Compression, Modeling, and Real-Time Rendering of Realistic Materials and Objects

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

des

Fachbereichs Mathematik und Informatik

der Philipps-Universität Marburg

vorgelegt von

Dipl.-Inform. Nicolas Menzel

Marburg, März 2011

Philipps-Universität Marburg
Fachbereich Mathematik und Informatik
Hans-Meerwein Straße 3, 35032 Marburg

Compression, Modeling, and Real-Time Rendering of Realistic Materials and Objects

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

des

Fachbereichs Mathematik und Informatik

der Philipps-Universität Marburg

vorgelegt von

Dipl.-Inform. Nicolas Menzel

Marburg, März 2011

Philipps-Universität Marburg
Fachbereich Mathematik und Informatik
Hans-Meerwein Straße 3, 35032 Marburg

Angefertigt mit Genehmigung des Fachbereichs Mathematik und Informatik der Philipps-Universität Marburg

Dekan: Prof. Dr. Manfred Sommer

1. Referent: Prof. Dr. Michael Guthe, Philipps-Universität Marburg
2. Referent: Prof. Dr. Carsten Dachsbacher, Karlsruher Institut für Technologie

To my wife,
Tina,
and my children,
Till and Alba.

Acknowledgments

The presented work has been produced within the scope of the *AG Grafik und Multimedia* at the Institute of Computer Science of the University Marburg. At this point I like to thank all people who were directly or indirectly involved in the creation of this work.

My thanks belong primarily to Prof. Michael Guthe whose ideas and support made this work possible. Dear Michael, not only have you tought me the fundamentals of computer graphics - now I also know the importance of skiing abilities for obtain a phD, and – of course – the right ingrediants to make a delectable "'willy"'.

I also like to thank Prof. Manfred Sommer and my second adviser, Prof. Carsten Dachsbacher, whose early works have been an important inspiration to me ever since I wrote my very first computer programs (instead of doing my homework after school).

Also, I would like to thank my friends and colleagues Mischa Dieterle, Stefan Jurack, Gerd Wierse, and Egbert Fohry. Special thanks belongs to Evgenij Derzapf for many discussions, good ideas and useful advices.

Last, but not least, I like to thank my working colleague Friedemann Rößler for constantly reminding me to finally finish my thesis!

Zusammenfassung

Der Realismus einer Szene basiert im Wesentlichen auf der Qualität der Modelle, der Beleuchtung und der verwendeten Materialien. Mittlerweile existieren zahlreiche Methoden für die Erstellung und Messung dreidimensionaler Modelle. Ebenso gibt es sehr effiziente Algorithmen für die Approximation von globaler Beleuchtung. Die Messung und Darstellung realistischer Materialien bleibt jedoch ein schwieriges Problem, für das bis heute keine einheitliche Lösung existiert.

Materialien sind für die Computergrafik unentbehrlich, denn sie beschreiben das Reflektanzverhalten von Oberflächen. Dieses Verhalten wird durch die Interaktion von Licht und Materie bestimmt. In der realen Welt existiert eine enorme Vielfalt von Materialien mit den verschiedensten Eigenschaften. Das Ziel der Computergrafik ist es, diese in all ihrer Vielfalt zu erfassen, zu formalisieren und zu simulieren.

Zu diesem Zweck existieren bereits analytische Repräsentationen, die auf der Verteilung von sogenannten Mikrofacetten beruhen. Diese sind jedoch aufgrund der hohen Zahl der Koeffizienten nur schwer zu parametrisieren. Eine alternative Vorgehensweise ist die Messung von Materialien durch herkömmliche Fotografien. Der Nachteil hierbei besteht jedoch im hohen technischen Aufwand zur Aufnahme und den großen resultierenden Datenmengen. Zwar existieren Kompressionsalgorithmen auf Basis statistischer Verfahren, diese erlauben jedoch keine nachträglichen Änderungen, beispielsweise der Farbe oder der Oberflächenstruktur. Kern dieser Arbeit ist eine Materialrepräsentation, welche gemessene Materialien einerseits komprimiert, andererseits aber auch nachträgliche Änderungen von bestimmten Eigenschaften erlaubt.

