PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
Ministry of Higher Education and Scientific Research
MOHAMED KHIDER UNIVERSITY - BISKRA
Faculty of Exacts Sciences, Nature and Life Sciences
Department of Computer Science

Order N°: IVA 07/M2/2020

Master Thesis

Presented for the academic master's degree in

Computer Science

Path : Image and Artificial Life

Cascaded Soft Shadow Maps Based On PCF Filtering

Presented by

Nadia AZRI

Submitted on 20/09/2020, before the board of examiners composed of:

- Aa BB	Prof	Chair
- Abd El Moumène ZERARI	MCA	Supervisor
- Aa BB	Prof	Examiner

Acknowledgments

All the thanks are for god, which gave me the force and the will to accomplish this work.

On this occasion, I would like to give a special thanks, to my teacher and supervisor Dr. ZERARI Abd El Moumène, not only for the invaluable input and suggestions on technical issues but also for making me feel like part of the team from day one.

Many thanks to all my teachers at the University of Mohamed Khider Biskra.

And finally, I'd like to thank my family and my friends who are always there for support, through thick and thin.

Nadia AZRI

Dedication

To my:

Mother and Father,

Brothers and Sisters.

Nadia AZRI

« Loyal as a shadow . . . »

Popular expression

Abstract

Soft shadows are one of the key features that distinguish realistic images from those that look unnatural. An ever-increasing number of algorithms have been proposed for the task, but the computation of physically-based soft shadows remains costly due to the large number of samples required from the area light source and is avoided in production whenever possible. In practice, soft shadow effects are most often produced using approximate methods from global illumination.

Physically accurate soft shadows in real-time applications can be imitated by subdividing the area light source into multiple samples and creating a shadow map for each sample and then accumulating them. In this work, we fitted in this context, and we proposed a new technique to reduce the rendering time required to achieve a high quality of soft shadows in real-time rendering. We generated physically accurate soft shadows using the Cascaded Shadow Maps technique (CSMs) to reduce further the samples of the area light source on far cascades, and use multiple implemented PCF filters on the GPU which allowed us reducing the samples by removing the artifacts and soften the resulting soft shadows. Both the CSMs technique and the PCF filtering freed-up significant computing time. The results show that our technique improves efficiency by creating high-quality soft shadows in real-time, and is undoubtedly faster than traditional techniques.

Keywords: soft shadow, real-time rendering, cascaded shadow maps, GPU.

Résumé

Les ombres douces sont l'une des principales caractéristiques qui distinguent les images réalistes de celles qui ne semblent pas naturelles. Un nombre croissant d'algorithmes a été proposé pour la tâche, mais le calcul des ombres douces physiquement correctes reste coûteux en raison du grand nombre requis d'échantillons de la source de lumière surfacique et est évité en production chaque fois que possible. En pratique, les effets d'ombre douce sont le plus souvent produits à l'aide de méthodes approximatives à partir de l'illumination global.

Les ombres douces physiquement précises dans les applications en temps réel peuvent être imitées en subdivisant la source de lumière surfacique en plusieurs échantillons et en créant une carte des ombres pour chaque échantillon, puis en les accumulant. Dans ce travail, nous nous sommes installés dans ce contexte, et nous avons proposé une nouvelle technique pour réduire le temps de calcule nécessaire pour obtenir une haute qualité d'ombres douces en temps réel. Nous avons généré des ombres douces physiquement précises en utilisant la technique Cascaded Shadow Maps (CSM) pour réduire davantage les échantillons de la source de lumière surfacique sur les cascades lointaines, et plusieurs filtres PCF implémentés sur le GPU qui nous ont permis de réduire les échantillons en supprimant les bandes d'artefacts et adoucissez les ombres douces qui en résultent. La technique CSMs et le filtrage PCF ont libéré un temps de calcul important. Les résultats montrent que notre technique améliore l'efficacité en créant des ombres douces de haute qualité en temps réel, et est nettement plus rapide que les techniques traditionnelles.

Mots clés: ombre douce, rendu en temps réel, shadow maps en cascade, GPU.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

2D	Two- Dimensional
3D	Three-Dimensional
APIs	Application Programming Interfaces
BRDF	Bidirectional Reflectance Distribution Function
CSMs	Cascaded Shadow Maps
CSSMs	Cascaded Soft Shadow Maps
DXR	DirectX Raytracing
FPS	Frames Per Second
GPU	Graphics Processing Unit
PCF	Percentage Closer Filtering
PLS	Point Light Source
PSSM	Parallel-Split Shadow Maps
RMSE	Root Mean Square Error
RT	Ray Tracing
SSM	Soft Shadow Mapping

Introduction

The field of computer graphics is a booming field. Computer generated images have become indispensable in many fields, such as cinema, the video game industry, and even the medical field. Generation of these images based on the realistic illumination of a 3D scene, such as transporting light in a virtual environment is accomplished through global illumination algorithms. Indirect lighting simulates complicated effects such as shadows, refractions, and caustics because it takes into account all the interactions between a light ray and a surface. It is practically impossible to find an analytical solution to the rendering equation, especially for complex geometric scenes. Thus, the researchers focused on using numerical methods such as the Monte Carlo method [Kaj86] to estimate the energy that arrives at the pixel. Currently, we can obtain sufficient image generation rates only by approximating global illumination.

Shadows play a crucial role in the understanding of shape, size, and relative location of geometry in three-dimensional space. Many properties of shadow casters and receivers can be derived by inspecting a shadow silhouette and its interaction with surfaces in the scene. Rendering shadows requires solving a point-to-surface visibility problem. In case of hard shadow, it amounts to solving a simple problem of visibility between two points for each pixel. However, rendering a soft shadow requires knowing the percentage of occlusion in each pixel. Global illumination methods are not suitable for real-time rendering, due to cost of visibility which is the main expense in most methods. However, global illumination approximation methods [RDGK12] can generate shadows in real time. Although, Real-time applications typically use shadow mapping [Wil78] and shadow volume [Cro77] algorithms. Shadow mapping is highly flexible in terms of scene geometry, but it has some important issues such as perspective aliasing. Recently, several approaches have been tried to reduce those aliasing artifacts, Cascaded Shadow Maps (CSMs) [Eng06] is one of them. In order to fix the aliasing problems, CSMs provide higher resolution of the depth texture near the viewer and lower resolution far away by splitting the camera view frustum into separate partitions and creating a depth-map for each partition.

1 Motivation

Generating a physically accurate soft shadow with multiple shadow maps required sampling the area light source into n samples. A shadow map is created for each point light from a view position in a scene render pass, therefore no render passes will be required to generate the soft shadow. The computing time increases if the number of the shadow maps is large, However, for the real-time applications this number should be reduced to reduce the computation time, several methods are created to speed up the generating of the soft shadows by reducing the number of samples. A method that achieves this has already been developed by Schwärzler et al. [SMSW12], but only by reducing the number of samples by the adaptive light source subdivision technique. this method starts by sampling the area light source to a small number of samples and generating additional sampling points if necessary. The problem with this method is that the size of the penumbra areas can change drastically in a short time, the number of samples needed can also vary considerably, making this approach unsuitable for real-time applications.

For real-time applications, the available computational power is strictly limited and may require a wide-range of approximations and assumptions such as the use of Monte-Carlo integration models in order to render soft shadows. It's for this reason preferable to reduce the computation time of shadows not only to produce shadows of higher quality for the same computational cost but also to free up rendering time for other effects which have not been tried before because of the volume of intensive computing which they generate in order to improve the overall image quality.

2 Aim and Contribution

The goal of this thesis project is to extend the physically accurate real-time soft shadow creation method of Schwärzler et al. [SMSW12] by further reducing the number of samples required and speeding up the computation without sacrificing image quality.

We provide a new technique for generating physically accurate soft shadows from an area light source based on CSMs technique and multiple PCF filters to reduce the number of shadow maps needed to achieve high quality soft shadows in real-time rendering.

3 Structure of this thesis

The organization of the thesis spread over three chapters:

The first chapter of this establish the calculation of illumination, where we will explain

in detail the two types of illumination namely: local and global illumination.; radiometric quantities, photometric quantities, propagation of light, up to the Kajiya rendering equation, Monte-Carlo method. Then, we describe some basic global illumination algorithms and the method of creating shadows in these algorithms.

The second chapter followed by an overview of the current state-of-the-art of the real-time shadow techniques. We focused on the physically accurate soft shadow and presenting some of its methods.

The third chapter presents our proposal, it introduces a new technique for creating physically accurate soft shadows using the CSMs technique and multiple PCF filters in real-time applications. We start with a general description of our technique, followed by the different details of each part of our technique. At the end, we present our results with a comparative study with the method of Michael Schwärzler et al. [SMSW12].

Chapter1

Illumination

1.1 Introduction

Smooth surfaces with uniform colors, brilliant lights with brilliant reflections. Here is perhaps the main defect of computer-generated images: they are too perfect. Fighting this perfection to get closer to the real has become one of the major challenges of image synthesis. The generation of so-called realistic photo images requires an exact calculation of the light distribution in the 3D scene. The creation of these images remains difficult because of the complexity of the illumination present in the real world where the light emitted by a source bounces on the objects which constitute this world, in fact, the calculation of the color of a point in a 3D scene requires the integration of all the incident lighting. Several methods have been developed for the calculation of lighting. Generally, we can classify them under two big classes: the first is local illumination which calculates the lighting by considering only the interactions between the light sources and the object, while the second is the global illumination which not only takes into account object-source interactions but also object-object interactions.

In this chapter, we first describe a local illumination model, and the light distribution methods, evoking the different basic concepts of radiometry and geometry, then we talk about BRDF and their models, also we are interested in defining the mathematical formulation of the rendering equation, as well as an overview of the global illumination algorithms and also shadows in offline applications.

1.2 Illumination

Illumination is the transport of luminous flux from light sources through both direct and indirect paths. In computer graphics, light usually consists of multiple components. The overall effect of a light source on an object is determined by the combination of the object's interactions with these components.

1.2.1 Local illumination

A local illumination model takes incident light and the properties of the material at one point on the surface. It models the reaction of the surface to light and produces the light returned by the object. Here the term light combines both color and light intensity [Lef05].

General Model of Local Illumination :

$$L(p \rightarrow e) = \rho_a L_a + \sum_k \rho(p, e, I_K)(n.I_k)L(p \leftarrow I_k) \tag{1.1}$$

Where:

- $\boldsymbol{e}$: is the direction of the view (gaze),
- $\boldsymbol{n}$: is normal at point p,
- $\boldsymbol{I}$: is the direction of the incident light,
- $\boldsymbol{L(p \rightarrow e)}$: is the outgoing radiance; how much energy gets to the eye,
- $\boldsymbol{L(p \leftarrow I_k)}$: is the incoming radiance; all the illumination coming from I and arriving at point p,
- $\boldsymbol{(n.I_k)}$: is the orientation of the surface; the dot product between the normal at the point and the direction of the light,
- $\boldsymbol{\rho(p, e, I_K)}$: are the material properties; how much energy the surface reflects towards the eye given an incident light I. They are called BRDF (Bidirectional Reflectance Distribution Function),
- $\boldsymbol{k}$:is the number of lights in the scene,
- $\boldsymbol{\rho_a L_a}$: is the ambient light (an approximation of indirect light).

1.2.1.1 Illumination models

An illumination model defines the reaction of light striking a surface. Depending on the complexity of the models used, the different effects of light behavior will be more or less well reproduced [PY06].

1.2.1.1.1 Ambient light

Ambient light has no spatial or directional characteristics. The amount of ambient light incident on each object is constant for all surfaces and overall directions.

It is entirely defined by its intensity. The surfaces reflect this light in proportion to a coefficient k_a. The reflected intensity is therefore worth [Fra02]:

$$L = k_a \times I_a \tag{1.2}$$

Where: $(0 \leq k_a \leq 1)$

1.2.1.1.2 Diffuse model

The simplest re-emission model is known as the Lambertian model. This is the ideal diffuse model (see Figure 1.1). The surface appears equally bright from all viewing directions [PY06].

The Lambert model is written as follows:

$$I_d = I_i \times K_d \times cos\theta \tag{1.3}$$

Where:

- $\boldsymbol{I_d}$: diffuse intensity; i.e. the intensity of outgoing light (whatever the direction, by isotropy),
- $\boldsymbol{I_i}$: incident intensity,
- $\boldsymbol{K_d}$: texture parameter setting the behavior, a low coefficient translates an absorbent object while a high coefficient translates a strong re-emission,
- $\boldsymbol{\theta}$: angle of incidence of light; calculated concerning the normal vector at the surface.

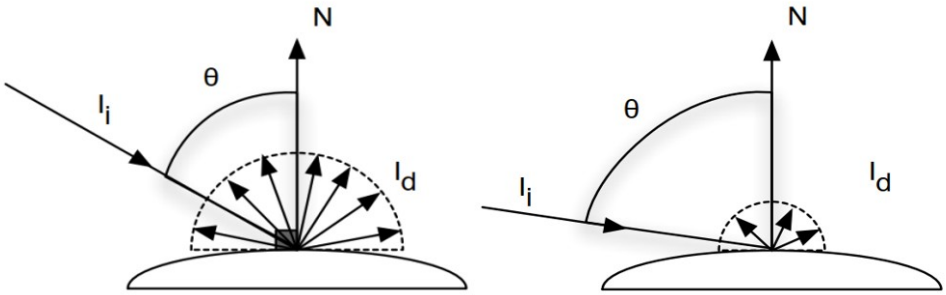

Figure. 1.1. The Lambertian model [PY06].

1.2.1.1.3 Specular reflection

Specular reflection is a type of surface reflectance often described as a mirror-like reflection of light from the surface. In specular reflection, the incident light is reflected into a single outgoing direction. The Phong model (see Figure 1.2) is a classic specular model; it is written as follows:

$$I_s = I_i \times K_s \times (cos\theta)^n \tag{1.4}$$

Where:

- $\boldsymbol{I_s}$: specular intensity; i.e. the intensity of outgoing light in a given direction,

- $\boldsymbol{I_i}$: incident intensity,
- $\boldsymbol{K_s}$: texture parameter setting the behavior; a low coefficient translates an object with a low specular component,
- $\boldsymbol{\theta}$: angle measured between the direction of exit considered and the ideal reflected ray,
- $\boldsymbol{n}$: texture parameter characterizing the polish of the object; a large n indicates acute specularity, the bright point tightening around the direction of the ideal reflected ray.

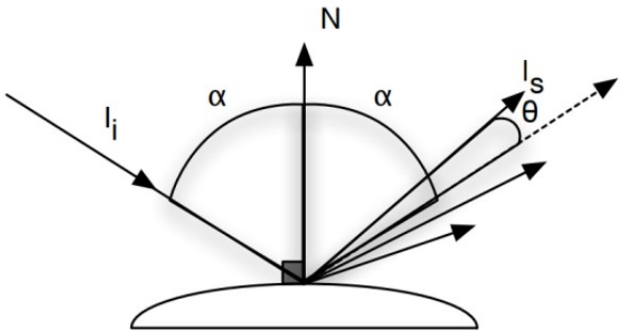

Figure. 1.2. The specular model of Phong: the re-emission is more or less intense around the ideal reflected ray [PY06].

1.2.1.1.4 Phong illumination model

The most used local illumination model is the model defined by B.T. PHONG in 1975 [Pho75]. It defines the light intensity reflected at a point P on a surface as a function of the intensity of the light sources. This model is a generalization of Lambert's law on the reflection of light. This law, also called the cosine law, describes the intensity scattered by a point on a matt surface as a function of the intensity and direction of the source taken into account [Zer11].

$$I(P)_d = K_d.cos\alpha.I_s \tag{1.5}$$

Where:

- $\boldsymbol{I(P)_d}$: intensity scattered at point P,
- $\boldsymbol{K_d}$: coefficient of scattering at point P,
- $\boldsymbol{I_s}$: source intensity,
- $\boldsymbol{\alpha}$: angle between normal and direction of the source.

Similarly, for a point located on a shiny surface, we can write the following law of reflection (equation 1.6):

$$I(P)_r = K_s(\alpha).cos^n\beta.I_s \tag{1.6}$$

Where:

- $\boldsymbol{I(P)_r}$: Intensity reflected at point P,
- $\boldsymbol{I_s}$: intensity of the source,
- $\boldsymbol{I_s}$: reflection coefficient at point P (depends on the angle α),
- $\boldsymbol{\alpha}$: angle between normal and direction of the source,
- $\boldsymbol{\beta}$: angle between the reflected direction and the direction of the observer,
- $\boldsymbol{n}$: exhibitor of Phong.

The complete PHONG model is deduced from equations 1.5 and 1.6, it is written for several light sources [Zer11]:

$$I(P) = K_d.I_a + \sum_{i \leq N} K_d.cos\alpha_i.I_i + \sum_{i \leq N} K_s(\alpha_i).cos^n\beta_i.I_i \tag{1.7}$$

Where:

- $\boldsymbol{I_a}$: estimated ambient intensity,
- $\boldsymbol{N}$: number of light sources in the scene.

1.2.2 Global illumination

Global Illumination refers to all the rendering methods, that include other objects in a scene when calculating the light arriving at a point in a scene (see Figure 1.3). Opposed to this we have the so-called local illumination, that just uses the lighted point and the light sources to calculate the light (object-object interactions)[Rad08].

The role of global illumination is to calculate the distribution of light energy in a scene. Global lighting allows you to:

- Calculate light-object, object-matter interactions,
- Improve the visual appearance of objects in the scene,
- Add the effects of shadows to the scene,
- Introduce the calculation of the rays of reflections, refractions.

Images rendered using global illumination techniques are more realistic than images rendered using local illumination techniques. However, they are also much slower and more expensive in terms of rendering time.

Figure. 1.3. Global illumination. Left image: the Box scene [Dav07]. Right image: the Crytek Sponza scene illuminated by one-directional and 12 medium-sized point lights.

1.2.3 Light sources

There are many different types lights in 3D scenes.

1. **Ambient light.** It is a diffuse (non-directional) light of which it is not possible to determine the source. Any object of the scene is immersed in this light [Pas05]. It can be an important contributor to the overall look and brightness of a scene.
2. **Point light.** A point light source is a light whose light rays are emitted from a single point in all directions [Pas05].

   It is characterized by its intensity I and its position P. At any point A, the direction of this point source is obtained with the vector $L = AP/ \mid AP \mid$ [Zer11].
3. **Spot light.** A spot light has a specified by their direction, their position, and a concentration factor, like a point light.
4. **Directional light.** Directional lights are always used to create effects such as sunlight in scenes. Directional lights can be thought of as distant light sources which exist infinitely far away, behaving in many ways like the sun. Directional light can be placed anywhere in the scene and so the light object does not have any identifiable source position.

   As if the light is always from the same direction, all objects in the scene are illuminated. The light does not diminish and so the distance of the light from the target object is not defined.

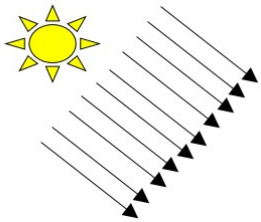

Figure. 1.4. Directional light.

5. **Area light.** A rectangle in space is how we define an area light. Only from one side of the rectangle, light is emitted in all directions uniformly across their surface area. Area light sources require an estimation of how much is the area of the light source visible from the current pixel [Tom12].

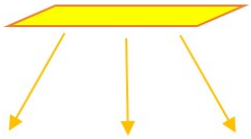

Figure. 1.5. Area light source.

The area light is similar to a studio soft-box style light, often used in professional photography because of the soft shadows they produce. The light comes from a rectangle or a round plane depending on the lighting parameters. The area light is great for creating photo-realistic lighting effects as it creates smooth shading and very natural light emission. They are generally more expensive, render time-wise, than other types of light because they require more samples due to their size, but their natural tendency to create shadows with soft edges is worth the trade-off in many situations.

Figure. 1.6. Soft shadow created by an area light source approximated to 1024 samples of point lights [HLHS03].

Area lights use a physically accurate light model so that the light intensity decreases with the inverse square of the distance to the light. e.g. an object is half as bright when the light is twice as far away. As a physically based light, the intensity is also increased as the size of the light increases. Increasing the size of the light compared to the model it affects also increases the amount of shadow spread (see Figure 1.6).

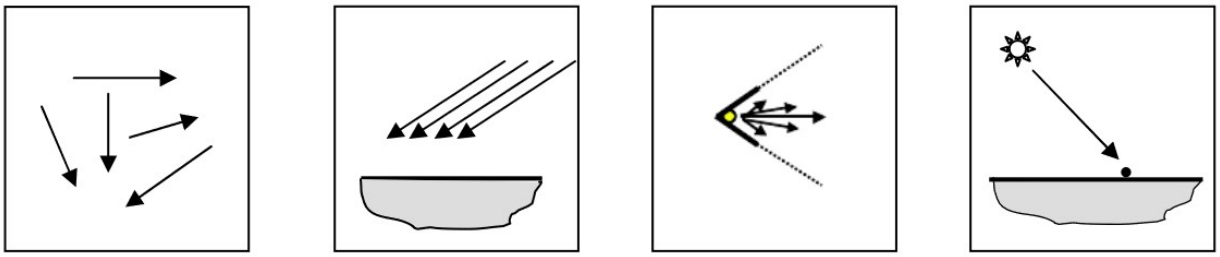

Figure. 1.7. Types of light sources [Zer11]. From left to right: ambient light, directional light, spot light, and point light.

1.3 Shading methods

Shading is a misleading term in computer graphics. In an illumination context, it designates the method of filling the polygons according to the normals used. There are three methods [Fra02]:

- Flat shading,
- Gouraud shading,
- Phong shading, not to be confused with the Phong illumination model.

