REAL-TIME HIGH QUALITY

HDR ILLUMINATION AND TONEMAPPED RENDERING

Despina Michael

University of Cyprus, 2010

Real-time realistic rendering of a computer generated scene is one of the core research areas in computer graphics as it is required in several applications such as computer games, training simulators, medical and architectural packages and many other fields.

The key factor of realism in the rendered images is the simulation of light transport based on the given lighting conditions. More natural results are achieved using luminance values near to the physical ones. However, the vast range of real luminances has a far greater range of values than what can be displayed on standard monitors. As a final step to the rendering process, a tonemapping operator needs to be applied in order to transform the values in the rendered image to displayable ones.

Illumination of a scene is usually approximated with the rendering equation which solution is a computational expensive process. Moreover, the computational cost increases even more with the increase in the number of light sources and the number of vertices of the objects in the scene. Furthermore, in order to achieve high frame rates, current illumination algorithms compromise the quality with assumptions for several factors or assume static scenes so that they can exploit precomputations.

In this thesis we propose a real-time illumination algorithm for dynamic scenes which provides high quality results and has only moderate memory requirements. The proposed algorithm is

based on factorization of a new notion that we introduce: *fullsphere irradiance*, which allows the pre-integration of contribution of all light sources within the same value for any possible receiver.

Recent illumination algorithms, including ours, usually use environment maps to represent the incident lighting in the scene. Environment maps enable natural environment lighting conditions to be used by using high dynamic range (HDR) values. Typically the HDR obtained result of the illumination needs to be tonemapped into LDR values that can be displayed on standard monitors.

Traditionally tonemapped techniques give emphasis either to frame rate (global operators) or to the quality (local operators) of the resulting image. In this thesis, we propose a new framework: selective tonemapping which addresses both requirements. The key idea of this framework is to apply the expensive computations of tonemapping only to the areas of images which are regarded as important.

A full rendering system has been developed which integrates HDR illumination computation and the selective tonemapping framework. Results show high quality images at real-time frame rates.

REAL-TIME HIGH QUALITY

HDR ILLUMINATION AND TONEMAPPED RENDERING

Despina Michael

A Dissertation

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

at the

University of Cyprus

Recommended for Acceptance

by the Department of Computer Science

July, 2010

© Copyright by

Despina Michael

All Rights Reserved

2010

APPROVAL PAGE

Doctor of Philosophy Dissertation

REAL-TIME HIGH QUALITY

HDR ILLUMINATION AND TONEMAPPED RENDERING

Presented by

Despina Michael

Research Supervisor _____
Yiorgos Chrysanthou

Committee Member _____
Christos Schizas

Committee Member _____
Constantinos Pattichis

Committee Member _____
Pere Brunet

Committee Member _____
Didier Stricker

University of Cyprus

July, 2010

to my beloved brother Gabriel

ACKNOWLEDGEMENTS

This thesis would not have been possible without the contribution of many people who I would like to acknowledge. Special thanks to my advisor, Yiorgos Chrysanthou, for his continued guidance all these years. His invaluable advice played an essential role in enabling me to complete this thesis. I would also like to thank him for introducing me to the amazing world of Computer Graphics.

Many thanks to the members of my PhD examination committee, Prof. Pere Brunet, Prof. Didier Stricker, Prof. Christos Schizas and Prof. Constantinos Pattichis for their guidance through their questions and comments on this thesis. My thanks to Alessandro Artusi and Benjamin Roch for their collaboration in the Tonemap project; their useful ideas and contributions were indeed very helpful. I would also like to thank all the members of Computer Graphics group at the Computer Science Department for their valuable feedback on my work. My thanks go also to all people in the Computer Science Department throughout my time at the University of Cyprus, who have made make this important part of my life an unforgettable and amazing experience.

Special thanks to my beloved family: to my parents George and Maria, and to my brothers Loizos, Aimilios and Gabriel for their love and support. Everyone was there, each in his own way, whenever I needed him. Last but not least, my grateful thanks to my friends Aimilia, Andreas, Costas, Georgios, Marios, Nearchos, Pyrros, Ritsa and Vicky who each in their unique way lightened the busy and difficult times with enjoyable moments.

CREDITS

Publications that lead to this thesis are given in this section. This thesis consists of two main parts: illumination and tonemapping. Concerning the work that has been done for illumination, the proposed *fullsphere irradiance factorization* algorithm [53] has been published at the Computer Graphics Forum, one of the major journals in the field. Work that has been done in this thesis for tonemapping is part of the work published at [4, 69]. Moreover, the *selective tonemapping framework* is to be submitted for publication to the Visual Computer journal.

Areas where the proposed illumination and tonemapping algorithms can be used include work of the author of the thesis in applications for museums, such as [54, 55, 62], and animation such as crowd simulation [52].

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Computer graphics and rendering

Computer graphics is the visual science that uses data in 3D representation and creates images on 2D displays. It has several applications in a variety of fields such as education, entertainment and science. Computer graphics involves modeling, geometric representation, animation and rendering and research is being carried out in these areas.

Rendering, which is the process of creating a 2D image from the 3D description of a virtual scene, is one of the core research areas in Computer Graphics. The creation of the final 2D image that is displayed on the screen is achieved through a number of steps called *the graphics rendering pipeline*. The rendering pipeline consists of three functional stages: application, geometry and rasterizer. In the last two stages several steps are performed, Figure 1. A description of each stage is given below.

The first stage of the rendering pipeline is the application stage and is executed on the CPU. The developer sets up the scene which is going to be rendered together with several parameters that will be used in the next stages and will affect the result of the rendered images. This stage involves

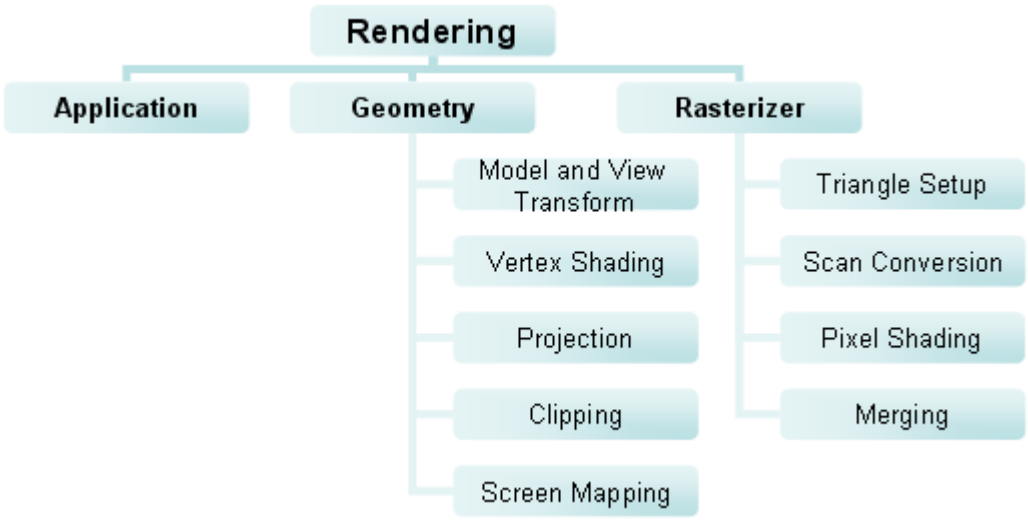


Figure 1: The rendering process.

many other processes, that can be implemented by the developer, which are not implemented in the next stages of the rendering pipeline. Such processes include collision detection between moving objects, interaction with the user, animation via transforms and many others.

The geometry stage receives the input data from the application stage and processes them through a series of steps. First, the coordinates that describe the geometry are transformed from the original *model space* to *viewing space*. Before any transformation each modeled object has its own local coordinates in the model space, the *model coordinates*.

We can apply transformations on an object in order to change its original position, orientation and scale, placing it in the *world space* based on a global coordinate system common for all objects in the scene. The transformation from local to world coordinates is called *model transform*. A second transformation, the *view transform*, occurs at this step, taking into account the position and orientation of the virtual camera. The *viewing coordinate system* has its origin at the position of the virtual camera and as y-axis the up-vector of the camera. World coordinates of models are transformed to viewing coordinates accordingly. At the end of this step, vertices and normals of scene geometry are located in *eye* or *viewing space*.

The vertex shading step is responsible for the appearance of the rendered objects. It takes into account light sources in the scene and their effect on objects' materials. At each vertex it computes information needed for shading. Outputs of the vertex shading stage, including color and texture coordinates, are passed to the rasterizer stage for interpolation.

In the projection step the view volume of the virtual camera is projected onto a unit cube. There are two types of projections; parallel and perspective. After the transformation, parallel lines remain parallel in the parallel projection, while in the perspective projection they converge in the horizon simulate the way we see in reality. Each type of projection suits different types of applications. For example, in architectural applications parallel projection is generally used whereas in virtual worlds perspective projection is preferable. The type of projection that is used at this stage is defined in the application stage.

Geometry which lies outside the viewing volume should not be rendered in the final image, so it should not be passed at the rasterizer stage; the clipping step is responsible for this task. Primitives entirely inside the viewing volume pass into the next stage, while primitives that lie totally outside the viewing volume are discarded. However, primitives which lie partially in the viewing volume are special cases. In these cases clipping is carried out against the planes of the unit cube. In order to keep only the part of the primitive that is within the viewing volume, vertices of the primitive that lie outside the viewing volume are discarded and new ones are generated.

The last step of the geometry stage is the screen mapping. Only the geometry that lies within the viewing volume is passed in this step. The x- and y-coordinate of each vertex that lies in the unit cube of previous step are transformed to the screen coordinates while the z-coordinate is not affected in any way. Window coordinates which are screen coordinates (x and y) and z-coordinates are passed on to the next stage: the rasterizer stage.

The rasterizer stage is the last functional stage of the graphics rendering pipeline and its purpose is to compute and set colors for the pixels covered by the rendered geometry. The rasterizer stage consists of four steps.

In the triangle setup step, vertices are collected and converted into triangles. Data for the triangle's surface that are needed for the scan conversion are computed. The scan conversion step identifies which pixels belong to each triangle based on which triangle covers the pixel's center. Data for each pixel are computed by interpolation of the data of the three vertices forming the triangle to which the pixel belongs. These data, which include the pixel's color, normal, and texture coordinates, will be used by the pixel shading step.

At the pixel shading step, the final color of each pixel is computed. Interpolated shading data are passed as input to this step and used to compute the color of each pixel.

The *color buffer* is an array of colors that stores the color of each pixel. This step is responsible for merging the color of the pixel computed in the pixel shading step with the color existing in the corresponding pixel of the color buffer. In the simplest case, the existing color in the color buffer is replaced by the computed pixel color wherever the current pixel is nearer to the user than the pixel stored in the color buffer. This test is done using pixels' z-values. However, in cases of transparent objects or blending effects, the existing and computed pixel colors are combined in a more complicated way. Final colors in the color buffer are displayed on the screen.

1.2 Motivation

Except for the application stage which needs the developer' intervention, all the steps of the rendering pipeline are executed and handled by the current *graphics processor units* (GPUs), commonly known as the graphics cards. In the fixed-function pipeline of both commonly used graphics APIs, OpenGL and Direct3D, the Blinn-Phong shading model [6] is used. Illumination is

computed on a per-vertex basis and the color values per pixel are computed with interpolation using Gouraud shading [26]. Due to the simplicity of this shading model, it runs fast and it is suitable for real-time graphics applications. However, it does not produce realistic results.

Real-time realistic rendering is needed in several applications such as computer games, training simulators, medical and architectural applications to name but a few. High frame rates give the user the ability to interact with the application in real-time while the realism makes the rendered scene look more natural.

Realism in the final rendered image is highly correlated with the faithful illumination of the virtual scene. An illumination algorithm that achieves realistic results in real-time frame rates is needed to be used in the rendering pipeline, in place of the simple Blinn-Phong model.

More natural results are achieved by using values for the intensity of the light near to the physical ones. However, the vast range of real luminances have a far greater range of values than can be displayed on standard monitors and in the range of values used in the fixed-function rendering pipeline. In cases where a vast range of values are used, an operation should be applied to transform the values in the rendered image to displayable ones. This operation is called *tonemapping*.

A trade-off exists between the degree of realism and the frame-rate. In order to increase the realism, computations need to be performed reducing the frame-rate and vice-versa. Thus, current techniques for real-time illumination and tonemapping compromise the quality in order to achieve the required timing performances.

A full rendering pipeline is needed that integrates illumination and tonemapping and thus gives high quality results in the real-time frame rate. Such a rendering pipeline would be used in interactive applications which require a high degree of realism.

1.3 Problem statement

Simulation of light transport in a computer generated scene is usually approximated with the rendering equation [34]. The rendering equation evaluates for a point in the scene, the radiance reflected towards the viewer, accounting for a number of factors, such as geometry of the scene, visibility, reflectance properties of the objects' materials and lighting conditions. Producing each frame requires a solution of the rendering equation at least for each visible point in the rendered image. More details for the rendering equation will be given in the next chapter.

Solving the rendering equation is a computationally expensive process. The cost increases with the number of light sources in the scene and the the geometric complexity of the scene. Traditionally, illumination methods could be placed into two broad categories: (a) those aiming at photorealism, where the emphasis is more on the quality of the final image and less on the computational cost, since they often run into minutes or hours, and (b) those aiming at real-time where the realism is traded off for speed, as essential in many interactive applications. Recently, we see a clear trend towards bringing the two competing aims together as recent techniques are increasingly aimed towards real-time photorealistic images.

However, despite the recent advances available in processing power for computer graphics, a real-time full solution to the rendering equation is still far off; thus it is usually approximated. In order to reduce the computations needed, existing illumination algorithms make several simplifications such as approximations to the environment lighting conditions [2, 38, 67, 77], the scene geometry [67], reflectance properties of materials etc. Consequently, this has a negative impact on the quality of the rendered image.

Moreover, most of the real-time illumination methods reduce run-time computations generally by using precomputations. Precomputations usually have huge memory requirements for storing

the preprocessing values [77] and need up-to several hours of calculations. The requirements of precomputations reduce the usefulness of the proposed algorithms. In addition, parameters that are used at the preprocessing, such as for instance the geometry of the receivers, must stay static at run-time [31] in order for precomputed values to be valid. This restricts the applications where these illuminations algorithm can be used.

Many recent illumination algorithms use *high dynamic range* (HDR) environment maps to represent light sources in the scene. This increases the realism in the scene in two ways: (a) light from all directions contributes to the illumination of the scene, simulating better what happens in reality, and (b) by using HDR values, the intensity of the incoming light from each direction is stored with greater accuracy. However, at the same time this increases the computation cost even more; (a) a huge number of light sources need to be considered, (b) operations with 16-bit or 32-bit floating point values per color channel used to store HDR intensities add an extra cost in comparison with standard case where 8-bit values are used.

By using HDR values for light source intensity, an HDR image is produced as the output of the illumination algorithm. HDR values cannot currently be displayed on standard monitors capable of displaying only *low dynamic range* (LDR) values, i.e. colors with representation using maximum 8-bit per color channel.

An additional pass, to convert the HDR values of the image to LDR values adds an extra computational overhead to the whole rendering process. This conversion is called *tonemapping* and is performed using a *tonemapping operator* that is applied on each frame. There are two categories of tonemapping operators, *local* and *global*. The former category uses more advanced operations giving high quality results but with a lower frame rate and so is not suitable for real-time application, while the latter uses more simplified operations able to run at real-time frame rates but with a compromisation in the quality of the results.

1.4 Approach

In this thesis, we aim for a unified rendering pipeline suitable for real-time realistic HDR renderings. Realistic illumination and tonemapping are two of the most significant bottlenecks in such a pipeline and work is needed to accelerate the two steps.

1.4.1 Illumination

An algorithm that illuminates the virtual scene at real-time frame rates with pleasing quality results suits the requirements of the unified rendering pipeline aimed to be developed in this thesis. In order to have more accurate results it should take into account all-frequency lighting conditions and use the geometry of the scene as it is without any simplifications.

In this thesis we propose a method that addresses the above requirements. It is based on a reformulation of the rendering equation that allows us to pre-integrate the contribution of all light sources in the scene in such a way that the computed values are independent of the geometry of the receiver. As a result, our algorithm's run-time complexity is independent of the number of vertices of the 3D objects and the number of light sources in the scene. This allows us to have high quality renderings since the algorithm can handle all-frequency environment maps and high detailed 3D models without performing any simplifications on them.

When all parameters of a dynamic scene are known, the precomputed values are used at run-time in order to evaluate the radiance at each pixel. This is done very rapidly with few operations thus making the algorithm suitable for real-time applications. The fact that only moderate memory requirements are needed to store precomputed values makes the algorithm practical to use.

The proposed algorithm is based on certain assumptions. We do not investigate the case of multiple bounces of light and we assume diffuse reflectance only in the scene.

1.4.2 Tonemapping

A tonemapping algorithm is needed that produces in real-time realistic results comparable to those achieved with high quality local operators. In this thesis we propose a new framework: *selective tonemapping*, which accelerated tonemapping process.

Selective tonemapping applies the computational expensive component of the local operator only to pixels that are perceptually important. In this way it gains the quality of the local operators but at a reduced cost while allowing real-time frame rates on high resolution images.

1.5 Dissertation structure

The dissertation consists of six chapters. In the next chapter (Chapter 2) we review existing techniques in areas of illumination and tonemapping and discuss their advantages and limitations. The two subsequent chapters, Chapters 3 and 4, contain the core contribution of this thesis and describe in detail the proposed algorithms in illumination and tonemapping, respectively. Extensive results for the two methods are given in Chapter 5. These results include timing and quality performances for testing of the proposed methods under several different parameters as well as results obtained when combining illumination and tonemapping. The final chapter, Chapter 6, presents a discussion of the proposed illumination algorithm and tonemapping framework together with a summary of the strengths and weaknesses of the proposed algorithms. Finally, we discuss future directions on these topics.

Chapter 2

Background knowledge

2.1 Introduction

This chapter aims to familiarize the reader with concepts related to this thesis. Definitions and notions of terms related to *illumination* and *tonemapping* are given. Then related work in each topic is described.

2.2 Definitions

Several terms are used through the literature of illumination and tonemapping algorithms. Very often terms are used in an interchange way and are not used with their accurate meaning. In this section we give the accurate definition of each term aiming their clarification. Measurement unit of each term that describes a quantity, is given in the international system of units (SI).

Radiant energy is the energy of electromagnetic waves. Its measurement unit is Joule (J).

Radiant flux or *radiant power* is the radiant energy that passes per unit time. It is measured in Watts (W).

Radiant intensity is a measure of the intensity of electromagnetic waves. It is defined as radiant power per unit solid angle. It is measured in Watts per steradian ($\frac{W}{sr}$).

Radiance is the amount of light that emitted or reflected from a unit area of a particular surface towards a specified direction that falls into a given solid angle. The measurement unit of the radiance is Watts per steradian per square meter ($\frac{W}{sr \cdot m^2}$).

Irradiance is the power per unit area of incident electromagnetic radiation at a surface. It is measured in Watts per square meter ($\frac{W}{m^2}$).

Radiant emittance or *radiant exitance* is the power per unit area of emerging electromagnetic radiation from the surface. Similar to the irradiance, is measured in Watts per square meter ($\frac{W}{m^2}$).

Radiosity is the emitted plus the reflected power leaving a unit area of a a surface towards all directions. It is also measured in Watts per square meter ($\frac{W}{m^2}$).

Besides the radiometric terms already given, that describe the physical amount of the light energy, there are corresponding terms for the perceived energy of light by the human eye, the photometric terms. The human eye sees only light with a wavelength between the visible spectrum, from about 400nm to 750nm and it has its highest sensitivity at 555nm. The sensitivity of the human eye is given by the luminosity function, Figure 2. The photometric terms are described below.

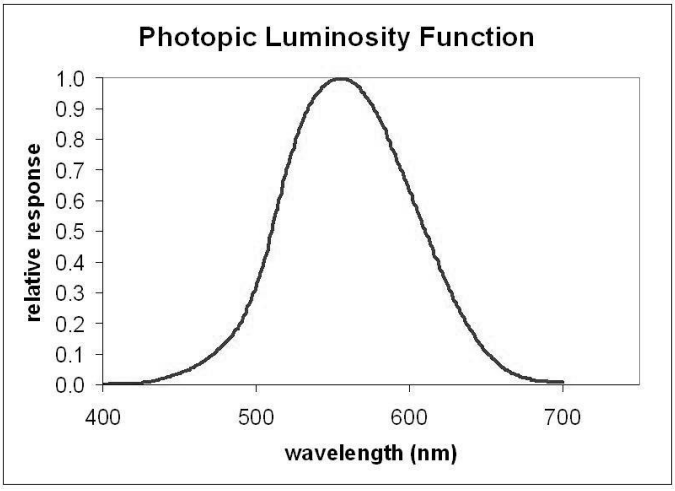


Figure 2: The luminosity function defines the sensitivity of the human eye for different wavelengths of light.

Luminous energy is the perceived energy of the light. It is measured in lumen second ($lm \cdot sec$).

Luminous flux or *luminous power* is the measure of the perceived power of the light, that is the perceived energy that passes per unit time. Its measurement unit is the lumen (lm).

Luminous intensity is a measure of the wavelength-weighted power emitted by a light source in a particular direction per unit solid angle. Its measurement unit is candela (cd) that is equal to lumen per steradian ($\frac{lm}{sr}$).

Luminance measures the luminous intensity of light emitted or reflected in a given direction from a unit area of surface. The measurement unit of luminance is candela per square meter ($\frac{cd}{m^2}$).

Illuminance is the total luminous power incident on a unit area of surface. It is measured in lumens per square meter ($\frac{lm}{m^2}$) or in lux (lx).

Luminance emittance or *luminance exitance* is the luminous flux per unit area emitted from a surface. It has the same measurement unit as illuminance, that is lumens per square meter ($\frac{lm}{m^2}$) or in lux (lx).

Luminosity is the luminous flux per unit area emitted and reflected from a surface. It is also measured in lumens per square meter ($\frac{lm}{m^2}$) or lux (lx).

Often in literature, there is a confusion in which notions have directivity, are normalized values regarding area or time and which regard light incident to or exiting from a surface. Table 1 summarizes this information for these terms.

Local illumination or *direct illumination* is the illumination of a surface taking into account only the light comes directly from the light sources.

Global illumination or *indirect illumination* is the illumination taking into account, besides the direct lighting, the bounces of light onto other surfaces of the scene.

Bidirectional reflectance distribution function (BRDF) is a function $f_p(\vec{\omega_i}, \vec{\omega_o})$ that defines how the light arriving at a point p from a specific direction $\vec{\omega_i}$ (irradiance) is reflected from that

Table 1: Summarization of radiometric and photometric terms.

Term	SI Unit	Per unit time	Per unit area	Per unit solid angle	Directivity
Radiant energy	J	no	no	no	both
Luminous energy	$lm \cdot sec$				
Radiant flux/power	$W = \frac{J}{sec}$	yes	no	no	both
Luminous flux/power	lm				
Radiant intensity	$\frac{W}{sr}$	yes	no	yes	both
Luminous intensity	$cd = \frac{lm}{sr}$				
Radiance	$\frac{W}{sr \cdot m^2}$	yes	yes	yes	exiting
Luminance	$\frac{cd}{m^2}$				
Irradiance	$\frac{W}{m^2}$	yes	yes	no	incident
Illuminance	$lx = \frac{lm}{m^2}$				
Radiant emittance/exitance	$\frac{W}{m^2}$	yes	yes	no	exiting
Luminous emittance/exitance	$lx = \frac{lm}{m^2}$				
Radiosity (emittance + reflectance)	$\frac{W}{m^2}$	yes	yes	no	exiting
Luminosity (emittance + reflectance)	$lx = \frac{lm}{m^2}$				

point towards direction $\vec{\omega_o}$ (radiance). Note that since each direction $\vec{\omega}$ is defined with spherical coordinates (θ, ϕ), BRDF is a four dimensional function. BRDF of an object defines its material properties.

Dynamic range is the ratio between the highest value of luminance that exists in a scene or in an image and the lowest luminance value. It is usually expressed in orders of magnitudes.

High dynamic range (HDR) images are images that encode a vast range of luminances. Most of them are able to represent tenths of orders of magnitudes of dynamic range. On the other hand *low dynamic range (LDR) images* are able to represent only less than two orders of magnitudes of dynamic range.

Tonemapping (TM) is the process of mapping high dynamic range of luminances to low dynamic range. This is done using *tonemapping operators (TMO)* that can be separated to global TMOs and local TMOs.