Dieser Ansatz basiert nicht ausschließlich auf Kompression, sondern auf einer Zerlegung der Oberfläche in mehrere Klassen von Materialien. Jede Klasse besitzt dabei unterschiedliche Eigenschaften. Auf dem Microfacetten-Modell beruhend, wird das Reflektanzverhalten dabei durch eine Funktion dargestellt, welche als herkömmliche zweidimensionale Textur gespeichert werden kann. Neben Tiefeninformationen werden zusätzlich die lokalen Rotationsparameter und die diffuse Farbe gespeichert. Da durch diese Zerlegung auch Informationen verloren gehen, werden im weiteren Verlauf der Arbeit Ansätze vorgestellt, diese verloren gegangenen Informationen nachträglich wieder analytisch hinzuzufügen.

Ein weiterer Beitrag dieser Arbeit ist eine wahrnehmungsbasierte Metrik

zur Simplifizierung von Dreiecksnetzen. Das Neuartige an dieser Metrik ist, dass sie das Material des Dreiecksnetzes berücksichtigt. Dies ist dadurch möglich, dass die Eigenschaften des menschlichen Sehsystems, beispielsweise die reduzierte Farbwahrnehmung oder räumliche Auflösung, mit einbezogen werden. Die vorgestellte Metrik erlaubt eine aggressivere Vereinfachung der Geometrie in Bereichen, in denen existierende geometrische Metriken zu schwach vereinfachen.

Abstract

The realism of a scene basically depends on the quality of the geometry, the illumination and the materials that are used. Whereas many sources for the creation of three-dimensional geometry exist and numerous algorithms for the approximation of global illumination were presented, the acquisition and rendering of realistic materials remains a challenging problem.

Realistic materials are very important in computer graphics, because they describe the reflectance properties of surfaces, which are based on the interaction of light and matter. In the real world, an enormous diversity of materials can be found, comprising very different properties. One important objective in computer graphics is to understand these processes, to formalize them and to finally simulate them.

For this purpose various analytical models do already exist, but their parameterization remains difficult as the number of parameters is usually very high. Also, they fail for very complex materials that occur in the real world. Measured materials, on the other hand, are prone to long acquisition time and to huge input data size. Although very efficient statistical compression algorithms were presented, most of them do not allow for editability, such as altering the diffuse color or mesostructure. In this thesis, a material representation is introduced that makes it possible to edit these features. This makes it possible to re-use the acquisition results in order to easily and quickly create deviations of the original material. These deviations may be subtle, but also substantial, allowing for a wide spectrum of material appearances.

The approach presented in this thesis is not based on compression, but on a decomposition of the surface into several materials with different reflection properties. Based on a microfacette model, the light-matter interaction is represented by a function that can be stored in an ordinary two-dimensional texture. Additionally, depth information, local rotations, and the diffuse color are stored in these textures. As a result of the decomposition, some of the original information is inevitably lost, therefore an algorithm for the efficient simulation of subsurface scattering is presented as well.

Another contribution of this work is a novel perception-based simplification metric that includes the material of an object. This metric comprises features of the human visual system, for example trichromatic color perception or reduced resolution. The proposed metric allows for a more aggressive simplification in regions where geometric metrics do not simplify

strong enough.

Vorwort

"Was glänzt, ist für den Augenblick geboren;
Das Echte bleibt der Nachwelt unverloren"
- Johann Wolfgang von Goethe (Faust I)

Üblicherweise steht an dieser Stelle etwas über den rasanten Fortschritt moderner Computergrafik, die ständig steigende Rechenleistung der Hardware und den wachsenden wirtschaftlichen Bedarf realistischer, künstlich generierter Bilder. Stattdessen möchte ich mit einer Frage beginnen: *Sind wir Maler*?

Als Computergrafiker können wir Stunden damit verbringen, das Licht einer Kerze, ihren Schattenwurf und die Brechung ihres Lichts durch ein Weinglas hindurch zu simulieren. Dabei stellen wir hohe Ansprüche an uns selbst: Der Farbverlauf der Kerzenflamme muss realistisch erscheinen, der Schattenwurf weich sein und die Bündelung des Lichts exakt der Krümmung, Dichte und Dicke des Glases entsprechen. Dazu nutzen wir unser Wissen über die wahren Vorgänge der Natur: Photonen, kleinste Teilchen von Licht, werden verschossen, reflektiert oder absorbiert, der Verlauf ihrer Flugbahn wird in jedem Punkt des Raumes berechnet bevor sie in räumlichen Datenstrukturen gespeichert und schließlich aufgesammelt werden. Millionen von Berechnungen werden durchgeführt und führen am Ende zu einem Helligkeitswert eines Punktes, dessen Fläche lediglich einem Bruchteil eines Quadratmillimeters entspricht.