1.3.1 Flat shading

The simplest shading method for polygonal facets is flat (or constant) shading (see Figure 1.8). The idea is to calculate a single illumination value for the whole facet. For example at the midpoint of the facet taking as normal on the surface that of the plane containing the facet. [Fra02]

Figure. 1.8. Flat shading of the sphere for 16×16 facets, 32×32 and 64×64 [Fra02].

1.3.2 Gouraud shading

The method developed by Gouraud in 1971 (see Figure 1.9). The idea of Gouraud's shading is to interpolate the light intensity values on a polygon based on the intensity of its vertices

[Pas05]. This method is widely used and is found in the majority of existing graphics hardware (libraries, graphics cards).

This method requires knowledge of the normal to the surface at the vertices of the polygonal facets. When the normals are known, the intensities at the vertices of the polygonal facets are calculated. Interpolation can then be performed using the line scanning algorithm used for polygon filling and the z-buffer [Fra02].

The main disadvantage of this model is that the specular reflections will be very much influenced by the decomposition into polygons at the place where the specular reflection occurs.

This model has the advantage of being simple, fast, and gives good results for simple illumination models that do not integrate specular reflection. [Pas05]

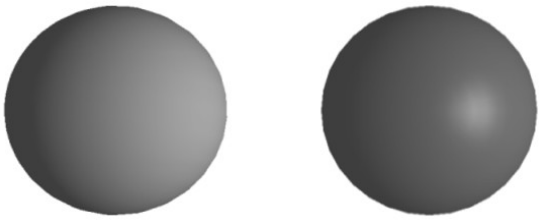

Figure. 1.9. Gouraud shading on the 32×32 sphere: diffuse reflection on the left, specular reflection on the right [Fra02].

1.3.3 Phong shading

The shading of Phong (1975) constitutes an important improvement of the Gouraud model but is more demanding in terms of calculation time (see Figure 1.10). The principle is the same that of interpolation; except that instead of interpolating the intensities, we interpolate the normals on the surfaces. This principle amounts to reconstructing a surface from the interpolation of normals [Pas05].

The interest of this approach compared to Gouraud's shading lies mainly in its ability to process specular reflections. Gouraud does not allow specular reflections to be taken into account when these are located at the center of a facet [Fra02].

Figure. 1.10. Phong shading and Phong illumination model [Fra02].

1.4 Basic definitions

Before we start, we highlight the difference between the two main concepts important in the field of illumination, namely: photometry and radiometry. Photometry is the consideration of the human perception of light energy while radiometry represents the formalism for the study of the physical quantities of light energy. The aim of studying these two quantities is to calculate the distribution of light energy in a scene. In the calculation of the lighting, only the quantities of the radiometry are used.

1.4.1 Geometric quantities: solid angle

A solid angle is an angular 3D volume which is defined in a similar way to the definition of a flat angle in two dimensions; it makes it possible to measure the surface which would be directly lit if this source were placed in the center of a sphere of unit radius. It is the ratio of the area of a part of a sphere to the radius squared. The unit of the solid angle is the steradian [Tak09].

Defined by the ratio of the section (dA) (elementary surface), cut by the intersection of the surface of a sphere and a cone whose apex is at the center of the sphere (S), squared by the radius (r) and as a function of the angle (θ) formed by the straight line joining (S) and the center of (dA) (see Figure 1.11).

$$d\omega = \frac{dAcos\theta}{r^2} \tag{1.8}$$

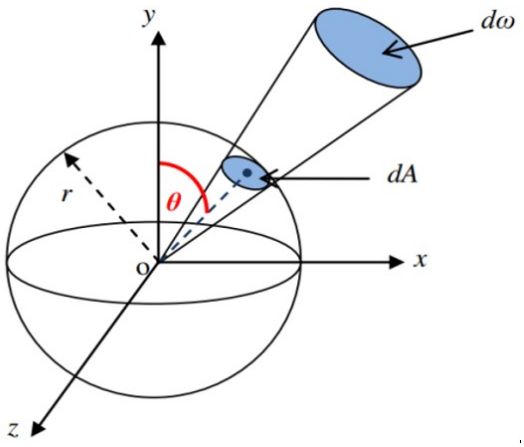

Figure. 1.11. Solid angle concept [Tak09]

1.4.2 Radiometric quantities

1. **Radiant flux:** it allows us to express in Watts the total amount of energy on a surface compared to the time unit.

$$\phi = \frac{d\phi}{dt} \tag{1.9}$$

2. **Irradiance:** is the incident radiant flux in a surface relative to a section, it is expressed in $Watts/m^2$.

$$E = \frac{d\phi}{dA} \tag{1.10}$$

3. **Luminance (radiance):** is the radiant flux compared to the surface section compared to the solid angle, it is expressed by $watts/(steradian \times m^2)$, in other words, the luminance describes how much flux emits or received at a certain surface point concerning a solid angle and for a surface section. The luminance varies with the position x and the directions θ of the observer [DBB06].

$$L(x \to \omega) = \frac{d^2\phi}{d\omega dA} = \frac{d^2\phi}{d\omega dAcos\theta} \tag{1.11}$$

4. **Light intensity:** the light intensity I, the unit of which is the candela (cd), is a measure of the brightness of a light source as perceived by the human eye. It indicates the luminous flux leaving a point x, emitted by the unit of solid angle ω in a given direction. The greater the light intensity, the brighter the illuminated surface. It is presented by the function [Tak09]:

$$L(x \to \omega) = \frac{d\phi}{d\omega} \tag{1.12}$$

1.5 The rendering equation

The rendering equation is probably one of the most popular in the computer graphics industry. Although far from perfect, it allows us to understand several aspects of what makes up this crucial step of rendering a 3D scene. Despite its decades of seniority, it is still one of the most used to understand some key concepts. Several implementations are based on this equation to propose new innovations in the field of graphic rendering.

In this section, we present the origins of the rendering equation that physically models the problem of lighting distribution in a 3D scene, starting by introducing the BRDF equation and mentioning its models, then we simplify the rendering equation, and we end up with integration by the Monte Carlo method.

1.5.1 Bidirectional reflectance distribution function (BRDF)

A surface can reflect radiation of different intensities in different directions; the intensity of the reflection varies according to the position of the light source and the angle at which the surface is observed. BRDF is used to define the relationship between the incident and reflected luminance at the same surface point.

The BRDF in a point x is defined as being the ratio between the luminance reflected in a given direction and the illumination coming from an incident direction (see Figure 1.12), its unit is the inverse of steradian.

$$f_r(x, \omega_i, \omega_0) = \frac{radiance}{irradiance} = \frac{dL_r(x \rightarrow \omega_0)}{dE(x \leftarrow \omega_i)} = \frac{dL_r(x \rightarrow \omega_0)}{L_i(x \leftarrow \omega_i) cos(N_x, \omega_i) d\omega_i} \tag{1.13}$$

BRDF shown in Figure 1.12 is an important property because it influences the overall lighting of the scene [Tak09].

1.5.1.1 BRDF models

A variety of models have been proposed to represent BRDFs. The first BRDF model applied to perfect diffusers is the Lambert model. This model proposed in 1760 by Lambert [Lam82], is widely used in computer graphics. The Lambertian surface for reflection is a surface that appears uniformly bright from all directions of view and reflects the entire incident light. Among the empirical specular reflection models which are not perfect for a smooth surface, we find the model proposed by Phong [Pho75] improved by Blinn [Bli77, Bli82], the Lewis model [Lew94] or that proposed by Lafortune et al. [LFTG97]. These models are based on a limited number of intuitive parameters and the BRDF is easy to implement. Like the

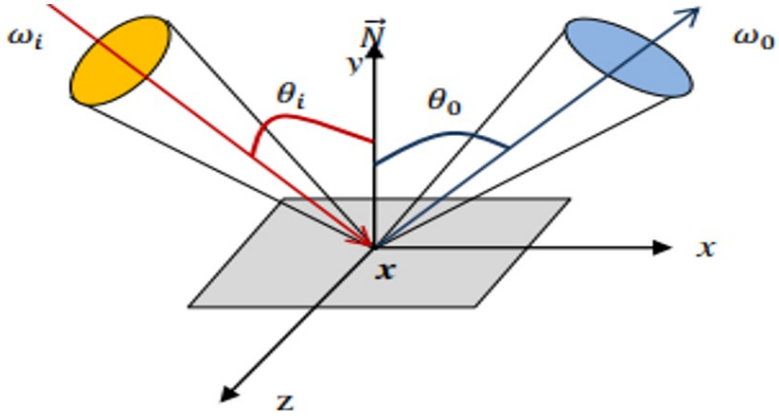

Figure. 1.12. Illustration of the bidirectional reflectance distribution function (BRDF) [Tak09].

Ward model [War92], the Oren and Nayar model [ON95] offers a new approach suitable for both smooth and rough surfaces, which are neither Lambertian nor reflective, like clay or sand. The surfaces are modeled by small diffusing elements that follow Lambert's law. Torrance and Sparrow [TS67] and Cook-Torrance [CT82] are more complex to implement because they rely on the physical and geometric properties of materials to model specular reflection on a rough surface.

1.5.2 Rendering equation

In graphics computing, the distribution of energy in a scene was first formulated mathematically by Kajiya [Kaj86] in 1986, as an equation called the rendering equation. The latter makes it possible to calculate the outgoing luminance of a point on the surface based on the luminance emitted by the surface, the incident luminance, and the properties of the materials of the surface.

$$L(x \to \omega) = L_e(x \to \omega) + L_0(x \to \omega) \tag{1.14}$$

Where:

- $\boldsymbol{L_e}$: the luminance emitted by the surface in the direction ω,
- $\boldsymbol{L_r}$: luminance reflected by the surface.

According to the definition of BDRF (equation 1.13):

$$f_r(x, \omega_i, \omega_0) = \frac{radiance}{irradiance} = \frac{dL_r(x \to \omega_0)}{dE(x \leftarrow \omega_i)} = \frac{dL_r(x \to \omega_0)}{L_i(x \leftarrow \omega_i) cos(N_x, \omega_i) d\omega_i}$$

Therefore:

$$L_r(x \to \omega_0) = \int_{\Omega_x} f_r(x, \overrightarrow{\omega_i}, \overrightarrow{\omega_0}) L_i(x \leftarrow \omega_i) cos\theta_i d\omega_i \tag{1.15}$$

Alternative formulation:

Let us consider the surfaces of the scene which contribute to the incident luminance at point x.

- The solid angle deduced from the surfaces:

$$d\overrightarrow{\omega_l} = cos(\overrightarrow{n_{x'}}, \overrightarrow{\omega_l}) \frac{dA_{x'}}{|| \, x - x' \, ||^2} \tag{1.16}$$

- By replacing the equation 1.16 in equation 1.15, we got :

$$dL_0(x \to \omega_0) = \int_A f_r(x, \overrightarrow{\omega_i}, \overrightarrow{\omega_0}) L_i(x \leftarrow \omega_i) V(x, x') \frac{cos(\overrightarrow{n_x}, \overrightarrow{\omega_l}) cos(\overrightarrow{n_{x'}}, \overrightarrow{\omega_l})}{|| \, x - x' \, ||^2} dA_{x'} \tag{1.17}$$

Such as:

— $V(x, x')$: visibility (1 if visible, 0 if not);

— $G(x, x') = \frac{cos(\overrightarrow{n_x}, \overrightarrow{\omega_l}) cos(\overrightarrow{n_{x'}}, \overrightarrow{\omega_l})}{||x - x'||^2}$ (Geometric term G).

- The equation then becomes:

$$dL_0(x \to \omega_0) = \int_A f_r(x, x \to x', \omega_0) L_i(x', x' \to x) V(x, x') G(x, x') dA_{x'} \tag{1.18}$$

So our rendering equation is:

$$L(x, \overrightarrow{\omega_0}) = L_e(x, \overrightarrow{\omega_0}) + \int_{\Omega_x} f_r(x, \overrightarrow{\omega_i}, \overrightarrow{\omega_0}) L_i(x, \overrightarrow{\omega_i}) cos(\overrightarrow{n_x}, \overrightarrow{\omega_l}) d\omega_i \tag{1.19}$$

The rendering equation is physically based and describes the balance of energy in a scene. Although it is a good model for lighting transport, solving the equation is analytically difficult. Photorealistic rendering aims to find effective ways to approximate and solve this equation, which inherently depends on itself because the light coming out at a given moment could end up being the light coming in for another point. This dependence makes the calculation particularly difficult. It is practically impossible to find an analytical solution to the rendering equation, especially for complex geometric scenes [Zer18]. But we can use Monte Carlo integration [Kaj86] to solve the rendering equation, which can be a robust solution because it is based on random sampling.

1.5.3 Integration by the Monte Carlo method

The numerical resolution of an integral is carried out by a summation of the form:

$$\chi = \int_b^a f(x)d_x \approx \sum_{i=1}^{N} \propto_i f(x_i) \tag{1.20}$$

Where the $(\propto_i, i = 1..N)$ are weights, and the $(x_i, i = 1..N)$ are determined discretization weights. When these weights are distributed evenly.

$$\propto_i = (b - a)/N \tag{1.21}$$

For a probabilistic integration, the samples $(x_i, i = 1..N)$ are chosen randomly according to a probability density $p(x)$. The estimate of the integral has a similar form:

$$\chi \approx \chi_N = \sum_{i=1}^{N} \frac{1}{Np(x_i)} f(x_i) \tag{1.22}$$

This estimate converges in $N^{-1/2}$ whatever the dimension of the problem. This makes it a very well suited method for global illumination phenomena since these phenomena integrate light paths which can contain many reflections, thus increasing the dimension of the problem [PB07].

1.6 Global illumination algorithms

Global illumination is a group of algorithms used in 3D computer graphics that are intended to add more realistic lighting to 3D scenes. The role of these algorithms is to simulate the propagation and interaction of light in large-scale geometry scenes for image synthesis. Global illumination involves the processes of light emission, reflection, redistribution, shadowing, and, ultimately, absorption in an environment, but it finds limited to use on interactive applications due to the computational cost for calculating these interactions of light and visibility in an environment.

In this section, we discuss the most popular global illumination algorithms such as radiosity, path tracing, and photon mapping, and how these algorithms evaluate shadows.

1.6.1 Radiosity

Solving the problem of global illumination is often simplified by solving only radiosity, which represents the integration of radiance on an encompassing sphere. Consequently, a

radiosity calculation generally reduces the illumination model to the ideal diffuse model and neglects the complex behaviors represented by the BRDFs [PY06]. The radiosity method consists of calculating the light transfer between the patches. The system could be solved numerically, using Gaussian elimination, Jacobi, and Gauss-Siedel iteration [CG85], or the use of Southwell relaxation [GCS94]. Radiosity is an interesting solution for a convincing overall rendering but often proves to be ineffective in representing details on a small scale. Also, the ideal diffuse model deprives the radiosity of specular effects, as common as reflection and refraction [PY06].

1.6.2 Path tracing

Path Tracing [Kaj86] is a rendering algorithm similar to ray tracing in which rays are projected from a virtual camera and traced through a simulated scene. Path tracing uses random sampling to progressively calculate a final image. The random sampling process makes it possible to make certain complex phenomena that are not dealt with in regular raytracing, but it generally takes more time to produce a high-quality path traced image. Path tracing works by taking a pixel from the output image frame and sending a virtual ray from the camera, through that pixel, and out into the scene. The ray bounces off objects until it runs out of energy, hits a light source, or reaches a predefined number of reflections (bounces). When this happens, the ray is traced back to the camera through all the objects it has touched and the final "color" of this pixel sample is determined.

1.6.3 Photon mapping

The photon map algorithm [JC95] was developed in 1993{1994} and the first papers on the method were published in 1995. It is a versatile algorithm capable of simulating global illumination including caustics, diffuse interreflections, and participating media in complex scenes. It provides the same flexibility as general Monte Carlo ray tracing methods using only a fraction of the computation time [JC00]. Throwing photons involves sending photons from light sources in random directions. The photons are sent infinite numbers in the scene and the power of the source is divided equally for each of them. When a photon meets a face, it is stored in a data structure called photon map then is randomly restarted in the scene having lost part of its energy; that which the face has absorbed. There are many heuristics for the death of photons: when their remaining energy is below a certain threshold when a maximum number of bounces is reached, or even by drawing a random number to know if it continues to live. These treatments are independent of the point of view and the creation of an image can then be carried out either using a ray-tracing or by

Zbuffer-based methods [Zer18].

1.6.4 Shadows in global illumination algorithms

Interactive global illumination remains an elusive goal, despite an abundance of heuristic solutions in recent years. The cost of visibility is the main expense in most methods. Real-time applications typically use shadow mapping [Wil78] and shadow volume [Cro77] algorithms, while offline rendering applications use ray tracing for shadow evaluation [Whi79]. Shadow mapping is highly flexible in terms of scene geometry, but it has some important issues:

- Perspective aliasing, which shows as jaggy shadows, due to the poor use of the shadow map area or its insufficient resolution,
- Shadow acne (self-shadowing artifacts) and Peter Panning (disconnected shadows),
- Soft shadows (lack of penumbras),
- Lack of support for semitransparent occluders.

Many techniques have been developed to solve these issues [ESAW11, EAS+13]. Generally, making an efficient implementation of shadow mapping complicated and achieving a good result requires a combination of some of these techniques and manual fine-tuning by the scene designer, and certainly, different solutions are required for different scenes.

Ray tracing [Whi79] is a flexible rendering paradigm that can compute accurate shadows with a simple algorithm and can handle complex lighting (area lights, semi-transparent occluders) in a scalable way. However, due to limited hardware resources as well as the complexity of implementing the underlying algorithms required for real-time ray tracing, such as the fast construction and maintenance of spatial data structures, it was difficult to achieve sufficient ray tracing performance for real-time applications. There was also no explicit ray tracing support in popular graphics APIs used for real-time applications [BWB19].

It is now straightforward to exploit ray tracing using DirectX and Vulkan APIs with the introduction of NVIDIA RTX and DirectX Raytracing (DXR). The recent NVIDIA Turing graphics architecture greatly improve ray tracing performance, since it provides hardware support for DXR using the dedicated RT Cores. These new features combine greatly with emerging hybrid rendering methods [HHM18] that use rasterization to resolve primary-ray visibility and ray tracing to compute shadows, reflections, and other illuminations effects [[BWB19]].

However, we have to use our resources wisely when rendering high-quality shadows using ray tracing, even with the new powerful hardware support. A naive algorithm might

easily cast too many rays to sample shadows from multiple light sources and/or area light sources, leading to low frame rates (see Figure 1.13) [BWB19].

Figure. 1.13. Naive ray-traced shadows using 256 samples per pixel running at 200 ms per frame. Times measured using a GeForce RTX 2080 Ti GPU. Top: visibility buffer. Bottom: final image [BWB19].

1.6.5 Summary

In the following (table 1.1) we present a summary of the global illumination algorithms, mentioning their advantages and their limits as well as the shadows in each of them.

	Advantages	Disadvantages	Shadow
Radiosity	▪ Very realistic images from lighting are created, ▪ Once the energy balance has been calculated, the display of a scene is very fast (Speed of Z-Buffer or ray tracing), ▪ You don't have to recalculate the balance if you move the observer inside the scene. As long as the scene is not changed, animation calculations are extremely fast.	▪ Radiosity only takes into account "diffused" light. ▪ If an object moves, we must recalculate everything. ▪ Calculating form factors and solving the system of linear equations, require large quantities of calculation.	▪ Shadows are generated implicitly, ▪ Shadows not present for objects that are too thin, ▪ Impossible to get shadows in real-time.
Path tracing	▪ The images it produces can be almost indistinguishable from reality, ▪ Path tracing allows you to incorporate global illumination calculations into the ray tracing, ▪ It has the advantage of naturally capturing the way light moves through a scene.	▪ Images require very long calculation times, to obtain very realistic images, ▪ The computed image is never finished, the computation being of convergent type.	▪ Shadows are generated implicitly, ▪ Impossible to get shadows in real-time.
Photon mapping	▪ Does not require Lambert surfaces (Lambertian or perfectly diffuse), ▪ Can statistically model the behavior depends on the wavelength of light and materials, ▪ Can model specular interreflection effects (e.g. caustic).	▪ We only store a photon after they have undergone refraction or simple reflection, ▪ Images require a long calculation time to get realistic images.	▪ Shadows are generated implicitly, ▪ Impossible to get shadows in real-time.

Table. 1.1. Synthesis on global illumination algorithms.

1.7 Conclusion

In this first chapter, we have highlighted the fundamental concepts for understanding the illumination complexity, which constitutes the first start to the generation of photo-realistic images with a high quality of shadows. To do this, we studied the two main classes of 3D scene illumination, namely, local illumination and global illumination. In the latter, we discussed the different basic notions of radiometry, light-surface interaction and finally the rendering equation, as well as an overview of global illumination algorithms and how we create shadows in these algorithms which are a native element of Whitted-style ray tracing require a long calculation time to get realistic images.

This study helped us to understand the phenomena of lighting a scene, generate shadows using global illumination algorithms, the advantages and the limits of the existing techniques, and will guide us thereafter in the choice of the algorithm to render real-time shadows with high quality.

Chapter2

Real-time shadows

2.1 Introduction

Shadows are a crucial topic in 3D computer graphics. They have a major influence on the realism and on the quality of rendered pictures. Moreover, without them, the relative depth of the objects in a scene can often not be understood. However, this is still an open research subject, as the complexity of simulating physically realistic shadows is a complex task for real-time applications.