A *global tonemapping operator* uses the same mapping function for all the pixels in an image. Global TMOs are usually inexpensive to be applied, however they are not able to handle well images with high dynamic range.

A *local tonemapping operator* uses a different mapping function for each pixel or a group of pixels in an image. Local TMOs are able to preserve better the local contrast within an image, however they are more computationally expensive than global TMOs.

2.3 The rendering equation

Illumination in a scene is usually approximated with the rendering equation, Equation 1.

$$L_{out,p}(\vec{\omega_o}) \quad = \quad L_{emit,p}(\vec{\omega_o}) + \int_{\Omega^+_{\vec{N_p}}} L_{in,p}(\vec{\omega_i}) f_p(\vec{\omega_i}, \vec{\omega_o}) \cos < \vec{N_p}, \vec{\omega_i} > d\vec{\omega_i} \qquad (1)$$

$L_{out,p}(\vec{\omega_o})$ is the outgoing radiance from point p towards the direction $\vec{\omega_o}$. Since the rendering equation is usually computed in order to shade the scene in the way that is seen by the observer, $\vec{\omega_o}$ in that case is the direction from point p towards the viewer. Both outgoing direction $\vec{\omega_o}$ and $\vec{\omega_i}$, which is an incident direction to point p, are defined with respect to the surface normal $\vec{N_p}$.

$L_{emit,p}(\vec{\omega_o})$ is the emitted radiance from point p towards the direction $\vec{\omega_o}$. Light is emitted from emitting surfaces such as light sources in the scene. For all the other surfaces $L_{emit,p}$ is a zero value.

The integration evaluates the reflected radiance from point p due to the irradiance arriving from the environment, Figure 3. The integral is over all the directions in the positive hemisphere of the receiver as it is defined by its normal direction $\vec{N_p}$. $f_p(\vec{\omega_i}, \vec{\omega_o})$ is the BRDF of the surface that point p lies on. The cosine within the integral between the incident direction $\vec{\omega_i}$ and the normal direction $\vec{N_p}$ exists in order to take into account Lambert's law [47] that implies that.

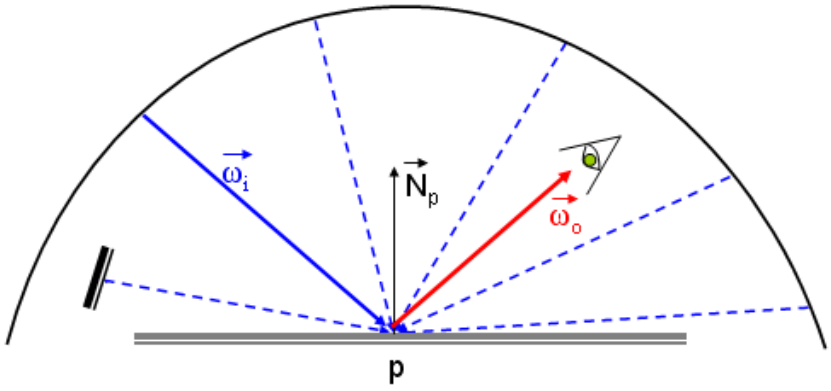


Figure 3: The radiance reflected towards viewer direction $\vec{\omega_o}$ is an integration of the irradiance arriving at point p over all incident directions $\vec{\omega_i}$ at the positive hemisphere of the receiver.

2.4 Related work on illumination

Many illumination algorithms have been proposed over the last decades, giving this area one of the largest bibliography in the computer graphics field [15, 21, 32, 35, 89]. A trade off between frame rate and realism exists for illumination algorithms.

In one extreme we have those methods that aim realistic results providing global illumination of the scene usually at offline frame rates. On the other extreme we have those techniques aiming for high frame rates for rendering, without giving emphasis on the quality. These techniques provide only direct lighting usually assuming only few point light sources in the scene. Between these two categories, several methods exist aim towards pleasing quality results in real-time frame rates.

2.4.1 Global illumination

Ray-tracing and radiosity are the two most known traditional offline global illumination methods. Both make assumptions for the BRDF of the objects of the scene.

Ray-tracing in its original form [99] is an image based algorithm that uses rays to evaluate in a recursive way indirect illumination at each pixel. It assumes perfect specular reflection

and transparent objects in the scene. The literature is rich with algorithms based on ray-tracing technique, mainly focused on acceleration of it.

Hierarchical data structures are one of the classic tools used for rendering acceleration. Such data structures include bounding volume hierarchies (BVH), octrees, binary space partitioning (BSP). Effectiveness of those techniques closely depends on actual scene that will be used. Recently we have seen renewed interest in this area especially in the context of ray-tracing. We have seen work on kd-tree based ray tracers [23, 68], grids [65], and even for beam tracers [61]. However building such acceleration structures requires seconds to minutes so they can not be used for animated scenes. More recent works build or modify acceleration structures in every frame in order to be able to use them for animated scenes [87] or deformable scenes [86]. In [104] in order to handle dynamic scenes perform selective restructuring operations to reconstruct small portions of a BVH instead of the entire BVH.

Few years after classic ray-tracing was proposed, a new algorithm called *radiosity* was developed at the Cornell University. It was aiming to reveal the main limitation of the ray-tracing that was the inability to handle diffuse surfaces. Radiosity is a scene based algorithm, where the scene is subdivided into surface elements, usually called patches. It is based on the exchange of radiance between the patches until the equilibrium of light distribution in the scene is achieved. Similar to the ray-tracing, the radiosity algorithm received a lot of attention and many research papers came out that were based on the original algorithm.

Multipass methods aiming to take the advantages of both ray-tracing and radiosity have been proposed [10, 75, 90]. Such methods usually have a first pass using radiosity method and then a ray-tracing pass follows that pick up values computed at the first pass. Moreover, in order to speed up the process *monte carlo techniques*, aiming to give a stochastic solution to the rendering equation, have been used among both ray-tracing and radiosity [71, 72].

A totally different approach for global illumination is ambient occlusion [106]. In this class of methods instead of solving the full recursive illumination equation on the surfaces, surfaces are shaded by approximation, where the global part is replaced by the visibility over the hemisphere. Ambient occlusion, in its standard form, precomputes the visibility at each point in the scene and shades the model by multiplying the visibility value by a constant factor. More recent methods focus on handling inter-object ambient occlusion, increasing more the realism in the result such as the work of [40, 105] for moving rigid objects. Another goal of the work at this area is to compute ambient occlusion for deformable models [39, 70].

2.4.2 Local illumination using point light sources

The literature is rich with many different approaches for computations of soft-shadows [28]. Some of the most popular ones are extensions of the shadow volume [12] or shadow buffer ideas [100] taking into account multiple samples on an area light source. An alternative is the fake shadow methods [9, 103] aiming to produce plausible rather than accurate soft-shadows. In all of these shadow methods, the source is assumed to be of a relatively small finite extent. It is rather hard to apply these soft-shadows methods to illuminate a scene using a full environment map. To account for environment maps, clusterization techniques are applied on light sources to make them more manageable [29, 38].

2.4.3 Methods aiming real-time high quality results

In this section we describe methods aiming real-time high quality results, taking into account lighting arriving from all over the directions, in contrast with methods mentioned in previous section, increasing the quality but making the computations more expensive. To speed-up the

process several acceleration techniques are used that are described below. A common assumption in these cases is that lighting is far enough and it's treated as directional lighting.

2.4.3.1 Reduced number of light sources

Assuming direct lighting only, the rendering equation becomes an integral over all light sources in the environment. In order to have realistic results many light sources have to be taken into account. Some times the number of light sources reaches thousands or even millions such as in case where a high resolution environment map is used that represents a huge number of directional light sources. Methods to compute representative light sources, have been recently proposed in order to reduce the number of light sources used. Kolling et al [38] use Voronoi diagrams to compute representative light sources, for a high dynamic range image, without any manual intervention. [94] adaptively samples the environment based on an importance metric (e.g. brighter areas are more important), by iteratively subdivide the sphere into quads. [7] not only takes into account light source energy in order to sample the environment, but also the surface's BRDF. At [29] they compute representative light sources for high dynamic range video environment map and not only for a static image representing an environment map.

Recently lightcuts a clustering method of light sources was proposed by [93]. Although lightcuts do not achieve real-time frame rates are mentioned here, since they are scalable to large numbers of light sources and give realistic results and they hold a lot of promise. At run-time it is chosen based on error metrics which lights clusters will be used to compute illumination. This technique was extended in order to handle effects such as motion, depth of field, temporal and spatial anti-aliasing at [92]. Another clustering algorithm for environment lights proposed at [33], where the clustering is based on similarities of light transfer vectors (i.e. radiance transfer function of all vertices with respect to one specific light source). They can achieve interactive rates even

with local changes to the environment map, by recompute radiance at vertices only from clusters affected from environment changes.

In our algorithm soft-shadows are computed using the whole environment map with no clusterization and no assumptions on light sources number, size or intensity.

2.4.3.2 Simplification of BRDF

Some methods achieve lighting from the environment by simplify the BRDF of the receiving objects. By keeping the distant lighting in an environment map, one can easily produce specular reflections in real-time by simply using a texture mapping technique such as cube mapping and map the environment on the scene. [1] and [84] create at a preprocessing step a diffuse map so they are able to have diffuse reflections as well. However in these works they do not consider occlusions to compute shadows, as we do.

2.4.3.3 Occlusion precomputation

Occlusion computation is typically the most time consuming part of any high quality illumination algorithm. It is thus not surprising that the idea of pre-sampling the occlusion of objects for dynamic scenes, has been around for a while in various applications: radiosity [60], ambient occlusion [40, 50], shadows [105] and inter-reflections [51]. Among these, the method of Zhou et al. [105] is closer to ours, since it also allows the inclusion of all-frequency illumination. In their method, for each sample view taken around an occluder, a complete occlusion field is stored in contrast our method where the occlusion field is only used at preprocessing as an intermediate step to compute a value that will be stored. Although their approach can also account for local lights, it requires storing hundreds of MBytes of memory per object and has a computational intensive run-time step. Our method reduces the memory requirement by two orders of magnitude and more

than an order of magnitude speed up at run-time, at the expense of having a fixed environment map. Ren et al. [67] can handle dynamic occluders. They achieve this by approximating the occluders with spheres and precompute the occlusion. However because of this approximation they are able to handle only low-frequency incident lighting and shading effects.

2.4.3.4 BRDF and environment radiance approximation

One approach that has been widely used is to approximate one or more terms of the rendering equation (such as BRDF, environment radiance or a combination) by breaking them down to few basis functions. At run-time the final result is computed by combining the basis functions using the appropriate coefficients for each one. In this way the rendering equation can be solved with fewer operations.

Sloan et al. [77] proposed representing radiance transfer functions (which are precomputed - PRT) and incident lighting on spherical harmonics basis functions. At run-time the appropriate coefficients are used to combine the basis functions representing the transferred radiance and make the convolution with BRDF in order to compute the exiting radiance at each point.

Since with SH only few coefficients are used to represent light information, high-frequency environments are difficult to be represented. In order to overcome this limitation other techniques have been proposed that encode lighting information as wavelets [27, 41, 58], or as radial basis functions [81], within a cube map [49]. In [95] they represent both the direct lighting and precomputed diffuse indirect transfer using a spectral mesh basis and they achieve multi-bounce indirect lighting in real-time.

One of the problems of these techniques is that basis functions are precomputed for every vertex in the scene, requiring expensive preprocessing and has high memory requirements. In contrast, our

method does precomputations per different type of object in the scene and its memory requirements is two orders of magnitude less than requirements of PRT methods.

Precomputed radiance transfer has been extended to dynamic radiance transfer. This has been used in dynamic scenes [36] achieving only interactive frame rates for scenes with low geometry complexity and in static scenes with dynamic local lighting conditions, simulated global illumination effects [41, 43]. Proposed method at [31], also takes into account inter-reflections of light between surfaces with diffuse and glossy BRDFs for moving rigid objects. It achieves this by considering the objects as secondary light sources.

Illumination algorithm that is proposed in this thesis, uses environment maps and assumes distant lighting with only directional light sources. However, it does not make any approximations in the incoming radiance, neither for the number of the light sources, nor on their intensity as the techniques described above. Although it assumes only diffuse objects in the scene, it considers occlusions and is able to compute soft-shadows.

2.4.3.5 Rendering equation reformulation

The idea of decoupling the receiver's normal from the incident radiance has been employed before [22, 76]. Everitt [22] proposed a method in the days when per-pixel illumination was not yet possible with graphics hardware. In this method, the three components of the normals (x, y, z) of the pixels needed to be stored in one texture each, therefore the geometry of the receivers needed to be known in advance, in contrast to our method where the receiver's geometry can be changed dynamically at run-time. Moreover the computation time was linear with the number of light sources in the scene. Our proposed algorithm has a static frame rate for arbitrary number of light sources. Sloan [76] used the idea of decoupling the normal of the receiver from incident irradiance in combination with PRT in order to handle normal maps. However since the radiance

transfer is precomputed, this technique can not handle fully dynamic receivers and works only for rigid objects.

2.4.4 Acceleration techniques

Besides the techniques that we have already mentioned and which are often tightly coupled with each proposed algorithm, there are some tools that any method can potentially use in order to decrease even more the computation time. Such techniques are stopping criteria based on perceptual methods, share computation on cluster of PCs, use of hardware acceleration by GPU programming or reusing already computed values. These methods are described below.

2.4.4.1 Perceptual methods

Since rendering with correct illumination is a very expensive process, rendering algorithms should not spent more recourses (time, memory etc) if the improvement on rendering can not be perceived, that is the resulted image is not noticeably different from the image produced with a full solution algorithm.

Work has been done on the development of perceptual metrics [78]. These metrics are used either as stopping criteria for illumination algorithms [64, 91] or in order to choose which factors are more important (affect noticeable things) in order to dedicate more recourses for those [18, 57, 79].

Recently [17] proposed a perceptual rendering framework that allows rendering algorithms to make decisions for rendering parameters such as use of appropriate LoD.

2.4.4.2 Multiple processors

The exponential increase in the processing power that we have been witnessing over the last decades was achieved mainly through the exponential increase in the number of processors in the

CPU. Recently due to heat and power consumption issues we see shift towards the use of multiple processing units. We have recently seen a number of CG papers that have adapted or developed new techniques to operate on such systems. Multi processor systems can in principle benefit most graphics applications. But in particular ray tracing researchers have taken advantage of them, either as multi-core systems [85] or clusters of PC's [88].

2.4.4.3 Graphics processors units

Due to recent advances in graphics hardware many proposed algorithms are either explicitly or partly implemented with shaders on GPUs in order to speed up the computation and rendering process. However due to some limitation of graphics hardware (such as maximum number of instructions), implementation on GPU is not always straight forward from the implementation of CPU. Besides that, programmers must consider the differences on architecture between GPU and CPU in order to really take advantage of GPU usage (e.g. parallelization). Many shading languages can be used to program the two and recently three programmable steps of modern GPUs' pipeline, by writing pixel, vertex and geometry shaders. Such shading languages are Cg, GLSL, HLSL or even more high level such as CUDA.

2.4.4.4 Caching and coherence

Reusing already computed values is another way to decrease the overall computation time. There are two ways to do that. One is to reuse values within the same frame (caching) and the other is to reuse values from one frame to the other (coherence).

Irradiance caching is a technique for reducing the computation time of indirect illumination in diffuse scenes [97]. It is based on the observation that incident illuminance varies smoothly over a surface. Irradiance is computed at sample points and cached in a data structure. The cached

values are interpolated to approximate irradiance at nearby surfaces, so expensive computations for illumination computation are reduced. [24] demonstrates how to avoid the nearest-neighbors queries and spatial data structures so the idea of irradiance caching can be efficiently implemented on GPUs that can not handle efficiently complex data structures. Radiance caching, although is not a real-time technique, [44] generalizes irradiance caching to glossy surfaces with low-frequency BRDFs. It was later extended [45] to an adaptive algorithm for guiding spatial density of the cached values in radiance and irradiance caching without decreasing in a noticeable way the performance.

Laine et. al. [46] use coherence techniques on standard instant radiosity [37] by reusing the virtual point lights (VPLs) from previous frame that are still valid. In [74] instead of only sampling VPLs from the light source they generate them also considering the camera's position, achieving interactive frame rates for instant radiosity methods.

Our technique does not use caching or coherence aiming higher accuracy in the computed values. The run-time part of the illumination algorithm proposed in this thesis has been explicitly developed on GPU using GLSL programming language. Our algorithm could also benefit from the other two acceleration techniques described here. Perceptual methods could be used for finding the minimum number of samples needed at the preprocessing. Perceptual experiments are demonstrated in the Chapter 5. Due to independence of illumination computation between pixels, provided by our algorithm, the algorithm could run on a cluster of PCs, distributed the processing power needed.

2.5 Related work on tonemapping

The concept of tonemapping (TM) was introduced by Tumblin and Rushmeier in [82], where they proposed a tone reproduction operator that preserves the apparent brightness of scene features. Subsequently many tonemapping operators (TMOs) have been proposed. Most TM methods

concern accurate operators that attempt to reproduce individual visual effects at non-interactive rates.

The interactive solutions to the TM problem can be classified in two main categories: direct GPU implementation of the original TMO, and definition of a general acceleration platform. The first one refers to the implementation of the original CPU implementation of the TMOs directly on the GPU. This often requires a significant modification of the original CPU implementation. As a result of this we have reduced quality when compared with the output obtained with the original TMO. In addition the time performances are rapidly decreasing as the resolution of the input frame increases [25]. The second category, aims to develop a framework that can be applied to the current state of art TMOs in order to achieve interactive rates. [3] proposed an efficient subdivision of the workload between CPU and GPU. The main advantage of this idea is that no modifications are required to the original TMOs, that are still implemented on the CPU.

Several authors, including Durand and Dorsey [19, 20] and Ward et al. [98], have proposed some acceleration methods in order to improve the computational performance of their TMOs. Global TMOs which achieve interactive rates, tightly coupled with current graphics hardware, have been proposed by Cohen et al. [11] and Scheel et al. [73]. Goodnight et al. [25] discussed the troublesome aspects of the GPU programming and presented a hardware implementation of the Reinhard operator [66].

The local TMOs are concerned with achieving some form of perceptual accuracy of individual visual effects. This requires significant computational effort and none of these TMOs currently can operate at high frame rates for very large images. Some other authors including [25, 42], implemented local TMOs tightly coupled with current graphics hardware achieving interactive rates only for low frame resolutions.

Cadik [8] presented an approach that validated the main idea of applying different TMOs on different areas of the HDR input image called hybrid approach. This technique must carefully select the local and global TMOs in order to reproduce high quality output images. Due to its behavior it requires a blending technique to smooth the boundaries between different regions (enhancement map). This work concentrated on producing pleasant output images without addressing any real-time issues.

Chapter 3

Real-time illumination

3.1 Introduction

A correct account for illumination and shadows produced by complex environmental lighting can greatly improve the visual realism of real-time applications such as training simulators, computer games, CAD programs etc. The process is computational intensive since it requires a weighted integral over all light sources in the scene, taking into account expensive occlusion calculations. The cost increases with the number of light sources in the scene and thus typically the quality is compromised by not considering all-frequency environments [38, 67, 77].

Real-time illumination methods, in order to reduce the computations needed at run-time, make use of precomputations that usually have heavy memory requirements for storing the precomputed values. In addition, the use of precomputations adds the limitation that static parameters must be used such as static or rigid receivers [31].

In this thesis we introduce a technique that can be used for real-time illumination with shadows from all frequency environment maps, for fully dynamic receivers, and with only moderate needs in memory space, Figure 4.

Figure 4: Real-time illumination with soft-shadows for fully dynamic receivers from all frequency environments.

The method is based on a reformulation of the irradiance computation that allows us to factor out the direction of the receiver's normal, for diffuse surfaces, [102, 101, 48]. We introduce a new notion, the *fullsphere irradiance* that is a modification of the irradiance in that it takes in account energy arriving at a point from all the light sources in the scene and not only from those that lie in the positive hemisphere of the receiver. Using a factorization, we are able to encode the fullsphere irradiance within a 3D vector, called *fullsphere irradiance vector* (FIV), whose values are valid for any direction of the receiver.

We applied our technique for computing in real-time inter- and intra-object soft-shadows by encoding also the occluded irradiance using FIVs. A FIV can be precomputed for the part of the environment hidden by each occluder, placing the occluder each time at a position of a dense set of sample points. Then, at run-time, given a point on a scene surface, we can use the FIV values to quickly compute the unoccluded irradiance using merely a dot product per occluder, per color channel, of the receiver's normal.

The contribution of this thesis, in research for illumination algorithms, is two-fold: (i) it introduces the notion of the fullsphere irradiance and fullsphere irradiance vector, which allow a pre-integration of light sources contribution, independently to the receiver's normal, (ii) it proposes a simple and scalable method that employs this novel idea and computes both soft and hard shadows in complex scenes from all-frequency lighting conditions (environment maps). Our results, show a

GPU implementation of the latter yielding high frame rates even for scenes with dozens of dynamic occluders and receivers.

3.2 Fullsphere irradiance vector

The reflected radiance $L_{out,p}$ in direction $\vec{\omega_o}$ at a point p, of a non-emitting surface, with surface normal $\vec{N_p}$ and BRDF $f_p(\vec{\omega_i}, \vec{\omega_o})$ is given by the following equation:

$$L_{out,p}(\vec{\omega_o}) = \int_{\Omega^+_{\vec{N_p}}} L_{in,p}(\vec{\omega_i}) f_p(\vec{\omega_i}, \vec{\omega_o}) \cos <\vec{N_p}, \vec{\omega_i}> d\vec{\omega_i} \tag{2}$$

where $L_{in,p}(\vec{\omega_i})$ is the radiance coming from direction $\vec{\omega_i}$ and $\Omega^+_{\vec{N_p}}$ is the positive hemisphere of the point p.

Assuming diffuse reflectance only, the BRDF is constant and the term $f(\vec{\omega_i}, \vec{\omega_o})$ can be substituted by a reflectance factor ϱ_{dif} which can be factored out from the integral and thus the radiance equation for diffuse surfaces becomes:

$$\begin{aligned} L_{out,p}(\vec{\omega_o}) &= \varrho_{dif} \int_{\Omega^+_{\vec{N_p}}} L_{in,p}(\vec{\omega_i}) \cos <\vec{N_p}, \vec{\omega_i}> d\vec{\omega_i} \\ L_{out,p}(\vec{\omega_o}) &= \varrho_{dif} I_p(\vec{N_p}) \end{aligned} \tag{3}$$

where $I_p(\vec{N_p})$ is the irradiance arrived at the point p from all directions in the positive hemisphere of p, i.e.

$$I_p(\vec{N_p}) = \int_{\Omega^+_{\vec{N_p}}} L_{in,p}(\vec{\omega_i}) \cos <\vec{N_p}, \vec{\omega_i}> d\vec{\omega_i} \tag{4}$$

We introduce here a new term, *fullsphere irradiance*, FI, that differs from the irradiance equation in that the integral is over **all** directions of the sphere,

$$FI_p(\vec{N_p}) = \int_{\Omega_{\vec{N_p}}} L_{in,p}(\vec{\omega_i}) \cos <\vec{N_p}, \vec{\omega_i}> d\vec{\omega_i} \tag{5}$$

where $\Omega_{\vec{N_p}}$ denotes all directions over the whole sphere.

Conceptually, the fullsphere irradiance differs from the irradiance, in that "we let" lighting behind the surface of the point p to affect the value of $FI_p(\vec{N_p})$. Note that $\cos < \vec{N_p}, \vec{\omega_i} >$ may have negative values.