Fertig und zufrieden mit unserer Arbeit sind wir erst, wenn das Bild *real* aussieht. Eben wie ein Maler, der seine Staffelei vor einem träumerischen Straßenkaffee, einer sonnengetränkten Naturlandschaft oder einem wunderschönen Modell aufbaut und jeden Pinselstrich mit der Realität abgleicht, so gleichen wir jeden Pixel ab. Der Bildschirm ist unsere Staffelei, die Programmiersprache und die Grafikkarte sind unsere Pinsel und Farbpalette. Was uns mit dem Maler eint ist die Tatsache, dass wir uns nicht damit zufrieden geben das Motiv für den Augenblick zu genießen. Stattdessen möchten wir es reproduzieren, es verstehen, es festhalten.

Sind wir Maler? Haben wir Pinsel und Farben durch moderne Technologie ersetzt? Sind wir einfach nur angepasst an ein neues Zeitalter und betreiben ein digitales "Malen nach Zahlen"?

Was wir mit Malern teilen, ist unser Antrieb und unser Wille, die Vorgänge der Natur zu durchschauen. Wenn ein Maler ein Motiv erblickt, so möchte er dieses festhalten. Die Leinwand dient dazu, das Gesehene auf Papier zu bannen und es über den Augenblick hinaus zu konservieren. Aber der Maler möchte auch *verstehen*: Durch die Art und Weise, wie er die Farben zu einer Komposition verarbeiten muss um ein Abbild der Realität zu erschaffen, lernt er die Vorgänge nachzuvollziehen. Dabei ist er auf sein handwerkliches Geschick, seine Augen und seine Vorstellungs- und Wahrnehmungskraft angewiesen.

Das Erzeugen eines realistischen Computerbildes beherbergt viele der Vorgänge, die auch beim Zeichnen eines Bildes von Bedeutung sind: Wir können die komplexen Vorgänge der Natur am besten *verstehen*, indem wir sie in Programmcode umsetzen. Wir können Schönheit *festhalten*, indem wir sie als Pixel auf den Bildschirm bannen. Digitale Technologie erlaubt uns außerdem, sie zu *reproduzieren*, zu verfielfältigen, zu manipulieren und die Realität *noch schöner* zu machen. Denn oft ist es erst die Überzeichnung, die uns in Erstaunen versetzt.

Als Goethes Faust in seiner Studierstube die Bilanz seines Lebens zieht, kommt er zu einem niederschmetterndem Fazit: Trotz seiner lebenslangen Studien ist er weit davon entfernt erklären zu können, *was die Welt im Innersten zusammenhält*. Vielmehr erkennt er, dass wir nichts wissen *können*, ein philosophisches Dilemma der fehlenden Beobachterperspektive. *Zwar weiß ich viel, doch möcht ich alles wissen*, fasst er das Bestreben eines Wissensschaftlers zusammen, bevor er sich Mephisto verspricht und am Ende seines Lebens erkennt, dass beim Streben nach dem höchsten Dasein sein Glück in wohltätiger Arbeit für andere Menschen besteht, eine Anlehnung an den philosphischen Kulturalismus.

Wie kommt man von Computergrafik über Malerei schließlich zu Faust? Als Erklärung sei eine Reiseschilderung Goethes aus dem Jahr 1788 gegeben:

> *Ich hatte nämlich zuletzt eingesehen, daß man den Farben, als physischen Erscheinungen, erst von der Seite der Natur beikommen müsse, wenn man in der Absicht auf die Kunst etwas über sie gewinnen wolle. Wie alle Welt war ich überzeugt, daß die sämtlichen Farben im Licht enthalten seien; nie war es mir anders gesagt worden, und niemals hatte ich die geringste Ursache gefunden, daran zu zweifeln.*

Goethe war ein begeisterter Maler.

Contents

Part I.

Introduction

Chapter 1.

Motivation

Before the first graphics cards became available, the infamous Phong shading[1] was the only real-time material affordable on hardware at that time[2]. Unfortunately, the flexibility of this shading model is very limited, as it is controlled by just a few parameters, making it only plausible for plastic-like materials. However, with the processing power and memory capacity of modern graphics cards, it became possible to compute more complex materials. For this purpose, several physically correct surface representations have been introduced: well-known examples are the Bidirectional Reflectance Distribution Function (BRDF), the Bidirectional Surface Scattering Distribution Function (BSSRDF) or the Bidirectional Texture Function (BTF). But for what do we need more sophisticated material representations?