In this chapter, we define the concept of shadows and in particular the concept of soft shadows in real-time used during the rendering process. We first present basic concepts of shadows such as definition and types of shadows; hard and soft shadows, then we expose the existing methods which use shadows in real-time in the field of local illumination. Finally, we end this chapter with a detailed study of the soft shadow concept in real-time rendering which is the objective of this chapter.

2.2 General information on shadows

In previous years, computer graphics was only known as flat, two-dimensional graphical representation. After three dimensional principles were introduced into computer graphics, objects were often rendered without shadows and do not appear to be anchored in the environment. Shadows convey a large amount of information because they provide what is essentially a second view of an object(see Figure 2.1).

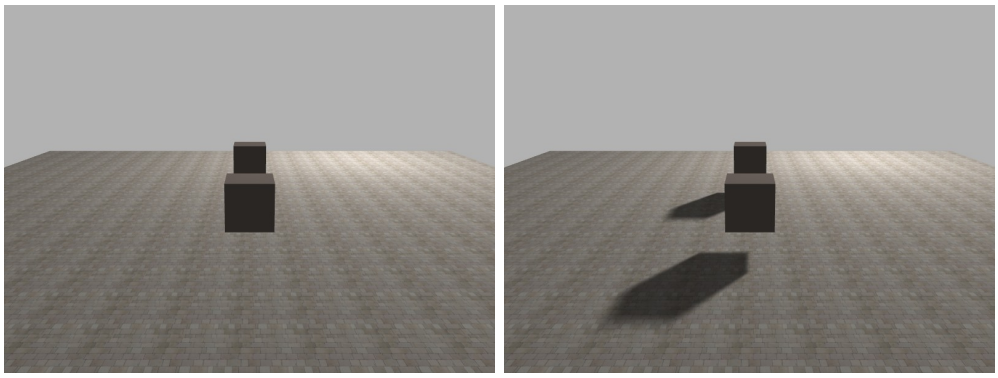

Figure. 2.1. Comparing two images rendered with and without shadows to see the shadow effect. Left image: image rendered without shadows. Right image: image rendered with soft shadows using an area light source.

Until the early 90's, shadows were often neglected in real-time applications. However, they are essential to the understanding of a scene. A user will immediately notice if a scene has been rendered without shadows (see Figure 2.1). Shadows are very important for the intuitive perception of an image and are crucial for the comprehension of its composition. A

lot of research was done to confirm this affirmation and it was proven that shadows provide the user with useful information to understand more easily the scene. Compared to light calculations, where we can have some (often quite a lot) deviation from reality, for shadows it is more easily detectable if something is wrong.

2.2.1 Definition of shadow

Shadows are a result of the interaction of three components: a light source, an occluder and a receiver; they are a result of the absence of light due to occlusion; when a light source's light rays do not hit an object because it gets occluded by some other object the object is in shadow.

Consider a light source L illuminating a scene. If a point P does not receive any light directly from the light source, it considered to be in the umbra. P is in the penumbra, if it can see a part of the light source. The union of the umbra and the penumbra is the shadow, the region of space for which at least one point of the light source is occluded (see Figure 2.2). Occluders are the objects that hide a point from the light source [HLHS03].

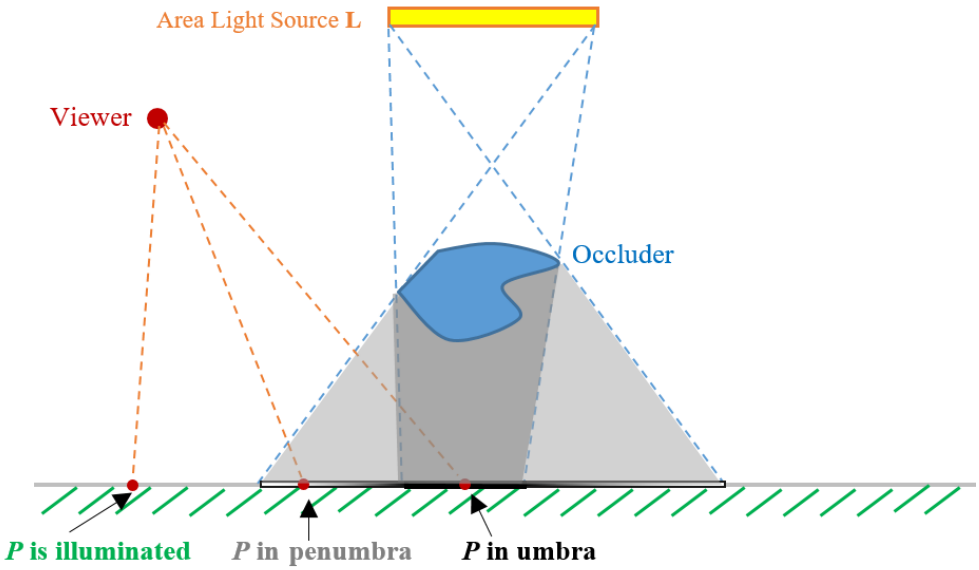

Figure. 2.2. The objects involved in the shadowing process.

2.2.2 Types of shadows

Shadows add a great deal of realism to a lighted scene and make it easier for a viewer to observe spatial relationships between objects. They give a greater sense of depth to our scene and objects (see Figure 2.1). We distinguish between three types of shadows:

- **Attached shadows:** appearing when the normal of the receiver is facing away from the light source [HLHS03];
- **Cast shadows:** appearing when a shadow falls on an object whose normal is facing toward the light source (see Figure 2.3).
  - **Self-shadows:** are a specific case of cast shadows that appear when the shadow of an object is projected onto itself, i.e. the occluder and the receiver are the same (see Figure 2.3) [HLHS03].
- **Penumbra:** is located at the periphery of shadows, when a point is illuminated only by a part of the light source (see Figure 2.3). It constitutes the transition between a lighted area and a shadow.

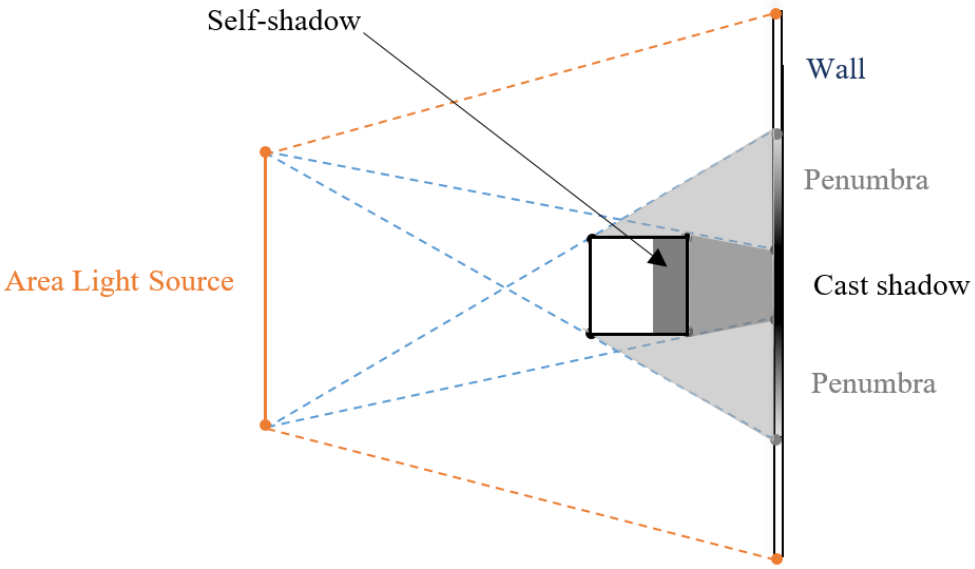

Figure. 2.3. Different shadows present in a scene.

2.2.3 Importance of shadow effects

Shadows has always been, and is still a hot topic in computer graphics, it have been studied for a long time, the first written characterizations are found in Leonardo da Vinci's Codex Urbinas on 1490. In fact, this is not surprising since the shadows cast are essential for the understanding of an image. Indeed, they provide several pieces of information [Zer18]:

- Shadows help to understand relative object position and size in a scene [Wan92, MKK98, KMK94, KMK97]. For example, without a cast shadow, we are not able to determine the position of an object in space (see Figure 2.4).
- Shadows can also help us understanding the geometry of a complex receiver [MKK98] (see Figure 2.5).

- Finally, shadows provide useful visual cues that help in understanding the geometry of a complex occlude [MKK98] (see Figure 2.6).

Figure. 2.4. Shadows provide information about the relative positions of objects [HLHS03].

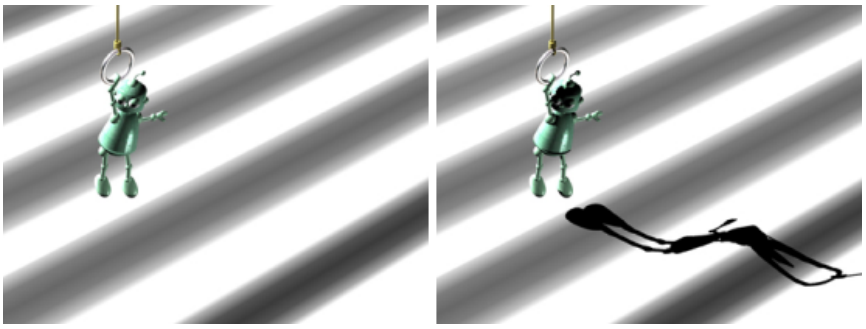

Figure. 2.5. Shadows provide information about the geometry of the receiver [HLHS03].

Figure. 2.6. Shadows provide information about the geometry of the occluder. Left robot holds nothing in his left hand. Robot in the middle holds a ring. Right robot holds a teapot [HLHS03].

2.2.4 Hard shadows vs. soft shadows

Given a point light source which emits light from single point in space, there's only one ray of light that can potentially hit the surface point, which is exactly the ray originating from the light source and directed towards the surface point.

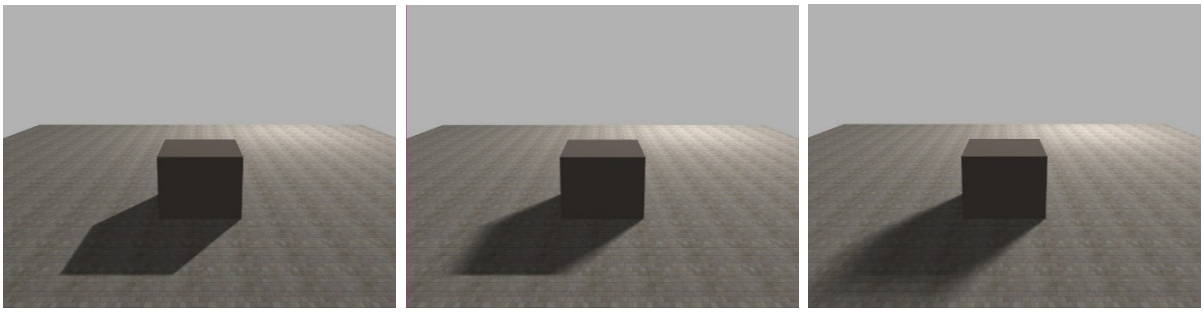

Figure. 2.7. Shadows from different types of light sources. Left image: shadow rendered from a point light source. Image in the middle: shadow rendered from a linear light source with 6 samples. Right image: shadow rendered from an area light source with 36 samples.

The corresponding intensity level can therefore only be 100% (fully lit) or 0% (fully shadowed), depending whether the ray is blocked or not. The resulting shadow will exhibit sharp boundaries where the intensity level changes as can be seen in the example scene shown in Figure 2.7 (left). We will refer to this type of shadow as hard shadow or umbra. Sharp shadow boundaries are common to all non-extended light sources, such as directional, point, and spot lights. However, point light sources do not exist in real life and hard shadows give a rather unrealistic feeling to images. As we see even the sun, probably can be the best source of light to create shadows in our daily life, has a significant angular extent and does not create hard shadows.

Using a linear light source, a number of light rays may hit a specific surface point, all originating from sample points on the line segment associated with the light source. Contribution therefore varies between 0% and 100%, resulting in a smooth shadow transition as shown in Figure 2.7 (middle). Linear light sources have a specific characteristic that can be observed at the top of the cast shadow in Figure 2.7 (middle): Smooth shadow regions are only generated for blocker edges that are not parallel to light source's line segment. In the parallel case a hard shadow transition will be visible, since all rays are blocked when crossing the given edge. We refer to the region in which the transition from lit to shadowed takes place as the penumbra. A shadow consisting of penumbra and umbra regions is called soft shadow.

For a linear light source we can easily visualize the geometric relationship between blocker, receiver, and light source and construct the shadow boundaries. Figure 2.8 shows such a 2D arrangement. On the left side, a point light source is used. As we can see,

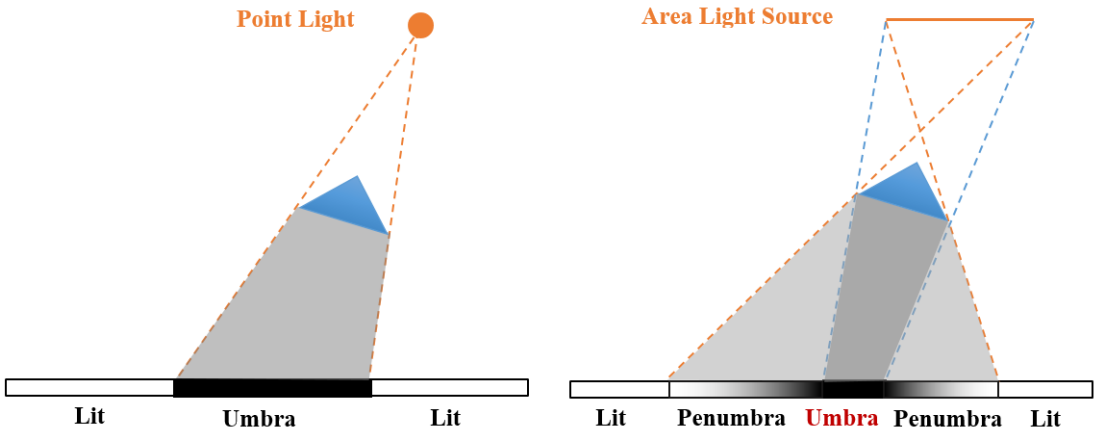

Figure. 2.8. Geometry of hard (left) and soft (right) shadows.

the contribution of the light source changes at two points on the receiver, resulting in a *lit-umbra-lit* transition. Using an extended light source, e.g. a linear one as shown on the right side of Figure 2.8, we can determine the relevant points at which illumination changes by drawing lines between the extremal points of the blocker and the end points of the linear light. This gives us the regions on the receiver in which the light source is fully visible, partially visible/occluded, and fully occluded, resulting in a *lit-penumbra-umbra-penumbra-lit* transition.

Overall smooth shadows can be obtained when using an area light source (see Figure 2.7 on the right). As explained before, an area light is of finite size where every point on the surface can emit rays of light. As in the linear case, each point on a surface may be hit by all, some, or none of the light rays, resulting in a soft shadow if blocking objects are in between.

Figure. 2.9. Illustration of hard (left) and soft (right) shadows [HLHS03].

2.3 Shadows in real-time

In the past 20 years, computer graphics researchers have made significant progress in real-time rendering. The search for new methods to make video games, virtual environments and interactive graphics applications more realistic, has seen a recent boom in work on the generation of shadows in real time [Ber86, WPF90, Kil01, HLHS03, AAM03, AW04, CD04, SSMW09].

Shadow computing in an infographic application is a very costly task in terms of computing time. It depends on the rendering method, the shadow algorithm used, the material available, the quality of the shadow, and the complexity of the scene. Generating a shaded image can be a very long task, it can take several milliseconds up to several minutes or even hours. Today's hardware is capable of rendering various shadow generation techniques in real-time [SSMW09]. Real-time shadow is a real challenge in computer graphics, and there is no general solution that can be applied to any scene [SSMW09].

Ray tracing and radiosity are methods for generating shadows, but they are not adapted to real-time, because they are very costly in computation time [Zer11]. But the methods for generating shadows such as the Shadow mapping (image-based) [Wil78] and Shadow volume (object-based) [Cro77] are adapted to real time.

2.3.1 Object space and image space

The object space is the coordinate system (or mathematical description space) associated with the geometric model of the object in which each point is represented by a triplet of coordinates. This coordinate corresponds to the virtual world in which the object is located. The objects are described with high precision there. In contrast, image space is associated with the projection of objects on the image plane. This space only corresponds to a partial representation of the object according to the definition of the screen [Zer18].

2.3.2 Shadows volumes

The Shadow Volume algorithm, introduced by Crow in 1977 [Cro77], was implemented with hardware acceleration in 1985 [FGH+85] but did not see an extended use until a version was suggested that could be hardware accelerated on consumer grade graphics hardware with the z-pass algorithm [Hei89]. The algorithm consists in finding the silhouette of occluders along the light direction, then extruding this silhouette along the light direction, thus forming a shadow volume. Objects that are inside the shadow volume are in shadow, and objects that are outside are illuminated.

The shadow volume is calculated as follows:

- The first step is to find the silhouette of the occluders, seen from the light. The easiest way is to identify the edges shared by one polygon facing the light and another in the opposite direction.
- We then construct the shadow polygons by extruding each silhouette edges along the direction from the light, thus forming a long shadow polygon (quadrilateral). All of these extrusions define a volume, and knowing whether a point is in the shadow is equivalent to knowing whether that point is in the volume.
- For each pixel in the image, we count the number of shadow polygons crossed from the image plane to the object to be rendered. The shadow polygons facing the camera increment the counter, the others decrement it. If the counter is positive at the very end, the point is in the shadow (see Figure 2.10).

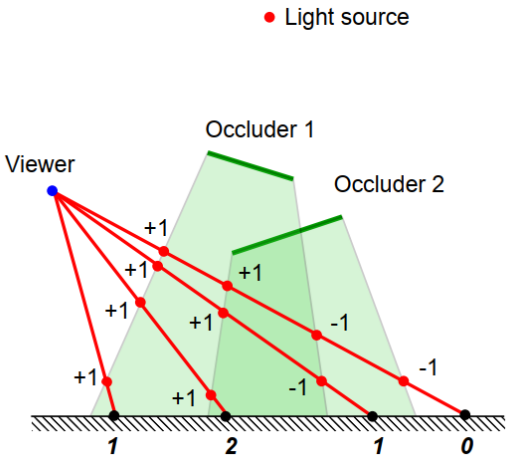

Figure. 2.10. Shadow volume [HLHS03]

2.3.3 Shadow Mapping

In computer graphics, shadow maps [Wil78] are a significant step in the evolution of shadow generation. They are image-based, therefore shadow maps don't need more calculation comparing to shadow volumes which are expensive in the case of rendering. They are fast enough in rendering then they are employed in different fields of computer graphics such as computer games, animations [IKSB13] and motion vision [KS12].

Identifying the parts of the scene that are hidden from the light source is the basic operation for computing shadows. Intrinsically, it is equivalent to visible surface determination, from the point-of-view of the light source. The first method to compute shadows

starts by computing a view of the scene, from the point-of-view of the light source. We store the z values of this image. This Z-buffer is the shadow map (see Figure 2.11).

The shadow map is then used to render the scene (from the normal point-of-view) in a two pass rendering process [Zer18]:

- First, the scene is rendered from the point of view of the light source. The values contained in the z-buffer are then saved; they form the shadow depth buffer also called the shadow map (see Figure 2.11).
- Next, the scene is rendered from the point of view of the camera, using the shadow map to determine which parts are lit or which are the shadows. To determine if a point is in the shadow, we project it into the shadow map of light and compare the depth resulting from this projection with the depth contained in the shadow map. If the depth of the shadow map is smaller, the point is in the shadow; otherwise it is lit.

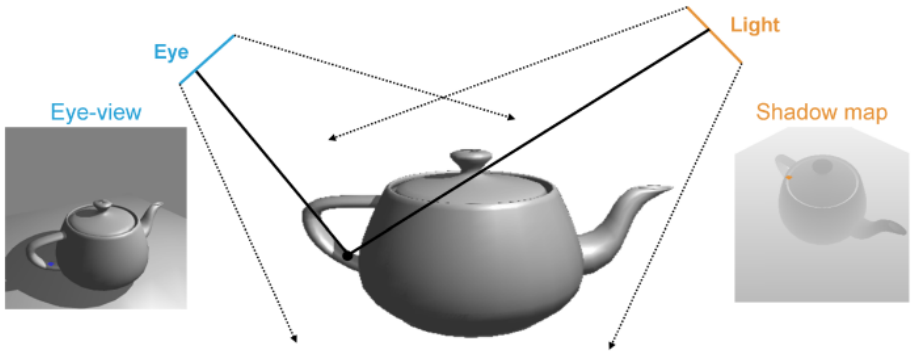

Figure. 2.11. The shadow mapping algorithm: The depth values as seen from the light source are stored in a shadow map, and are then used in a second pass to generate shadows on the objects [SMSW12].

The shadow map can be filtered to hide artifacts, simulating the effect of an area light-source, this is one of the reasons for the popularity. Filtering during lookup usually requires a large number of samples [RSC87, Fer05], whereas pre-filtering requires additional attributes, which increases storage and bandwidth requirements [DL06, AMS+08]. It can enable the use of relatively low resolution shadow maps while producing visually pleasing results. And The main advantage of shadow maps is that the complexity does not depend on the geometrical complexity of the scene but just on the image size and the number of light sources. Another important advantage is that it can be used with any renderable geometry.