Assuming distant lighting, light sources become directional and thus the computation of the irradiance becomes independent of the position's coordinates of the receiving point p. This is the case when the illumination is computed from an environment map [16]. We will refer to this environment map, as the *lighting environment*, Env. For M discrete directional light sources in the scene, from which the M^+ lie in the positive hemisphere of the receiver the equations of the irradiance and fullsphere irradiance become as below:

$$I_p(\vec{N_p}) = \sum_{\vec{\omega_i} \in M^+} (L_{in}(\vec{\omega_i}) \cos < \vec{N_p}, \vec{\omega_i} >) \tag{6}$$

$$FI_p(\vec{N_p}) = \sum_{\vec{\omega_i} \in M} (L_{in}(\vec{\omega_i}) \cos < \vec{N_p}, \vec{\omega_i} >) \tag{7}$$

Note that in cases when all light sources in the scene, are in the positive hemisphere of the receiver then the values of the irradiance and the fullsphere irradiance are equal:

$$I_p(\vec{N_p}) = FI_p(\vec{N_p}) \textit{ when } M = M^+ \tag{8}$$

In the irradiance equation, Equation 6, the normal $\vec{N_p}$ of the receiving point p is within the summation. This means that the irradiance can be accumulated only once the normal $\vec{N_p}$ is known. For dynamic receivers this implies that this costly computation will necessarily have to be computed online. The irradiance equation can be rearranged to factor out the normal from the summation. By replacing the cosine of the unit vectors $\vec{N_p}$ and $\vec{\omega_i}$ with the dot product of the two vectors the summation can be broken into three components.

$$
\begin{aligned}
I_p(\vec{N_p}) &= \sum_{\vec{\omega_i} \in M^+} (L_{in}(\vec{\omega_i}) \cos < \vec{N_p}, \vec{\omega_i} >) \\
&= \sum_{\vec{\omega_i} \in M^+} (L_{in}(\vec{\omega_i})(\omega_{i_x} N_x + \omega_{i_y} N_y + \omega_{i_z} N_z)) \\
&= N_x \sum_{\vec{\omega_i} \in M^+} (L_{in}(\vec{\omega_i})\omega_{i_x}) + N_y \sum_{\vec{\omega_i} \in M^+} (L_{in}(\vec{\omega_i})\omega_{i_y}) + N_z \sum_{\vec{\omega_i} \in M^+} (L_{in}(\vec{\omega_i})\omega_{i_z}) \\
&= (N_x, N_y, N_z)(\sum_{\vec{\omega_i} \in M^+} (L_{in}(\vec{\omega_i})\omega_{i_x}), \sum_{\vec{\omega_i} \in M^+} (L_{in}(\vec{\omega_i})\omega_{i_y}), \sum_{\vec{\omega_i} \in M^+} (L_{in}(\vec{\omega_i})\omega_{i_z})) \\
I_p(\vec{N_p}) &= \vec{N_p} * IV(Env_{M^+}) \qquad (9)
\end{aligned}
$$

where $IV(Env_{M^+})$ is the *irradiance vector* of an environment map or a scene Env with M^+ light sources lying in the positive hemisphere of p, similar to the vector irradiance used for radiosity computation in [48, 101, 102]. The irradiance vector is a 3D vector per color channel. Having computed the $IV(Env_{M^+})$ the irradiance can be calculated using only a dot product between the normal of the receiver $\vec{N_p}$ and the $IV(Env_{M^+})$, see Equation 9. However we still have the issue of determining the M^+ light sources which still depend on the orientation of the receiving surface.

To eliminate this constrain of the dependency on the normal of the receiver, we introduce here the term of *fullsphere irradiance vector*. The fullsphere irradiance vector is similar to the irradiance vector in the same way that the fullsphere irradiance is similar to the irradiance. That is, in the fullsphere irradiance vector we consider to the calculation all the light sources, even those that are in the negative hemisphere of the receiver.

$$
FIV(Env_M) = (\sum_{\vec{\omega_i} \in M} (L_{in}(\vec{\omega_i})\omega_{i_x}), \sum_{\vec{\omega_i} \in M} (L_{in}(\vec{\omega_i})\omega_{i_y}), \sum_{\vec{\omega_i} \in M} (L_{in}(\vec{\omega_i})\omega_{i_z})) \qquad (10)
$$

Note that in the same way we prove the Equation 9, we can prove that:

$$
FI_p(\vec{N_p}) = \vec{N_p} * FIV(Env_M) \qquad (11)
$$

Since FIV does not have any dependency on the normal of the receiver, FIV for any Env, can be precomputed once and be valid for any dynamic receiver.

3.3 Illumination using FIVs

3.3.1 Overview

Irradiance $I_p(\vec{N_p})$ at any given point p in the scene can be computed as the total irradiance from the whole environment assuming no occluders in the scene $I_{tot}(\vec{N_p})$ minus the irradiance intercepted by the occluders $I_{occ,p}(\vec{N_p})$ [13], see Figure 5.

$$I_p(\vec{N_p}) = \quad I_{tot,p}(\vec{N_p}) - I_{occ,p}(\vec{N_p}) \tag{12}$$

This can be proved as shown below:

M^+ : The set of all light sources in the scene in the positive hemisphere of the receiver

M^+_{occ} : The set of occluded light sources in the positive hemisphere of the receiver

M^+_{un} : The set of unoccluded light sources in the positive hemisphere of the receiver

$$M^+ = M^+_{occ} \cup M^+_{un}, \qquad M^+_{occ} \cap M^+_{un} = \emptyset$$

$$I_{tot,p} = \sum_{\vec{\omega_i} \in M^+} (L_{in}(\vec{\omega_i}) \cos < \vec{N_p}, \vec{\omega_i} >)$$

$$I_{occ,p} = \sum_{\vec{\omega_i} \in M^+_{occ}} (L_{in}(\vec{\omega_i}) \cos < \vec{N_p}, \vec{\omega_i} >)$$

Proof:

$$
\begin{aligned}
I_p(\vec{N_p}) &= \sum_{\vec{\omega_i} \in \{M^+_{un}\}} (L_{in}(\vec{\omega_i}) \cos < \vec{N_p}, \vec{\omega_i} >) \\
&= \sum_{\vec{\omega_i} \in \{M^+ - M^+_{occ}\}} (L_{in}(\vec{\omega_i}) \cos < \vec{N_p}, \vec{\omega_i} >) \\
&= \sum_{\vec{\omega_i} \in M^+} (L_{in}(\vec{\omega_i}) \cos < \vec{N_p}, \vec{\omega_i} >) - \sum_{\vec{\omega_i} \in M^+_{occ}} (L_{in}(\vec{\omega_i}) \cos < \vec{N_p}, \vec{\omega_i} >) \\
I_p(\vec{N_p}) &= I_{tot,p}(\vec{N_p}) - I_{occ,p}(\vec{N_p})
\end{aligned}
$$

In a similar way we can prove that:

$$
FI_p(\vec{N_p}) = FI_{tot,p}(\vec{N_p}) - FI_{occ,p}(\vec{N_p}) \tag{13}
$$

Note that $FIV(Env_M)$ which can be used to calculate $FI_{tot,p}(\vec{N_p})$ is independent of the normal of the receiver since for any N_p we sum up all M light sources of the environment map. This is not the case for $I_{tot,p}(\vec{N_p})$ since in $IV(Env_{M^+})$, the determination of the M^+ light sources require prior knowledge of $\vec{N_p}$.

Our algorithm uses the Equations 12 and 13 proved above, to compute at run-time the irradiance arrived at a point p of a dynamic receiver, for shadows and self-shadows respectively. The distinction between the way we compute shadows and self-shadows lies in the fact that in case of shadows we may have light sources in the negative hemisphere of the receiver that are not occluded by any object, while in case of self-shadows light sources in the negative hemisphere are definitely self-occluded. The proposed algorithm has two stages; the preprocessing and the run-time stage. At the preprocessing we precompute values that are independent of the dynamic parameters of the scene, such as occluders positions and receivers' geometry.

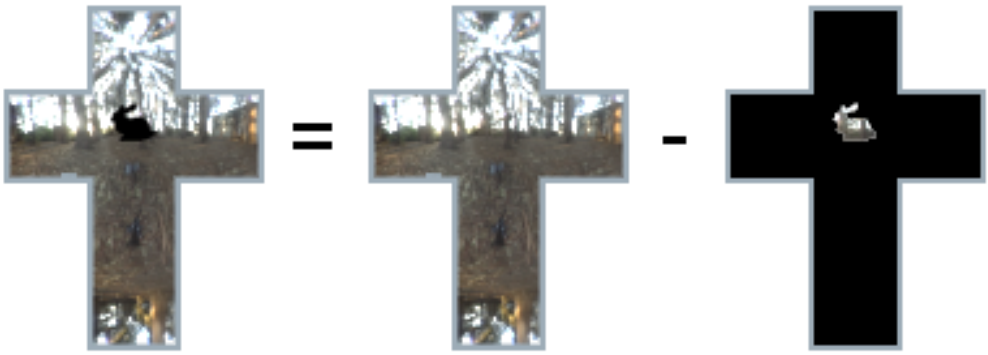


Figure 5: The irradiance I_p arriving at a point p that lies under the bunny (left), is equal to the total irradiance $I_{tot,p}$ (middle) minus the irradiance intercepted by the bunny $I_{occ,p}$ (right).

Precomputed data include: irradiance arriving for each possible normal direction assuming no occlusion, $I_{tot,p}(\vec{N_p})$, fullsphere irradiance vector for the whole environment, $FIV(Env_M)$ and $FIVs$ for occluded parts of the environment placing each occluder in a number of sample positions.

The precomputed data are used at run-time to compute irradiance $I_p(\vec{N_p})$, arriving at each point p of the geometry seen from each pixel. To evaluate irradiance $I_p(\vec{N_p})$, taking into account occlusions by other objects, precomputed values of $I_{tot,p}(\vec{N_p})$ and $FIVs$ are used. In case of self-shadows the $FIVs$ are used among the $FIV(Env_M)$ of the whole environment. In both cases, computation of shadows and self-shadows, the equality between fullsphere irradiance and irradiance, is used whenever is valid, see Equation 8, to transform from the notion of fullsphere irradiance for which we have precomputed information, to the notion of irradiance that is what we want to evaluate.

In the next sections we describe in detail the preprocess and run-time stages.

3.3.2 Preprocessing

At preprocessing we have to compute two categories of data that are needed at run-time:

Figure 6: The total diffuse irradiance I_{tot} from the whole environment map, for each possible normal direction $\vec{N_i}$ of the receiver (left) is precomputed and stored in a cube texture (right).

- Total diffuse irradiance for each sample normal direction, assuming no occlusion.

- A fullsphere irradiance vector, FIV for a number of environment maps (the *lighting* and *occluded environment maps*).

In both cases we treat the environment map as a distant scene [16]. The distant scene is considered far enough from the objects so its incoming light is seen as directional and the irradiance arriving at any point of the local scene is independent of its absolute position. The irradiance at a point is affected though by the relative position of the other objects in the scene (local scene), due to shadows. In other words, each point p is illuminated as the whole scene is translated to place point p, in the origin of the coordinate axes.

For the precomputation of the total diffuse irradiance we create the *diffuse environment irradiance map*, ($DifMap$) [56]. It is stored in an HDR cube texture, Figure 6. Each texel of the $DifMap$, stores the total irradiance $I_{tot,p}(\vec{N_p})$ arriving at the surface with normal $\vec{N_p}$ from the *lighting environment* Env, assuming no occluders in the scene (computed using Equation 6). The $DifMap$ is indexed by the direction of the surface normal $\vec{N_p}$.

The second category of the data precomputed, fullsphere irradiance vectors, are computed using Equation 10, for a number of environment maps. The $FIV(Env)$ of the lighting environment

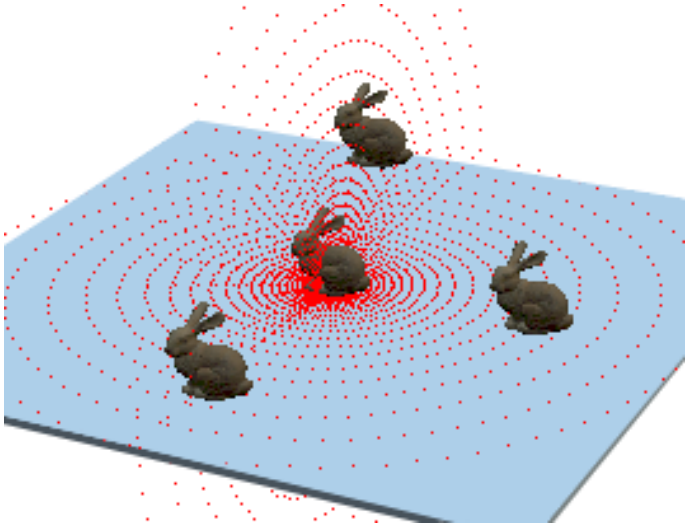


Figure 7: Sample positions of the occluders.

map Env is computed and stored in a structure of a 3D color vector. This will be used to evaluate $FI_{tot,p}$, that is the fullsphere irradiance arriving at a point p from all over the environment Env.

Moreover we compute the FIV of each occluded environment map, $EnvOcc$. $EnvOcc_{(\vartheta_i,\varphi_j,\varrho_k)}$ is the environment map representing the part of the lighting environment map, Env, occluded by an occluder placed at the position $(\vartheta_i, \varphi_j, \varrho_k)$ in spherical coordinates, see Figure 8, right. An $EnvOcc$ is computed at a number of sample positions for each occluder. The sample positions form a dense set (ϑ, φ) around the center of the scene and they are taken in concentric spheres of increasing distance ϱ, Figure 7, similar to [105]. The first sphere, that is used to take samples on it, tangents the inner outline of the occluder. Then the distances of the spheres increases logarithmically and thus we have more samples near the occluder and less samples away from it where shadows are imperceptible.

To create an occluded environment map, $EnvOcc_{(\vartheta_i,\varphi_j,\varrho_k)}$, we place the occluder at the sample position $(\vartheta_i, \varphi_j, \varrho_k)$ and compute the binary cube mask that defines the occluded part of the environment (Figure 8, left), similar to object occlusion field in [105]. Using an 'AND' boolean operation, pixel to pixel between environment map (Figure 8, middle) and the binary mask we get

Figure 8: The binary mask denoting the part of the environment map covered by the occluder (left) is ANDed with the environment map (middle) to get the occluded part of the environment map (right).

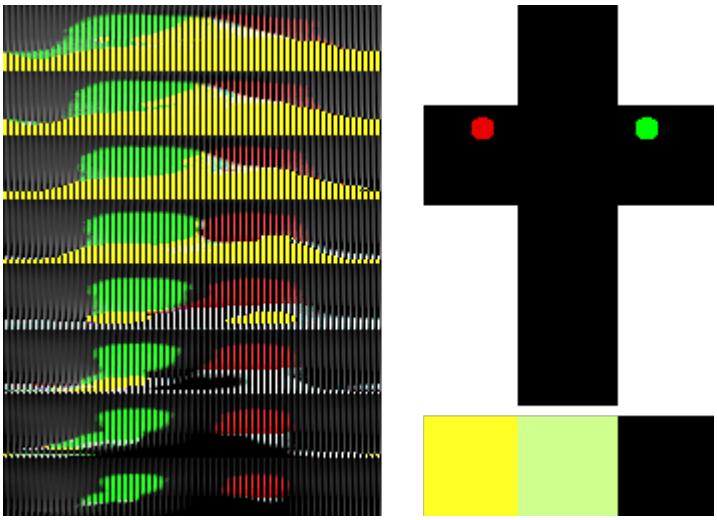


Figure 9: The part of the FIVs texture that stores the precomputed FIVs (left) for 8 different distance samples for the bunny in an environment map with one green and one red area light sources (right-top). The FIVs texture comprises of triplets of colors (right-bottom), each representing the precomputed FIV at a specific sample position of the occluder.

the part of the environment map that is occluded by the occluder in the specific sample position (Figure 8, right).

The FIV is computed for each one of the occluded environment maps created. All $FIVs(EnvOcc)$ precomputed, are encoded within a floating point 2D texture, that we call *FIVs texture*, Figure 9, left. The FIVs texture comprises of triplets of colors, each representing the three components of a precomputed FIV at a specific sample position of the occluder. The information in the texture is arranged in such a way that the texture is near to a square shape. Square textures have

the advantage of a faster look up [80]. In the horizontal axis, we have FIVs values for the different occluder types and different samples at ϑ direction. In the vertical axis of the FIVs texture we have FIVs for different samples at ϱ and φ direction.

3.3.3 Run-time

In order to illuminate our scene at run-time, we need to compute the radiance exiting each pixel, $L_{out,p}$. Since we assume only diffuse objects in the scene, the radiance at any pixel p is equal to the irradiance $I_p(\vec{N_p})$ multiplied by the diffuse coefficient of the material of the receiver, Equation 3. The run-time computation of $I_p(\vec{N_p})$ is done using the precomputed values of $FIVs$, taking into account shadows and self-shadows as described in the next subsection.

All run-time computations have been implemented on GPU to speed up the computation time. Moreover, the implementation in a fragment shader, allow us to have per pixel illumination.

3.3.3.1 Shadows

To compute the irradiance arriving at each pixel $I_p(\vec{N_p})$ we sum up the irradiance occluded $I_{occ,p}$ by all occluders and deduct it from the total irradiance, $I_{tot}(\vec{N_p})$, Equation 12.

Total irradiance

To get the $I_{tot}(\vec{N_p})$, within the pixel shader, we use the normal (in world coordinates) of the geometry visible through the pixel, to look up in the cube texture of the diffuse environment irradiance map $DifMap$.

Occluded irradiance

To get the total occluded irradiance $I_{occ,p}$ we sum up the occluded irradiance from each object in the scene. Since we assume a distant lighting environment, we only need to consider the relative position of each occluder to the receiver. There are three cases; the occluder may lie fully in the positive hemisphere of the receiver (Figure 10 case A), partially in the positive hemisphere (Figure 10 case B) or fully in the negative hemisphere of the receiver (Figure 10 case C). Cases C, do not occlude any irradiance so are completely ignored in the computations. Cases B are special cases; we describe how we handle these cases in the subsection Partial occluders.

For the occluders that lie fully in the positive hemisphere of the receiver (case A), we can compute the occluded irradiance using $FIVs(EnvOcc)$. Based on the relative position (in terms of distance and direction) of occluder from the receiver, we find the closest sample point and look up the corresponding $FIV(EnvOcc)$ in the $FIVs\ \ texture$. Knowing the normal of the receiver, the occluded fullsphere irradiance $FI_{occ,p}(\vec{N_p})$ is calculated as the dot-product of the normal $\vec{N_p}$, with the $FIV(EnvOcc)$, as defined in Equation 11. To further enhance the results, the 8 nearest samples of the $FIVs(EnvOcc)$ are trilinearly interpolated.

In case A, the occluder and all the light sources it occludes, are entirely in the positive hemisphere of the receiver. As a consequence, based on Equation 8, occluded fullsphere irradiance is equal to occluded irradiance, $FI_{occ,p}(\vec{N_p}) = I_{occ,p}(\vec{N_p})$. The computed occluded irradiance $I_{occ,p}(\vec{N_p})$ is deducted from the $I_{tot}(\vec{N_p})$ to get the irradiance arriving at the pixel $I_p(\vec{N_p})$. $I_p(\vec{N_p})$ is used to shade the pixel.

Partial occluders

In the preceding discussion we estimated the occluded irradiance using the precomputed $FIV(EnvOcc)$, under the assumption that all light sources occluded are in the positive hemisphere

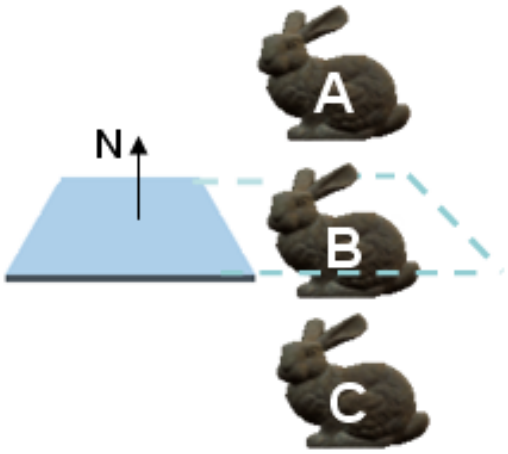


Figure 10: Depending on the relative position of the occluder with regard to the receiver, there are 3 different cases for computing the occluded irradiance: case A is computed using the FIV values, case C is totally ignored and case B is handled as special case.

of the receiver. However, this is not the case when an occluder is not fully in the positive half-space of the receiver, case B in Figure 10. In such cases the use of $FIV(EnvOcc)$, result in an underestimation of the occluded irradiance since the result would correspond to the irradiance due to the light sources in the positive hemisphere (marked with green color in Figure 11) minus the irradiance due to the light sources in the negative hemisphere (marked with orange color in Figure 11), $I_{FIV} = FI_{FIV_+} - FI_{FIV_-}$. The deduction of FI_{FIV} for the light sources in the negative hemisphere is because in these cases the dot product between receiver's normal and light direction gives negative value.

The correct value of the occluded irradiance would be equal to the occluded fullsphere irradiance computed using only FIV_+, Figure 11. However, the FIV_+ can not be precomputed since it would rely on a prior knowledge of the receiver's orientation. The FIV_+ can be approximated by scaling the $FIV(EnvOcc)$ based on the percentage of the occluder that lies in the positive hemisphere over p. The percentage of the occluder in the positive hemisphere, is approximated with $(d + R)/2R$,

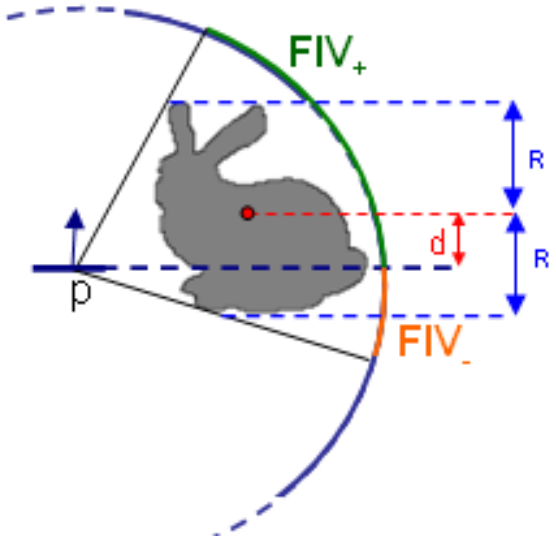


Figure 11: At border cases only a part of the occluder intercepts irradiance from the receiver.

at run-time; d is the distance of the center of the occluder from the plane of the receiver and R is the radius of the bounding sphere of the occluder, Figure 11. We count as occluded irradiance $I_{occ,p}(\vec{N_p})$ the approximation of FI_{FIV_+}. The fact that the irradiance is cosine weighted minimizes any error, that may occur by the approximation, see Figure 23 in the Results section.

Multiple occluders

In scenes with multiple occluders, pixels are shadowed by more than one occluder. In cases where two or more occluders hide a common part of the environment, the irradiance occluded by the overlapping part, should be deducted only once from the total irradiance. For clarity in the description, we explain in this section how we handle cases only for two overlapped occluders. The same concept can be used for an arbitrary number of overlapped occluders.

Occluded irradiance by two overlapping occluders $I_{occ,p}$ at the receiving point p, is equal to the summation of the occluded irradiance by each individual occluder $I_{occluder_i,p}$ minus the occluded irradiance by the overlapped part of the two occluders, $I_{overlapped,p}$.

$$I_{occ,p} = I_{occluder_1,p} + I_{occluder_2,p} - I_{overlapped,p} \tag{14}$$

The occluded irradiance by each occluder, $I_{occluder_i,p}$, is computed as already described in the section Occluded irradiance. The occluded irradiance of the overlapping part is approximated as the percentage $Overlapped_{\%}$ of the area of the overlapping part $Area_{overlapped}$, over the summation of the area of the parts covered by the two occluders $Area_{occluder_i}$, of the total irradiance occluded by the two.

$$Overlapped_{\%} = \frac{Area_{overlapped}}{Area_{occluder_1} + Area_{occluder_2}}$$

$$I_{overlapped,p} = Overlapped_{\%} * (I_{occluder_1,p} + I_{occluder_2,p}) \tag{15}$$

The area covered by each occluder, $Area_{occluder_i}$, is approximated as the area of the *unit bounding disc* of the occluder. We define the unit bounding disc as the disc centered along the line formed by the center of the bounding sphere of the occluder and point p, and translated in the space (perpendicular to the surface of p), such as to be away from the point p a unit distance. The overlapping area of the two occluders $Area_{overlapped}$, is approximated by the area of crescent formed by the overlapping of the two unit bounding discs, Figure 12.

3.3.3.2 Self-shadows

Self-shadows are also computed using $FIVs$. Note that in case of self-shadows, light sources that are definitely lie in the positive hemisphere of the receiving pixel p, are all **unoccluded** once. This is the main difference from the case of the shadows, where all **occluded** light sources lie in

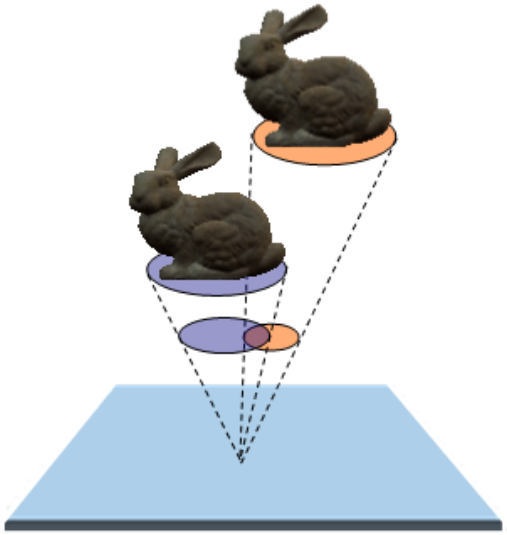


Figure 12: The occlusion part of overlapping occluders is taken into account only once.