If we look at the real world, we notice that the diversity of materials is absolutely overwhelming. To simulate a material, we have to analyze and understand what physical processes take place in that particular material. Subsurface scattering, for example, occurs for almost every material in nature, especially for organic matter. Another well-known effect is that polished surfaces reflect more light when we look at them at grazing angles. Transparent and translucent objects change the light's direction according to the refractive index. Nacre, an organic material that is produced by sea shells, is one of the most complex materials in nature: it changes the color as the angle of view changes, a phenomenon called *iridescence*. *Phosphorescence*, which exists in synthetic materials, describes when energy absorbed by a substance is released relatively slowly in the form of visible light. It is closely related to *fluorescence*, where light is absorbed and then reflected at a different wavelength. These are just a few examples that demonstrate the complexity of real-world light-matter interaction.

Of course, there is a vast industrial need for more realistic materials: in car design, measured materials are used to simulate the interior, seat covers, and car paint. A realistic simulation of the final product is extremely useful for designers and customers. The simulation of the appearance of cloths is another important area, since fabric exhibits extremely complex light-matter-interaction properties. Realistic materials are also important for flight simulators, as air can be considered as a material as well: atmospheric effects are created by the interaction of light with the participating media, containing particles such as dust

[1] Named after its inventor Bui Tuong Phong in 1973

[2] Of course besides Gouraud shading, which was presented by Henri Gouraud in 1971

and aerosols. For pilots, the color of the sky is an important clue of altitude, weather conditions, distance, and time. Not to mention that realistic materials are basic elements of every video game and computer-generated movie.

In the last years it became common practice to take reflection properties directly from the real-world. For this purpose, special devices for the acquisition of materials were built. Typically, they consist of two gantries, one holding a light source and the other one holding a digital camera[3]. If the camera remains static and only the light source changes its position, the result is a reflectance field. If this is repeated for several view positions, a dense representation of the material's reflection properties for every light and view combination is obtained. Unfortunately, this creates input data up to several gigabytes, which significantly exceeds the capacities of graphics hardware.

To address this problem, several approaches have been introduced to reduce the high dimensional input data. In most cases, statistical methods like principal component analysis (PCA) or matrix factorization have been used. With these methods, the input data can be transformed into a set of ordinary 2D textures in the range of several megabytes. The reconstruction is then computed on the GPU in the fragment shader in real-time. However, these approaches lack of *editability*. Once the materials are compressed, they cannot be altered anymore. This can be compared to a compressed text document: in the compressed representation, it is not possible to change specific letters or words in a meaningful manner. The document has to be decompressed first, then changes can be made, and then it can be compressed again[4]. One of the key ideas presented in this thesis is to perform a decomposition instead of a compression, resulting in a set of materials with very distinct properties. It is then possible to compress each material more efficiently, since only little deviations can be expected in the same material[5]. Unfortunately, one problem of material decomposition is, that some of the reflectance properties are lost, since the *geometric correspondence* can not be guaranteed[6]. This thesis addresses this problem by introducing efficients methods for the simulation of some of the material properties that are not included in the decomposition, such as self-shadowing or subsurface scattering.

In addition to the editable material representation, a perception-based metric for the simplification of meshes with arbitrary materials is presented as well. Simplification is important, because the rendering of models with high polygonal count is still a challenging task in real-time computer graphics. A very common way to improve rendering performance is to generate different levels of detail (LOD) of a model. These are computed by polygonal simplification techniques, which aim to reduce the number of polygons without a significant loss of visual appearance. The contribution of this thesis is a very fast simplification algorithm grounded in the human visual system. The key idea is to change the domain for the error computation from image-space to vertex-space. The advantage is that

[3] e.g. the Stanford Spherical Gantry developed by Marc Levoy.

[4] Note, that in the case of materials, this compression process can last several hours.

[5] e.g. deviations in color, depth etc.

[6] For example color bleeding or shadowing of one pixel onto the other.

the verification of a local simplification operation can be computed much faster compared to existing algorithms.

Chapter 2.