However, sharp shadows cannot be produced without resorting to a very high-resolution shadow map, leaving shadows artificially blurred or with obvious discretization artifacts. Several algorithms attempt to improve precision where it is most needed, without abhorrent

memory requirements, either by warping or by partitioning the shadow map. Warping techniques [WSP04, LGQ+08] can yield impressive results but suffer from special cases where they degenerate to ordinary shadow maps. Partitioning approaches can produce high quality sharp shadows quickly for some scenes [Arv04, LSO07, LSL11], but have difficulties if the scenes are very open with widely distributed geometry, leading to aliasing reappearing, unpredictable run-time performance, or escalating memory requirements.

2.3.3.1 Percentage Closer Filtering (PCF)

To reduce aliasing reeves et al. proposed a technique [RSC87], a technique which they called Percentage-Closer Filtering (PCF). This algorithm attempted to produce monotonic data in the outline of a shadow by filtering the depth map with interpolation of binary data. This method involved binary values. It would cause the shadow edges to be sharp, and soft anti-aliasing would not be possible if applied to depth maps.

The shadow map in the shadow mapping algorithm is evaluated in a second pass from the camera point of view using the data from the first pass for shadow calculation to calculate whether fragments are in shadow according to their occlusion. Standard shadow mapping features aliasing and sharp outlines. In contrast to standard shadow mapping, PCF follows these steps in reverse, which means that the data must be filtered first to produce new binary data. The next step is to produce the filtered data average. The new data can range between 0 and 1 without the requirement that the data be exactly 0 or 1. Figure 2.13 illustrates this as well. Some games and game engines use PCF directly to implement soft shadows [KSA+15].

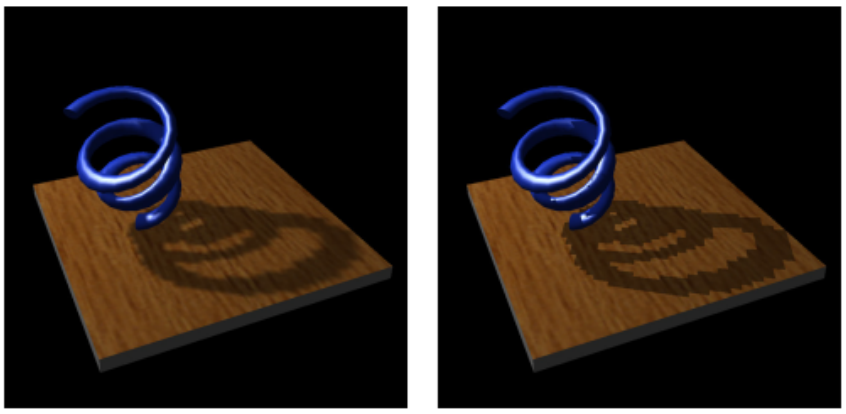

Figure. 2.12. Effects of percentage closer-filtering. Left image: image filtred using PCF. Right image: image rendered without filter [Kil02]

The difference between ordinary texture map filtering and percentage closer filtering is shown schematically in Figure 2.13. In this example, the distance from the light source to

the surface to be shadowed is $z = 54$. The region in the depth map that it maps onto is a square measuring 3 pixels by 3 pixels. Ordinary filtering would filter the depth map values to get 73 and then compare that to 54 to end up with a value of 1 meaning that 100% of the surface was in shadow. Percentage closer-filtering (PCF) compares each depth map value to 54 and then filters the array of binary values to arrive that 44% of the surface is in shadow.

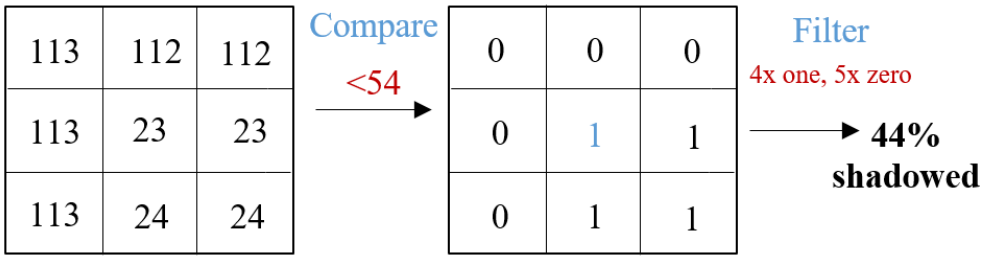

Figure. 2.13. Percentage Closer Filtering (PCF).

We can thus generate a form of soft shadows instead of modulating the color of the fragment by 0 or 1, we thus have a whole range of values making it possible to avoid the edges of shadows that are too sharp [Zer18].

2.3.3.2 Cascaded Shadow Maps (CSMs)

Cascaded shadow maps (Cascaded Shadow Maps (CSM) [Eng06], Parallel-Split Shadow Maps (PSSM) [ZSXL06], or even z-partitioning [LTY+06]) is a more effective solution against problems of perspective aliasing, consists of partitioning the view frustum into n sub-frustums, each associated with a dedicated shadow map. The partitions close to the user receive a denser sampling since the surface it covers is smaller than for distant partitions (see Figure 2.14), this provided of course that the shadow map frustums are calibrated for the associated scene portions.

Assigning a separate shadow map for the different areas of the view frustum redistributes the error on the screen and makes the artifacts less noticeable. The effectiveness of cascade, however, depends on the distribution of partitions along the view frustum, and there is no fixed partitioning strategy that works for all viewpoints (whether it be uniform, logarithmic, or a combination of both , and also arbitrarily(see Figure 2.15)). In the field of video games, the task has often been delegated to artists [LSL11] who had to manually adjust the parameters for different points of view. Even in this case, not all points of view can be taken into account, and partitioning always ends up becoming suboptimal. Lauritzen et al. [LSL11] therefore propose to automate the partitioning process. They use the depth buffer

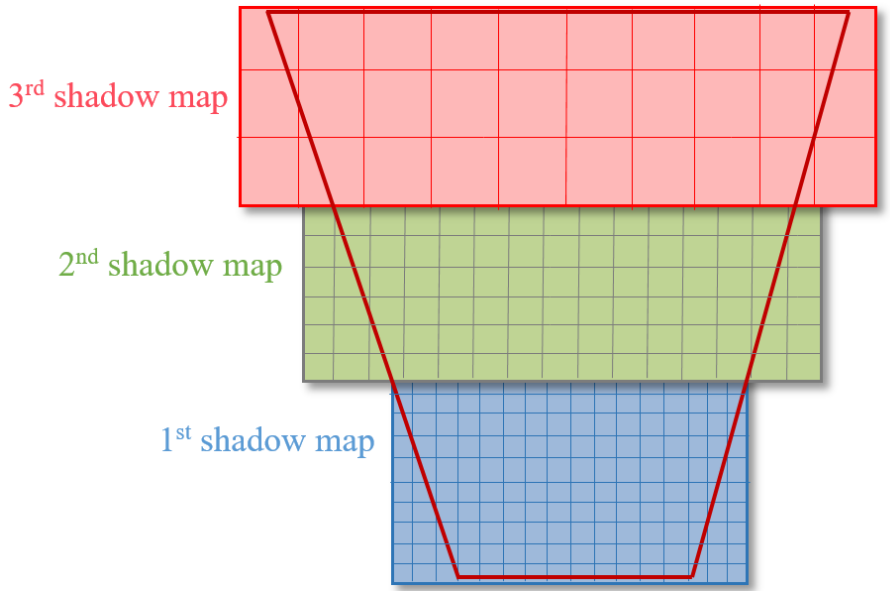

Figure. 2.14. Cascaded Shadow Maps (CSMs).

to get the distribution of samples along the view frustum, allowing them to efficiently place shadow maps [Dev19].

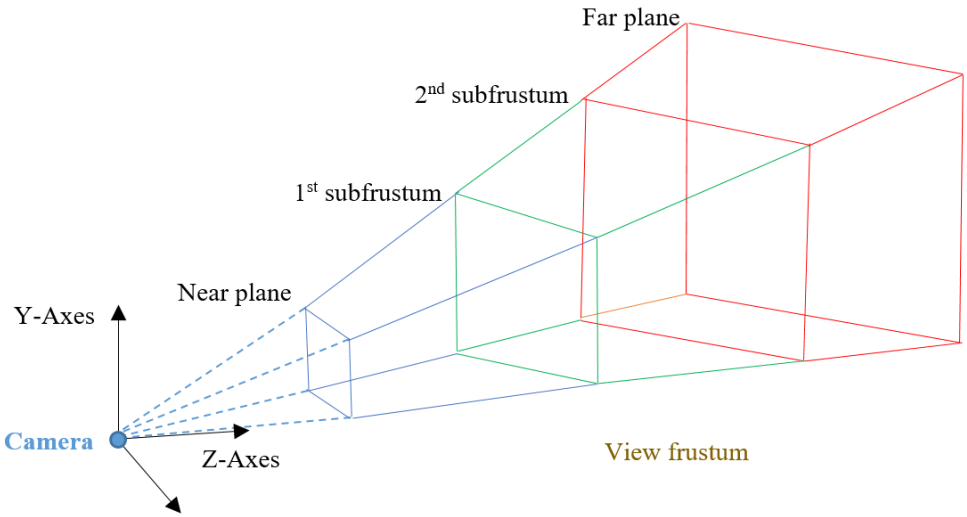

Figure. 2.15. Partitioning the view frustum into subfrustum.

2.3.4 Summary

In the following (Table 2.1) we present a summary of the volume shadows and shadows mapping, mentioning their advantages and their limits as well as their common categories.

	Advantages	Disadvantages
Volume Shadows	▪ Processes omnidirectional sources, ▪ High precision of shadows, ▪ Generation and interactive rendering for dynamic scenes, ▪ Treats cases of self shading, ▪ Robust if well programmed.	▪ Calculation time depends on the complexity of the blackout (calculation of the silhouette, extrusion), ▪ Requires the calculation of the silhouette of the blockers, ▪ Creation of large polygons (extrusion), ▪ Two renderings of the scene, plus rendering of the volume.
Sadows Mapping	▪ It does not require any geometrical calculation, since shadow mapping is an image space technique, working automatically with objects created or altered on the GPU, ▪ It does not need stencil buffer. Just one single texture is required to hold shadowing information for each light, ▪ It avoids the high fill requirement of shadow volumes.	▪ Bad quality in outline of shadow or aliasing, especially when light source is far from occlude, ▪ For each light, the scene must be rendered once, in order to generate the shadow map for a spotlight, and more times for an omnidirectional point light, ▪ At least 2 passes are required for rendering.

Table. 2.1. Synthesis on volume shadows and shadows mapping.

2.4 Important issues in computing soft shadows

In this section, we will discuss several popular issues of soft shadow rendering which have been solved and improved by the researchers to keep pace with their researches.

2.4.1 Composition of multiple shadows

Care must be taken to allow for more complex situations while the creation of a shadow is easily described for a (light source, occluder, receiver) triple. If we know how to deal with a single source, shadows produced by multiple light sources are relatively easy to obtain. We simply sum the contribution of each light for each wavelength or color band due to the linear nature of light transfer. For point light sources, shadows due to different occluders can be easily combined since the shadow area (where the light source is invisible) is the union of all individual shadows [HLHS03].

Combining the shadows of several occluders is more complicated with an area light

source. Recall that the lighting contribution of the light source on the receiver involves a partial visibility function, a major problem is that the partial visibility function of the of the occluders considered together cannot be given by a simple combination of the partial visibility functions of separate occluders [HLHS03]. For instance there may be points in the scene where the light source is not occluded by any object taken separately, but is totally occluded by the set of objects taken together. The correlation between the partial visibility functions of different occluders cannot be predicted easily, but can sometimes be approximated or bounded [ADMAM03, SS98]. As a consequence, the union of the shadows of the objects can be smaller than the shadow of the union of the objects (see Figure 2.16). This effect is quite real, but is not very visible on typical scenes, especially if the objects casting shadows are animated [HLHS03].

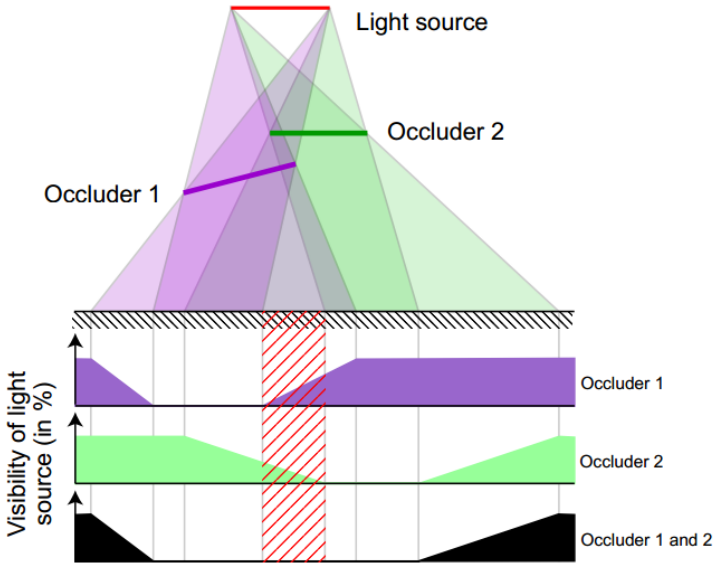

Figure. 2.16. The shadow of two occluders is not a simple combination of the two individual shadows. Note in particular the highlighted central region which lies in complete shadow (umbra) although the light source is never blocked by a single occluder [HLHS03].

2.4.2 Related work of physically based soft shadows

Soft shadows come from spatially extended light sources. In computer graphics, a set of point light source can represent a light source area. Several samples of the surface light source must be considered, in order to simulate a surface light source, a shadow map is created for each point light source. As soon as these shadow maps create, in a single rendering pass, finally we calculate the value of the shadow but before that it is necessary to perform the depth test, thus generating a physically correct soft shadow. Generating soft shadows with multiple shadow maps by light is costly due to the high sampling density that is required to make regions smooth. Artifacts will likely appear If the density is too low, and the human visual system will no longer perceive a soft shadow, but several hard shadows (see Figure 2.17 in the left) [Zer18].

Physically based soft shadow algorithms strive at computing the visibility between a given point and an area light source. Akenine-Möller and Assarsson [AAM03] proposed penumbra-wedges as an extension of the well known shadow volume algorithm [Cro77]. This algorithm has been extended many times, both in terms of accuracy [LAA+05, SEA08], speed [FBP08], and generality to support alpha textured polygons [FBGP09]. Nevertheless, all these variants require the extraction and rasterization of object based primitives making them limited to clean polygonal meshes, and impracticable for complex scenes. Soft Shadow Mapping (SSM) techniques replace the complex geometry of the scene by a simple shadow map (SM) [Wil78], and employ back-projection to integrate the visibility [GBP06, AHL+05]. Continuous reconstruction methods [MM07] have been proposed to overcome the artefacts caused by the discrete nature of the SM. Yang et al. [BFGL09] showed that SSM performance can be significantly improved by exploiting the light and screen space coherence via sophisticated hierarchical algorithms. However, it requires the construction of an expensive multi-scale data structure which is impracticable for high resolution shadow maps. Moreover, discretization artefacts is still an open and challenging issue for such approaches [LGYF11].

Figure. 2.17. Comparing two images rendered with different samples of the area light source. Left image: soft shadow computed using 4 samples. Right image: soft shadow computed using 1024 samples [HLHS03].

2.5 Conclusion

In this second chapter, we have described the important issues when working with shadows, we focused mainly on physically accurate soft shadows in interactive applications. We have presented existing algorithms that produce soft shadows in real-time. Two main categories of approaches have been reviewed, based on shadow maps and shadow volumes. Each one has advantages and disadvantages, and none of them can simultaneously solve all the problems we have mentioned.

This study helped us to understand how to create shadows in real-time using different techniques and that generating shadows using shadow maps algorithms are easy, fast, and widely used, and the efficient use of shadow maps create soft shadows of high quality and higher frames per second compared to geometrical based algorithms. Even the CSMs technique is easy, fast, and creates soft shadows of high quality and higher FPS.

In the next chapter, we introduce our proposed technique which ensures the surface visualization with high quality based on the Monte-Carlo sampling methods and CSMs technique with the exploitation of the bilateral filter carried out on the GPU.

Chapter3

Design, implementation, and results

3.1 Introduction

Shadows are important to a virtual environment. They can not only provide geometry and position information of objects, but also improve the realism of the environment. Since high-quality shadows require lots of computational resources, research on real-time shadow rendering is still significant. Recently games or real-time applications don't have a high-quality shadows system that works flawlessly on both static and dynamic objects unless we use some crazy high resolutions for light maps which greatly increase both the calculation times and the map size. Also, shadows on models just don't work properly since they are all vertex lit. So we needed some sort of dynamic and a very high-quality shadow system (CSM), which is very common nowadays in a real-time rendering application or game.

In this chapter, we are interested in cascaded physically accurate soft shadows calculated by direct lighting techniques. Physically accurate soft shadows in image synthesis applications can be simulated by taking and accumulating multiple samples of the entire area light source, each sample represent a point light source.

We will describe the objectives of the different stages of implementation of our application, for this we will start by describing the technique to be carried out, which will be detailed by describing the sub-objectives. Then, we will expose some obtained results to present the quality and performance of the proposed technique. In order to validate our contribution, a comparative study is established between our technique and others techniques, by highlighting the reduction in the number of samples using the cascaded shadow map technique [Eng06] and the use of multiple PCF filters.

3.2 Description of our cascaded soft shadow maps technique in real-time based on multiple PCF

In this section, we describe our proposition (see Figure 3.1) which is based on shadow maps associated with each point light source for each cascade and which is also based on multiple PCF filters, both cascaded shadow maps technique and PCF are used to accelerate the calculation of soft shadows in real-time rendering.

The proposed technique is inspired by the work carried out by Schwärzler et al. [SMSW12] and consists in reducing the number of area light source samples to the maximum for soft shadows that are physically accurate in real-time. Our contribution consists in subdividing the area light source into a smaller number of samples than that of the Schwärzler's et al. method [SMSW12] using the Zerari technique [ZBNK14], and using

the cascaded shadow maps technique [Eng06] that allows us to reduce more samples, as well as, we use multiple PCF filters to remove the artifacts and soften the resulting soft shadows. We propose to realize and implement the CSMs technique in a new way, which allows providing higher resolution of the depth texture with the high number of point light samples near the viewer and lower resolution with a smaller number of samples far away by splitting the camera view frustum into separate partitions and creating a depth-maps of the point light samples for each partition to increase quality and reduce the calculation time.

We apply the proposed technique as follows: first, we subdivide the area light source into n point light sources for each cascade. Then, we render the scene from the point of view of each point light source to generate the shadow maps (see section 2.3.3). Next, we partition the view frustum into multiple frusta (splits). After that, in another pass of rendering, we evaluate the soft shadow value for each subfrustum (cascade). Then, we use multiple PCF filters to soften the resulting shadows. Finally, we apply a shading model to light up the scene.

The diagram in Figure 3.1 shows a general overview of the proposed technique.

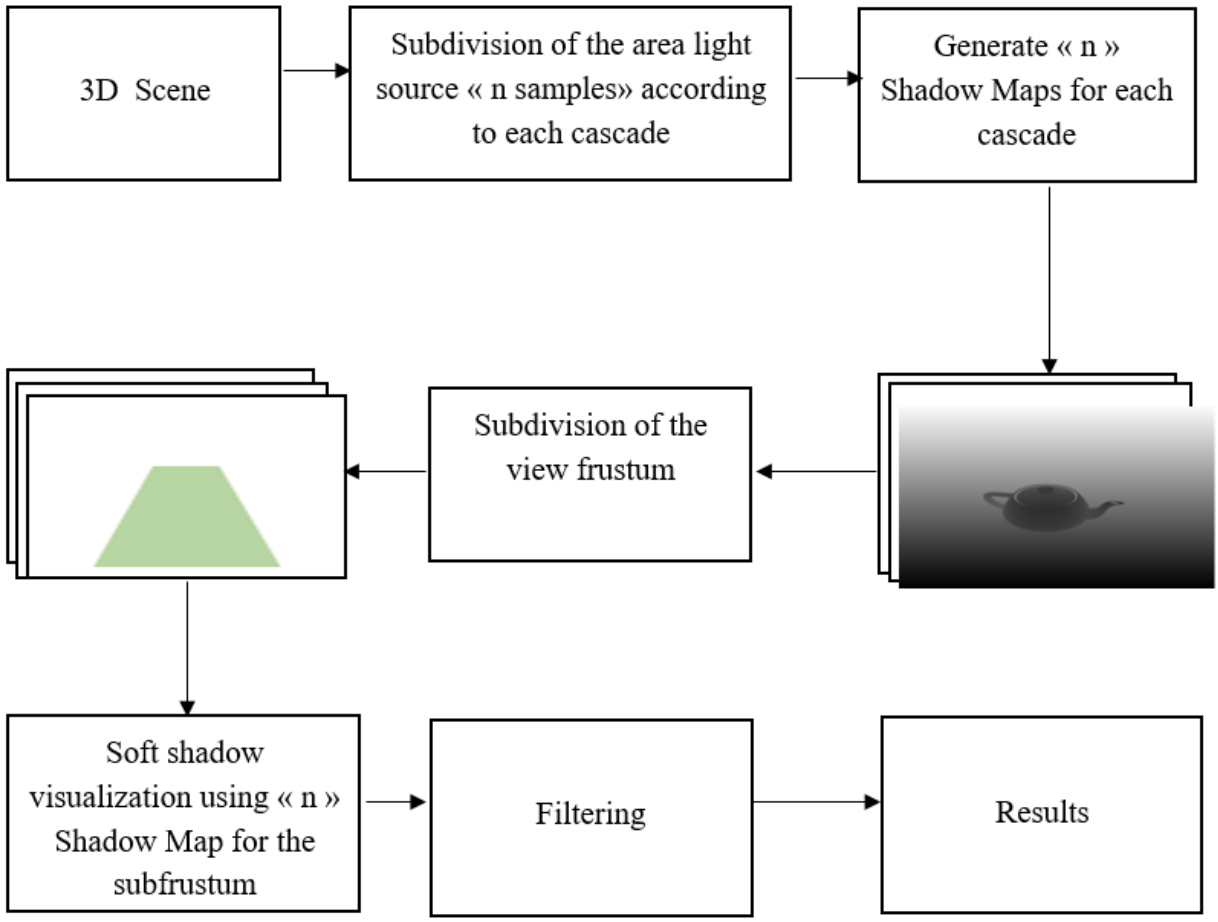

Figure. 3.1. Overview of our cascaded soft shadow maps technique using multiple PCF filters.