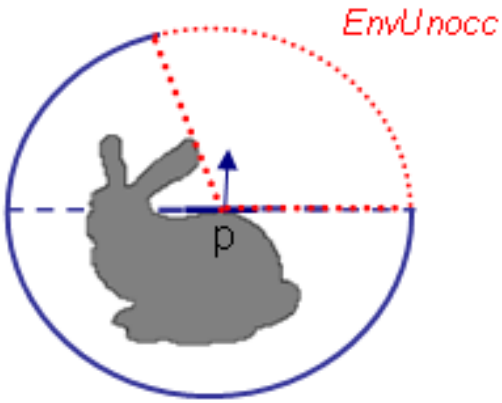


Figure 13: In case of self-shadows, all the unoccluded part of the environment $EnvUnocc$, always lies in the positive hemisphere of the receiving point.

the positive hemisphere, and not the unoccluded. This is because in case of self-shadows all light sources in the negative half space of the receiving point p, are occluded at least from the surface that p lies on. Figure 13 shows with dotted red line, the part of the environment map the object does not self-occlude, $EnvUnocc$.

Exploiting the fact that all unoccluded light sources $EnvUnocc$ which contribute to the illumination are in the positive hemisphere of the receiving point p, based on Equation 8, the irradiance at a point p is equal to the fullsphere irradiance at that point, $I_p(\vec{N_p}) = FI_p(\vec{N_p})$.

$FI_p(\vec{N_p})$ is computed by deducting occluded fullsphere irradiance $FI_{occ,p}(\vec{N_p})$ from the total fullsphere irradiance $FI_{tot,p}(\vec{N_p})$, Equation 13. $FI_{tot,p}(\vec{N_p})$ is the same for all $\vec{N_p}$ and has been computed once and stored at preprocessing. Thus, we don't have to do anything else than just use the precomputed value.

To compute $FI_{occ,p}(\vec{N_p})$ we use the nearest sample of FIV precomputed with a dot product with the normal of the receiver $\vec{N_p}$, Equation 11. In the same way as in case of shadows, we trilinearly interpolate up to 8 nearest samples of $FIVs$ to enhance the results. Samples that fall within the geometry of the object are not taken into account.

Chapter 4

Real-time tonemapping

4.1 Introduction

Illumination of a scene, using accurate description of the environment as in case of HDR environment maps, produces HDR illuminated scenes. Computed luminance values are near to those that can be measured in a similar real scene. However the wide range of luminances that exist in the real world can not be displayed on standard monitors which currently are capable of displaying only low dynamic range values. The conversion from high dynamic range to displayable luminance values is known as tonemapping (TM). TM is a very important last step in the production of realistic images and many operators have been proposed. Tonemapping operators (TMOs) are a key part of the process of high fidelity image synthesis, as they attempt to generate images visually similar to a real scene by carefully mapping to a set of luminances that can be printed or displayed on a low contrast ratio display. However, because of the computational requirements of a complex tonemapping operator (TMO), it' s not still possible to achieve high quality results in real-time. In this thesis we followed two different approaches to achieve this goal. Both approaches are described in the following sections while firstly we explain what type of tonemapping operator is suitable according to our goals.

4.2 Suitable tonemapping operator

Existing TMOs can be subdivided in two basic categories: global and local operators. Global TMOs apply the same operation to all pixels of the input image, while local operators take into consideration the local properties of individual pixels and use this information to preserve the local contrast reproduction.

A trade-off exists between local and global operators. Local operators generally give better quality results but they are computationally more expensive. On the other hand, global operators that can achieve high frame rates give only moderate quality results, and thus are not suitable for applications where realism is needed.

In the past, several local TMOs have been presented which can be classified as either $separable$ or $non\text{-}separable$. The separable class comprises of those local TMOs which contain separate local and global components. The local component is the gain control that defines which neighborhood pixels are influencing the luminance adaptation computation of the image pixel. Afterwards, this information is used to compute at each pixel the luminance adaptation. The global component compresses the high dynamic range of luminance adaptation, previously computed, into the low dynamic range available in the display, [5, 66]. In the non-separable class this separation is not possible. Non-separable operators are not suitable for interactive applications due to the fact that they require several parameters to be tuned, that can be different for each input frame [83]. These operators also involve highly complex mathematical models with a significant computational costs, especially for high resolution frames. For these reasons both approaches that we follow in this thesis are based on a separable local TMO.

The illumination algorithm that is proposed in this thesis (see Chapter 3) can handle all-frequency environment maps. This means that in the environment maps that are used for the

illumination we may have huge differences in luminances between nearby pixels. For this reason we selected the Ashikhmin [5] operator to use in our proposed methods, which is suitable for high contrast images. Moreover the Ashikhmin operator achieves visually appealing results which obviously is necessary in applications where realistic illumination is needed.

4.3 GPU local tonemapping

The first approach that we follow in order to implement our aim is to use a hardware implementation of a local tonemapping operator. In this way the frame rate of a high-quality tonemapping operator is increased.

The graphics hardware, currently available, is becoming more and more flexible and suitable for general purpose programming, but there are still several limitations that restricts the possibility of implementing complex algorithms such as local TMOs. In fact a difficult aspect of GPU programming, as discussed in Goodnight et al. [25], is that it requires exceedingly careful optimization in order to achieve the performance that is expected. Several factors contribute to this problem, such as: memory bandwidth, driver overhead, etc. Some of these problems can be reduced, but not completely avoided [25].

Recently, several high-level programming languages for GPUs were introduced, which help the programmer to speed up the programming phase, but on the other hand the limited number of assembly instructions, still reduces the possibility to implement a sophisticated algorithm without significantly modifying it.

In this thesis, we use the hardware implementation of a state of the art local TMO, Ashikhmin operator [5], as it is proposed at [69] where it is shown how it is possible to overcome the limitations and drawbacks that still affect the direct implementation of a state of art TMO on the GPU. This

implementation is able to deliver in real-time the results of the original TMO (CPU implementation), maintaining intact its quality reproduction. No trade-off between quality and speed is required.

4.3.1 Implementation

The Ashikhmin operator is a multipass algorithm. The proposed GPU implementation comprises of 3 passes as described below. Throughout the description uppercase and lowercase notation represent HDR and LDR values respectively. This GPU implementation of the Ashikhmin operator, algorithmically performs the same operations as the original CPU implementation [5]. Below we describe briefly the GPU implementation. More details for the implementation can be found at [69].

- Conversion of $RGB(x, y)$ to $L(x, y)$: The original frame, with HDR illuminated scene, is binded on a floating point texture in order to be able to pass the data to the fragment shader and process them. The luminance $L(x, y)$ of each pixel (x, y) of HDR frame is computed. In this GPU implementation the alpha channel of the pixel is exploited in order to store the luminance $L(x, y)$ value. The resulting image with (R, G, B, L) information per pixel is stored in a floating point framebuffer object (FBO), called the *luminance FBO*.

- Computation of the local adaptation level $L_a(x, y)$: The local adaptation level $L_a(x, y)$ is computed as an average luminance over some neighborhood around the pixel position (x, y). As an input to this step we use the *luminance FBO*, computed in the previous step. $L_a(x, y)$ of each pixel is computed. L_a image, that is the result of this step, is rendered in a floating point FBO, called the *adaptation FBO*.

- Computation of $rgb(x, y)$: In the final step of the proposed GPU implementation several operations are performed in order to compute the final (r, g, b) triple per pixel, i.e. the

tonemapped low dynamic range image. Firstly a tonemapped (i.e. displayable) value of luminance of each pixel $L_d(x, y)$ is computed, as it is explained in [5]. $L_d(x, y)$ is a function of the HDR luminance $L(x, y)$ of each pixel and the local adaptation level of each pixel $L_a(x, y)$, Equation 16. For this reason, as input to the shader that is responsible for this step, both results of *luminance FBO* and *adaptation FBO* are needed. Both FBOs are binded as floating point textures and passed as input to the shader.

$$L_d(x, y) = f(L(x, y), L_a(x, y)) \tag{16}$$

$L_d(x, y)$ value is then used to scale each component of the RGB triplet by $L_d(x, y)/L(x, y)$. RGB values are accessed with a texture lookup. A standard gamma correction follows as the final step to obtain pixel values for the display.

The OpenGL shading language has been used for the implementation described above. Frame-buffer objects (FBOs) are used to provide a fast way for floating point textures, as well as floating point render targets. Every step uses one or more floating point textures as input and outputs the result to the double-buffered FBO result buffer. All computations are implemented as fragment shaders.

4.4 Selective tonemapping

The GPU implementation of the local tonemapping operator increased drastically the frame rate that can be achieved, compared with the original CPU implementation. Real-time frame rates, using the GPU implementation of a local tonemapping operator, are only feasible for medium image resolutions as demonstrated in the Chapter 5. As larger resolution high dynamic range (HDR) textures/images are becoming common in computer graphics applications, a more advanced

technique should be used to achieve real-time frame rates for such images. Here, we introduce a novel technique called *selective tonemapping* for accelerating the tonemapping step on large resolution HDR images.

4.4.1 Approach

One of the main tasks of a tonemapping operator (TMO) is to preserve local contrast and details that are available in the original HDR input image. Strong contrast is localized along the transition regions between bright and dark areas and details are typically localized in textured regions. Knowledge of where the strong contrast and details are located may be exploited to limit the use of the computational expensive local luminance adaptation computation only on these regions.

In this section we introduce the novel concept of the *selective tonemapping* which automatically detects the visually important regions in an HDR input image and directs the local luminance adaptation computation at these salient parts of the scene. For the remainder parts of the image the original luminances of the scene are kept. After this step the global TM curve of the original local TMO is applied to the updated luminances of the whole scene in order to reduce the dynamic range.

We propose a general framework, Figure 14, that may be implemented either as a hybrid solution (CPU and GPU) or fully implemented on the GPU. The GPU implementation of our framework exploits current graphics hardware in order to achieve perceptually high quality tonemapping at real-time frame rates for large resolution HDR images, while preserving contrast and details of the input image.

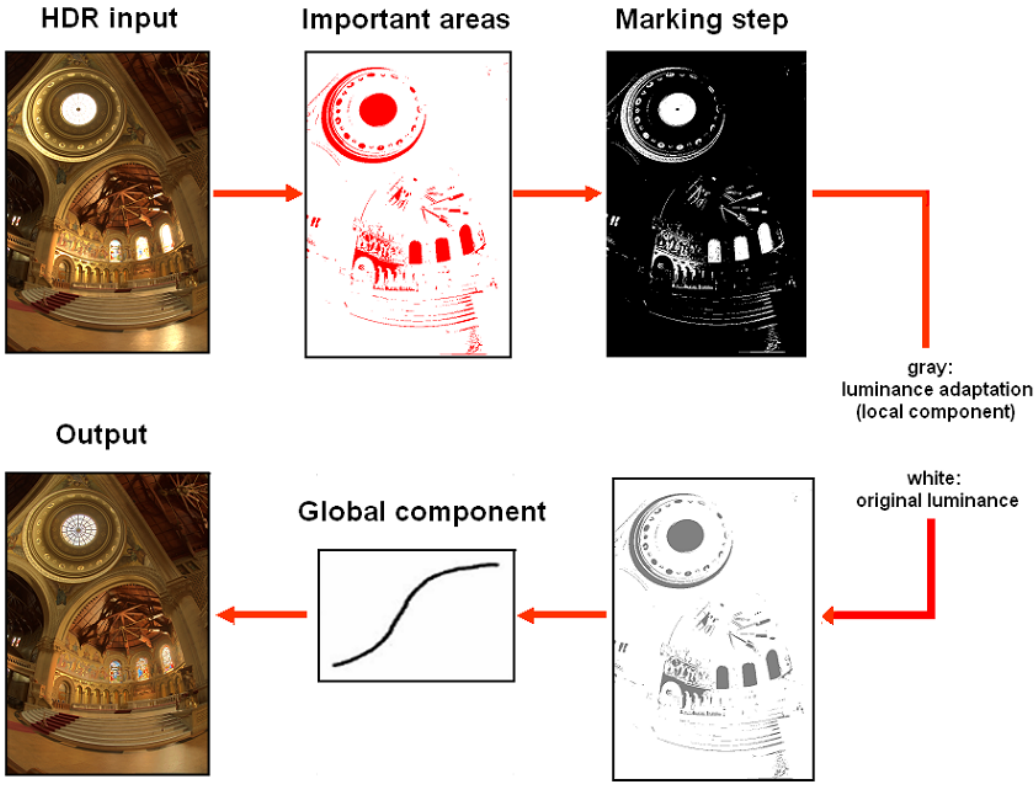


Figure 14: Scheme representing the behavior of selective tonemapping framework.

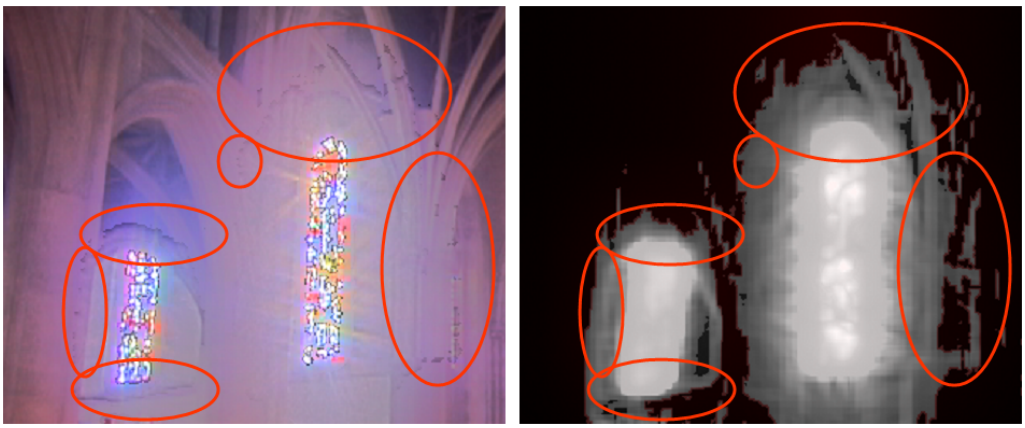


Figure 15: Output obtained using a saliency map for localizing the areas in an HDR image with strong contrast and details. (left) Several artifacts are visible (red circles) in the output image (right).

4.4.2 Framework

4.4.2.1 Description

In this section we describe the steps used in the selective tonemapping framework.

- *Important areas identification*

  In this step we identify the areas that are important for the local component of a tonemapping operator. We define as important those areas that are affected mainly by the local component. Local component of a tonemapping operator is responsible to preserve the contrast between nearby pixels in an image. After application of local component luminance values in areas where there is high contrast are greatly changed while in the rest areas luminance values are about the same with original ones. Based on this, important areas of an image are those areas that have high contrast.

  High contrast areas can be detected using an edge detection algorithm. We selected to use sobel operator for edge detection, since it's a simple operator and thus can run relatively fast. This is a requirement in our case that we need to achieve high frame rates for the whole tonemapping process.

  Another possibility for this step is to identify visually perceptual important areas in the frame. Validated visual perception models such as a saliency map can be used to correctly localize the perceptually important or salient regions [30]. However these methods are computationally expensive and often produce visible artifacts as shown in the Figure 15. This is because saliency map models often do not correctly localize the boundary regions, see Figure 15 (right image), where strong contrast occurs. However due to the high computational costs of these techniques, and their failing in localizing the boundary regions are not suitable for our purpose.

To reduce the computation time for the determination of the important areas map, it is possible to apply the important areas identification algorithm on a down scaled version of the input frame and up scale it before applying the next step (marking). We explored this possibility, but preliminary results show that important small features may be lost resulting in incorrect determination of saliency regions. This causes details to be lost. Based on this, we decided to apply important areas identification step to the full scale frame. However for extremely high resolution images this technique can be used to speed up the whole process, with a cost in quality.

- *Marking step*

  Once the important areas of the input frame are localized, it is necessary to define a strategy that properly applies the two components of a separable local TMO (local and global components). This step requires in theory a simple conditional constructor (or test) to decide which component of the operator we should apply in each pixel of the input frame.

- *Tonemapping application*

  This step applies the local component of the original TMO only on important areas as they have been defined by the marking step. Then the global component is applied to the whole image of adapted luminances.

In the next subsection we describe the implementation of these steps.

4.4.2.2 Implementation details

- *Important areas identification*

  The important areas map is computed using sobel operator filter on the original HDR luminance values of the input HDR frame. The magnitude of each pixel of the image,

resulted after applying the sobel operator, has been shown that is a good indicator of contrast on gray scale images [59] as luminance image $L_a(x, y)$.

The output of the sobel operator is a gray scale image. In order to get a binary image that defines which pixels are important and which are not, we apply a filter using a threshold value, $importance_{threshold}$. Any pixels with magnitude bigger than the threshold, assumed to be important and marked with white color, while the rest are marked with black color. This filtering step also helps to identify more uniform important and un-important regions, avoiding the presence of singular pixels inside these regions.

Captured important regions are based on $importance_{threshold}$ which affects in this way the quality of the output image. Figure 16, shows the resulting images obtained from the important areas localization step when varying the threshold level. When increasing the threshold level several important areas are not identified, Figure 16, right, resulting in the loss of important details in the final output image, Figure 17. From these results it is clear that using lower threshold level we obtain better quality results.

The output of this step is a binary image that is binded as a texture, $importance_{texture}$.

- *Marking step*

  Based on the output of the important areas identification step, that is used as input to this step, we should find an efficient way for marking. That is, a way to mark the pixels that have been important as pixels that are going to be processed by the next step (local component application).

  The most obvious way to do marking and the first approach that we followed, is to use a conditional branching within the pixel shader that is responsible for applying local component of tonemapping operator. Branching condition tests whether the corresponding pixel in the

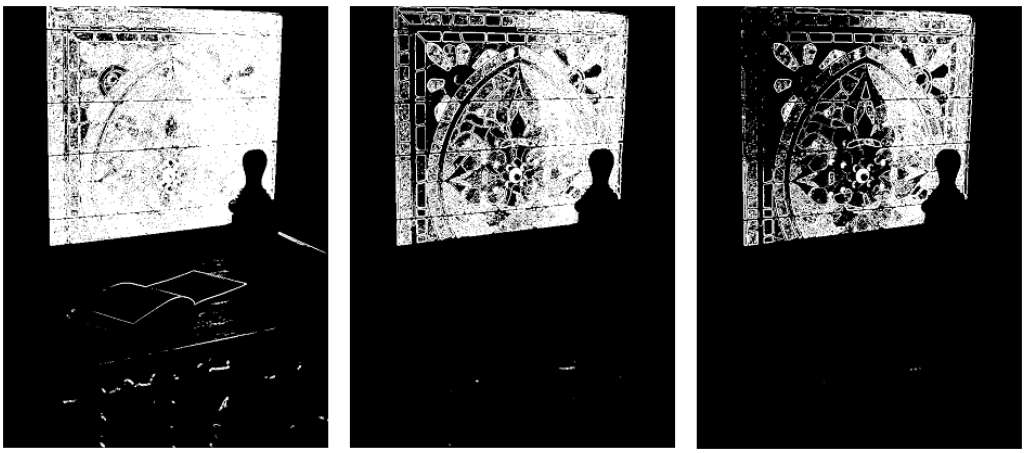


Figure 16: Important areas localization varying the threshold level: threshold value 0.1 (left), 0.5 (middle) and 1.0 (right). White and black pixels indicate important and un-important pixels respectively.

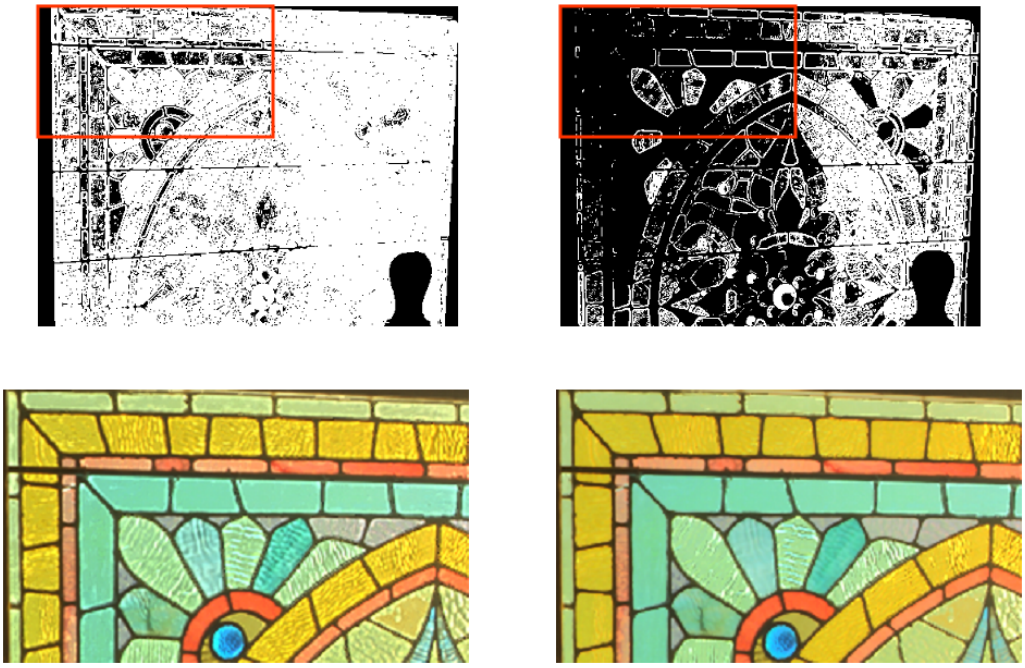


Figure 17: A close-up of important areas localization map (top) and the tonemapped output results (bottom). Value that has been used for $important_{thresh}$ is 0.1 for the left images and 1.0 for the right images.

$importance_{texture}$ is white or black. For white pixels, assumed as important, the rest of the local application pixel shader is executed. Black un-important pixels are skipped using texkill operation.

Preliminary results of using the branching and texkill operation, showed that this step was the bottleneck of our framework and real-time frame rates were not feasible. To overcome this bottleneck we investigated to separate the important areas output in tiles (smaller images) and execute the marking step and application of local component on them, in a parallel way. However, the overhead added by separation in tiles and reconstruction of them, makes this solution not suitable for high frame rates performances.

We verified that the use of the early Z-buffer is a suitable solution to gain improvement in computation performance of our framework. Early Z-test occurs before the fragment stage. In the standard rendering pipeline its purpose is to discard any pixels that are hidden in order not to go through the pixel shader.

In our framework, the important areas map of each frame is written to the depth buffer. An early Z-test is performed to discard the pixels that are not important from going through the pixel shader that applies local tonemapping. We do this by initialize all the pixels of the depth buffer to 1.0. The important pixels are written with value 0.0 in the Z-buffer while the un-important pixels with value 1.0. The depth function that defines which pixels pass the early-Z test, is set to GL_LESS, that means any pixels that have values less than the corresponding value in the Z buffer will pass the test. In this way, important pixels pass the test and processed by the pixel shader, while the un-important pixels assumed to be hidden and are not further processed.

- *Tonemapping application*

The application of the tonemapping operator is straightforward. The computational expensive shader that is responsible for applying the local component of the TMO, makes use of the early Z-buffer. Local component of the TMO is used only on the pixels that pass the early Z-test. The output of the shader is the computed local adaptation luminance for the pixels belong to the important areas of the input frame. Afterwards the global TM function of the operator is applied on the whole modified luminance input frame.

Due to the temporal discontinuities between frames, in real-time tonemapping applications, temporal artifacts may appear (flickering). To solve this problem a temporal adaptation process that smooths these discontinuities must be used. To handle this we use the solution proposed in [3] and [25], which is based on interpolation of local adaptation between consecutive frames. In this way any temporal artifacts introduced by small differences between the edge maps generated for each frame, are also removed.

Chapter 5

Results

5.1 Introduction

In this chapter we demonstrate the results of the proposed algorithms. Both timing and quality results are given. Firstly we give the results for the proposed illumination algorithm. We show that our algorithm can run at real-time frame rates for complex scenes giving high quality results with only moderate requirements in memory. Then we give the results for the tonemapping. It is shown that our aim to have real-time high quality results have been achieved. This allow us to apply tonemapping at run-time on HDR illuminated scenes that are produced by our proposed illumination algorithm. Finally we show how the application of tonemapping affects what is assumed perceptually as adequate quality for the output of the illumination algorithm. Knowing this we can adjust the resources given (e.g. memory) for illumination computation.