Basics

"What if angry vectors veer,
Round your sleeping head, and form,
There's never need to fear
Violence of the poor world's abstract storm."
- Robert Penn Warren

The basics of this thesis are manifold and from a wide range of areas of computer graphics. They are divided into a theory and an implementation section: in the theory, the principles of high dynamic range imaging (HDRI), the human visual system (HVS), material representations, and mesh simplification are explained. The implementation section is divided into shader programming and a brief discussion about the more general parallel languages Compute Unified Device Architecture (CUDA) and Open Computing Language (OpenCL).

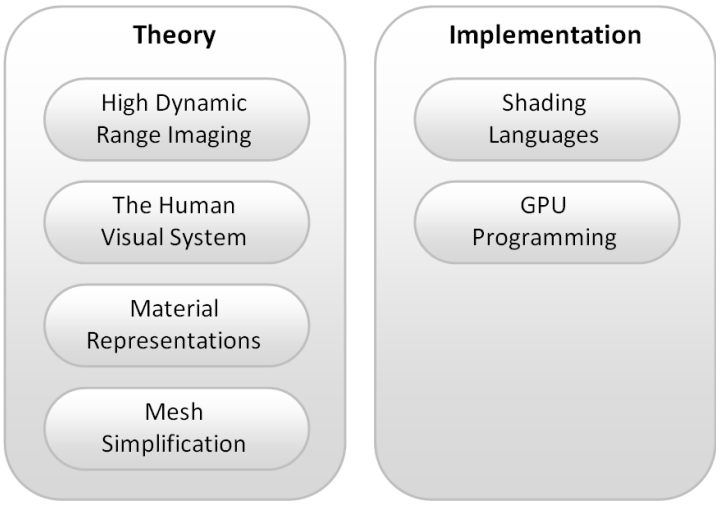

Figure 2.1.: Structure of the Basics chapter.

This chapter begins with high dynamic range imaging, where the quality of display devices, frame buffer representations, and quantities for different levels

of light intensities are presented. After this, the principles of the human visual system are explained, including the structure of the eye, adaptation mechanisms, contrast sensitivity, color perception, and image metrics.This section is important for the visual metric that is presented later in this work. It is followed by a survey of material representations, starting with the rendering equation, the fundamentals of light-matter interaction, and finally the BRDF, the BSSRDF, and the BTF. The theory part ends with a brief overview of the most important simplification algorithms, including vertex clustering, mesh removal, and quadric error metrics.

The implementation section starts with a general introduction to GPU programming and is followed by a closer look at the High Level Shading Language (HLSL), followed by a quick glance at CUDA and OpenCL.

1 High Dynamic Range Imaging

The perceived quality of a displayed image depends on several things: first, the resolution of the display device. The more pixels are available on an area of fixed size, the more details can be shown in relation of the distance of the viewer to the image. Second, the contrast of the display device determines how realistic the lighting of the the image appears. Third, but less important in the context of this thesis, is the temporal resolution of a sequence of images.
This section gives an technical overview of the most important components of realistic image synthesis, such as display resolution, contrast, color precision, gamma correction, and high dynamic range rendering in real-time computer graphics.

1.1. Display Resolution

The display resolution depends on the technical quality of the device and increases ever since the first presentation of the cathode ray tube. Standard display resolution in the year 2010 is 1920x1080[1]. Highest available resolutions are 2560x1600 WQXGA for consumer market liquid crystal displays (LCD) and 3280x2048 WQSXGA for medical diagnostic devices. A high resolution greatly improves the visual quality for the viewer, since it allows to show more detail and reduces aliasing effects. Note, that the display resolution is independent of the resolution on which the image is actually formed.

1.2. Screen Resolution

The screen resolution depends on resolution of the framebuffer in the memory of the graphics device. In television, it depends on the signal of the television provider, which is 768x576 for the European PAL standard and the American NTFS standard and unchanged since the 1950s. The screen resolution of games and applications is continuously growing: these days, it is common to play games in HD.

[1] (High Definition or HD)

1.3. Contrast

As explained in more detail in section 2, the human eye adapts to a great variety of light intensity levels. In an image, the relation of the darkest to the brightest intensity value is called the dynamic range. The real world comprises a dynamic range of nearly twelve magnitudes, reaching from only a few photons to full sunlight. An ideal monitor would emit no light at all if it displays black and it would be very bright if it displays white. In computer graphics, the synthesis of images done in larger dynamic range is called high dynamic range rendering. Video games and computer-generated movies benefit from this as it allows to create more realistic images[2].