3.2.1 Cascaded Soft Shadow Maps algorithm

In this section, we describe the technique that provides a solution of physically accurate soft shadow. We will generate physically correct soft shadows based on an adaptive sampling of the light source and using a cascaded shadow maps algorithm. The general algorithm that we want to develop is as follows: dividing the area light source into several point light sources according to each cascade. Creating a depth map (shadow map) for each light source, and then partitioning the view frustum into n subfrustum. Once these depth maps are created, in a rendering pass, do the depth test for each subfrustum (cascade), and finally, we use adaptive PCF filtering, thus generating a physically correct soft shadow.

Algorithm 1 General technique of our application

Result: Image with high-quality shadows

foreach *Cascade* **do**

  Sampling of the area light source into n point light sources;

  foreach *Point light source* **do**

    Generate a shadow map;

    Render the scene from the point of view of that point light source;

    Save this shadow map (depth);

  fin

fin

Rendering the scene from the camera point of view;

With a shader program divide the view frustum into m subfrustum (cascade);

foreach *Subfrustum* **do**

  Do the depth test;

  Use filtering with Multiple PCF;

  Screen display;

fin

3.2.2 Frustum subdivision

Partitioning the frustum is the act of creating subfrusta. One technique for splitting the frustum is to calculate intervals from zero percent (0%) to one hundred percent (100%) in the Z-direction (see Figure 3.2). Each interval then represents a near plane and a far plane as a percentage of the Z-axis.

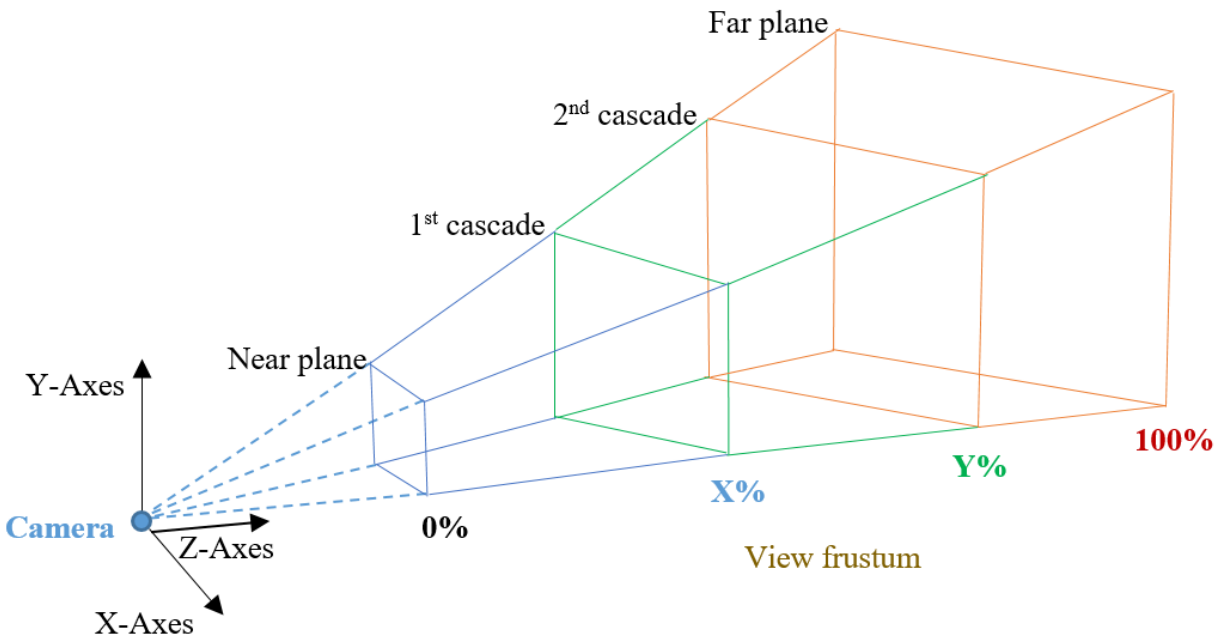

Figure. 3.2. Splitting the view frustum from 0% to 100% in the z-direction.

The subdivision of the camera frustum is done either linearly (each subfrustum has the same length) or logarithmically (the length of the first subfrustum is smaller than the length of the last one). It can also be a combination of linear and logarithmic splitting or it can be partitioned arbitrarily as we did in this work (see Figure 3.3).

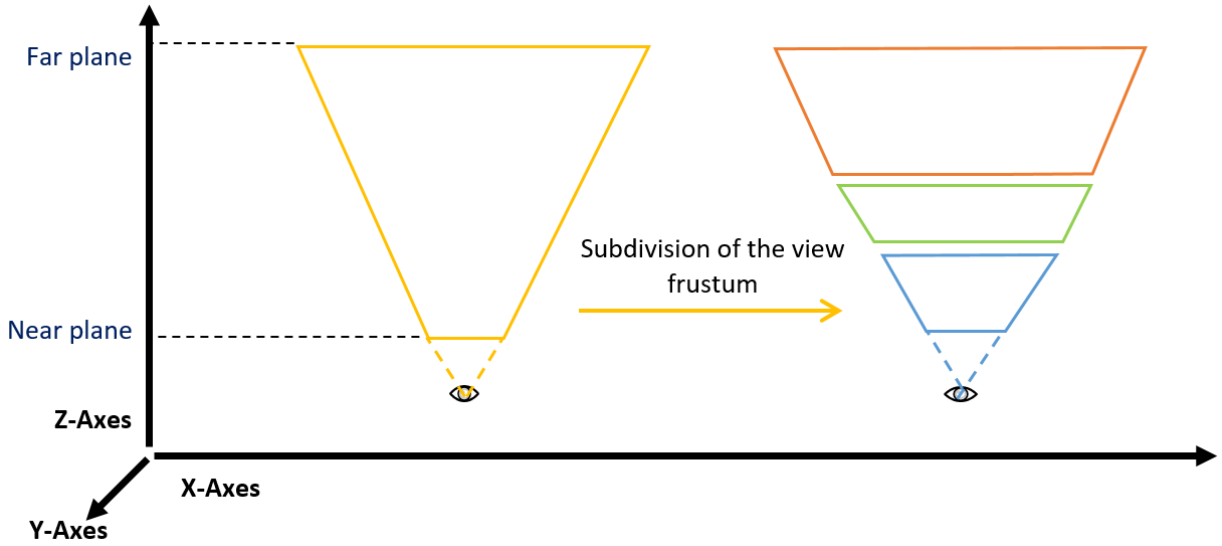

Figure. 3.3. Arbitrarily partitioning of the view frustum.

3.2.2.1 Computing the shadow level

From a high-level view, we are going to split the view frustum into several cascades (see Figure 3.4). Every cascade will be rendered into its own private shadow maps (a shadow map for each point light sample). The shadow algorithm itself will remain the same but when sampling the depth from the shadow maps we will need to select the appropriate maps based on the distance from the viewer.

Figure. 3.4. South gallery of cloister elne scene rendered using Cascade Shadow Map (CSM) technique.

For the purpose of this work, we used three cascades: near, middle and far. In the first cascade, we sampled the area light source into a high number of samples, a high-resolution shadow map is generated for each point light source to ensure that the closest objects have a high quality of shadows (see Figure 3.5). In the second cascade, we sampled the area light source into a medium number of samples, a medium-resolution shadow map is generated for each point light source. Finally, in the last cascade, we sampled the area light source into a small number of samples, a small-resolution shadow map is generated for each point light source to reduce the calculation time because we don't have to use high quality for objects that are far from the camera, but if we did, it will be a waste of calculation time.

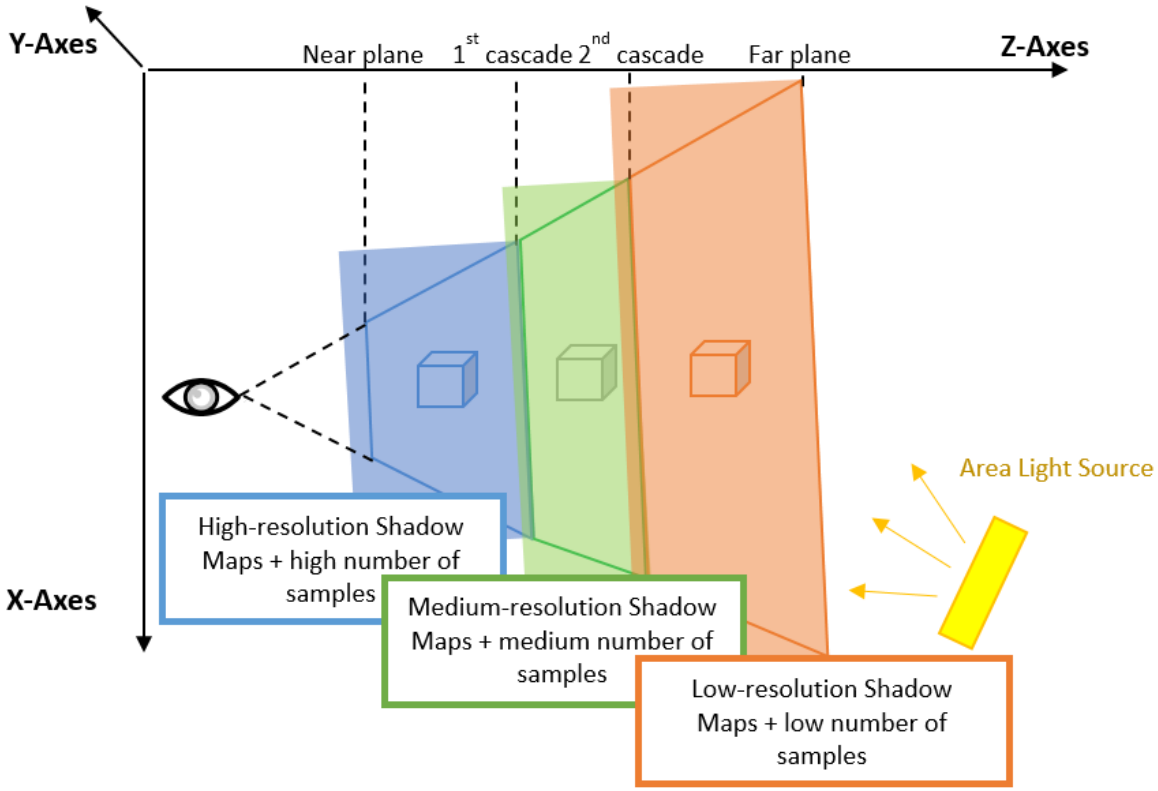

Figure. 3.5. Overview of our proposed technique.

3.2.3 Soft shadow approximation with sampling the area light source

In this section, we will demonstrate the origins of the soft shadow equation from the Kajiya rendering equation [Kaj86], and then approximate the soft shadows by the area light source subdivision technique.

3.2.3.1 From the rendering equation to soft shadow approximations

To calculate the shadow, we can simplify the fundamental equation in computer-graphics; the rendering equation (equation 1.18 of chapter 1), so that a light source L is a set of points l forming the surface of the light. We will refer to these points as light samples (source).

If it's assumed that only direct illumination is of interest, then the recursive part of the integral can be removed (L_0 is replaced with L_e, since $L_0 = 0$ for non-light sources). And thus only points in the light source l are considered. In addition, we assume a single light source, meaning that $S = L$ and the missive part L_e of the surface is ignored.

This results in the direct-lighting equation:

$$L_0(x \to \omega_0) = \int_L f_r(x, x \to l, \omega_0) L_e(l, l \to x) V(x, l) G(x, l) dl \tag{3.1}$$

Where V denotes a visibility function, being either 1, for visible surfaces, or 0 for occluded surfaces.

Additionally, if it can be assumed that the distance from the light source is large, and the light shape is simple, the geometric term G varies little, and if all the surfaces of the scene are Lambertian (perfectly diffuse), which causes that the BRDF become independent of the directions, i.e. $f_r(x, x \to l, \omega_0) = \rho(x)/\pi$. The integral can be split in to a product of shading and visibility (shadow):

$$L_0(x \to \omega_0) = \frac{\rho(x)}{\pi} \int_L G(x, l) dl . \frac{1}{\mid L \mid} \int_L L_e(l, l \to x) V(x, l) dl \tag{3.2}$$

Where:

- $\boldsymbol{L_0(x \to \omega_0) = \frac{\rho(x)}{\pi} \int_L G(x, l) dl}$ = The shading.
- $\boldsymbol{\frac{1}{|L|} \int_L L_e(l, l \to x) V(x, l) dl}$ = The shadow.

If the emissive part L_e can be considered constant for all directions and points on the surface, what is left to evaluate is the visibility integral:

$$V_L(x) = \frac{1}{\mid L \mid} \int_L V(x, l) dl \tag{3.3}$$

The visibility integral in equation 3.3 is what is typically meant by soft shadow computation, shrinking the integration area L to a single point is what produces hard shadows, thus, in our work, we are interested in the visibility factor $V_L(x)$ to calculate the soft shadow.

Computing soft shadows require solving the integration of equation 3.3, but it is very complex to solve analytically. Thus, we are particularly interested in stochastic methods of global illumination of the Monte-Carlo type [Kaj86] to approximate this integration.

The estimate of the integral function is obtained by the weighted average of the values of the function at a set of points sampled in a stochastic manner. Thus equation 3.3 becomes:

$$V_L(x) \approx \sum_{i=1}^{N} w_i V(x, l_i) \tag{3.4}$$

A visibility test $V(x, l_i)$ is required to determine whether point x is in the shadow area or where visibility is only determined in N point light samples l_i with associated weights w_i, where $\sum_{i=1}^{N} w_i = 1$ illuminated by point light i.

We can produce a soft shadow by sampling the area light source to N point light. This soft shadow can be estimated by the ratio S_N of shadow maps:

$$S_N = \frac{1}{N}\sum_{i=1}^{N} V(x, l_i) \tag{3.5}$$

3.2.3.2 Area light source subdivision

We will use the Zerari technique [ZBNK14] to subdivide the area light source uniformly. We assume that the area light source is square. The first step is always to subdivide the area light source into four initial point lights in the corners of the square area light source, then other point lights will be created by following equation 3.6 if this area light needs to be subdivided (see Figure 3.6).

$$n = k^2 \tag{3.6}$$

Where:

- $\boldsymbol{n}$: is the number of point light sources,
- $\boldsymbol{k \times k}$ **:** is the kernel of uniform subdivision,$k \geq 2$.

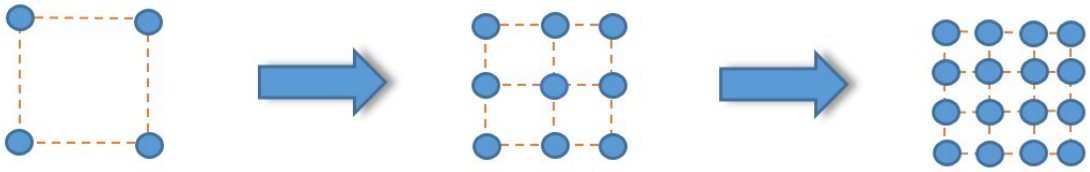

Figure. 3.6. Area light source subdivision.

3.2.4 Shadow map rendering

The shadow calculation consists of identifying the parts of the scene which are hidden from the light source. This therefore intrinsically amounts to a calculation of visibility from the point of view of light.

In the following, the shadow maps are produced from each point light (sample) of each cascade using the shadow mapping technique with an orthographic projection.

For each point light of each cascade, a shadow map is created, after subdividing the area light source into a set of n point lights, so n shadow maps will be created.

Generating soft shadows with multiple shadow maps per light is computationally expensive due to the high sampling density which is required to render smooth, visually appealing penumbra regions. If the density is too low, banding artifacts are likely to appear,

and the human visual system does not perceive a soft shadow anymore, but several hard shadows.

3.2.5 Assessing the Shadow Map Information

The shadow maps must be assessed in an illumination render pass, this step is identical to the second render pass in the standard shadow mapping algorithm and it comes after having all the necessary shadow maps. In this section, we show how to solve obstacles that can appear due to different subdivision depths, and due to the large number of depth textures that have to be sampled.

3.2.5.1 Assigning Shadow Map Contribution Weights

We, therefore, apply weights on the sampling points. In areas of many subdivisions, the individual samples are assigned a smaller weight, and will not contribute as much to the darkness of the penumbra as the ones with a large weight.

In the case of an area light source which is subdivided using the Zerari technique [ZBNK14] as mentioned in section 3.2.3.2, the weights w_i assigned to the i^{th} shadow map are calculated with:

$$w_i = \frac{1}{k^2} \tag{3.7}$$

Where:

- $\boldsymbol{k^2}$: is the number of samples,
- The sum of all the weights is 1.

Otherwise, the weights have to be normalized to make sure the final accumulated shadow values lie between 0 (fully lit) and 1(fully shadowed).

3.2.5.2 Soft shadow visualization using n Shadow Maps

The shadow map in the shadow mapping algorithm is evaluated in a second pass from the camera point of view using the data from the first pass for shadow calculation to calculate whether fragments are in shadow according to their occlusion. Back-project each pixel in the pixel shader to each of the point light's frustum comparing the depth of that particular pixel from light's points of view to calculate soft shadow. We get a value (0 or 1) on whether the pixel in under shadow or not with respect to a particular point light source i, these values are multiplied with their weight w_i and summed-up, resulting in an estimate for the percentage of occlusion.

Using a high number of shadow maps can lead to problems due to technical limitations: For example, in some older rendering APIs, it is not possible to sample more than 16 textures in a single rendering pass. This limits the theoretical number of shadow maps to 16.

Deferred rendering system introduced by [DWS+88], as well as a so-called accumulation buffer, is a way to solve this problem, which is a screen-space buffer with a single data channel. For each shadow map, we render the scene in a separate render pass. Instead of using the obtained hard shadow value of a screen space fragment $f(x, x')$ directly for illumination, we multiply it with its weight and add it to the accumulation buffer at the position $f_{accumulationBuffer}(x, x')$.

All shadow maps have been evaluated and the accumulation buffer is filled after n render passes. Now, in a final rendering pass, the scene is illuminated: For each screen space fragment $f(x, x')$, the corresponding accumulation buffer value $f_{accumulationBuffer}(x, x')$ is sampled and used as the occlusion percentage.

3.2.5.3 Soft Shadow evaluation using Texture arrays

One of the useful extensions introduced in OpenGL is "Array Textures". This extension introduces the idea of one and two-dimensional array textures. A Texture array is a collection of one and two-dimensional images of identical size and format, organized in layers. TexImage2D is used to specify a one-dimensional array texture, where the height specifies the number of layers for the array texture. Similarly, TexImage3D is used to declare a two-dimensional array texture, where the depth specifies the number of layers of 2D textures. Array Textures allows us to send up to 512 textures.

This way, evaluating the shadow maps in multiple passes is no longer necessary. Making all the shadow maps accessible in a single shader using the Arrays Texture functionalities perfectly serves our purpose.

In case of 2D array textures, a layer is selected by specifying the r texture coordinate. Further it is accessed as though it were a one or two-dimensional texture. Texture lookups do not filter between layers, though the same can be achieved in the shaders. Rendering to 2D textures can only be achieved by binding a 2D array texture to a frame buffer object. Furthermore, using an accumulation buffer is not necessary anymore: We obtain the fragment's occlusion value by simply summing up the hard shadows values multiplied with their corresponding weights. This value can be directly used in the same pass to modify the illumination. The current fragment's occlusion value can thus be obtained without adding more passes, saving n read/write operations as well as the memory previously consumed by the accumulation buffer.

3.2.5.4 PCF filtering

Generating soft shadows with multiple shadow maps per light is computationally expensive due to the high sampling density which is required to render smooth, and banding artifacts are more likely to become visible if we generate a few shadow maps. Similar problems occur if the camera is extremely close to a penumbra so that the maximum number of shadow maps is not sufficient to generate an appealing penumbra region.

We suggest sampling the transitions between the individual shadow maps using a small PCF kernel in such situations, to improve the smoothness. PCF filtering softens the shadow boundaries, and a version with a 3×3 kernel can be used on modern graphics hardware without a performance hit.

We apply a reconstruction and re-sampling filter to the area of pixel values. Inspired by equation 3.5, and given a point p (the orange point in Figure 3.7) to perform the search, and a two-dimensional filter kernel f, the filtered visibility value of S_N. The visibility factor is estimated as an average value:

$$V_{p,S_N} = \frac{1}{m} \sum_{i=1}^{m} V_{p,v_i} \tag{3.8}$$

Where m is the number of visible point light sources at point p.