5.2 Illumination

5.2.1 Proposed algorithm results

The results were taken on an Intel Core 2 Duo 2.4 GHz machine, with 2GB Ram and an Nvidia GeForce 8800 GTS graphics card. A CPU implementation was used for the precomputations and a GPU implementation for run-time calculations. All images are taken with a window resolution of 512×512, with trilinear interpolation of the FIVs values and per pixel illumination. The environment maps used for illumination are 6×32×32 pixels. The FIVs are computed considering the whole environment map and not only samples on it. Using a higher resolution environment map would have caused longer precomputation times but no difference at the run-time frames per second.

In Table 2, we show the precomputation statistics for the individual objects used in our experiments. We demonstrate the results of objects with different complexity with regards to the number of vertices, (column *Vertices*), and their type of geometry (column *Object*). The *Precmp* column shows the time, in minutes, taken to precompute the FIVs. Computation of the *diffuse environment irradiance map* takes approximately about 1 minute. Here it is worth noting that our precomputation was computed entirely on CPU. A GPU implementation would run considerably faster. As expected, the table indicates that the precomputation time is linear to the number of sample points. Under the *Memory* column we show the memory usage in MBytes. Memory needed to store the precomputations is also linear to the sample points that were used and independent of the complexity of the object. This is illustrated by the fact that any object needs the same memory capacity for constant number of samples.

Table 3 shows the statistics for combinations of different objects. The *Objects* and *Vertices* columns show the total number of objects and number of vertices in the scene. Memory that is

Table 2: Precomputation times and memory requirements for different objects at various sampling rates.

Object	Vertices	Samples	Precmp (mins)	Memory (MB)
Bunny	35947	256×129×30	397	34.0
		128×65×30	99	8.56
		64×33×30	24	2.17
		64×33×15	12	1.08
		32×17×15	3	0.28
Tree	20614	256×129×30	222	34.0

Table 3: Statistics for combinations of objects for constant samples $(\vartheta, \varphi, \varrho)$ = (128,65,30) .

Scene	Objects	Vertices	Memory (MB)	FPS
1 bunny	1	35 947	8.56	112
1 tree	1	20 614	8.56	112
10 bunnies	10	359 470	8.56	68
10 trees	10	206 140	8.56	68
5 × (bunny+tree)	10	282 805	17.12	68
10 × (bunny+tree)	20	565 610	17.12	46
15 × (bunny+tree)	30	848 415	17.12	35
20 × (bunny+tree)	40	1 131 220	17.12	29
30 × (bunny+tree)	60	1 696 830	17.12	22

needed is increased linearly with the number of different objects in the scene but having multiple times the same object does not increase the memory requirements. Frames per second (illumination only) depend only on the number of objects but not on their type, as it is illustrated by the different scenes with 10 objects. FPS demonstrate that the proposed method scales well and achieves real-time frame rates even for complex scenes with millions of vertices and tenths of occluders.

Note that run-time frames per second (FPS) are independent of the number of sample points used, and independent of the geometry complexity of objects. This allows us to increase the realism in the scene by using high quality models and a dense FIV sample set without additional cost at run-time for the illumination process.

Figure 18: The bunny at different sampling rates ($\vartheta, \varphi, \varrho$): Samples = (32,17,15) (NRMSE = 0.023) (left), Samples = (64,33,30) (NRMSE = 0.008) (middle), Samples = (256,129,30) (NRMSE = 0.007) (right). The small images show the shadow only of the result of the proposed technique (left), the ground truth shadow (middle) and the difference of the two (right).

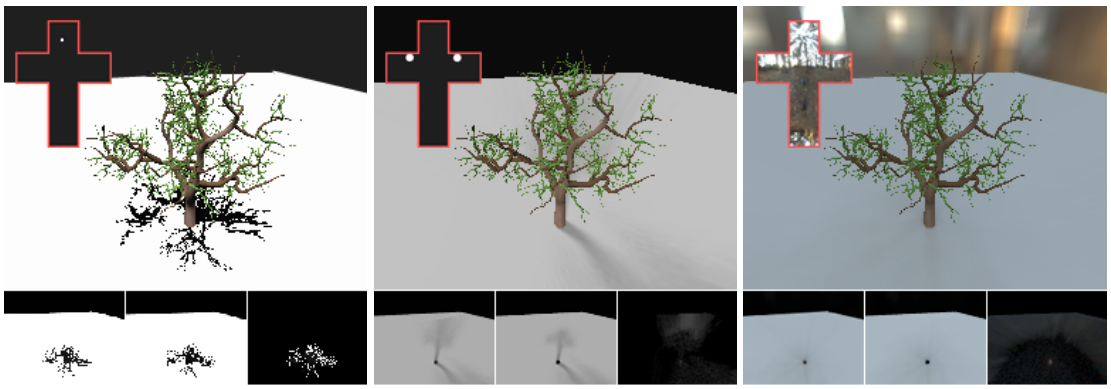


Figure 19: All-frequency environment maps can be used with our technique: 1 point light source (NRMSE = 0.1) (left), 2 area light sources (NRMSE = 0.035) (middle), Eucalyptus grove environment map (NRMSE = 0.022) (right).

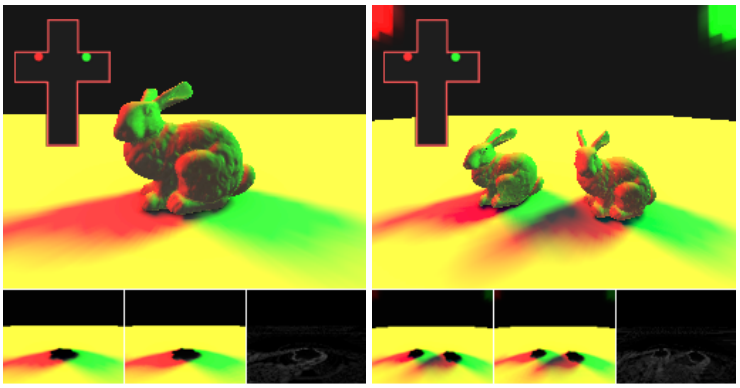


Figure 20: Illumination from area lights of different color: 1 occluder (NRMSE = 0.019) (left), 2 occluders with overlapping shadows (NRMSE = 0.019) (right).

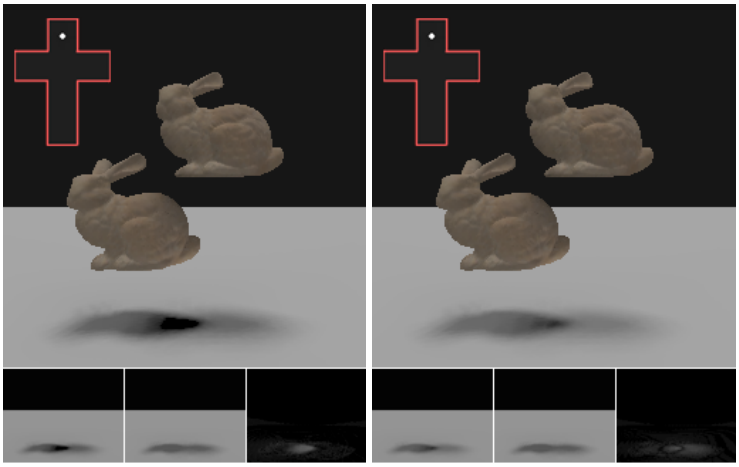


Figure 21: Illumination of multiple occluders: Correction for multiple occluders is disabled (NRMSE = 0.027) (left), enabled (NRMSE = 0.019) (right).

Figure 22: Illumination of multiple occluders in different parameters: 1 area light source (NRMSE = 0.019) (left), Eucalyptus grove environment map (NRMSE = 0.015) (middle), occluders in shorter distance (NRMSE = 0.013) (right).

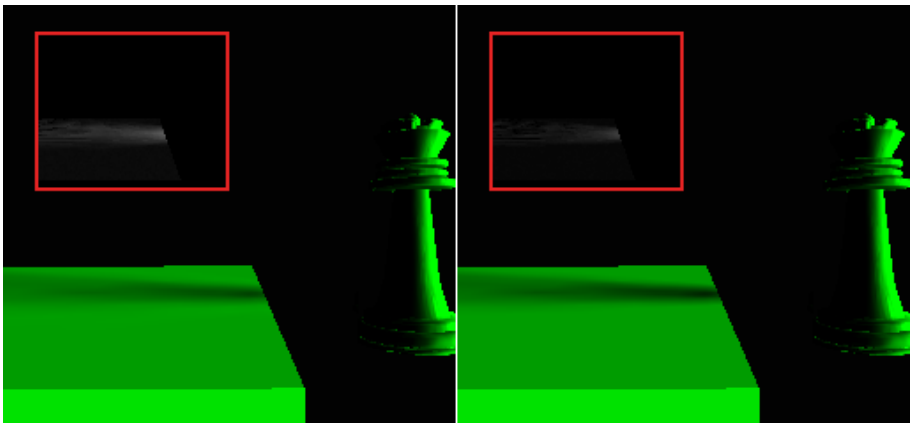


Figure 23: The queen lies at $\frac{1}{5}$ in the negative half space of the receiver. Illumination of the receiver having the correction for partial occluders disabled (NRMSE = 0.026) (left) and enabled (NRMSE = 0.019) (right). Small images show the difference of the corresponding result with the reference solution.

Figure 24: Self-shadows are also computed at run-time using precomputed FIVs. Only direct illumination without considering self-occlusions (left), direct illumination and self-shadows (middle), isolated self-shadow (right).

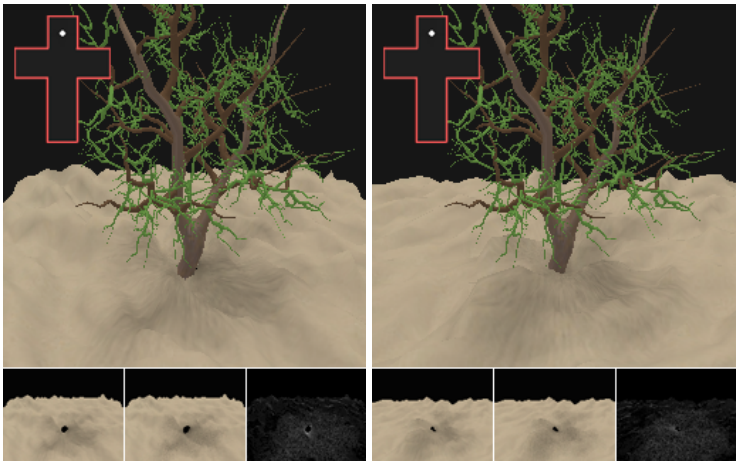


Figure 25: Illumination of a fully dynamic (deformable) receiver in two different times in left and right images.

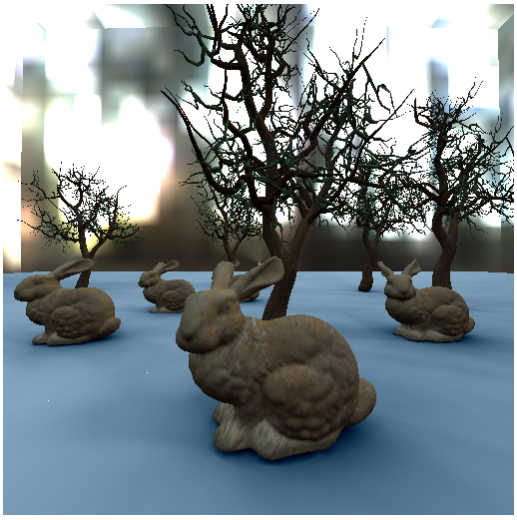


Figure 26: Complex scenes can be shaded in real-time. This scene with multiple objects and more than 280K thousands vertices runs at 68 fps.

Figures 18 and 19 demonstrate the quality of the results of the proposed technique. The three small images at the bottom show: the shadow on the ground plane using the proposed method (left), the shadow using the reference (brute force) solution (middle) and the difference of the two images (right). The difference image is displayed with high exposure in order even pixels with small error to be distinguishable. The reference solution has been evaluated using the standard rendering formula, Equation 3, assuming diffuse reflectance only, with an overall integration of the unoccluded part of the environment map as it is seen from each pixel. A numerical evaluation of the difference of the two shadow images (proposed technique and reference solution) is given. It is evaluated using the *normalized root mean square error* (NRMSE) as:

$$NRMSE = \frac{\sqrt{\frac{\sum [p_{ref}(x,y) - p_{ill}(x,y)]^2}{n}}}{p_{max} - p_{min}} \tag{17}$$

where n is the number of pixels in each image, $p_{ref}(x, y)$ is the irradiance at pixel (x, y) of the reference image, $p_{ill}(x, y)$ is the irradiance at pixel (x, y) of the image computed using the proposed illumination algorithm, and p_{max} and p_{min} are the maximum and minimum irradiance values of all

pixels in the images, respectively. For each image, a NRMSE is computed per color channel and the average of the three computed NRMSEs is given as the NRMSE for the computed image.

Figure 18 shows how the number of sample points used to precompute FIVs affect the quality of the images. As it was expected, increasing number of the sample points reduces the error. However, even with few samples, the NRMSE is only 0.023 as it is shown in Figure 18, left. Note that for this image, 3 minutes of precomputations and 0.28 MBytes of memory for the FIVs are enough for a good quality result, see Table 2.

Figure 19 demonstrates that our technique can handle all-frequency environment maps; 1 point light source (left), 2 area light sources (middle), Eucalyptus grove environment map (right). The error (NRMSE) varies based on the type of environment map, from 0.1 in case of 1 point light to only 0.022 in case of the Eucalyptus grove environment map. Even for the 1 point light source that has the highest NRMSE (mainly because of aliasing), it is hard for a viewer to perceive the difference with the reference solution.

Figure 20 demonstrates that the algorithm works not only for symmetric (or almost symmetric) environment maps. Figure 20, left, shows the shadow of one occluder of an an environment map with one green and one red area light sources and Figure 20, right, shows overlapping shadows from two occluders using the same environment map.

The same object may act both as an occluder and as a receiver at the same time, see the bunny in Figure 27, left. A receiver may be shadowed by more than one occluders. Note that in Figure 27 left, the plane is shadowed by the tree and the bunny.

Figure 23 demonstrates that our method works also well for the cases that the occluder does not lie fully in the positive hemisphere of the receiver. Figure 23, left, shows the results using the standard proposed algorithm that uses the values of FIVs as they precomputed. The right image of

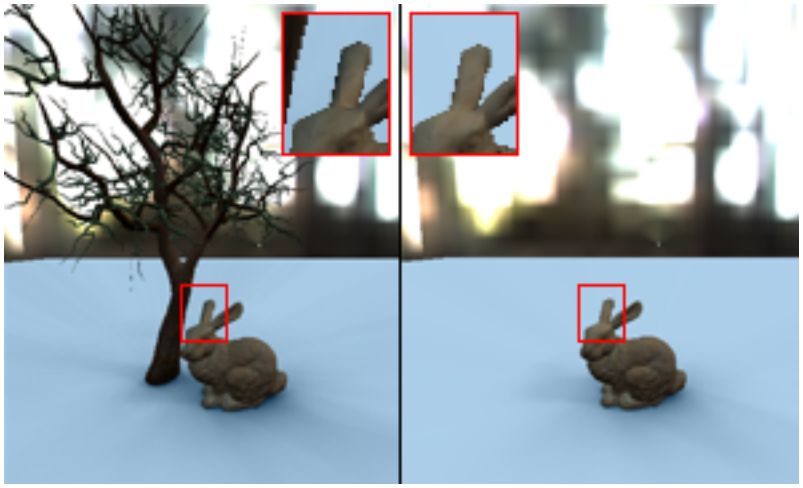


Figure 27: An object (bunny) may act as both occluder and receiver. The bunny as a receiver (left) has shadows cast on it, in contrast to the bunny (right) which act only as an occluder.

the Figure 23, shows the results using the modified algorithm for the partial occluder as described in the subsection 3.3.3.1. In the latter case the error is minimized to only 0.019.

Multiple occluders are also handled by our proposed technique. Figure 21 shows that the error is minimized when the common occluded part by the overlapped occluders, is taken into account only once (correction enabled). Results for different conditions in the scene with multiple occluders are demonstrated in the Figure 22.

Self-shadows result is demonstrated in the Figure 24. The image on the left shows illumination without considering any self-occlusions, the image in the middle is the illuminated object with self-shadows, and the image at the right is the difference of the two previous, showing the shadow isolated from the object.

5.2.2 Comparison with other methods

Our method is compared with state of the art methods for real-time illumination as it is demonstrated in Table 4. We compare our algorithm with PRT methods originally proposed by Sloan [77], precomputed shadow fields technique proposed by Zhou [105] and with proposed

algorithm of Ren for soft-shadows in dynamic scenes [67]. Details for these three methods are given in the Chapter 2.

Methods are compared among different parameters. Our illumination algorithm, FIV, supports all-frequency environment maps while the other three methods support only low-frequency environment maps for dynamic scenes. High-frequency environment maps are supported by precomputed shadow fields algorithm in case of static scenes only.

Our proposed method supports fully dynamic and deformable receivers but only moving rigid occluders. Ren et al. [67] demonstrate dynamic occluders as well, however in order to achieve this, they approximate the geometry of the scene using spheres while we don't perform any approximations on the geometry of the occluders nor on the receivers'.

Regarding memory requirements the comparison table shows the needs in memory giving the number of different objects in the scene, total number of vertices in the scene and level of frequencies of the environment maps. We demonstrate requirements of orders of magnitude less memory for an arbitrary number of vertices and all-frequency environment maps.

Frame rate achieved with our method ranges from about two times faster in case of many objects in the scene to more than ten times faster for few objects comparing to Zhou's and Ren's methods. Comparing to PRT method we demonstrate similar frame rates however the frame rate achieved at the PRT methods depends on the number of vertices of the object while we demonstrate fixed frame rate for any geometrical complexity of the object.

5.3 Tonemapping

In this section we demonstrate results for the real-time tonemapping methods described in this thesis. Firstly, results for the GPU implementation of local tonemapping are demonstrated and then results for the proposed selective tonemapping framework.

Table 4: Comparison of the proposed illumination algorithm, FIV, with state of the art methods.

	PRT Sloan SIGGRAPH	**Precomputed shadow fields** Zhou SIGGRAPH	**Soft-shadows in dynamic scenes** Ren ACM TOG	**FIV** this thesis
Environment map frequencies	Low	Low (dynamic scenes) High (static scenes)	Low	All
Dynamic scenes	Rigid	Rigid	Deformable	Deformable receivers & rigid occluders
Geometry approximation	No	No	Yes	No
Memory requirements dif. obj. / vert. / freq.	>GB	50MB 7 / 30K / low 500MB 2 / 70K / high	N/A	<1MB 1 / ∞ / all $\sim$10MB (dense sampling) 1 / ∞ / all $\sim$10MB (adeq. sampling) 10 / ∞ / all
Frame rate obj. / vert. / freq.	$\sim$100 fps 1 / $\sim$40K / low	0.1 - 10 fps 2 / 70K / high 20 fps 3-32 / 28K / low	$\sim$25 fps 2 / 65K / low $\sim$12fps 8 / 120K / low	112 fps 1 / ∞ / all 35 fps 30 / ∞ / all

5.3.1 GPU local tonemapping

In this subsection we describe the experimental results done for testing the GPU implementation of the Ashikhmin local operator. Results have been taken on still images in order to get quality results and on a real-time rendering system that produces HDR frames, in order to get timing results. The experiments were conducted on a PC with graphics card Nvidia Go6800.

5.3.1.1 Timing results

A system that produces at real-time frame rates HDR images was implemented in order to measure timing performance. The GPU implementation of the Ashikhmin local tonemapping operator was integrated with this system. Each HDR frame that is produced by the system is used as an input to the local tonemapping operator. The tonemapped LDR image is rendered on the screen.

Table 5 shows the results for this experiment for different window resolutions, as given under column *Frame Resolution*. *HDR Scene Rendering* column shows the number of frames per second achieved for the rendering only of HDR scene without applying tonemapping. Column *Scene Rendering + GPU TM* shows how the frame rate changes when tonemapping is applied.

Real-time performances are achieved with frame resolution of 512×512 and smaller, see Table 5. We can also observe how the overhead, added by tonemapping application to the rendering process, is decreased with the increase of the frame resolution. For high frame resolution the overhead is becoming imperceptible in terms of fps. This can be easily noticed on the graphical representation of Table 5, Figure 28.

Comparing our results with those demonstrated at Goodnight et al. [25], we have better performance timings. In the case of [25] for the same quality in the tonemapped image as ours, at resolution 512×512 the frame rate has already dropped down at 5 fps. In order to keep the

Table 5: Frames per second achieved for only rendering the scene without applying tonemapping (HDR Scene Rendering) and for rendering after applying GPU local tonemapping (Scene Rendering + GPU TM). Statistics are given for varying sizes of frame resolution.

Frame Resolution	HDR Scene Rendering	Scene Rendering + GPU TM
256 x 256	60	39
512 x 512	29	28
1024 x 1024	9	8
2048 x 2048	9	7

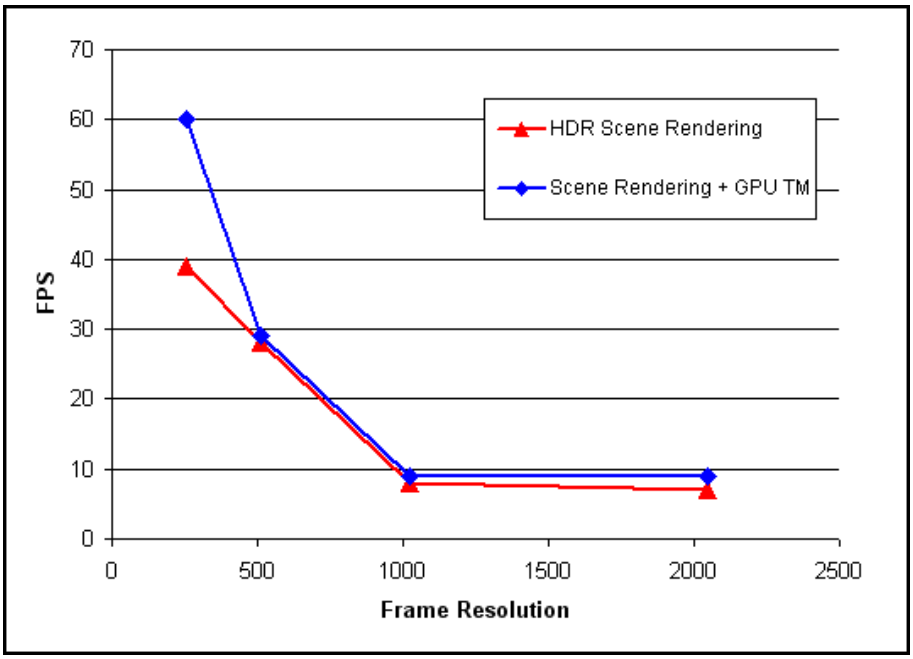


Figure 28: Graphical representation of Table 5.

rendering time at interactive frame rates, they need to compromise between quality and speed, while this is not needed in our case.

Krawczyk et al. [42] also claim in their work real-time performances. However real-time are results obtained only for low image resolution (320×240 pixels). When the image resolution is increased, the frame rate is drastically decreased to non interactive rates. Furthermore, their time performances were obtained with approximated results, in terms of quality of the TMO.

5.3.1.2 Quality results

In order to validate the quality of tonemapping results, we compared the tonemapped images produced by the GPU implementation of the Ashikhmin local operator described in this thesis with the tonemapped images produced by the original CPU implementation of the same TMO. The resolution of the still HDR images used in these experiments are given on the Table 6.

Table 6: Dimensions of the HDR images used in the experiments.

HDR Image	Resolution (pixels)
Aeroporto	1024×705
Lamp	400×300
Memorial	512×768
Nave	720×480
Rosette	720×480
Stilllife	1240×846

The *root mean square* (RMS) percentage error and *mean* percentage error between the CPU and GPU implementations are computed using the equations given at Equation 18 and Equation 19 respectively.

$$error_{RMS_{\%}} = \sqrt{\frac{1}{n}\Sigma[\frac{p_{cpu}(x,y) - p_{gpu}(x,y)}{p_{cpu}(x,y)}]^2} \tag{18}$$

$$error_{mean_{\%}} = \frac{1}{n}\Sigma \mid \frac{p_{cpu}(x,y) - p_{gpu}(x,y)}{p_{cpu}(x,y)} \mid \tag{19}$$

In the above two equations, n is the number of pixels in the image and $p(x, y)$ is the luminance value of the (x, y) pixel at the tonemapped image.