1.4. Temporal Resolution

The rate at which images are displayed is measured in frames per second (fps) or Hertz (Hz). At one fps, there is only little sense of interactivity. At around 6 fps, a sense of interactivity starts to grow. An application at 24 fps can be considered as real-time, as the user is able to focus on action and reaction[3]. From about 3×24 fps and up, differences in the display rate are effectively undetectable.

1.5. Integer Color Precision

Traditionally, a pixel is encoded with 24 bits using 8 bits per color channel, which confines lighting integer precision to intensities in the range [0..255]. The color is encoded as an RGB triplet (r, g, b), where each component can vary from zero to a defined maximum value. If all of the color values are zero, the result is black. If all are maximum, the result is the brightest white representable on the display device. For obvious reasons, concerning the dynamic range that the real world provides, 256 intensity levels are not enough.

1.6. Floating-Point Color Precision

When Microsoft released DirectX 9.0, lighting precision was not limited to 8-bit anymore on software side. On the hardware side, the R300 chip of ATI's Radeon 9700 was finally designed for floating point precision with fractional color values in the range [0..1][4]. Modern graphics hardware provides full support for floating point precision framebuffers and textures(including texture filtering).

1.7. Measuring Luminance

The luminance of a device is measured in candela per square metre (cd/m^2). It is defined as the amount of light that passes through or is emitted from a particular area and is defined as

$$L_v = \frac{d^2F}{dAd\Omega cos\Theta}$$

[2]Like the display resolution, the contrast of a display device is technically limited.

[3]This is also the norm of the PAL television standard.

[4]However, the R300 supported only 96-bit (FP24) color precision instead of DirectX 9.0's maximum of 128-bit (FP32).

where L_v is the luminance, F is the luminous flux (or luminous power), Θ is the angle between the surface normal and the specified direction, A is the area of the surface (m^2), Ω is the solid angle in steradian[5] and d is the distance for consideration of the attenuation.
The smallest luminance corresponding to the intensity value $(0, 0, 0)$ displayable by a LCD monitor lies usually between 0.15 to 0.8 cd/m^2, on a CRT monitor below 0.1 cd/m^2. The brightest values lie between 150-500 cd/m^2 on a LCD and between 80-200 cd/m^2 on a CRT. This means that both LCD and CRT devices always have a residual amount of light, even if they display black. Also, they are very limited in the maximum brightness.

1.8. Limitations and Compensations

Modern graphics hardware is able to represent and process color values in full floating-point precision at high resolution in real-time. Yet, no monitoring device on the consumer market is able to show the full gamut and range of an HDR image. Whereas synthesis is performed in HDR and textures can be stored in special HDR file formats, the final image is always *squeezed* to 256 intensity levels.
To compensate for this, algorithms exist that map colors from the HDR image range to a lower dynamic range that matches the capabilities of the desired display device. Such a method is called tone reduction or tone-mapping operator (TMO). Essentially, tone mapping addresses the problem of strong contrast reduction from the scene values to the displayable range. The most simple TMO would be just to clamp all color values above the maximum displayable value (e.g. > 1.0). More sophisticated TMO try to preserve image details and color appearance to appreciate the original scene content. Well-known TMO were presented by Reinhard et al. [RSSF02] and Ashikhmin and Ferwada [AG06, RSSF02].

1.9. Gamma Correction and Linear Color Values

Gamma correction is the practice of applying the inverse of the monitor transformation to the image pixels before they are displayed. That means that if the pixel values are raised to the power $\frac{1}{\gamma}$ before display, then the display implicitly raising to the power gamma will exactly cancel it out, resulting, overall, in a linear response:

$$V_{out} = V_{in}^{\gamma}$$

CRTs do not behave linearly in their conversion of voltages V into light intensities. And LCDs, although they do not inherently have this property, are usually constructed to mimic the response of CRTs. A typical gamma of 2.2 means that a pixel at 50 percent intensity emits less than a quarter of the light as a pixel at 100 percent intensity. Gamma is different for every individual display device, but typically it is in the range of 2.0 to 2.4. Adjusting for the effects of this nonlinear

[5] solid angle in the international system of units

characteristic is called gamma correction.
Note that renderers, shaders, and compositors operate with linear data in most of the cases. They sum the contributions of multiple light sources and multiply light values by reflectances. Unfortunately, if the pixel's numerical value is doubled, the CRT, LCD, or similar device display will not display a pixel twice as bright. The nonlinearity is subtle enough that it is often unintentionally or intentionally ignored, particularly in real-time graphics, which makes it worth noting in the context of this thesis.

