de reprojection. Implémentée sur GPU, cette reprojection s'avère beaucoup moins coûteuse que l'évaluation réelle de l'éclairement futur.

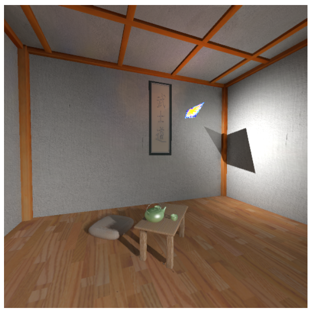
Enfin, nous proposons des gradients temporels, représentant le changement d'illumination indirecte pendant la durée de vie d'un enregistrement. Ces gradients permettent de réaliser une interpolation temporelle de qualité éliminant les clignotements.

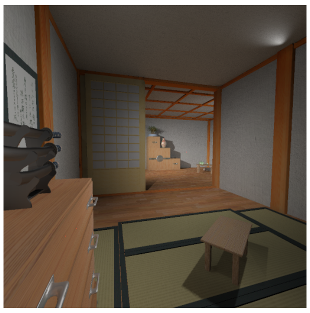
Les résultats démontrent une accélération significative ainsi qu'une augmentation de qualité en comparaison avec un calcul d'illumination globale indépendant pour chaque image. Notre méthode étant d'implémentation très simple, nous pensons qu'elle pourrait être facilement intégrée dans des logiciels professionnels pour un rendu d'animations efficace et de qualité.

Les travaux futurs incluent une estimation plus précise de l'éclairement futur, éliminant totalement les discontinuités dans le cas des gradients extrapolés. Un autre axe de recherche serait le définition d'une méthode efficace de détection de changements temporels, ce qui rendrait inutile la durée de vie maximale donnée par l'utilisateur.


(a) Cube in a Box


(b) Lumière Mobile


(c) Cerf-Volant


(d) Intérieur Japonais

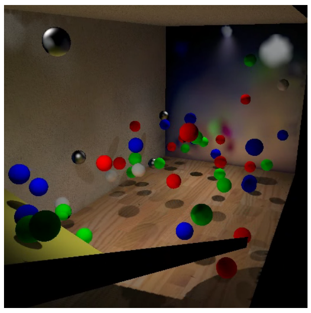

(e) Sphères

Figure 37: Images des scènes de test

## Rendu par Cartes de Photons et Cache d'Éclairement

La dernière contribution de cette thèse est le travail de Master de Jonathan Brouillat sur le rendu de cartes de photons par cache d'éclairement. Cette contribution est résumée dans la version anglaise de cette thèse. Ce travail est décrit *in extenso* en français dans [Bro06].

## Conclusion et Perspectives

Dans cette thèse, nous nous sommes intéressés au calcul d'illumination globale de haute qualité. Plus précisément, les recherches effectuées portent sur la reformulation et l'extension des algorithmes existants de cache d'éclairement et de luminance. Les méthodes proposées sont facilement implémentables sur cartes graphiques.

Les calculs d'illumination globale impliquant fréquemment des calculs sur l'hémisphère, nous avons développé une nouvelle base de fonctions pour la représentation de fonctions hémisphériques : les harmoniques hémisphériques. Obtenues par décalages de polynômes de Legendre associés utilisés par les harmoniques sphériques, notre base de fonctions permet d'obtenir une qualité de représentation supérieure aux méthodes existantes. L'utilité des harmoniques hémisphériques a été présentée dans le cadre d'éclairage par cartes d'environnements ainsi que pour le calcul d'éclairage global par cache de luminance.

La seconde contribution de cette thèse est la reformulation des algorithmes de cache d'éclairement et de luminance pour l'implémentation sur cartes graphiques. Cette méthode de splatting de luminance permet de remplacer le lancer de rayons et la structure hiérarchique des caches d'éclairement et de luminance par les opérations de *rasterization* et de mélange alpha câblés sur les cartes graphiques. L'utilisation de programmes dédiés exécutés pour chaque pixel de l'image permet de calculer simplement la contribution de chaque enregistrement à l'éclairage indirect des points visibles. De plus, nous avons proposé une méthode rapide et précise d'échantillonnage de l'hémisphère sur GPU pour le calcul des enregistrements. Les résultats obtenus montrent une accélération d'un facteur 30 à 40 en comparaison du logiciel Radiance basé sur le lancer de rayons.

De manière générale, les algorithmes classiques de calcul d'éclairage global ne considèrent que des scènes statiques. Bien que de nombreuses méthodes aient été proposées pour l'extension à des scènes dynamiques, ces méthodes impliquent le plus souvent des structure de données complexes et encombrantes en termes de mémoire. Dans le contexte des algorithmes de cache d'éclairement et de luminance, nous avons étendu le principe d'échantillonnage et d'interpolation de ces algorithmes au domaine temporel. Pour cela, nous avons défini une fonction de pondération temporelle ainsi que des gradients temporels. En réutilisant les enregistrements durant plusieurs pas de temps, nous avons réduit les temps de calcul de manière importante. De plus, la qualité de rendu des animations est significativement augmentée.