Table 7 demonstrates the quality results for all images used in the experiments, using the above mentioned error metrics. Comparing our results with Goodnight's [25], we demonstrate much smaller error since in their results they show an average 1.051% and 0.177% for $RMS_{\%}$

Table 7: *RMS* and *mean* percent errors of the proposed GPU implementation, of the local Ashikhmin operator. These values are computed considering the CPU implementation of the operator as the ground truth values.

HDR Image	$RMS_{\%}$ error	$mean_{\%}$ error
Aeroporto	0.053 %	0.008 %
Lamp	1.063 %	0.023 %
Memorial	0.041 %	0.025 %
Nave	0.201 %	0.091 %
Rosette	0.043 %	0.035 %
Stilllife	0.074 %	0.051 %

error and $mean_{\%}$ error respectively. Figure 29 collated the tonemapped output of described GPU implementation with the original CPU implementation. Results show that there is no perceivable difference between the two.

5.3.2 Selective tonemapping

In this section we demonstrate results obtained with a GPU implementation of the proposed selective tonemapping framework. In the same way as we did for testing local tonemapping hardware implementation, we tested the proposed selective tonemapping framework, on still images and on a real-time rendering system that produces HDR frames. We benchmark the selective tonemapping on two generations of GPUs on the same workstation: on an Nvidia GeForce 8800 GTX and on a GeForce 6600. Our CPU is an Intel Duo Pro 2.33GHz with 2.0 GB RAM. In order to validate our framework we applied it using the Ashikhmin TMO [5] which is a separable local TMO and thus it has a local and a global component.

First, we demonstrate timing performances of the GPU implementation integrated in the real-time setting and on a number of different still images. Afterwards the quality results are analyzed and presented in section 5.3.2.2.

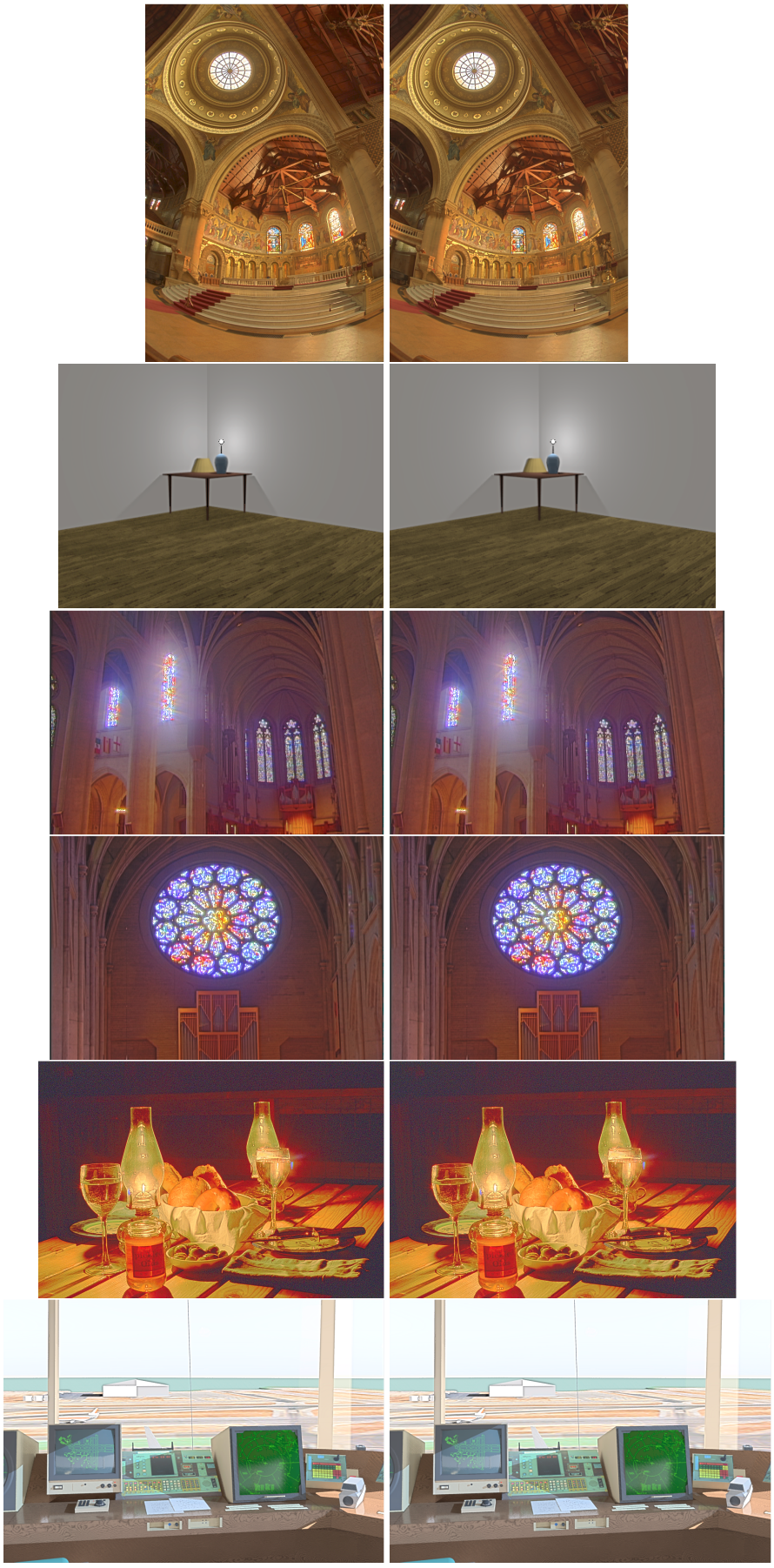


Figure 29: Images obtained with the GPU implementation of the Ashikhmin local tonemapping operator (left) and with the original CPU implementation (right).

5.3.2.1 Timing results

To evaluate the computational performances of our framework we integrated the selective tonemapping with a system that we have developed that creates HDR frames at real-time.

Table 8 shows the results for frame rates that can be achieved, for varying frame size and varying $importance_{threshold}$ level used for edge filtering at important areas identification step. These results were taken on GeForce 6600.

Table 8: Frames per second achieved on a real-time setting on GeForce 6600, for varying HDR frame size and varying threshold used at important areas identification step.

HDR Frame	0.1	0.2	0.5	1.0
320×240	141	145	145	145
640×480	78	79	82	82
800×600	55	58	59	59
1024×768	37	38	40	40

Results that are presented on Table 8 show that using smaller values for the $importance_{threshold}$ level, the frame rate is reduced. This is true for any resolution of HDR frame. This validates what was expected, since in case of small values for threshold level, more pixels are identified as important, thus the computational expensive local component of the TMO is applied on more pixels. However, we can observe that the frame rate is only slightly reduced. Based on this and on the fact that the low threshold level improves the identification of the important areas in the input frame, see section 4.4.2.2, the small threshold level of 0.1 is used in the rest of the experiments.

Table 9 and Table 10 demonstrate the frame rates that can be achieved using GPU implementation of global TMO, *GTM*, local TMO, *LTM*, and proposed selective tonemapping, *STM*. Table 9 and Table 10 report the frames per second achieved on the GeForce 6600 and Geforce 8800 GTX respectively varying the frame size.

Table 9: Frames per second achieved on a real-time setting on GeForce 6600 for the global GPU TMO, local GPU TMO and the selective tonemapping. The degree of acceleration of selective tonemapping is reported in the last column.

HDR Frame	GTM	LTM	STM	$\frac{STM}{LTM}$
320×240	226	83	141	1.68
640×480	111	28	78	2.79
800×600	80	19	55	2.89
1024×768	55	12	37	3.08

Table 10: Frames per second achieved on a real-time setting on GeForce 8800 GTX for the global GPU TMO, local GPU TMO and the selective tonemapping. The degree of acceleration of selective tonemapping is reported in the last column.

HDR Frame	GTM	LTM	STM	$\frac{STM}{LTM}$
512×512	1719	360	503	1.40
1024×1024	584	115	190	1.65
2048×2048	195	33	64	2.00
4096×4096	56	9	20	2.22

The two tables highlight the acceleration that is achieved by selective tonemapping framework over the local TMO. The degree of acceleration is given at the column $\frac{STM}{LTM}$. One important advantage of the framework is that for bigger resolutions of input HDR frames, the degree of acceleration is increased, giving space to other steps of rendering, that may be more costly for high resolution frames (e.g. per pixel illumination), to gain from this fact.

We also performed experiments on different HDR images with different dynamic range of luminances and image resolutions, that include indoor, outdoor and daylight scenes, Figure 30. Their characteristics are presented on Table 11.

Results in Tables 12 and 13 show that the frame rate of global TM and local TM is mostly affected by the frame size and not in that degree by the dynamic range of image. This conclusion is extracted from about the same frame rates of *Nave* and *Rosette* images that have the same resolution. Memory caches may give slightly different frame rates in the two cases. That was expected for global and local TM, since in both cases the same operations are applied on all pixels.

Table 11: Still HDR images used in the experiments. First column gives the image resolution and the second column the dynamic range in logarithmic scale. Images are listed in increasing number of total pixels.

HDR Image	Resolution	Dynamic Range (log10)
Nave	720×480	6.0
Rosette	720×480	4.4
Memorial	512×768	4.8
Desk	644×874	5.2
Belgium	1025×768	4.1
16RPP	900×900	2.0
FogMap	751×1330	3.6

Table 12: Frames per second achieved for HDR images in Table 11, using GeForce 6600, for the global GPU TMO, local GPU TMO and the selective tonemapping. The degree of acceleration of selective tonemapping is reported in the last column.

HDR Image	GTM	LTM	STM	$\frac{STM}{LTM}$
Nave	147	29	94	3.24
Rosette	147	29	78	2.69
Memorial	136	25	82	3.28
Desk	95	18	31	1.7
Belgium	66	13	30	2.31
16RPP	63	12	44	3.76
Fogmap	63	12	37	3.08

In contrast in case of the selective tonemapping, frame rate is not only affected by the resolution of input image but from its dynamic range as well. Differences in dynamic range of luminances implies indirectly different results of important identification step. Thus the local component of TMO, is applied on different number of pixels, giving different frame rates.

5.3.2.2 Quality results

To evaluate the quality of the proposed framework we performed experiments on several still images, with different characteristics. (Table 11, Figure 30). The aim of the selective tonemapping is not the production of a tonemapped image with the same absolute values of a reference tonemapped image, but the production of a tonemapped image which is perceptual similar to that. Based on

Table 13: Frames per second achieved for HDR images in Table 11, using GeForce 8800 GTX, for the global GPU TMO, local GPU TMO and the selective tonemapping. The degree of acceleration of selective tonemapping is reported in the last column.

HDR Image	GTM	LTM	STM	$\frac{STM}{LTM}$
Nave	1115	295	650	2.20
Rosette	1129	236	483	2.05
Memorial	975	210	520	2.48
Desk	746	154	251	1.63
Belgium	554	114	244	2.14
16RPP	547	111	298	2.70
Fogmap	537	107	285	2.66

this, perceptual metrics are used to evaluate the quality of the proposed framework. In all cases we assume as reference image, the tonemapped image produces by the GPU local tonemapping.

We have used two well known perceptual quality metrics: *visual difference predictor*, (VDP) [14] and *structural similarity index*, (SSIM) [96]. The two metrics take into account the properties of HVS to evaluate the similarities between two images. The VDP metric gives as output the number of pixels that are perceptually different on the two images that are compared. The SSIM index gives a decimal number, between 0 and 1, where 0 indicates totally different images and 1 identical images.

We performed two classes of experiments. Firstly, we performed experiments in order to verify that the usage of sobel filter at the important areas identification step, gives better results than usage of visual attention models such as saliency map. Then we performed experiments to evaluate the quality of tonemapped images produced using our proposed framework.

For the first class of experiments we compared the results given by the selective tonemapping using sobel filter and results given by the selective tonemapping using saliency map with reference image of local tonemapping. In all cases GPU implementations have been used.

The results for the comparisons are given on Table 14. For each image used in the experiments, the percentage of number of different pixels is given using VDP metric. Difference percentages

Figure 30: Colour plate of the images used in the experiments for selective tonemapping evaluation. From left to right and top to bottom: FogMap, Desk, Memorial, Nave, Rosette, Belgium and 16RPP.

Table 14: Percentages of perceptual differences using VDP metric. Results are given for the output of selective tonemapping using sobel filter and for the output of selective tonemapping using saliency map. In both cases the comparison is done with the ground truth output of local operator.

HDR image	STM-Sobel vs LTM	STM-Saliency vs LTM
Nave	1.47	2.81
Rosette	0.38	0.09
Memorial	2.41	3.62
Desk	2.72	2.05
Belgium	0.21	0.35
16RPP	0.04	0.07
FogMap	0.07	0.18

Table 15: Percentages of perceptual similarities using SSIM index for the comparison of output of selective tonemapping using sobel filter with the ground truth output of local operator.

HDR Image	STM-Sobel vs LTM
Nave	0.9761
Rosette	0.9982
Memorial	0.9876
Desk	0.9925
Belgium	0.9853
16RPP	0.9968
FogMap	0.9972

are given for the comparison between the result of selective tonemapping using sobel filter with reference image, *STM-Sobel vs LTM* and for the comparison between the results of selective tonemapping using saliency map with the reference image *STM-Saliency vs LTM*.

Table 14 demonstrates that the images obtained using the two different maps (sobel and saliency) as input for driving the selective tonemapping, are not perceived by the HVS to be different from the reference solution. Moreover, in most cases, the percentage of perceptual different pixels for the images using sobel filter is indeed smaller than that of the images obtained using saliency map.

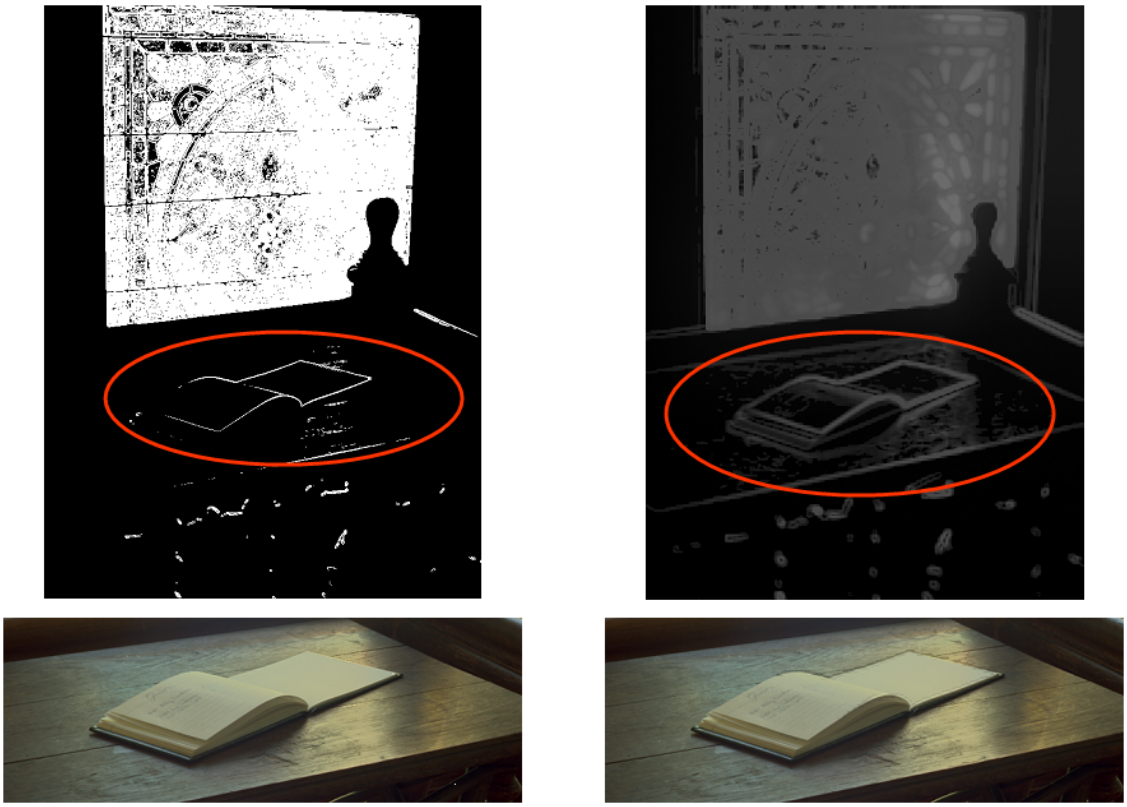


Figure 31: Important areas identification step output (top) and final output of selective tonemapping (bottom) for the highlighted area. Sobel operator and saliency map have been used to identify the important areas in left and right images respectively.

The experiments were repeated and evaluated with SSIM metric, Table 15. The SSIM confirms the results of the VDP metric, that images obtained using sobel filter at the selective tonemapping, are perceptually similar to the reference local tonemapped images.

In the Figure 31 we give a visual example of the important areas identification step using sobel filter (Figure 31, left) and saliency map (Figure 31, right). Black pixels indicate where the global component of the TMO will be applied while on the rest image the local component will be apply. Although the two resultant important area maps are quite similar, the saliency map identifies redundant saliency areas that do not add any quality improvement at the final tonemapped image. As it is shown in Figure 31, bottom, similar tonemapped results obtained in both cases,using either sobel filter or saliency map. However, for the same area, marked with red color, saliency map

Figure 32: Tonemapped results of Desk image; close-up at the book area (top) and full resolution image (bottom). Results are given for selective tonemapping using sobel filter (left), local tonemapping (middle), selective tonemapping using saliency map (right). Artifacts in the transition areas between dark and bright regions, are strongly visible in the output obtained when saliency map is used (red marked areas).

detects more pixels as important. This implies that the computational expensive local component of the TMO will be applied on more pixels. This reduce the timing performance comparing with the one that can be achieved by applying the local component only on the pixels identified by sobel filter.

Moreover, as it is shown in Figure 32, in some cases the use of saliency map introduce visible artifacts in the transition areas between bright and dark regions. The artifacts are highlighted with red color around the book in Figure 32.

Based on the experiments and results described above, it has been proved that the sobel filter is more appropriate to be used for the important areas identification step. For this reason sobel operator

has been integrated in the proposed selective tonemapping framework. In all the experiments that are described below, sobel operator filter has been used.

The second class of the experiments aims to evaluate the quality of the results of the whole selective tonemapping framework. Numerical results have been already given in Table 14 and in Table 15 are demonstrating that images obtained using the proposed selective tonemapping framework give perceptually similar results to the reference images obtained using local tonemapping operator on the whole image.

Visual results are demonstrated below. Figure 33 collate the results of the selective tonemapping with the results of ground truth GPU local tonemapping. We observe that the two tonemapping methods have perceptually similar results.

Figures 34 and 35 demonstrate that the proposed algorithm works well also in areas with strong contrast and many details. Figure 36, a close-up for Rosette image, shows clearly that the selective tonemapping is performing well in the transition areas between dark and bright. This is validated by considering the similarity of the output of the selective tonemapping (Figure 36, middle) with the ground truth output of local tonemapping (Figure 36, right) and its better visual result comparing with the output of global tonemapping (Figure 36, left).

Figure 34 shows the result for the close-up of the area that is highlighted in the important areas map (left image). The output obtained with our framework (right-top image) shows the achieved degree of accuracy (highlighted red zones) with respect to the output obtained with the ground truth GPU local tonemapping (right-bottom image).

However, there are cases that the selective tonemapping gives perceptually slightly different results from the ground truth solution. In Figure 37, the marked area highlights where the selective tonemapping (top-right image) under-performs comparing with the results of the local tonemapping (bottom-right image). This is done due to the small black regions inside the large white region

Figure 33: Quality comparison of the selective tonemapping (left) versus the ground truth local TMO (right).

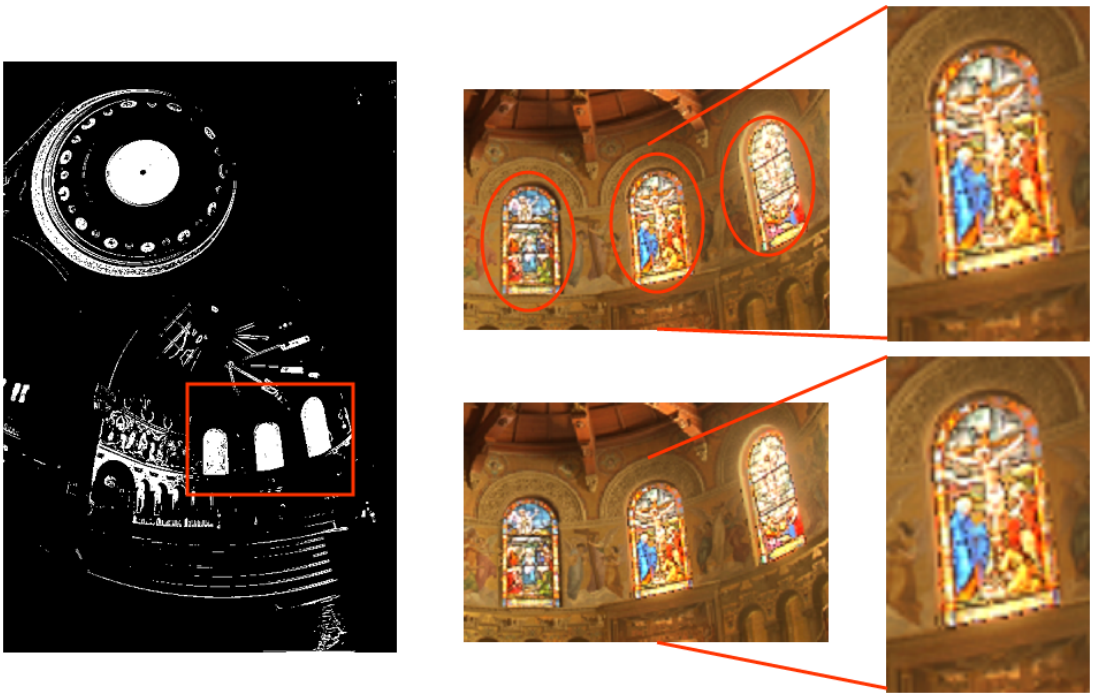


Figure 34: Example of the use of the selective tonemapping framework. Strong contrast and details regions identified by important areas identification step. A close-up of the the output obtained with selective (top) and local (bottom) tonemapping.

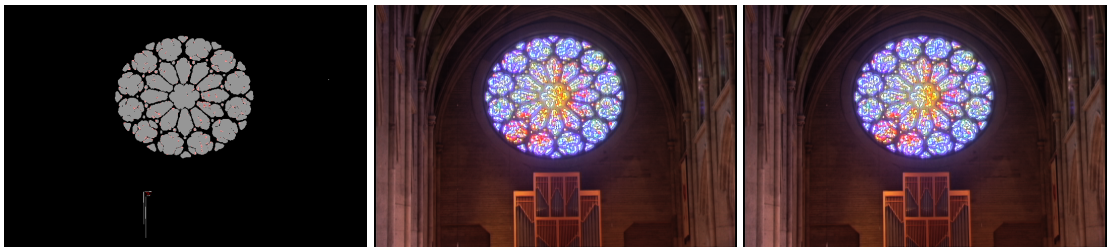


Figure 35: Selective tonemapping applied on a high contrast image. Important areas map with superimposed VDP map (left) and tonemapped images obtained with selective tonemapping (middle) and ground truth local tonemapping operator (right).

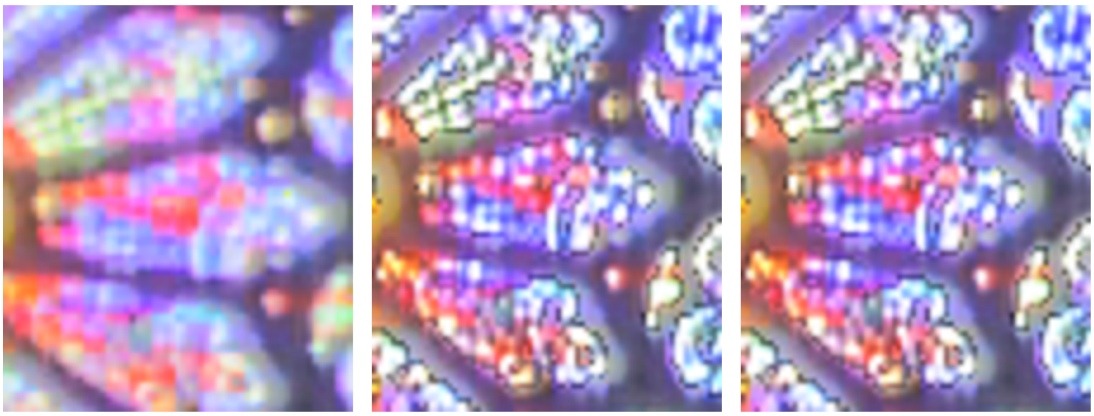


Figure 36: A close-up of the tonemapped Rosette image: global TMO (left), selective tonemapping (middle) and ground truth GPU local TMO (right).