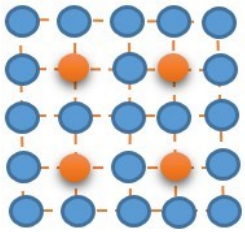

Figure. 3.7. Multiple PCF filters (with a 3×3 kernel) applied to each point light source.

3.3 Implementation

Our experiments are implemented on a graphics station which has the following configurations:

Hardware configuration: The graphics station is equipped with an Intel(R) Core i3-5005U 2.00GHZ CPU, a 4.00 GO RAM, and an Intel(R) HD Graphics 5500 graphics card.

Software configuration: For the realization of our application, we used the following software tools and platforms:

- **Operating system:** Windows 10 professional, 64-bit.
- **Programming language:** Visual Studio C ++ 2010.
- **Graphics API:** OpenGL 2.02.

We implemented the cascaded soft shadow maps technique (CSSM) described in section 3.2, and both rendering methods described in section 3.2.5 (3.2.5.2 and 3.2.5.3) using a two-dimensional rectangular light source and different shadow map resolutions (1024^2 for the first cascade, 512^2 for the second cascade, 128^2 for the last cascade) . The implementation of texture arrays using a single render pass to evaluate the shadow illumination performs slightly better (approximately 10% faster) than the deferred rendering solution. In systems that use the older APIs, the deferred rendering solution appears to be a sufficient choice for the application of our algorithm.

We declared 2D-Array Textures as follows :

```
//Generate an opengl texture
glGenTextures(1, &ShadowMap);
glBindTexture(GL_TEXTURE_2D_ARRAY, ShadowMap)
//Setting up the parameters.
glTexParameteri(GL_TEXTURE_2D_ARRAY, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D_ARRAY, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D_ARRAY, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_2D_ARRAY, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_2D_ARRAY,
GL_TEXTURE_COMPARE_MODE, GL_COMPARE_R_TO_TEXTURE);
glTexParameteri(GL_TEXTURE_2D_ARRAY, GL_TEXTURE_COMPARE_FUNC, GL_LEQUAL);
glTexParameteri(GL_TEXTURE_2D_ARRAY, GL_DEPTH_TEXTURE_MODE, GL_INTENSITY);
//Create space for the array texture.
/*Depth in our case is the number of shadow maps (each point light source has a shadow map)*/
glTexImage3D(GL_TEXTURE_2D_ARRAY, 0, GL_DEPTH_COMPONENT24, ShadowResolution,
ShadowResolution, Samples , 0, GL_DEPTH_COMPONENT, GL_FLOAT, NULL);
```

We evaluate the shadow maps in a second rendering pass using the data from the first pass for shadow calculation at the fragment shader. After checking in which cascade this current pixel has to be rendered using the z position (gl_Position.z), we evaluate whether the fragment is in shadow according to its occlusion for each point light source in that cascade using the lookup function. While assessing the occlusion percentage of the current fragment we filter the hard shadows of each point light source using a PCF filter with 3×3 kernel to blur the edges of the hard shadows to remove the artifacts. At the end, we get soft shadows with high quality in the first cascade, and in the second and last cascade we

get soft shadows with less quality due to the number of samples in that cascades and the shadow maps resolution.

Here we represent a part of our fragment shader code corresponding to what we explained previously:

```
#extension GL_EXT_texture_array : enable // enabling the Texture Array
//declaration of sampler
uniform sampler2DArray ShadowMapCascade[3];
//fetching color from layers ShadowTexCoord.z from the 2D
//location ShadowTexCoord.xy from sampler
if ( ZdepthViewSpace ≤ cascadedSplits[c] )
{
    for( int i = 0; i ≤ cascadeSamples[c]; i++)
        ShadowTexCoord[i] = LightTextureMatricesCascade[i]  vertex;
    shadow = 0;
    for( int i = 0; i ≤ cascadeSamples[c]; i++ )
    {
        // PCF filter using 3 × 3 kernel
        for (y = -1.0 ; y ≤ 1.0 ; y+=1.0)
        for (x = -1.0 ; x ≤ 1.0 ; x+=1.0)
            shadow += shadow2DArray( ShadowMapCascade[c], ShadowTexCoord[i]).r; //lookup fct
        shadow /= 9.0 // 3 × 3
        ShadowCascade += shadow;
    }
    ShadowCascade /= cascadeSamples[c]; // persentage occlution
    diffuse = diffuse  ShadowCascade; //diffuse effect
    Speculaire = speculaire  ShadowCascade; //speculaire effect
    gl_FragColor = vec4((diffuse + speculaire),(diffuse + speculaire),(diffuse + speculaire), 1.0);
    // screen display
}
```

3.4 Results and evaluation

In this section, we will present the results of the implementation where different measurements of performance and visual quality will be compared and reviewed.

To compare our results, we chose the standard criterion of FPS (Frame Per Second) and the time required to generate an image. We evaluated the proposed technique using the RMSE (Root Mean Square Error) metric [WM05]. We validate our results qualitatively and

quantitatively on real-time scene images. Our test scenes, which contain various objects, are illustrated in [CL96, McG17, Ske12].

We present some results obtained with the proposed technique of rendering cascaded physically correct soft shadows based on the sampling of the area light source and splitting the camera view frustum into separate partitions to increase the shadow quality for near objects and reduce the calculation time. We compare our images generated by the proposed technique to the images generated with the method described in [SMSW12] as well as the reference image (Ground Truth) calculated using an area light source of 256 samples which is the ideal expected result (image similar to the reality).

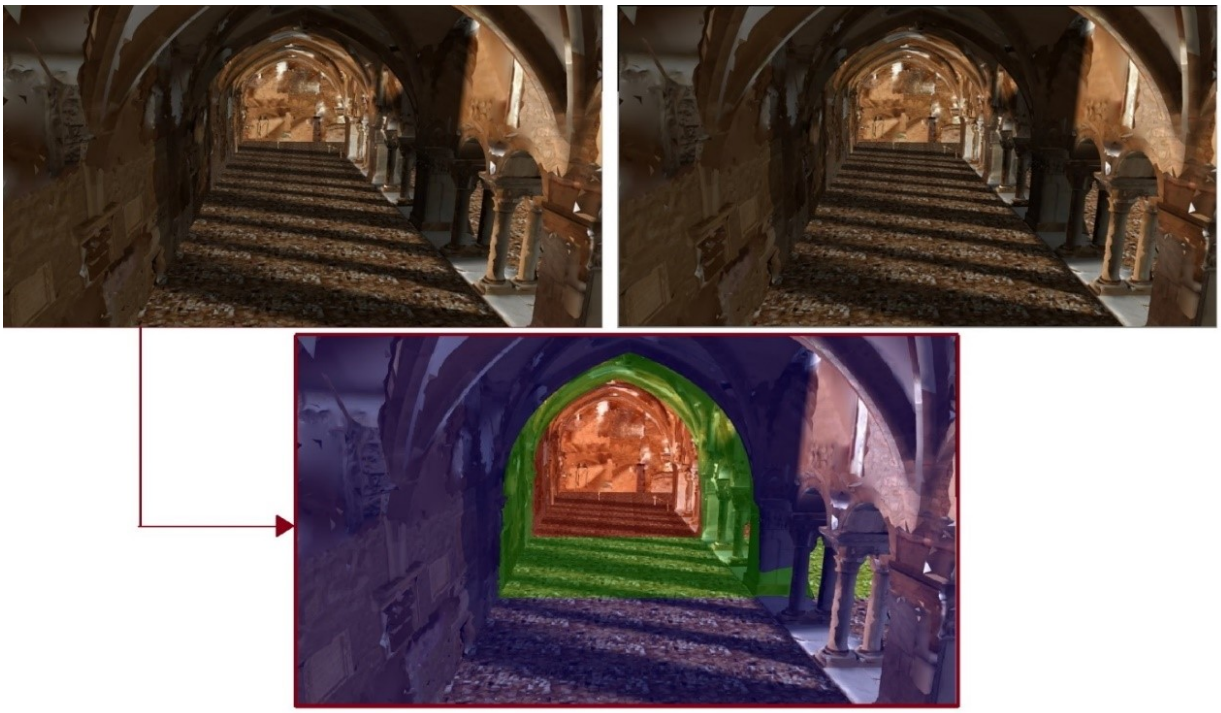

Figure. 3.8. South gallery of cloister elne scene rendered with two different configurations.

Figure 3.8 shows the South gallery of cloister elne scene rendered with two different configurations. The image on the left is rendered by our proposed technique using 3 cascades with PCF filter with a time equal to 1.4 seconds. The image on the right is rendered by Michael Schwärzler's method which uses 36 point lights with a time equal to 2.7 seconds. The image in the middle is the same scene, rendered with color in each cascade.

3.4.1 Visual results

In this part, we present below the results of our algorithm on different scenes using different configurations. We also test the validity of the proposed technique in terms of quality and calculation time and we finally discuss the strong and weak points using perceptual metrics.

To verify the precision of the proposed technique, we compare the final results obtained by our technique to images synthesized by Schwärzler's method [SMSW12].

Figure 3.9 on the right presents the result of Schwärzler's method which shows 3 cubes illuminated by an area light source approximated by 36 point lights, soften the resulting soft shadows using PCF with an FPS equal to 14. Our proposed technique (left image) uses 3 cascades, we put a cube in each cascade, in the first cascade the cube illuminated by 16 point light sources using a shadow map per a point light with 1024 resolution. In the second cascade the cube illuminated by 9 point light sources and shadow maps of 512 resolution and in the last cascade it illuminated by 4 point light sources using shadow maps of 128 resolution, with an FPS equal to 48.

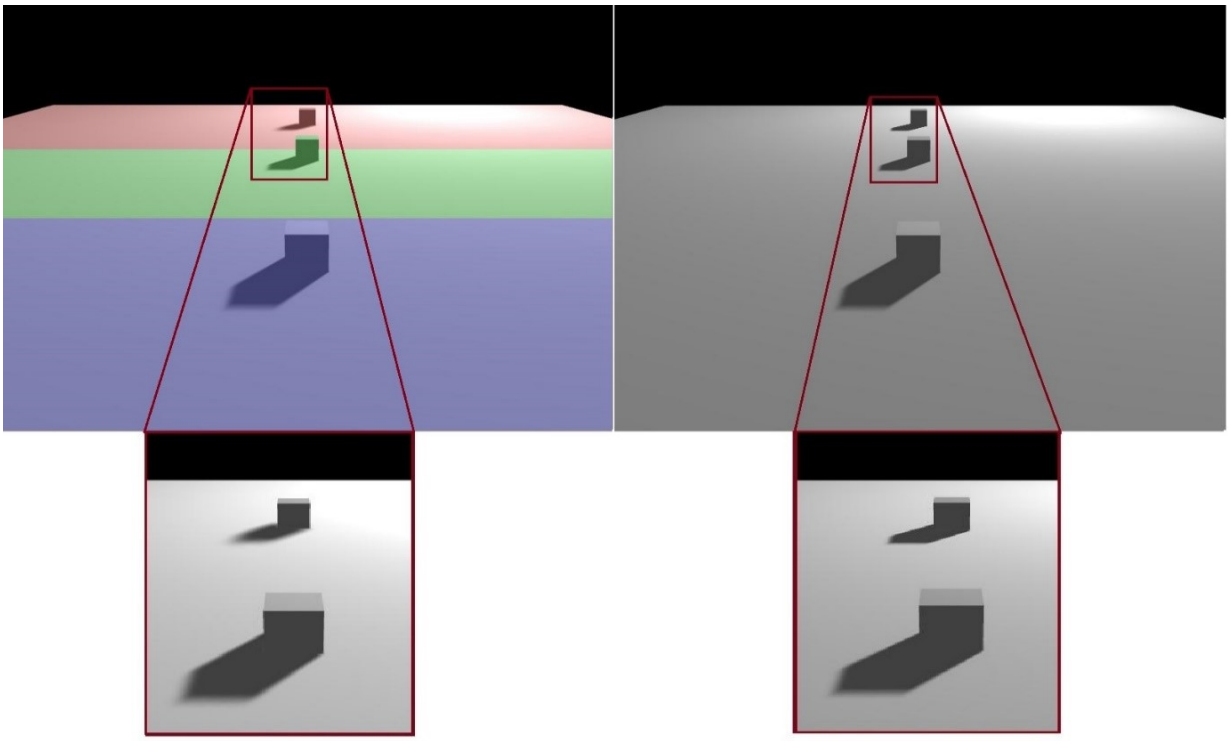

Figure. 3.9. Visual comparison. Both techniques use a multiple PCF filter. The right image is created by the method of [SMSW12]: the area light source sampled in 36 samples (FPS = 14). Left image is created by our technique: 3 cascades (16 PLS,9 PLS,4 PLS) (FPS = 48).

The visual quality of our solution proposed with multiples cascades and different shadow maps resolutions is almost identical to the images with a significantly higher amount of samples and a high shadow maps resolution for all over the scene. In order to validate our technique, we also calculate the mean square error (RMSE) between these two images which is 0.0075 (see Figure 3.10), which means that the two images in Figure 3.9 are almost identical.

To better visualize our images in composition 3.9, we have displayed the scene without

the specular effect.

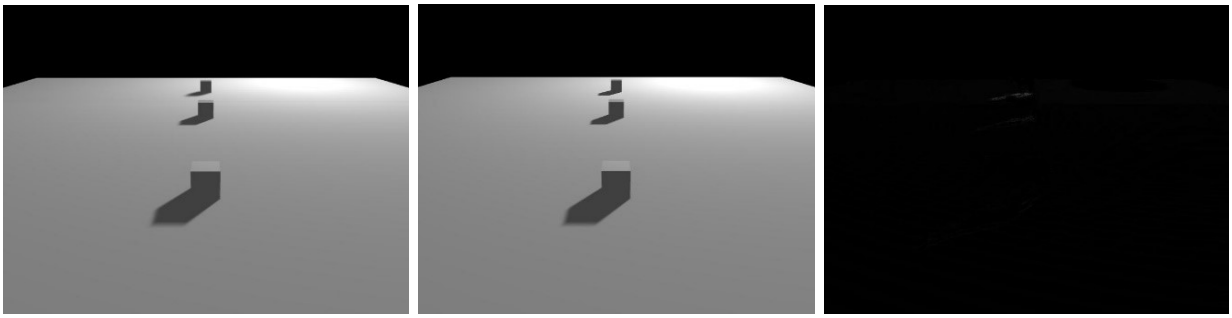

Figure. 3.10. Visual comparison. Left image: is created by our technique. The image in the middle: is created by the method of [SMSW12]. Right image: difference image between the image generated by Schwärzler's method and the image generated by our proposed technique.

In Figure 3.11, we compare the image quality of the Bunny model lit by an area light source using the proposed technique and the Schwärzler's method. The image generated by the Schwärzler's method uses the same number of samples in the previous model (36 shadow maps) with an FPS equal to 6. The image generated by the proposed technique uses the same cascades too with the same number of samples (16 shadow maps in the first cascade, 9 shadow maps, 4 shadow maps) with an FPS equal to 12.

We reduced the number of samples from 36 samples to 16 samples where the multiple PCF filters replaced the effects of the shadow maps in our algorithm but we kept the resolution of the shadow maps on that cascade (1024) and we also reduced the number of samples from 36 samples to 9 samples and the resolution of the shadow maps from 1024 to 512 on the second cascade where the camera cannot notice the difference of quality that far. The same thing on the last cascade, we reduced the number of samples from 36 samples to 4 samples and the resolution of the shadow maps from 1024 to 128, which allowed us to avoid a large number of rendering passes. This leads directly to a significant reduction in the computing time.

In Figure 3.11 on the right, the second and the last bunny have the same shadow quality of the first one, but in fact we can't notice if the quality is the same when the object is far from the camera. To validate this, we calculate the mean square error (RMSE) between these two images which is 0.0077 (see Figure 3.12), this value clearly demonstrates the effectiveness of our technique.

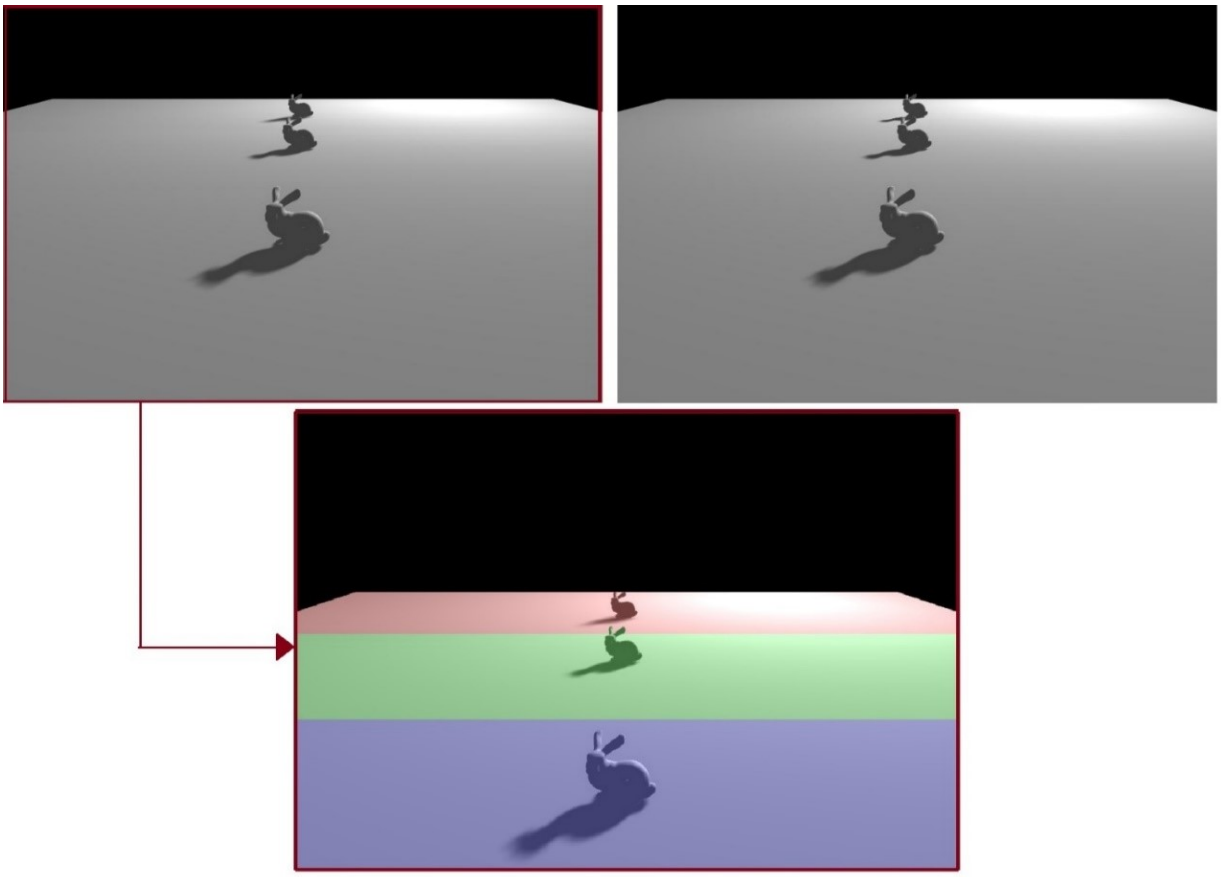

Figure. 3.11. Visual comparison. Both techniques use a multiple PCF filter. The right image is created by the method of [SMSW12]: the area light source sampled in 36 (FPS = 6). Left image is created by our technique: 3 cascades (16 PLS,9 PLS,4 PLS) (FPS = 12). The image in the middle is the same scene, rendered with color in each cascade.

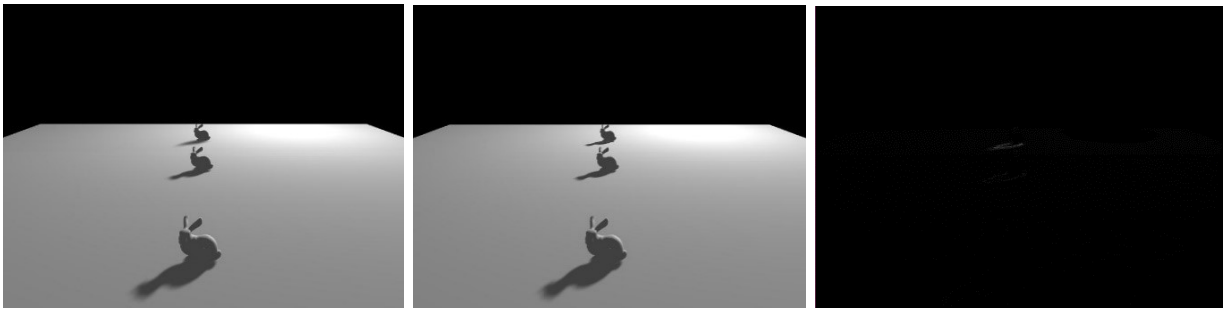

Figure. 3.12. Visual comparison. Left image: is created by our technique. The image in the middle: is created by the method of [SMSW12]. Right image: difference image between the image generated by Schwärzler's method and the image generated by our proposed technique.

In Figure 3.13, we compare the image quality of the proposed technique with a reference image which is considered to be the ideal case for our comparisons. The reference image is generated using an area light source sampled by 256 samples (256 shadow maps) with a

calculation time equal to 4 seconds.