1.10. HDR in Real-Time Graphics

HDR Rendering has several advantages compared to traditional 8-bit rendering. First of all, light sources with very different intensities can be used. A dim candle in a dark room can as well be represented as the sun[6]. It is barely possible to model this relation with 256 intensity levels, especially because surfaces reflecting the sunlight (e.g. shiny or wet surfaces, water) will have high luminance values after reflection as well.
A second advantage are spatial filtering operations such as Gaussian blur, motion blur or other filter types: for a normalized kernel and 8-bit precision, a surface that has the maximum intensity of 1.0, will have an intensity < 1.0 if only one neighbored pixel has a value < 1.0. This means that filter operations directly influence the dynamic range, which is not intended by the programmer. This is no longer a problem when HDR and floating point render targets are supported.

[6]The sun's luminance on a bright day is $100,000$ times higher than a candle.

2 The Human Visual System

The human visual system can be divided into the following components: the eyes - consisting of the lens, iris, the retina and the optic nerve - and the visual pathways in the brain, along which these signals are transmitted and processed. The following sections explain the biological fundamentals of the human vision, which are essential for the development of the perceptual metric presented later in this work.
Note, that the HVS is based on biological and psychological processes which are not yet fully understood. A so called perceptual model, which is used to quantify visual quality, is just an approximation of the real processes occurring in the eye and brain and simulates the behavior of this very complex system. A typical perceptual model includes features such as

- Light adaption
- Lack of color-resolution
- Masking
- Low-pass filter characteristics
- Motion sensitivity

For the development of a perception-based metric, it is important to know the principles of human vision, therefore the most relevant elements are introduces in this section. A very comprehensive review can be found in [Win05].

2.1. The Eye

The eye serves as sensor for electromagnetic radiation. We see things because each object has its specific electromagnetic spectrum, which is the characteristic distribution of radiation that is emitted, absorbed or reflected by that particular object. The eye only sees radiation in the range $380 - 750$ nm, with ultra-violette (UV) at the shorter end and infra-red (IR) radiation at the longer end. Working very similar to the principles of a photographic camera, the eye comprises a system of lenses and a variable aperture to focus images on the retina. The optics of the eye rely on the physical principles of refraction, as a beam of light changes its direction as it is transmitted through media with different refractive indices.

2.2. The Pupil

The pupil is the visible part of the lens. The size of the pupil is controlled by the iris muscle. It is able to regulate the amount of light over a 10 log unit range, changing its diameter from approximately 8.0 mm down to about 1.5 mm. The pupil is only able to produce a little more than a log unit change, so pupillary action alone is not sufficient for visual adaptation. More than adaptation, the pupil's size serves to mitigate the visual consequences of aberrations in the eye's optical system. At high levels where there is plenty of light to see by, the pupil

stops down to limit the effects of these aberrations. At low levels where catching enough light to allow detection is more essential than optimizing the resolution of the retinal image, the pupil opens to allow more light into the eye.

2.3. The Retina

The retina consists of a large number of photoreceptor-cells, which are arranged in several layers which the light has to pass before it is finally absorbed in the pigment layer, located at the very back of the retina. A small area on the retina is called the fovea: it is responsible for sharp central vision. The human fovea has a diameter of about 1.0 mm and contains a very high number of photoreceptor cells. The high spatial density of cones accounts for the high visual acuity in the fovea.

2.4. Photoreceptors

The photoreceptors are specialized neurons that contain a particular protein called rhodopsin. Two types of opsin are involved in vision, namely rods and cones. Rods are responsible for scotopic vision at low light levels, and cones are responsible for photopic vision at high light levels. When both the rod and cone systems are active, this is called mesopic range. Rods are found primarily in the periphery of the retina. Though rods sample the retina very finely, their acuity under scotopic conditions is poor, because signals from many rods converge onto a single neuron, which improves sensitivity but reduces resolution. There are between 75 and 150 million rod and 6 to 7 million cone photoreceptors in each retina.
There are three types of cones that differ in wavelength of light they absorb. These three types are referred to as L-cones, M-cones and S-cones, according to their sensitivity to long, medium and short wavelengths. They are used primarily to distinguish color and other features of the visual world at normal levels of light.