Figure 37: Depending on the level of $importance_{threshold}$ used, the selective tonemapping (top-right) may under-performs comparing to the ground truth local tonemapping output. Important areas map with superimposed VDP map (red dots) is given on the left.

Figure 38: Full resolution output of the image in Figure 37 using threshold value of 0.1 (left) and 0.05 (right), showing that the under-performance has been overcome by reducing the $importance_{threshold}$ level.

in the important areas map (left image). In these regions the global component of the local TMO is applied in our framework reducing slightly the contrast. The solution to this problem is to reduce the $importance_{threshold}$ level used to filter the magnitude of the sobel filter during the important areas identification step. In this way, more pixels will be marked as important and will be processed by the high quality local component of the tonemapping operator. Figure 38, shows the improvement in these regions when the threshold value is decreased.

Finally, we have defined a gain function that accounts the frame rate and quality that a tonemapping method can achieve. We define the gain function as:

$$Gain_{TM} = speed(fps) \times quality(VDP) \tag{20}$$

where the speed(fps) is the number of frames per second that the tonemapping algorithm can run on a real-time setting. Quality(VDP) is defined as the similarity between the results of tonemapping method used and the ground truth solution. Since from VDP metric we get the percentage of differences between two images, $VDP_{\%}$ we define the quality(VDP) as:

Table 16: Gain function results of global operator (GTM) and selective tonemapping (STM).

	$VDP_{\%}$		speed (fps)		$Gain_{TM}$	
HDR image	GTM	STM	GTM	STM	GTM	STM
Nave	3.70	1.47	147	94	39.72	63.95
Rosette	6.25	0.38	147	78	23.52	205.26
Memorial	5.44	2.41	136	82	25.0	34.02
Desk	17.66	2.72	95	31	5.38	11.40
Belgium	1.63	0.21	66	30	40.49	142.86
16RPP	0.08	0.04	63	44	787.5	1100.0
FogMap	0.23	0.07	63	37	273.91	528.57

$$quality(VDP) = \frac{1}{VDP_{\%}} \tag{21}$$

Bigger values of gain function, Equation 20, $Gain_{TM}$ indicate better tonemapping method. Table 16 shows the values for gain function of the global operator tonemapping , *Gain (GTM)* and selective tonemapping framework. *Gain (STM)*, when applying them on a number of still images. Results show that in all cases, selective tonemapping is by far better than global operator tonemapping.

5.4 Combining HDR illumination and tonemapping

Finally we obtained experiments to examine whether the application of tonemapping on the HDR output of our proposed illumination algorithm, makes the illumination algorithm more or less tolerant to errors.

To do that we find the perceptual difference (using the VDP metric) between the image of illuminated scene, resulted by our algorithm and the ground truth image resulted by the brute force solution, used to evaluate illumination algorithm, both before applying tonemapping. Then we compare that, with the perceptual difference of the two images after applying tonemapping.

Given that HDR images are viewed on a display at a specific exposure, we should choose the proper exposure to use in our experiments. Under- or over- exposed images would give erroneous results since differences between comparing images would be hidden in total black or total white areas.

The exposure that we use in each case of our experiments has chosen to be the one the user would most probably used to see the HDR image; that is the illuminated scene in HDR image is clearly visible, without consider as a matter any over- or under- exposed areas in background.

Experiments has obtained in scenes without rendering the occluders. In this way we guarantee that the errors we measure would be the maximum possible ones. This is because in case that occluders are also rendered, they hide part of the illuminated receivers and consequently pixels with possible errors, decreasing the actual error.

The first experiment aims to show how the perceptual error varies for different sampling rates at illumination precomputations, before and after applying tonemapping. Figure 39 shows the scene on which this experiment was performed. Table 17 gives the percentage of the number of perceptual different pixels. Statistics are given for different number of samples used for FIVs computations at illumination algorithm precomputations. Table 17, *before TM* column, shows the percentage perceptual error between the HDR images produced by our illumination algorithm comparing to HDR ground truth solution, both at the chooses exposure, while Table 17, *after TM* column, shows the percentage error between the corresponding images after applying tonemapping.

The perceptual difference is decreased by increasing the number of samples used in FIVs computations. That was obviously expected in case of HDR images, since more accurate values for illumination are computed. The results validate that also in case of tonemapped images, the perceptual error decreases by using more samples, even if luminance values are mapped to another range (LDR).

Table 17: Percentage errors using VDP metric, for the output illumination algorithm comparing with the ground truth solution. The errors are given for the comparison of images before and after applying tonemapping. Results are given for a varying number of sample points used for FIVs computations.

Samples	before TM	after TM
$32 \times 17 \times 15$	5.94	2.59
$64 \times 33 \times 30$	0.20	0.32
$256 \times 129 \times 30$	0.13	0.20

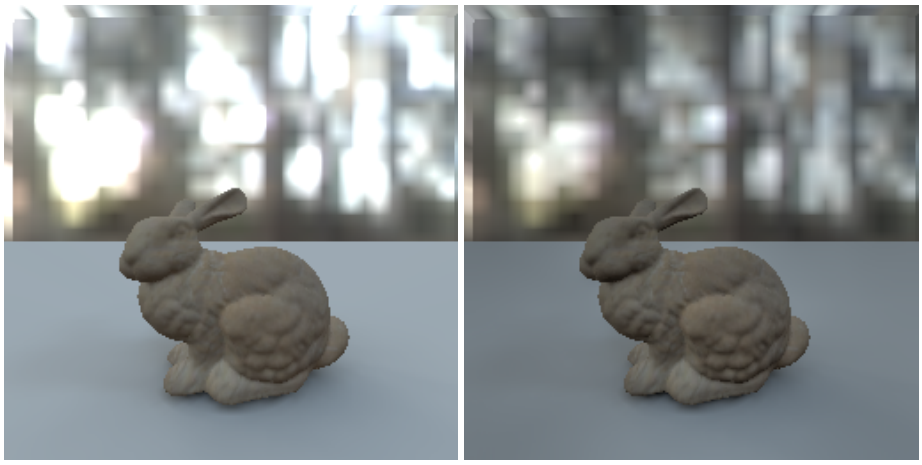


Figure 39: HDR output of illumination algorithm (left) and its tonemapped image (right) for the scene used in experiments for testing how the tonemapping affects the perceptual error for different number of sample points used.

Results also show that the perceptual error in tonemapped luminances is drastically smaller than the error in original luminances in case of few samples. In case of more samples we can assume that the error retains the same since it's only slightly increased (only about 0.1%) in case of tonemapped images.

Figure 40 demonstrates visually the results of this experiment. Images are given for ground truth solution and for the different number of FIVs samples used in our illumination algorithm. Images in the figure show HDR images (at the chosen exposure) of illuminated scene, *before TM*, and corresponding tonemapped images, *after TM*. VDP maps which show the pixels that are perceptually visual different between the images compared, are also given. VDP map, of the

comparison between HDR image of the ground truth solution for illumination and HDR result obtained by our illumination algorithm, is given under column *VDP map before TM*. VDP maps for the comparison of the corresponding tonemapped images are given on the *VDP map after TM* column of the same figure. VDP maps are given for different number of FIVs samples used.

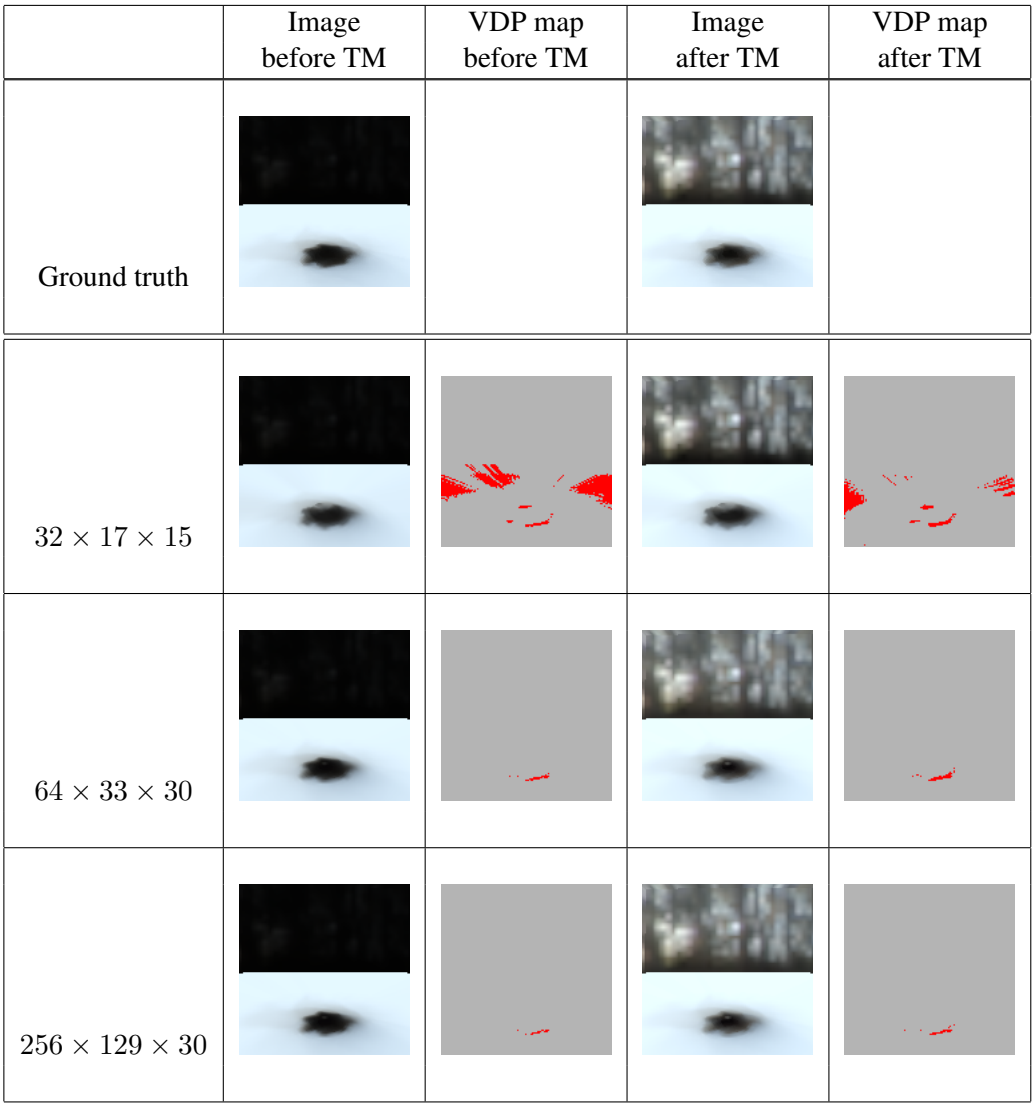


Figure 40: Visual perceptual differences (VDP maps) between the ground truth solution and results obtained with our illumination algorithm. Results are given for different number of samples. Comparisons are performed between the images before applying tonemapping and between the images after applying tonemapping.

The next experiment that performed, was about to examine how tonemapping affects the perceptual difference for different environment maps. The measures were taken on images produced

Table 18: Percentage errors using VDP metric, for the output illumination algorithm comparing with the ground truth solution. The errors are given for the comparison of images before and after applying tonemapping. Results are given for different environment maps used to represent incident lighting in the virtual scene.

Environment map	before TM	after TM
1 point light source	1.70	1.70
2 area light sources	0	0
Eucalyptus grove	11.42	8.31

by the illumination algorithm using a constant number of sample points ($256 \times 129 \times 30$), same viewing parameters and same scene for all the different environment maps used. Figure 41 show the illuminated scene before and after applying tonemapping, for the different environment maps used in this experiment.

Table 18 demonstrates that after applying tonemapping the perceptual difference with ground truth solution never increases. It remains static in cases of 1 point light source and 2 area light sources, and decrease in case of Eucalyptus grove environment map. Figure 42 illustrates the results of this experiment.

Based on the above experiments we conclude that the application of tonemapping makes the perceptual error smaller in most cases. This can be explained as described below. Pixels covering the background in all compared images (before/after applying tonemapping, FIV illumination/ ground truth result) do not have any error, since in that part of the image the environment map appears, which is not affected by illumination. The error appears in the foreground part of the images where the shades objects exist.

For comparison of images before tonemapping application, such exposure was chosen so the values of pixels in the foreground lie in the displayable range. This means that about all pixels in the foreground have values between 0.0 and 1.0. In contrast in the tonemapped images, all pixels

of images have values between 0.0 and 1.0 that means the values of pixels at the foreground have values in a smaller range.

Since values of pixels that may have error (foreground pixels) varying in a greater range in case of images before application of tonemapping than in case of tonemapped images, the difference in values of corresponding pixels is expected to be bigger in the former case; thus the error in comparison of images after applying tonemapping is reduced. We can exploit this fact to save resources in illumination computation.

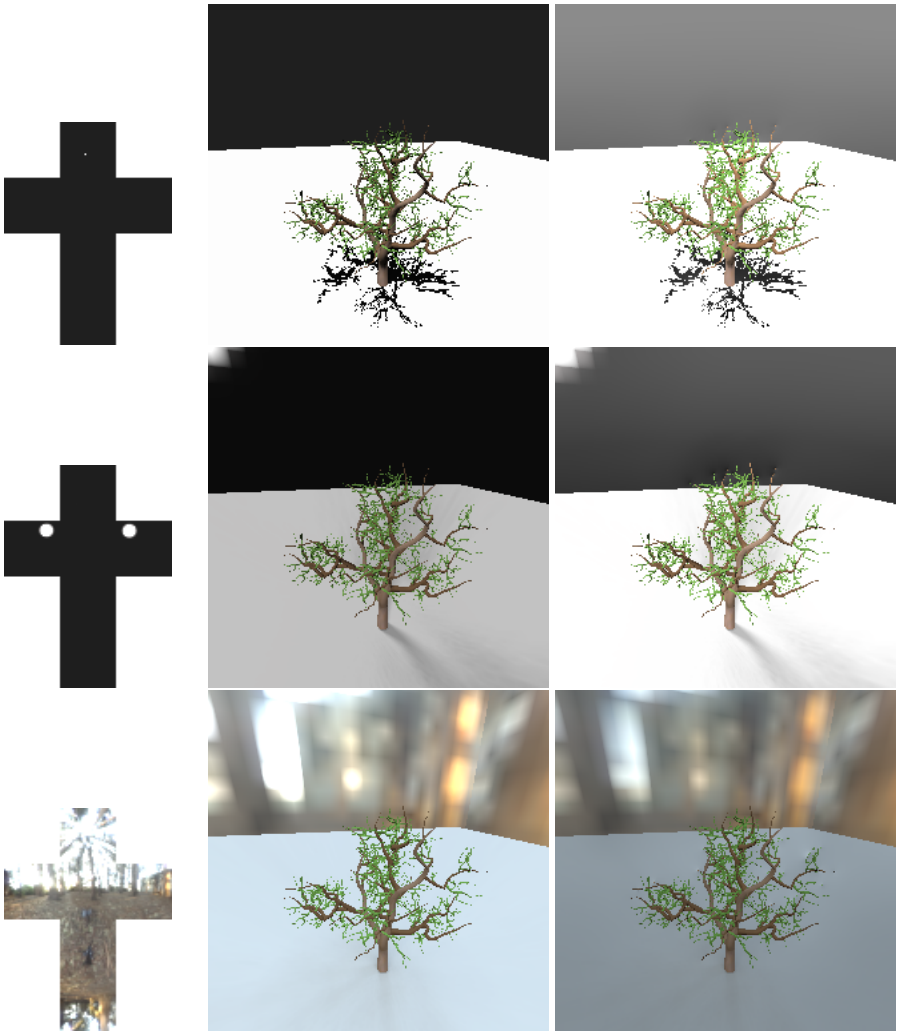


Figure 41: HDR output of illumination algorithm (left) and its tonemapped image (right) for the scene used in experiments for testing how the tonemapping affects the perceptual error for different environment maps. Environment maps used are: 1 point light source (top), 2 area light sources (middle), Eucalyptus grove (bottom).

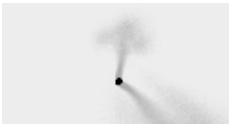
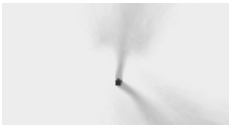
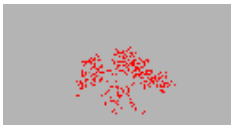
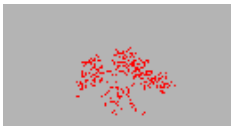
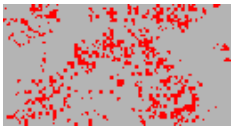
	1 point light source	2 area light sources	Eucalyptus grove environment map
Before TM			
Ground truth			
Illumination algorithm			
VDP map			
After TM			
Ground truth			
Illumination algorithm			
VDP map			

Figure 42: Visual perceptual differences between the ground truth solution and results obtained with our illumination algorithm. Results are given for different environment maps. Comparisons are performed between original HDR images at chosen exposure *before TM* and between tonemapped images *after TM*. Corresponding VDP maps that indicate the pixels that are perceptual different are given. Numerical values for the differences are given on the Table 18.

Chapter 6

Conclusions and future work

6.1 Introduction

In this chapter we discuss the strengths and weakness of the algorithms proposed in this thesis for *illumination* and *tonemapping*. Finally we discussed the application of tonemapping on HDR images produced by the illumination algorithm. Furthermore we propose future work that can be done in the three directions.

6.2 Illumination

The proposed method can compute diffuse illumination with soft-shadows for fully dynamic receivers, Figure 25, moving occluders and all-frequency environment maps at real-time frame rates. Frame rate and memory requirements are independent of the number of light sources and the number of vertices of the occluders. Frame rate depends only on the number of occluders in the scene while the memory requirements depends on the number of sample points used and the different type of objects in the scene. The size and the complexity of the objects in the scene do not affect the computations for illumination at run-time. Moreover, run-time computations are not

affected by the number of light sources. The last two properties allow us to increase the realism in the rendered image, by having high-resolution models and an arbitrary number of light sources without any cost at run-time computation. Our technique requires only moderate memory space and has relative fast precomputations. It can run real-time for very complex scenes, Figure 26, with low measured error, Figures 18, 19, 22 & 23.

The main contribution of our technique is the factorization of a new notion that we introduce, the *fullsphere irradiance*. The factorization allows us to precompute and store, in only a 3D color vector, the contribution to illumination from an arbitrary number of light sources on a reference base system, without requiring the knowledge for the receiver's normal in advance. The precomputed values can be transformed, in a very fast way (using only a dot product), to the irradiance arriving from the light sources, once the normal of the receiver is known.

The main limitation of our technique stems from the fact that the precomputed values of FIVs encode the environment map information, rendering our technique applicable only for scenes with static light sources.

Another limitation is that occluders can only be moved in the space, but they can not rotate, since the placement at sample points of the occluder at the preprocessing, define the occluded light sources which their contribution is encoded within the FIV. This limitation can be eliminated by taking samples, not only at different positions of each occluder, but with different rotations as well. However, this solution will make the precomputations more expensive and increase the memory requirements of the algorithm. A more sophisticated solution is needed for this.

The proposed method can be improved in a number of various ways. It currently accounts only for diffuse reflectance. Consideration of the specular reflection component would increase the realism of the illumination. Our technique could be extended to account for Phong-style specularities, for a fixed view direction. In such case the bisector between the incoming illumination and the

viewing direction is known. We could do that by computing fullsphere irradiance factorization in terms of the bisecting vector *h* and receiver normal *N*, storing a FIV for each of the cosine power, in a similar way that we factorized intensity of each direction L_i with *N*. For higher powers of the cosine, more factorizations would be needed. For specularities for arbitrary viewing direction, it does not seem to be a straight forward way to do that using only the factorization. A more challenging effort would be required to extend the method to handle this cases as well.

Precomputed FIVs are stored in a texture in order to pass the values within the shader. It seems to be a pattern between FIVs for different samples, as it can be seen in Figure 9. Finding a pattern between samples, it can be used to approximate the values of FIVs of positions between the samples for which FIVs have not been computed. With this approach we would be able to have continuous values of FIVs and not only those at sample positions, without increasing the memory requirements by computing FIVs at more sample positions. This will allow us to precompute FIVs only for a small number of sample positions.

Another direction that this work can be extended is using adaptive sampling for the sample positions that occluders are placed at preprocessing for FIVs computation. Occluders geometry should be consider in such case. Instead of taking samples on a number of positions at different concentric spheres, as we do with current implementation, the samples should be taken based on the outline of occluder, Figure 43. The environmental lighting should also be taken into account for adaptive sampling. Areas around the occluder that are affected by high frequent parts of the environment map should be sampled in a more dense way than other areas.

Another way that algorithm can be improved is the elimination of precomputations. Online computations of FIVs can be computed only at samples needed (i.e. a receiving point exists), reducing the total time needed to compute FIVs. Considering that the precomputation of FIVs for all sample positions for the Figure 18 left, with current CPU implementation of precomputations,

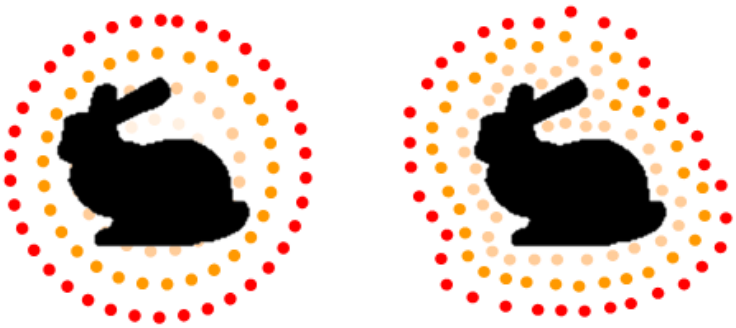


Figure 43: Sampling at concentric spheres (left) and adaptive sampling (right) based on the outline of the occluder.

took only 3 minutes, we speculate that computations of FIVs at run-time, at least at interactive rates, is feasible using a GPU implementation. For real-time computation of the *diffuse irradiance environment map* the proposed technique of [63] can be used. By having no precomputations, we will be able to have fully dynamic conditions such as deformable occluders and dynamic lighting conditions.

6.3 Tonemapping

6.3.1 GPU local tonemapping

The first approach, we followed in this thesis, in order to have real-time tonemapping, was to use a hardware implementation of a state of the art local TMO [5]. Despite the level of acceleration we achieved, the quality of the output image is maintained when compared with the output image obtained with the corresponding original CPU implementation. We also showed that the overhead introduced by the GPU implementation of the TMO to the rendering process is imperceptible, for high resolution frames, in term of fps. We compared the results we obtained with two previous works [25, 42], and showed that the implementation used in this thesis has superior quality and speed.

Even though using this approach we are able to have real-time frame rates, in case of high resolution frames, speed is decreased to only interactive rates. A more sophisticated method to do that should be used. This led us to the concept of the selective tonemapping.

6.3.2 Selective tonemapping

In this thesis we introduced the concept of *selective tonemapping* and outlined a framework which is able to accelerate a GPU implementation of existing local TMOs by exploiting knowledge of the HVS. Selective tonemapping applies the computational expensive local component of the TMO only on the areas that are assumed as important. The proposed framework has been included into a real-time setting allowing an easy integration into the rendering pipeline. Our method does not require any modification of the rendering pipeline and provides an efficient solution for real-time tonemapping.

The selective tonemapping gives a speed-up over existing local TMOs and achieves computational performances close to those gained with the corresponding global component of the local operators. We demonstrate the ability of our technique to achieve real-time performances on large resolution HDR images while being able to keep the quality of the ground truth tonemapping operator. We presented an exhaustive quality evaluation of the output obtained by our framework; showing how it is able to achieve quality comparable to the original local TMO.

An important feature of our framework is that there is no need to modify the GPU implementation if a different TMO is applied. The modularity of the framework enables us to concentrate on improving each of its individual parts independently.