The image generated by the proposed technique uses 3 cascades each cascade uses different samples of the area light source (cascade 1: 16 samples, cascade 2: 9 samples, cascade 3: 4 samples), with a different resolution of the shadow maps (cascade 1: 1024, cascade 2: 512, cascade 3: 128), and a calculation time equal to 0.29 seconds.

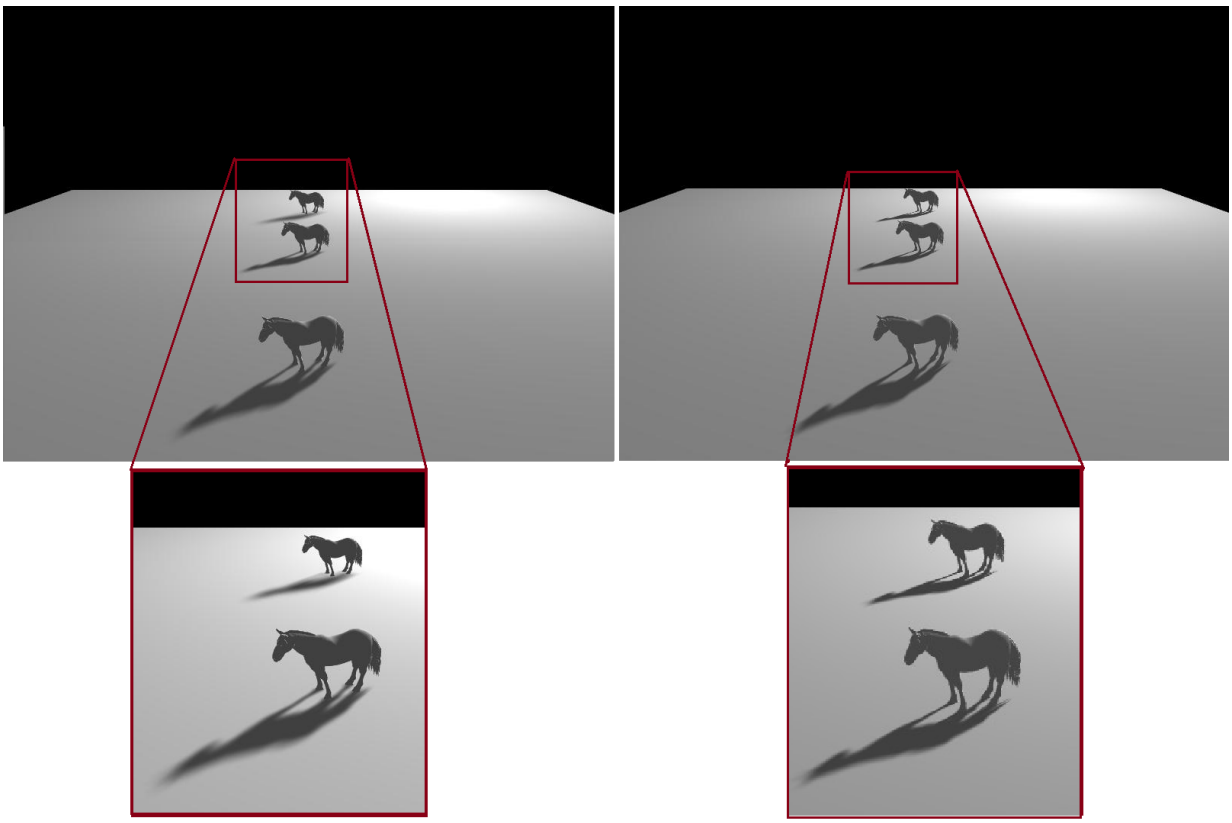

Figure. 3.13. Visual comparison between our result and the reference image. Left image: the proposed technique with 29 rendering passes and multiple PCF filtering (0.29 seconds).Right image: image acting as Ground Truth with 256 samples (4 seconds).

The mean square error (RMSE) between these two images is 0.0091 (Figure 3.14 in the right). However, the proposed technique can generate almost identical results as the reference image (Figure 3.14 in the middle). Compared to 256 samples, our technique achieves an acceleration of 14 times using 29 rendering passes. We can judge that our results are satisfactory because they are interactive, they have good visual quality, and are close to the reference images.

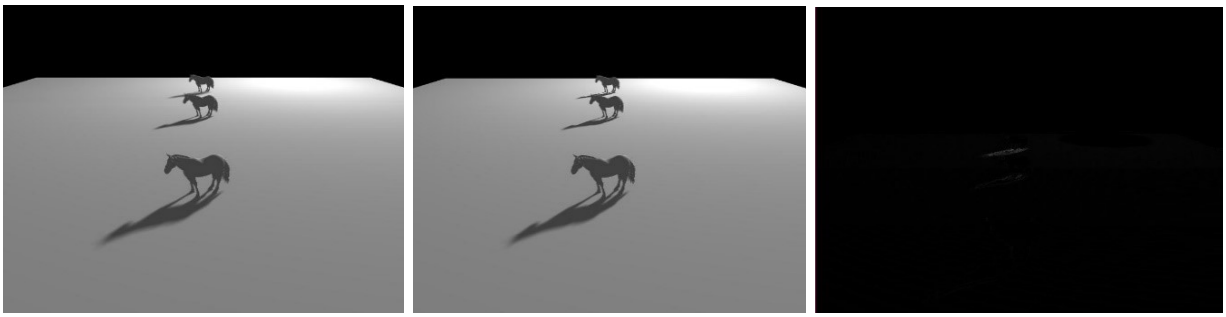

Figure. 3.14. Visual comparison between our result and the reference image. Left image: the proposed technique with 29 rendering passes and with multiple PCF (0.29 seconds). Image in the middle: image acting as Ground Truth with 256 rendering passes (4 seconds). Right image: difference image between the Ground Truth image and the image generated by the proposed technique.

In Figure 3.15, we compare the image quality of the Dragon model lit by an area light, using the proposed technique and the Schwärzler's method. The image generated by the Schwärzler's method uses 81 samples of the area light source (81 shadow maps), and with a time equal to 5.09 seconds. The image generated by the proposed technique uses three cascades(25 shadow maps in the first cascade, 16 shadow maps in the second cascade, 4 shadow maps in the last cascade), with a time equal to 2.57 seconds. The mean square error (RMSE) between these two images is 0.0085. However, the proposed techniques with those different samples in each cascade generate almost identical results as the Schwärzler's image that uses a high number of samples (81 samples) with the advantage of reducing the calculation time.

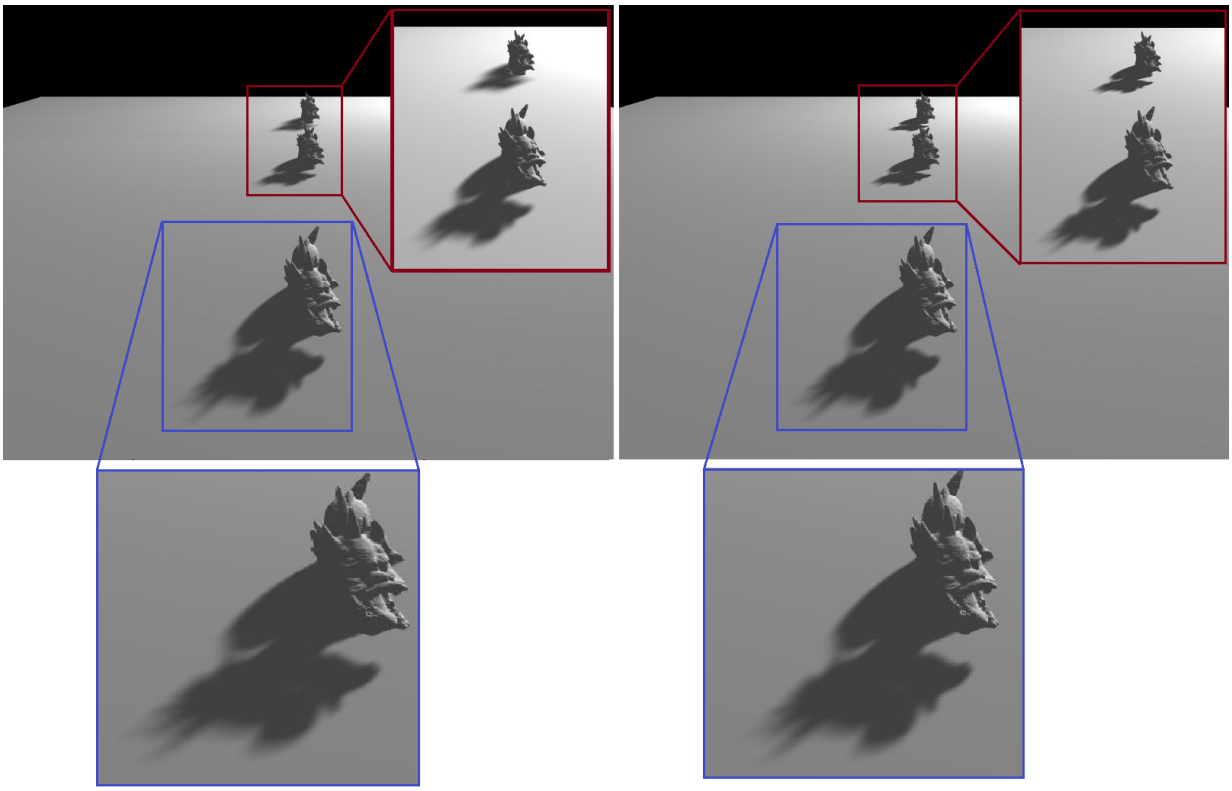

Figure. 3.15. Visual comparison. The left image is created by our technique: 3 cascades (25 PLS,16 PLS,4 PLS) with PCF filter (2.57 seconds). The right image is created by the method of [SMSW12]: the area light source sampled in 81 samples and with PCF filter (5.09 seconds).

All figures from Figure 3.8 to Figure 3.15 show a comparison between our technique and Michael Schwärzler's method [SMSW12]. It is clear that the proposed technique faster than that of Michael Schwärzler, even the quality of the soft shadows of the proposed technique is almost the same as of Michael Schwärzler.

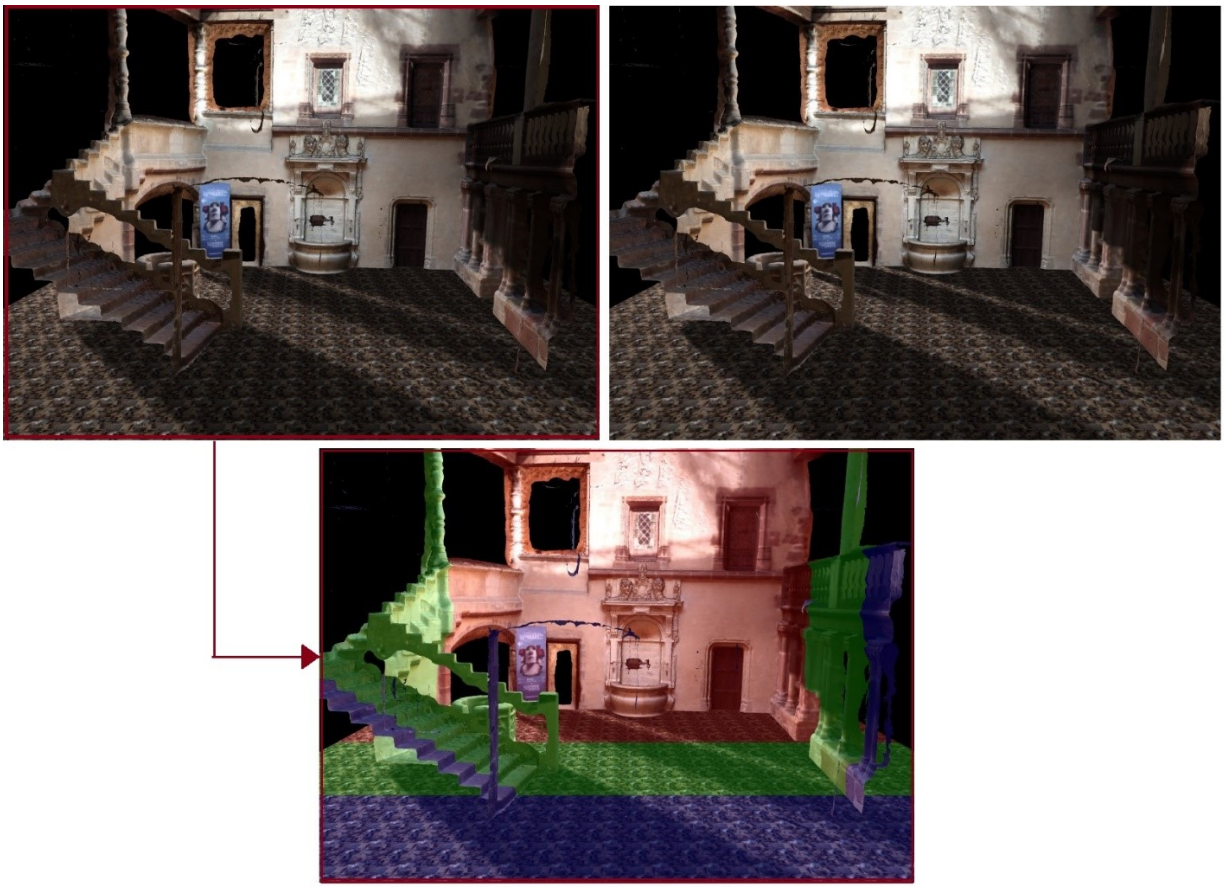

Figure. 3.16. Joury rodez hotel scene rendered with two different configurations. Image on the right is rendered by our proposed technique using 3 cascades (16 PLS,9 PLS,4 PLS) with PCF filter and a time equal to 3.9 seconds. Image on the left is rendered by Michael Schwärzler method which uses 36 point lights with a time equal to 5.9 seconds. Image in the middle is the same scene, rendered with color in each cascade.

In the following (Table 3.1) we present a summary of the results of our technique and Michael Schwärzler's method, mentioning the FPS and/or the rendering time of each model we used in both methods as well as the size of each model.

Model	Size of the model	Our technique : CSSM	Schwärzler's method
Cube	1octet glutSolid-Cube(1)	▪ **Samples** = (16,9,4) ▪ **FPS** = 48 ▪ **Time** = 0.03 seconds	▪ **Samples** = 36 ▪ **FPS** = 14 ▪ **Time** = 0.085 seconds
Bunny	315 Ko	▪ **Samples** = (16,9,4) ▪ **FPS** = 12 ▪ **Time** = 0.12 seconds	▪ **Samples** = 36 ▪ **FPS** = 6 ▪ **Time** = 0.22 seconds
Horse	1.31 Mo	▪ **Samples** = (16,9,4) ▪ **FPS** = 4 ▪ **Time** = 0.29 seconds	▪ **Samples** = 36/256 ▪ **FPS(36)** = 12 ▪ **FPS(256)** = / ▪ **Time(36)** = 0.54 seconds ▪ **Time(256)** = 4 seconds
Dragon	8.66 Mo	▪ **Samples** = (25,16,4) ▪ **FPS** = / ▪ **Time** = 2.57 seconds	▪ **Samples** = 81 ▪ **FPS** = / ▪ **Time** = 5.09 seconds
CloisterElne	20 Mo	▪ **Samples** = (16,9,4) ▪ **FPS** = / ▪ **Time** = 1.4 seconds	▪ **Samples** = 36 ▪ **FPS** = / ▪ **Time** = 2.7 seconds
JouryRodez hotel	62 Mo	▪ **Samples** = (16,9,4) ▪ **FPS** = / ▪ **Time** = 3.9 seconds	▪ **Samples** = 36 ▪ **FPS** = / ▪ **Time** = 5.9 seconds

Table. 3.1. Summary of the results of our technique and Schwärzler's method.

As explained previously, we made a direct comparison with the results presented in [SMSW12]. We used the same models, data, and the same graphic card. Figure 3.17 and Figure 3.18 show a histogram of the rendering time and FPS results achieved by Schwärzler's method and our approach. As we can see Schwärzler's algorithm is not as fast as our algorithm on these test scenes.

With the Cube model, the Schwärzler algorithm is almost 4 times slower if we compare the FPS results. Using the Dragon scene (3 dragon models of 8.66 Mo), Schwärzler's method needs from 5.09 to 7 seconds which is 2 to 3 times slower. On Bunny, Horse and Cloister Elne scenes, Schwärzler's algorithm is almost 2 times slower.

In these tests, our approach was always faster than Schwärzler's method which means that we increased the performance and efficiency.

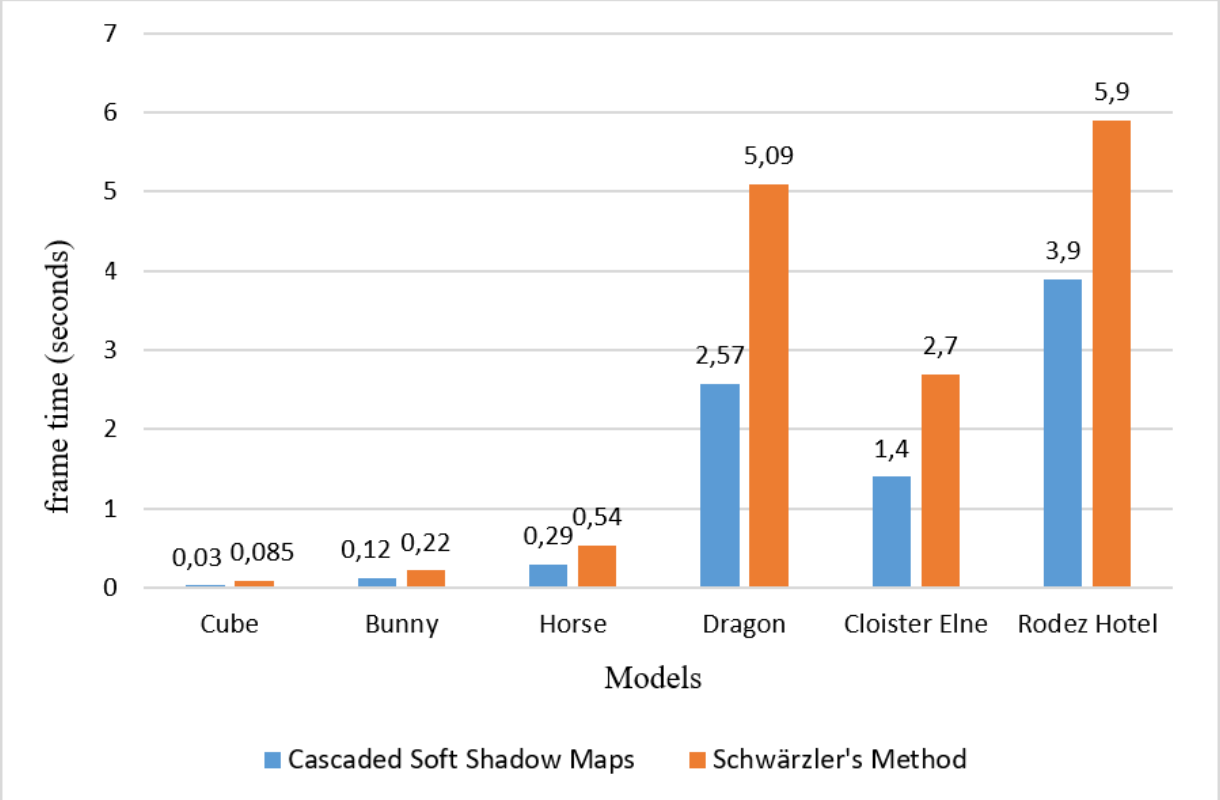

Figure. 3.17. Comparing the rendering time between our cascaded soft shadow maps technique and Schwärzler's method using different models with different sizes.

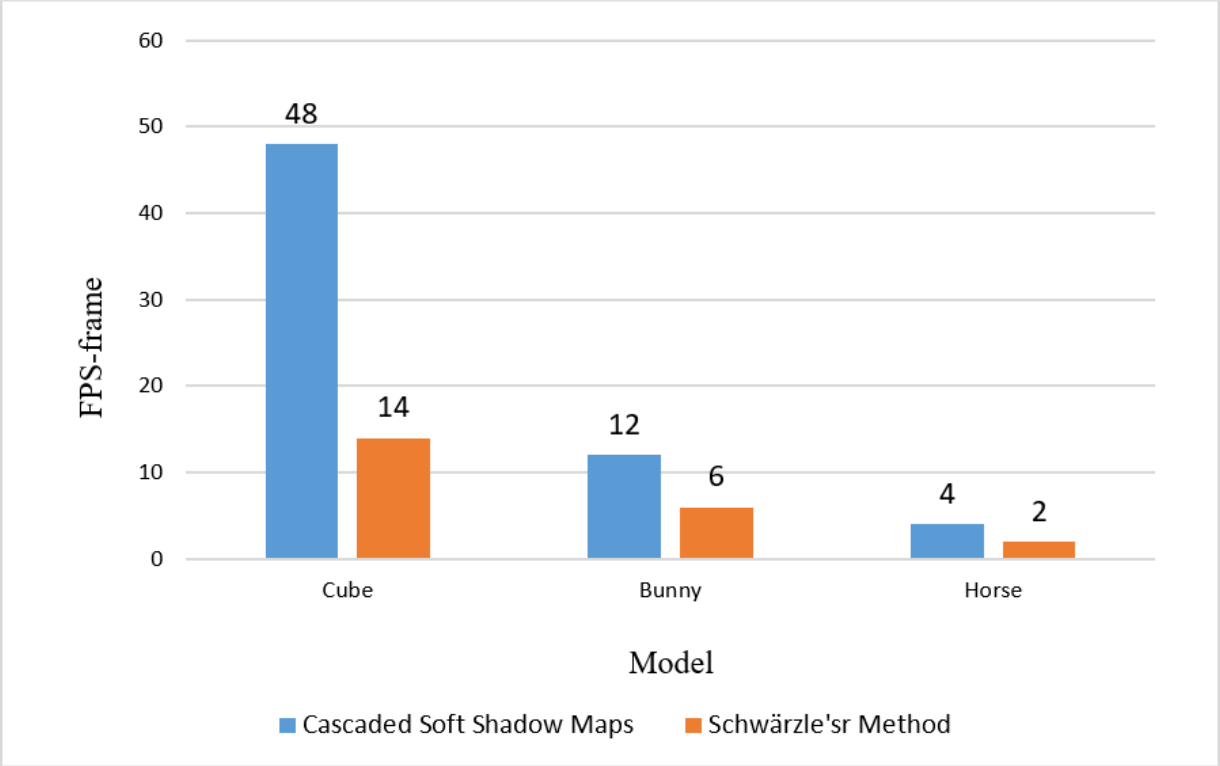

Figure. 3.18. Comparing the FPS between our cascaded soft shadow maps technique and Schwärzler's method using different models with different sizes.