La dernière contribution de cette thèse a été réalisée lors de l'encadrement du stage de Master de Jonathan Brouillat. Cette méthode vise à exploiter directement une carte de photons pour la génération d'un cache d'éclairement de qualité. Basée sur une estimation de densité précise, le coût de création des enregistrements est significativement réduit.

Il existe de nombreux axes de recherche pour augmenter la qualité et l'efficacité des méthodes proposées. Parmi celles-ci, nous pouvons mentionner le remplacement des harmoniques hémisphériques par des fonctions de base localisées similaires aux ondelettes sphériques [SS95]. Toutefois, les opérations de rotation nécessaires à l'algorithme de cache de luminance doivent être définies. Ces bases de fonctions localisées permettraient une représentation efficace des fonction de haute fréquence et réduiraient le phénomène de Gibbs. Une extension importante de l'algorithme de splatting de luminance serait le calcul de multiples interréflexions, par exemple à l'aide de splatting récursif. Notre méthode prenant en compte la cohérence temporelle de l'éclairage indirect peut également être améliorée en définissant une méthode d'interpolation temporelle non linéaire. Cette nouvelle méthode permettrait de simuler des changements d'éclairage rapides, qui peuvent ne pas être pris en compte correctement dans notre méthode. Toutefois, cette méthode d'interpolation nécessite une connaissance avancée de l'éclairage indirect futur, dont la méthode de calcul reste à déterminer.

## Résumé

La modélisation et la simulation des interactions lumière/matière sont des aspects essentiels du rendu photoréaliste. Bien que de nombreux algorithmes de calcul d'illumination aient été proposés au cours des dernières décennies, ces méthodes ont généralement des coûts élevés en termes de mémoire et temps de calcul. Dans cette thèse nous nous intéressons au calcul rapide d'illumination globale dans des scènes animées. A cette fin, nous utilisons les performances des processeurs graphiques récents. Les contributions de cette thèse visent à se rapprocher du calcul d'illumination globale en temps-réel. Nous proposons une nouvelle base de fonctions pour la représentation précise et compacte de fonctions basse fréquence définies sur l'hémisphère : les harmoniques hémisphériques. Nous appliquons notre base de fonctions à la représentation de fonctions de luminance incidente et de réflectance bidirectionnelle dans le cadre de l'algorithme de cache de luminance. Nous introduisons également une reformulation des algorithmes de rendu par cache d'éclairement et de luminance. Cette reformulation permet une implémentation simple et efficace sur carte graphique. Notre algorithme permet d'obtenir une solution d'illumination globale jusqu'à 40 fois plus rapidement qu'un logiciel de rendu classique basé sur le cache d'éclairement. L'algorithme d'illumination globale par cache d'éclairement étant basé sur un échantillonnage épars et sur l'interpolation de l'éclairage indirect, nous proposons une extension de cette méthode au domaine temporel. Notre implémentation sur cartes graphiques permet d'obtenir des temps de calcul très inférieurs à la méthode classique, tout en augmentant la qualité de rendu. La dernière contribution de cette thèse est une nouvelle méthode de calcul d'illumination globale par cartes de photons et cache d'éclairement. Basée sur une technique précise d'estimation de densité, cette méthode permet d'obtenir un rendu de qualité de la carte de photons sans nécessiter de passe de regroupement final.

**Mots-Clés : infographie, simulation d'éclairage, processeurs graphiques**

## Abstract

An accurate modeling and simulation of the interactions between light and matter is crucial in the context of photo-realistic rendering. While many approaches have been considered for high quality lighting simulation, those methods often involve high computational costs and/or memory consumption. In this thesis we investigate the field of fast high quality global illumination computation in dynamic scenes using graphics hardware. Our contributions are steps towards realtime global illumination. We first propose the hemispherical harmonics basis functions for efficient representation of low-frequency functions defined on the hemisphere. We apply our basis to the representation of incoming radiance and reflectance functions for efficient global illumination using the radiance caching algorithm. We also propose a reformulation of the irradiance and radiance caching algorithms for fast rendering using graphics hardware. This method proves significantly faster (up to 40×) than a classical renderer based on irradiance caching. The irradiance caching interpolation scheme aims at a sparse sampling and interpolation of the indirect lighting. We extend this approach to the temporal domain for fast and accurate rendering of animated scenes. Our implementation on graphics hardware exhibits a significant quality improvement while reducing the computational cost compared to the classical method. The last contribution of this thesis is a novel method for photon map rendering using irradiance caching. This method avoids the costly final gathering by performing an accurate density estimation, which is directly used to generate irradiance records.

**Keywords: computer graphics, lighting simulation, graphics hardware**