In addition, we showed that the framework is able to localize automatically the strong contrast and detail regions in a rapid manner and obtain results that preserve the contrast and details of the input image. These results are then compared with those obtained using more sophisticated visual

attention models, such as the one used by Cadik [8], showing that our technique can overcome the artifacts introduced by visual attention models. Moreover, this technique presents an efficient memory usage, when compared with traditional methods, due to the reduced information required to localize the strong contrast and details areas.

Future work needs to investigate how to exploit temporal coherence between multiple frames in order to enable at least significant parts of the importance areas map to be reused. Other direction that future work can be driven is to generalize the concept of importance in the computation of important areas map in order to cover conceptual important pixels as well. For example in a computer game the main character may be assumed as important and pixels covered by it can be processed with the high quality local component of the tonemapping operator, even though it may not have high contrast areas.

6.4 Combining HDR illumination and tonemapping

In this thesis a real-time illumination algorithm for dynamic scenes has been developed. Using HDR environment maps, to represent the lighting conditions in the scene, the algorithm produces images of illuminated scenes with HDR values. In order to display the HDR images that are produced by the illumination algorithm, on an LDR display, we use the proposed selective tonemapping, that is able to run at real-time frame rates. A full rendering system, that is able to compute illumination in high dynamic range of luminances and display tonemapped frames, has been developed.

We evaluated the way that tonemapping affects the tolerance to errors of the proposed illumination algorithm. Results show that in most cases, the application of tonemapping reduces the visual error that is perceived. We can exploit this fact, to save resources at illumination computation. For example, less samples can be used for FIVs computations and still have adequate quality in results.

Future work needs to determine specifically how usage of memory and spending time for computations should be adapted, because of the tonemapping, at illumination algorithm. Moreover adapting sampling for FIVs computation can be used in a selective way as in case of selective tonemapping. Important areas around an occluder should be defined and costly computations should be apply only on these areas. In case of illumination algorithm, this is interpreted as using more dense sample set in these areas and a rear sample set on the rest.

Bibliography

[1] Kusuma Agusanto, Li Li, Zhu Chuangui, and Ng Wan Sing. Photorealistic rendering for augmented reality using environment illumination. In *ISMAR '03: Proceedings of the The 2nd IEEE and ACM International Symposium on Mixed and Augmented Reality*, page 208, Washington, DC, USA, 2003. IEEE Computer Society.

[2] Thomas Annen, Zhao Dong, Tom Mertens, Philippe Bekaert, Hans-Peter Seidel, and Jan Kautz. Real-time, all-frequency shadows in dynamic scenes. *ACM Transactions on Graphics*, 27(3):34:1–34:8, August 2008.

[3] A. Artusi, J. Bittner, M. Wimmer, and A. Wilkie. Delivering interactivity to complex tone mapping operators. pages 38–44, 2003. Eurographics Symposium on Rendering 2003.

[4] Alessandro Artusi, Benjamin Roch, Despina Michael, Yiorgos Chrysanthou, and Alan Chalmers. Selective tone mapper. *Technical Report, University of Cyprus, TR-05-07*, 2007.

[5] Michael Ashikhmin. A tone mapping algorithm for high contrast images. In *Rendering Techniques '02 (Proceedings of the 13th Eurographics Workshop on Rendering)*, pages 145–156, June 26–28 2002.

[6] James F. Blinn. Models of light reflection for computer synthesized pictures. In *Siggraph '77: Proceedings of the 4th annual conference on Computer graphics and interactive techniques*, pages 192–198, New York, NY, USA, 1977. ACM.

[7] David Burke, Abhijeet Ghosh, and Wolfgang Heidrich. Bidirectional importance sampling for direct illumination. In *Rendering Techniques 2005: 16th Eurographics Workshop on Rendering*, pages 147–156, June 2005.

[8] M. Cadik. Perception motivated hybrid approach to tone mapping. 2007. Proceedings of WSCG 2007.

[9] E. Chan and F. Durand. An efficient hybrid shadow rendering algorithm. *Proceedings of the Eurographics Symposium on Rendering*, pages 185–195, 2004.

[10] Shenchang Eric Chen, Holly E. Rushmeier, Gavin Miller, and Douglass Turner. A progressive multi-pass method for global illumination. In *Computer Graphics (Proceedings of Siggraph 91)*, pages 165–174, July 1991.

[11] Jonathan Cohen, Chris Tchou, Tim Hawkins, and Paul Debevec. Real-Time high dynamic range texture mapping. In *Rendering Techniques '01 (Proceedings of the 12th Eurographics Workshop on Rendering)*, pages 313–320, 2001.

[12] Franklin C. Crow. Shadow algorithms for computer graphics. In *Computer Graphics (Proceedings of Siggraph 77)*, pages 242–248, July 1977.

[13] Carsten Dachsbacher, Marc Stamminger, George Drettakis, and Frédo Durand. Implicit visibility and antiradiance for interactive global illumination. *ACM Transactions on Graphics (Siggraph Conference Proceedings)*, 26(3), August 2007.

[14] Scott Daly. The visible differences predictor: an algorithm for the assessment of image fidelity. pages 179–206, 1993.

[15] Cyrille Damez, Kirill Dmitriev, and Karol Myszkowski. State of the art for global illumination in interactive applications and high-quality animations. *Computer Graphics Forum*, 22(1):55–77, March 2003.

[16] Paul Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings of Siggraph 98*, Computer Graphics Proceedings, Annual Conference Series, pages 189–198, July 1998.

[17] George Drettakis, Nicolas Bonneel, Carsten Dachsbacherand, Sylvain Lefebvre, Michael Schwarz, and Isabelle Viaud-Delmon. An interactive perceptual rendering pipeline using contrast and spatial masking. In *Rendering Techniques (Proceedings of the Eurographics Symposium on Rendering)*. Eurographics, June 2007.

[18] Reynald Dumont, Fabio Pellacini, and James A. Ferwerda. Perceptually-driven decision theory for interactive realistic rendering. *ACM Transactions on Graphics*, 22(2):152–181, April 2003.

[19] Fredo Durand and Julie Dorsey. Interactive tone mapping. In *Rendering Techniques '00 (Proceedings of the 11th Eurographics Workshop on Rendering)*, pages 219–230, 2000.

[20] Frédo Durand and Julie Dorsey. Fast bilateral filtering for the display of high dynamic range image. In *Proceedings of Siggraph 2002*, pages 257–265, 2002.

[21] Philip Dutre, Kavita Bala, and Philippe Bekaert. *Advanced Global Illumination*. A. K. Peters, Ltd., Natick, MA, USA, 2002.

[22] Cass Watson Everitt. High-quality, hardware-accelerated per-pixel illumination for consumer class opengl hardware. Master's thesis, Mississippi State University, Mississippi State, Mississippi, May 2000.

[23] Tim Foley and Jeremy Sugerman. Kd-tree acceleration structures for a GPU raytracer. In *Graphics Hardware 2005*, pages 15–22, July 2005.

[24] Pascal Gautron, Jaroslav Kriv´anek, Kadi Bouatouch, and Sumanta Pattanaik. Radiance cache splatting: A GPU-friendly global illumination algorithm. In *Rendering Techniques 2005: 16th Eurographics Workshop on Rendering*, pages 55–64, June 2005.

[25] N. Goodnight, R. Wang, C. Woolley, and G. Humphreys. Interactive time-dependent tone mapping using programmable graphics hardware. In *EGSR03, Eurographics Symposium on rendering 2003*, pages 26–37, 2003.

[26] Henri Gouraud. Continuous shading of curved surfaces. pages 87–93, 1998.

[27] Miloš Hašan, Fabio Pellacini, and Kavita Bala. Direct-to-indirect transfer for cinematic relighting. *ACM Transactions on Graphics*, 25(3):1089–1097, July 2006.

[28] J.M. Hasenfratz, M. Lapierre, N. Holzschuch, and F. Sillion. A survey of real-time soft shadows algorithms. *Computer Graphics Forum*, 22(4):753–774, 2003.

[29] Vlastimil Havran, Miloslaw Smyk, Grzegorz Krawczyk, Karol Myszkowski, and Hans-Peter Seidel. Interactive system for dynamic scene lighting using captured video environment maps. In *Rendering Techniques 2005: 16th Eurographics Workshop on Rendering*, pages 31–42, June 2005.

[30] L. Itti and C. Koch. A saliency-based search mechanism for overt and covert shifts of visual attention. In *Vision Research Journal*, volume 40, pages 1489–1506, 2000.

[31] Kei Iwasaki, Yoshinori Dobashi, Fujiichi Yoshimoto, and Tomoyuki Nishita. Precomputed radiance transfer for dynamic scenes taking into account light interreflection. In *Rendering Techniques 2007: Eurographics Symposium on Rendering*, pages 35–44, june 2007.

[32] Katrien Jacobs and Celine Loscos. Classification of illumination methods for mixed reality. *Computer Graphics Forum*, 25(1):29–51, March 2006.

[33] Henry Johan and Tomoyuki Nishita. Clustering environment lights for an efficient all-frequency relighting. *IEICE - Trans. Inf. Syst.*, E89-D(9):2562–2571, 2006.

[34] James T. Kajiya. The rendering equation. In *Computer Graphics (Proceedings of Siggraph 86)*, pages 143–150, August 1986.

[35] Jan Kautz. Hardware lighting and shading: a survey. *Computer Graphics Forum*, 23:85–112(28), March 2004.

[36] Jan Kautz, Jaakko Lehtinen, and Timo Aila. Hemispherical rasterization for self-shadowing of dynamic objects. In *Rendering Techniques 2004: 15th Eurographics Workshop on Rendering*, pages 179–184, June 2004.

[37] Alexander Keller. Instant radiosity. In *Proceedings of Siggraph 97*, Computer Graphics Proceedings, Annual Conference Series, pages 49–56, August 1997.

[38] Thomas Kollig and Alexander Keller. Efficient illumination by high dynamic range images. In *Eurographics Symposium on Rendering: 14th Eurographics Workshop on Rendering*, pages 45–51, June 2003.

[39] Janne Kontkanen and Timo Aila. Ambient occlusion for animated characters. In *Rendering Techniques 2006: 17th Eurographics Workshop on Rendering*, pages 343–348, June 2006.

[40] Janne Kontkanen and Samuli Laine. Ambient occlusion fields. In *SI3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 41–48, New York, NY, USA, 2005. ACM Press.

[41] Janne Kontkanen, Emmanuel Turquin, Nicolas Holzschuch, and François X. Sillion. Wavelet radiance transport for interactive indirect lighting. In *Rendering Techniques 2006: 17th Eurographics Workshop on Rendering*, pages 161–172, June 2006.

[42] G. Krawczyk, K. Myszkowski, and H.-P. Seidel. Perceptual effects in real-time tone mapping with the support of graphics hardware. In *Spring Conference in Computer Graphics*, pages 195–202, 2005.

[43] Anders Wang Kristensen, Tomas Akenine-Möller, and Henrik Wann Jensen. Precomputed local radiance transfer for real-time lighting design. *ACM Transactions on Graphics*, 24(3):1208–1215, August 2005.

[44] J. Krivanek, P. Gautron, S. Pattanaik, and K. Bouatouch. Radiance caching for efficient global illumination computation. *IEEE Transactions on Visualization and Computer Graphics*, 11(5):550–561, September/October 2005.

[45] Jaroslav Krivanek, Kadi Bouatouch, Sumanta Pattanaik, and Jiri Zara. Making radiance and irradiance caching practical: Adaptive caching and neighbor clamping. In *Rendering Techniques 2006: 17th Eurographics Workshop on Rendering*, pages 127–138, June 2006.

[46] Samuli Laine, Hannu Saransaari, Janne Kontkanen, Jaakko Lehtinen, and Timo Aila. Incremental instant radiosity for real-time indirect illumination. In *Rendering Techniques 2007: Eurographics Symposium on Rendering*, pages 277–286, june 2007.

[47] J. Lambert. Photometry, or, on the measure and gradations of light, colors, and shade: translation from the latin photometria sive de mensura et gradibus luminus, colorum et umbrae. 1760.

[48] Gregory Lecot, Bruno Levy, Laurent Alonso, and Jean-Claude Paul. Master-element vector irradiance for large tessellated models. In *Graphite '05: Proceedings of the 3rd international conference on Comp uter graphics and interactive techniques in Australasia and South East Asia*, 2005.

[49] Xinguo Liu, Peter-Pike Sloan, Heung-Yeung Shum, and John Snyder. All-frequency precomputed radiance transfer for glossy objects. In *Rendering Techniques 2004: 15th Eurographics Workshop on Rendering*, pages 337–344, June 2004.

[50] Mattias Malmer, Fredrik Malmer, Ulf Assarsson, and Nicolas Holzschuch. Fast precomputed ambient occlusion for proximity shadows. *Journal of Graphics Tools*, 12(2):57–71, 2007.

[51] C. Mei, J. Shi, and F. Wu. Rendering with Spherical Radiance Transport Maps. *Computer Graphics Forum*, 23(3):281–290, 2004.

[52] D. Michael and Y. Chrysanthou. Automatic high level avatar guidance based on affordance of movement. In *Proceedings Interactive Demos and Posters, Eurographics 2003 short paper*, pages 221–226, 2003.

[53] D. Michael and Y. Chrysanthou. Fullsphere irradiance factorization for real-time all-frequency illumination for dynamic scenes. *Computer Graphics Forum, DOI: 10.1111/j.1467-8659.2010.01818.x*, 2010.

[54] D. Michael, N. Pelekanos, I. Chrysanthou, P. Zaharias, L. Hadjigavriel, and Y. Chrysanthou. Comparative study of interactive systems in a museum. In *Proceedings of Euromed 2010*, 2010.

[55] D. Michael, P. Zaharias, and Y. Chrysanthou. A virtual tour of the walls of Nicosia: An assessment of childrens' experience and learning effectiveness. In *VAST 2010, short paper*, 2010.

[56] Gene S. Miller and C. Robert Hoffman. Illumination and reflection maps: Simulated objects in simulated and real environments. *Course Notes for Advances Computer Graphics Animation, Siggraph 1984*, 1984.

[57] Karol Myszkowski, Takehiro Tawara, Hiroyuki Akamine, and Hans-Peter Seidel. Perception-guided global illumination solution for animation rendering. In *Proceedings of ACM Siggraph 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 221–230, August 2001.

[58] Ren Ng, Ravi Ramamoorthi, and Pat Hanrahan. All-frequency shadows using non-linear wavelet lighting approximation. *ACM Transactions on Graphics*, 22(3):376–381, July 2003.

[59] J. M. Ogden, E. H. Adelson, J. R. Bergen, and P. J. Burt. Pyramid-based computer graphics. *RCA Engineer*, 30, 1985.

[60] M. Ouhyoung, Y.Y. Chuang, and R.H. Liang. Reusable radiosity objects. *Computer Graphics Forum*, 15(3):347–356, 1996.

[61] Ryan Overbeck, Ravi Ramammoorthi, and William R. Mark. A real-time beam tracer with application to exact soft shadows. In *Rendering Techniques 2007: Eurographics Symposium on Rendering*, pages 85–98, june 2007.

[62] P. Patias, Y. Chrysanthou, S. Sylaiou, Ch. Georgiades, D. Michael, and S. Stylianidis. The development of an e-museum for contemporary arts. In *Proceedings of the 14th International Conference on Virtual Systems and Multimedia*, 2008.

[63] Ravi Ramamoorthi and Pat Hanrahan. An efficient representation for irradiance environment maps. In *Siggraph '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 497–500, New York, NY, USA, 2001. ACM.

[64] Mahesh Ramasubramanian, Sumanta N. Pattanaik, and Donald P. Greenberg. A perceptually based physical error metric for realistic image synthesis. In *Proceedings of Siggraph 99*, Computer Graphics Proceedings, Annual Conference Series, pages 73–82, August 1999.

[65] Erik Reinhard, Brian Smits, and Chuck Hansen. Dynamic acceleration structures for interactive ray tracing. In *Rendering Techniques 2000: 11th Eurographics Workshop on Rendering*, pages 299–306, June 2000.

[66] Erik Reinhard, Michael Stark, Peter Shirley, and Jim Ferwerda. Photographic tone reproduction for digital images. In *Proceedings of Siggraph 2002*, pages 267–276, 2002.

[67] Zhong Ren, Rui Wang, John Snyder, Kun Zhou, Xinguo Liu, Bo Sun, Peter-Pike Sloan, Hujun Bao, Qunsheng Peng, and Baining Guo. Real-time soft shadows in dynamic scenes using spherical harmonic exponentiation. *ACM Transactions on Graphics*, 25(3):977–986, July 2006.

[68] Alexander Reshetov, Alexei Soupikov, and Jim Hurley. Multi-level ray tracing algorithm. *ACM Trans. Graph.*, 24(3):1176–1185, 2005.

[69] Benjamin Roch, Alessandro Artusi, Despina Michael, Yiorgos Chrysanthou, and Alan Chalmers. Interactive local tone mapping operator with the support of graphics hardware. In *Spring Conference on Computer Graphics 2007*, 2007.

[70] M. Sattler, R. Sarlette, G. Zachmann, and R. Klein. Hardware-accelerated ambient occlusion computation. In *9th International Fall Workshop Vision, Modeling and Visualization (VMV)*, pages 119–135, Stanford (California), USA, November16–18 2004.

[71] Mateu Sbert. *The Use of Global Random Directions to Compute Radiosity: Global Monte Carlo Techniques*. PhD thesis, Universitat Politecnica de Catalunya, Barcelona, Spain, 1997.

[72] Mateu Sbert, Xavier Pueyo, Lazlo Neumann, and Werner Purgathofer. Global multipath monte carlo algorithms for radiosity. *The Visual Computer*, 12(2):47–61, 1996.

[73] A. Scheel, M. Stamminger, and H.-P. Seidel. Tone reproduction for interactive walkthroughs. In *Computer Graphics Forum (Proceedings of Eurographics 2000)*, volume 19(3), pages 301–312, 2000.

[74] Benjamin Segovia, Jean Clau de Iehl, Richard Mitanchey, and Bernard Péroche. Bidirectional instant radiosity. In *Rendering Techniques 2006: 17th Eurographics Workshop on Rendering*, pages 389–398, June 2006.

[75] François X. Sillion and Claude Puech. A general two-pass method integrating specular and diffuse reflection. In *Computer Graphics (Proceedings of Siggraph 89)*, pages 335–344, July 1989.

[76] Peter-Pike Sloan. Normal mapping for precomputed radiance transfer. In *SI3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 23–26, New York, NY, USA, 2006. ACM Press.

[77] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Transactions on Graphics*, 21(3):527–536, July 2002.

[78] William A. Stokes, James A. Ferwerda, Bruce Walter, and Donald P. Greenberg. Perceptual illumination components: a new approach to efficient, high quality global illumination rendering. *ACM Transactions on Graphics*, 23(3):742–749, August 2004.

[79] Parag Tole, Fabio Pellacini, Bruce Walter, and Donald P. Greenberg. Interactive global illumination in dynamic scenes. In *Siggraph '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 537–546, New York, NY, USA, 2002. ACM Press.

[80] Pedro Trancoso and Maria Charalambous. Exploring graphics processor performance for general purpose applications. In *DSD '05: Proceedings of the 8th Euromicro Conference on Digital System Design*, pages 306–313, Washington, DC, USA, 2005. IEEE Computer Society.

[81] Yu-Ting Tsai and Zen-Chung Shih. All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation. *ACM Transactions on Graphics*, 25(3):967–976, July 2006.

[82] Jack Tumblin and Holly E. Rushmeier. Tone reproduction for realistic images. *IEEE Computer Graphics and Applications*, 13(6):42–48, November 1993.

[83] Jack Tumblin and Greg Turk. LCIS: A boundary hierarchy for detail-preserving contrast reduction. In *Computer Graphics (Siggraph '99 Proceedings)*, pages 83–90, 1999.

[84] Jonas Unger, Magnus Wrenninge, and Mark Ollila. Real-time image based lighting in software using hdr panoramas. In *Graphite 2003*, pages 263–264, 2003.

[85] I. Wald, S. Boulos, and P. Shirley. Ray tracing deformable scenes using dynamic bounding volume hierarchies. *ACM Transactions on Graphics (TOG)*, 26(1), 2007.

[86] Ingo Wald, Solomon Boulos, and Peter Shirley. Ray tracing deformable scenes using dynamic bounding volume hierarchies. *ACM Trans. Graph.*, 26(1):6, 2007.

[87] Ingo Wald, Thiago Ize, Andrew Kensler, Aaron Knoll, and Steven G. Parker. Ray tracing animated scenes using coherent grid traversal. *ACM Transactions on Graphics*, 25(3):485–493, July 2006.

[88] Ingo Wald, Thomas Kollig, Carsten Benthin, Alexander Keller, and Philipp Slusallek. Interactive global illumination using fast ray tracing. In *Rendering Techniques 2002: 13th Eurographics Workshop on Rendering*, pages 15–24, June 2002.

[89] Ingo Wald, William R Mark, Johannes Günther, Solomon Boulos, Thiago Ize, Warren Hunt, Steven G Parker, and Peter Shirley. State of the art in ray tracing animated scenes. In *STAR Proceedings of Eurographics 2007*. Eurographics Association, September 2007. to appear.

[90] John R. Wallace, Michael F. Cohen, and Donald P. Greenberg. A two-pass solution to the rendering equation: A synthesis of ray tracing and radiosity methods. In *Computer Graphics (Proceedings of Siggraph 87)*, pages 311–320, July 1987.

[91] B. Walter, S. N. Pattanaik, and D. P. Greenberg. Using perceptual texture masking for efficient image synthesis. *Computer Graphics Forum*, 21(3):393–399, 2002.

[92] Bruce Walter, Adam Arbree, Kavita Bala, and Donald P. Greenberg. Multidimensional lightcuts. *ACM Transactions on Graphics*, 25(3):1081–1088, July 2006.

[93] Bruce Walter, Sebastian Fernandez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P. Greenberg. Lightcuts: a scalable approach to illumination. *ACM Transactions on Graphics*, 24(3):1098–1107, August 2005.

[94] Liang Wan, Tien-Tsin Wong, and Chi-Sing Leung. Spherical Q2-tree for sampling dynamic environment sequences. In *Rendering Techniques 2005: 16th Eurographics Workshop on Rendering*, pages 21–30, June 2005.

[95] Rui Wang, Jiajun Zhu, and Greg Humphreys. Precomputed radiance transfer for real time indirect lighting using a spectral mesh basis. In *Rendering Techniques 2007: Eurographics Symposium on Rendering*, pages 13–21, june 2007.

[96] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assesment: From error visibility to structural similarity. In *IEEE Transaction on Image Processing*, volume 13, pages 600–612, 2004.

[97] Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A ray tracing solution for diffuse interreflection. In *Computer Graphics (Proceedings of Siggraph 88)*, pages 85–92, August 1988.

[98] Gregory Ward Larson, Holly Rushmeier, and Christine Piatko. A Visibility Matching Tone Reproduction Operator for High Dynamic Range Scenes. *IEEE Transactions on Visualization and Computer Graphics*, 3(4):291–306, October 1997.

[99] Turner Whitted. An improved illumination model for shaded display. *Communications of the ACM*, 6(23):343–349, 1980.

[100] Lance Williams. Casting curved shadows on curved surfaces. In *Computer Graphics (Proceedings of Siggraph 78)*, pages 270–274, August 1978.

[101] Andrew J. Willmott. *Hierarchical Radiosity with Multiresolution Meshes*. PhD thesis, Computer Science, Carnegie Mellon Univerity, 2000.

[102] Andrew J. Willmott, Paul S. Heckbert, and Michael Garland. Face cluster radiosity. In *In Eurographics Workshop on Rendering*, pages 293–304. Springer, 1999.

[103] C. Wyman and C. Hansen. Penumbra maps: approximate soft shadows in real-time. *Proceedings of the 14th Eurographics workshop on Rendering*, pages 202–207, 2003.

[104] Sung-Eui Yoon, Sean Curtis, and Dinesh Manocha. Ray tracing dynamic scenes using selective restructuring. In *Rendering Techniques 2007: Eurographics Symposium on Rendering*, pages 73–84, june 2007.

[105] Kun Zhou, Yaohua Hu, Stephen Lin, Baining Guo, and Heung-Yeung Shum. Precomputed shadow fields for dynamic scenes. *ACM Transactions on Graphics*, 24(3):1196–1201, August 2005.

[106] Sergej Zhukov, Andrej Iones, and Grigorij Kronin. An ambient light illumination model. In *Eurographics Rendering Workshop 1998*, pages 45–56, June 1998.