3.5 Conclusion

We presented a new technique that can render physically accurate soft shadows using cascaded shadow maps technique.

In our test application, we were able to render soft shadows of a quality similar to the ones generated with 256 samples, but at interactive or even real-time frame rates. Performance can even be further increased by reducing the rendering time using cascaded shadow maps technique which reduces both shadow maps resolution and the number of samples in cascades that are far from the camera, besides, the use of multiple PCF filters in rendering soft shadows makes it possible to reduce a large number of shadow maps and to speed-up the calculation time. During our various experiments we have shown that the proposed technique is more efficient in terms of rendering time than that of Schwärzler's method.

Conclusions

Realistic rendering has become essential in industry, simulation and video games. This requires simulating the interaction between light and objects in a 3D scene, a computation known as global illumination is often very computationally expensive. Indeed, to generate an image, the main calculation has been formalized in an equation so-called rendering equation. This equation reflects the conservation of energy in the transport of light. it can only be achieved by approximations. One of the most importatnt complicated effects simulated by these approximations is soft shadows.

Soft shadows are, in contrast to hard shadows, not cast by point lights without extents, but by area light sources. They do therefore consist of umbra and penumbra regions, and despite the problem of increased computational costs that caused by the large number of samples required to achieve a good quality of soft shadows, using them is worth the effort. Nearly every shadow in reality has soft boundaries, so using soft shadows in rendering applications significantly increases the realism of the generated images.

Physically accurate soft shadows in image synthesis applications can be simulated by taking and accumulating multiple samples of the entire area light source, such as each sample represent a point light source. In this work, we have proposed a solution to the problems of physically accurate soft shadows in real-time applications. The contribution that we have presented is to design a technique to efficiently compute physically accurate soft shadows generated by sampling an area light source into a small number of samples using the Zerari technique which is based on subdividing the area light source uniformly, and replacing the effect of a large number of samples by a multiple PCF filter, besides, using the CSMs technique to reduce the samples of the area light source in the objects that are far from the camera to both reduce the computing time and improving the quality of the rendered image. Experimental results have shown that our contribution can improve the efficiency, and produce high-quality shadows in large scale with real-time performance.

As future work, we suggest reducing the computation time required for the comparison step which is the second pass in the shadow map algorithm that costs too much by reducing the number of occlusion requests. We can achieve this by limiting the lookups on the shadow maps of the area light source samples for only penumbra zone which needs the percentage occlusion to get soft shadows, and render the shadow of the umbra zone from

one point light source of the area light source. This automatically leads to limiting the filtering to only regions with artifacts. This suggestion will increase the performance of rendering physically accurate soft shadow.

Bibliography

[AAM03] Ulf Assarsson and Tomas Akenine-Möller. A geometry-based soft shadow volume algorithm using graphics hardware. *ACM Transactions on Graphics (TOG)*, 22(3):511–520, 2003. https://doi.org/10.1145/1201775.882300.

[ADMAM03] Ulf Assarsson, Michael Dougherty, Michael Mounier, and Tomas Akenine-Möller. An optimized soft shadow volume algorithm with real-time performance. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 33–40, 2003.

[AHL+05] Lionel Atty, Nicolas Holzschuch, Marc Lapierre, Jean-Marc Hasenfratz, François X. Sillion, and Hansen Charles. Soft Shadow Maps: Efficient Sampling of Light Source Visibility. Research Report RR-5750, INRIA, 2005. http://dx.doi.org/10.1111/j.1467-8659.2006.00995.x.

[AMS+08] Thomas Annen, Tom Mertens, Hans-Peter Seidel, Eddy Flerackers, and Jan Kautz. Exponential shadow maps. In *Graphics Interface*, pages 155–161. ACM Press, 2008. https://doi.org/10.5555/1375714.1375741.

[Arv04] Jukka Arvo. Tiled shadow maps. In *Proceedings Computer Graphics International, 2004.*, pages 240–246. IEEE, 2004. https://doi.org/10.1109/cgi.2004.1309216.

[AW04] Graham Aldridge and Eric Woods. Robust, geometry-independent shadow volumes. In *Proceedings of the 2nd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 250–253, 2004. https://doi.org/10.1145/988834.988877.

[Ber86] Philippe Bergeron. A general version of crow's shadow volumes. *IEEE Computer Graphics and applications*, 6(9):17–28, 1986. https://doi.org/10.1109/MCG.1986.276543.

[BFGL09] Yang Baoguang, Jieqing Feng, Gael Guennebaud, and Xinguo Liu. Packet-based Hierarchal Soft Shadow Mapping. *Computer Graphics Forum*, 28(4):1121–1130, 2009.

[Bli77] James F Blinn. Models of light reflection for computer synthesized pictures. In *Proceedings of the 4th annual conference on Computer graphics and inter-*

active techniques, pages 192–198, 1977. https://doi.org/10.1145/563858.563893.

[Bli82] James F Blinn. Light reflection functions for simulation of clouds and dusty surfaces. *Acm Siggraph Computer Graphics*, 16(3):21–29, 1982. https://doi.org/10.1145/800064.801255.

[BWB19] Jakub Boksansky, Michael Wimmer, and Jiri Bittner. Ray traced shadows: maintaining real-time frame rates. In *Ray Tracing Gems*, pages 159–182. Springer, 2019. https://doi.org/10.1007/978-1-4842-4427-2_13.

[CD04] Eric Chan and Frédo Durand. An efficient hybrid shadow rendering algorithm. *Rendering Techniques*, 2004:15th, 2004.

[CG85] Michael F Cohen and Donald P Greenberg. The hemi-cube: A radiosity solution for complex environments. *ACM Siggraph Computer Graphics*, 19(3):31–40, 1985. https://doi.org/10.1145/325334.325171.

[CL96] B Curless and M Levoy. The stanford 3d scanning repository, 1996. http://graphics.stanford.edu/data/3Dscanrep/.

[Cro77] Franklin C Crow. Shadow algorithms for computer graphics. *Acm siggraph computer graphics*, 11(2):242–248, 1977. https://doi.org/10.1.1.424.6834.

[CT82] Robert L Cook and Kenneth E. Torrance. A reflectance model for computer graphics. *ACM Transactions on Graphics (ToG)*, 1(1):7–24, 1982. https://doi.org/10.1145/357290.357293.

[Dav07] Roger David. Lancer de rayons et anti –aliasing, 2007.

[DBB06] Philip Dutre, Philippe Bekaert, and Kavita Bala. Advanced global illumination, ak peters. *CRC Press, Boca Raton, FL*, 10:b10632, 2006.

[Dev19] Francois Deves. *Volumes d'ombre en rendu temps réel: Complexité géométrique et stratégie de partitionnement*. PhD thesis, Limoges, 2019.

[DL06] William Donnelly and Andrew Lauritzen. Variance shadow maps. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 161–165, 2006. https://doi.org/10.1145/1111411.1111440.

[DWS+88] Michael Deering, Stephanie Winner, Bic Schediwy, Chris Duffy, and Neil Hunt. The triangle processor and normal vector shader: a vlsi system for high performance graphics. *ACM Siggraph computer graphics*, 22(4):21–30, 1988. https://doi.org/10.1145/54852.378468.

[EAS+13] Elmar Eisemann, Ulf Assarsson, Michael Schwarz, Michal Valient, and Michael Wimmer. Efficient real-time shadows. In *ACM SIGGRAPH 2013 Courses*, pages 1–54. 2013.

[Eng06] Wolfgang Engel. *Shader X5: Advanced Rendering Techniques*. Charles River Media, Inc., 2006.

[ESAW11] Elmar Eisemann, Michael Schwarz, Ulf Assarsson, and Michael Wimmer. *Real-time shadows*. CRC Press, 2011. https://doi.org/10.1201/b11030.

[FBGP09] Vincent Forest, Loïc Barthe, Gael Guennebaud, and Mathias Paulin. Soft textured shadow volume. In *Computer Graphics Forum*, volume 28, pages 1111–1120. Wiley Online Library, 2009. https://doi.org/10.1111/j.1467-8659.2009.01488.x.

[FBP08] Vincent Forest, Loïc Barthe, and Mathias Paulin. Accurate Shadows by Depth Complexity Sampling. *Computer Graphics Forum*, 27:663 – 674, 2008. https://doi.org/10.1111/j.1467-8659.2008.01164.x.

[Fer05] Randima Fernando. Percentage-closer soft shadows. In *ACM SIGGRAPH 2005 Sketches*, pages 35–es. 2005. https://doi.org/10.1145/1187112.1187153.

[FGH+85] Henry Fuchs, Jack Goldfeather, Jeff P Hultquist, Susan Spach, John D Austin, Frederick P Brooks Jr, John G Eyles, and John Poulton. Fast spheres, shadows, textures, transparencies, and imgage enhancements in pixel-planes. *ACM SIGGRAPH Computer Graphics*, 19(3):111–120, 1985. https://doi.org/10.1145/325334.325205.

[Fra02] Faure François. Illumination, 3 2002. http://ebook-cours.com/cour/informatique/infogra/illumination.pdf.

[GBP06] Gaël Guennebaud, Loïc Barthe, and Mathias Paulin. Real-time soft shadow mapping by backprojection. pages 227–234, 06 2006. https://doi.org/10.2312/EGWR/EGSR06/227-234.

[GCS94] Steven Gortler, Michael F Cohen, and Philipp Slusallek. Radiosity and relaxation methods. *IEEE Computer Graphics and Applications*, 14(6):48–58, 1994. https://doi.org/10.1109/38.329094.

[Hei89] Tim Heidmann. Real shadows real time. *Iris Universe*, 18:28–31, 1989.

[HHM18] Eric Heitz, Stephen Hill, and Morgan McGuire. Combining analytic direct illumination and stochastic shadows. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 1–11, 2018. https://doi.org/10.1145/3190834.3190852.

[HLHS03] Jean-Marc Hasenfratz, Marc Lapierre, Nicolas Holzschuch, and François X Sillion. A survey of real-time soft shadows algorithms. *Computer Graphics Forum*, 22(4):753–774, 2003. https://doi.org/10.1111/j.1467-8659.2003.00722.x.

[IKSB13] Ismahafezi Ismail, Hoshang Kolivand, Mohd Shahrizal Sunar, and Ahmad Hoirul Basori. An overview on dynamic 3d character motion techniques in virtual environments. *Life Science Journal*, 10(3), 2013.

[JC95] Henrik Wann Jensen and Niels Jørgen Christensen. Photon maps in bidirectional monte carlo ray tracing of complex objects. *Computers & Graphics*, 19(2):215–224, 1995. https://doi.org/10.1016/0097-8493(94)00145-O.

[JC00] Henrik Wann Jensen and Niels Jørgen Christensen. A practical guide to global illumination using photon maps. *SIGGRAPH 2000 Course Notes CD-ROM*, 2000.

[Kaj86] James T Kajiya. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 143–150, 1986. https://doi.org/10.1145/15922.15902.

[Kil01] Mark J Kilgard. Robust stencil shadow volumes. *CEDEC Presentation, Tokyo*, 115:116–119, 2001.

[Kil02] Mark J Kilgard. Shadow mapping with todays opengl hardware. *SIGGRAPH 2002 Course*, 6, 2002.

[KMK94] Daniel Kersten, Pascal Mamassian, and David C Knill. Moving cast shadows and the perception of relative depth. 1994.

[KMK97] Daniel Kersten, Pascal Mamassian, and David C Knill. Moving cast shadows induce apparent motion in depth. *Perception*, 26(2):171–192, 1997. https://doi.org/10.1068/p260171.

[KS12] Hoshang Kolivand and Mohd Shahrizal Sunar. An overview on base real-time shadow techniques in virtual environments. *Telkomnika*, 10(1):171, 2012. http://dx.doi.org/10.12928/telkomnika.v10i1.775.

[KSA+15] Hoshang Kolivand, Mohd Shahrizal Sunar, Ayman Altameem, Amjad Rehman, and Mueen Uddin. Shadow mapping algorithms: Applications and limitations. *Applied Mathematics and Information Sciences*, 9(3):1307–1315, 2015. http://dx.doi.org/10.12785/amis/090325.

[LAA+05] Samuli Laine, Timo Aila, Ulf Assarsson, Jaakko Lehtinen, and Tomas Akenine-Möller. Soft shadow volumes for ray tracing. In *ACM SIGGRAPH 2005 Papers*, pages 1156–1165. 2005.

[Lam82] JH Lambert. Photometria sive de mensura de gratibus luminis, colorum et umbrae. eberhard klett, augsburg, 1760. *W. Engleman, Lambert's Photometrie. Leipzig*, 127(128):2, 1982.

[Lef05] Sylvain Lefebvre. *Modèles d'habillage de surface pour la synthèse d'images.* PhD thesis, 2005.

[Lew94] Robert R Lewis. Making shaders more physically plausible. In *Computer Graphics Forum*, volume 13, pages 109–120. Wiley Online Library, 1994. https://doi.org/10.1111/1467-8659.1320109.

[LFTG97] Eric PF Lafortune, Sing-Choong Foo, Kenneth E Torrance, and Donald P Greenberg. Non-linear approximation of reflectance functions. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 117–126, 1997. https://doi.org/10.1145/258734.258801.

[LGQ+08] D Brandon Lloyd, Naga K Govindaraju, Cory Quammen, Steven E Molnar, and Dinesh Manocha. Logarithmic perspective shadow maps. *ACM Transactions on Graphics (TOG)*, 27(4):1–32, 2008. https://doi.org/10.1145/1409625.1409628.

[LGYF11] Shen Li, Gael Guennebaud, Baoguang Yang, and Jieqing Feng. Predicted Virtual Soft Shadow Maps with High Quality Filtering. *Computer Graphics Forum*, 30(2), 4 2011. https://doi.org/10.1111/j.1467-8659.2011.01875.x.

[LSL11] Andrew Lauritzen, Marco Salvi, and Aaron Lefohn. Sample distribution shadow maps. In *Symposium on Interactive 3D Graphics and Games*, pages 97–102, 2011. https://doi.org/10.1145/1944745.1944761.

[LSO07] Aaron E Lefohn, Shubhabrata Sengupta, and John D Owens. Resolution-matched shadow maps. *ACM Transactions on Graphics (TOG)*, 26(4):20–es, 2007. https://doi.org/10.1145/1289603.1289611.

[LTY+06] Brandon Lloyd, David Tuft, Sung-eui Yoon, Dinesh Manocha, et al. Warping and partitioning for low error shadow maps. In *Rendering Techniques*, pages 215–226, 2006. http://dx.doi.org/10.2312/EGWR/EGSR06/215-226.

[McG17] Morgan McGuire. Computer graphics archive.(july 2017), 2017. http://casualeffects.com/data/index.html.

[MKK98] Pascal Mamassian, David C Knill, and Daniel Kersten. The perception of cast shadows. *Trends in cognitive sciences*, 2(8):288–295, 1998. https://doi.org/10.1016/S1364-6613(98)01204-2.

[MM07] Schwarz Michael and Stamminger Marc. Bitmask soft shadows. *Computer Graphics Forum*, 26(3 (Proceedings of Eurographics 2007)):515–524, 9 2007. https://doi.org/10.1111/j.1467-8659.2007.01074.x.

[ON95] Michael Oren and Shree K Nayar. Generalization of the lambertian model and implications for machine vision. *International Journal of Computer Vision*, 14(3):227–251, 1995. https://doi.org/10.1007/BF01679684.

[Pas05] Mignot Pascal. Initiation à la synthèse d'images, 2005.

[PB07] Bernard Péroche and Dominique Bechmann. *Informatique graphique, modélisation géométrique et animation*. 2007.

[Pho75] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, 1975. https://doi.org/10.1145/360825.360839.

[PY06] Chatelier Pierre Y. *Une approche de la radiosité par voxels,application à la synthèse d'images.* PhD thesis, Auvergne University, Computer Science, 12 2006.

[Rad08] Ingo Radax. Instant radiosity for real-time global illumination. *Institute of Computer Graphics and Algorithms, Vienna University of Technology. https://old. cg. tuwien. ac. at/research/publications/2008/radax-2008-ir/radax-2008-ir-paper. pdf [Accessed February 2016]*, 2008.

[RDGK12] Tobias Ritschel, Carsten Dachsbacher, Thorsten Grosch, and Jan Kautz. The state of the art in interactive global illumination. In *Computer Graphics Forum*, volume 31, pages 160–188. Wiley Online Library, 2012. https://doi.org/10.1111/j.1467-8659.2012.02093.x.

[RSC87] William T Reeves, David H Salesin, and Robert L Cook. Rendering antialiased shadows with depth maps. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 283–291, 1987. https://doi.org/10.1145/37401.37435.

[SEA08] Erik Sintorn, Elmar Eisemann, and Ulf Assarsson. Sample based visibility for soft shadows using alias-free shadow maps. In *Computer Graphics Forum*, volume 27, pages 1285–1292. Wiley Online Library, 2008. https://doi.org/10.1111/j.1467-8659.2008.01267.x.

[Ske12] Sketchfab community : To publish and find 3d content online, 2012. https://sketchfab.com/feed.

[SMSW12] Michael Schwärzler, Oliver Mattausch, Daniel Scherzer, and Michael Wimmer. Fast accurate soft shadows with adaptive light source sampling. In *Proceedings of the 17th International Workshop on Vision, Modeling, and Visualization (VMV 2012)*. Citeseer, 2012. http://dx.doi.org/10.2312/PE/VMV/VMV12/039-046.

[SS98] Cyril Soler and François X Sillion. Fast calculation of soft shadow textures using convolution. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 321–332, 1998. https://doi.org/10.1145/280814.280927.

[SSMW09] Daniel Scherzer, Michael Schwärzler, Oliver Mattausch, and Michael Wimmer. Real-time soft shadows using temporal coherence. In *International Symposium on Visual Computing*, pages 13–24. Springer, 2009. https://doi.org/10.1007/978-3-642-10520-3_2.

[Tak09] Nawel Takouachet. Utilisation de critères perceptifs pour la déterminatin d'une condition d'arrêt dans les méthodes d'illumination globale. Littoral, 2009.

[Tom12] Barak Tomas. *Global Illumination via Instant Radiosity*. PhD thesis, Czech Technical University in Prague, 5 2012.

[TS67] Kenneth E Torrance and Ephraim M Sparrow. Theory for off-specular reflection from roughened surfaces. *Josa*, 57(9):1105–1114, 1967. https://doi.org/10.1364/JOSA.57.001105.

[Wan92] Leonard Wanger. The effect of shadow quality on the perception of spatial relationships in computer generated imagery. In *Proceedings of the 1992 symposium on Interactive 3D graphics*, pages 39–42, 1992. https://doi.org/10.1145/147156.147161.

[War92] Gregory J Ward. Measuring and modeling anisotropic reflection. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 265–272, 1992. https://doi.org/10.1145/142920.134078.

[Whi79] Turner Whitted. An improved illumination model for shaded display. In *Proceedings of the 6th annual conference on Computer graphics and interactive techniques*, page 14, 1979. https://doi.org/10.1145/358876.358882.

[Wil78] Lance Williams. Casting curved shadows on curved surfaces. In *Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, pages 270–274, 1978. https://doi.org/10.1145/800248.807402.

[WPF90] Andrew Woo, Pierre Poulin, and Alain Fournier. A survey of shadow algorithms. *IEEE Computer Graphics and Applications*, 10(6):13–32, 1990.

[WSP04] Michael Wimmer, Daniel Scherzer, and Werner Purgathofer. Light space perspective shadow maps. *Rendering Techniques*, 2004:15th, 2004.

[ZBNK14] Abd El Mouméne Zerari, Med Babahenini, Bahi Naima, and Sarra Khemliche. Ombres douces avec source de lumière surfacique. 12 2014.

[Zer11] Abd Al Mouméne Zerari. *VOLUME D'OMBRE EN RENDU TEMPS RÉEL*. PhD thesis, Image Synthesis and Artificial Life option, Université Mohamed Khider Biskra, 2011.

[Zer18] Abd Al Mouméne Zerari. *Occultation ambiante basée sur l'échantillonnage préférentiel multiple pour la génération des ombres douces en temps réel*. PhD thesis, Université Mohamed Khider Biskra, 2018.

[ZSXL06] Fan Zhang, Hanqiu Sun, Leilei Xu, and Lee Kit Lun. Parallel-split shadow maps for large-scale virtual environments. In *Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications*, pages 311–318, 2006.