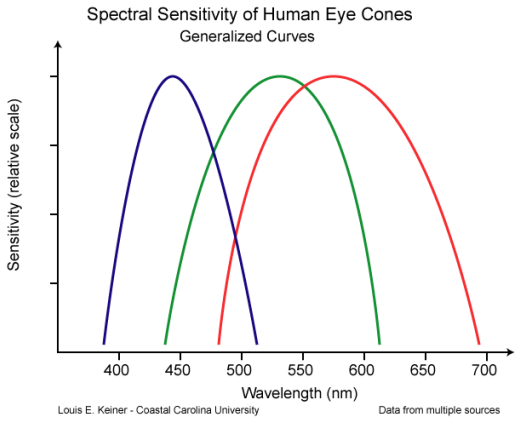

Figure 2.2.: Sensitivity levels of the three cone types.

As shown in Figure 2.2, peak sensitivities occur around 440 nm, 540 nm, and 570 nm. The absorption spectra of the L- and M-cones are very similar, whereas

the S-cones exhibit a significantly different sensitivity curve[7].

2.5. Light Adaptation

The range of light energy we experience is vast: the light of the noonday sun can be as much as 10 million times more intense than moonlight. Figure 2.3 displays the range of luminances we encounter in the natural environment and summarizes some visual parameters associated with this luminance range.
The way our visual system copes with this huge range of luminances is by changing its sensitivity, depending on the luminance prevailing in the visual field. This process is called adaptation. During the adaptation, the system becomes accustomed to processing higher or lower light levels in its environment than it was exposed to before. The human visual system is capable of adapting to an enormous range of light intensity levels. Adaptation allows us to better discriminate relative luminance variations at every light level. The HVS provides three different systems to control adaptation:

- **Pupillary aperture**: The pupil diameter can be varied between 1.5 mm and 8 mm, corresponding to a 30-fold change of the light quantity.
- **Chemical processes in the photoreceptors**: Existing in both cones and rods, this process changes the concentration of photochemicals in the receptors[8].
- **Adaptation at neural level**: This mechanism involves all neurons in all layers of the retina. While neural adaptation is less powerful, it is faster than adaptation in the photoreceptors.

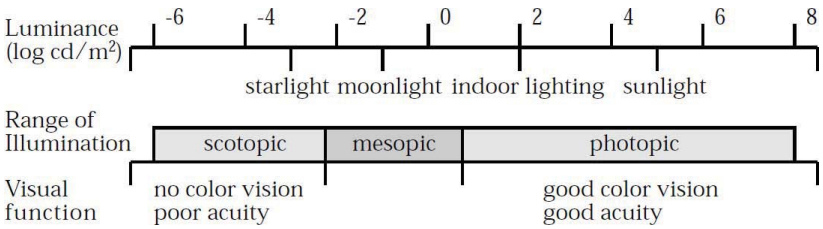

Figure 2.3.: The range of luminances in the natural environment and associated visual parameters. From Ferweda et al. [FPSG96].

Although adaptation provides visual function over a wide range of ambient intensities, this does not mean that we see equally well at all intensity levels. For example, under dim conditions our eyes are very sensitive and are able to detect even small differences in luminance. However, at low lighting conditions, our acuity for pattern details and the ability to detect colors are poor.

[7] The overlapping regions are essential to discriminate fine color differences.
[8] This adaptation mechanism is rather slow, as it may take up to an hour.

2.6. Contrast Sensitivity

Much like the contrast of display devices, the contrast in the HVS describes the ratio between the color of an object and its background. The higher the contrast, the more distinguishable is the object from the background. Contrast is depending on both color and luminance. Because the human visual system is more sensitive to contrast than to absolute luminance, we are capable of perceiving the world regardless of the huge changes in illumination. This property is known as the Weber-Fechner law. Mathematically, Weber contrast is expressed as

$$C^W = \frac{\Delta L}{L}$$

where ΔL is the difference between the foreground object and the background, and L is the luminance of the background.

2.7. Color Perception and Trichromacy

The color matching experiment by Brainard [Bra95] showed that human color vision is based on trichromacy, this means that any color can be expressed by the linear equation

$$t = Cx$$

where t is a three-dimensional vector whose coefficients are the intensities of the three primary lights, x is the color to be matched, and N is a matrix, where the rows consist of N samples of the so-called color-matching-functions as investigated by [Bay87].

2.8. Disadvantages of Pixel-based Metrics

The mean square error (MSE) is the mean of the squared differences between the gray-level values of pixels in two pictures I and I':

$$MSE = \frac{1}{XY} \sum_x \sum_y [I(x,y) - I'(x,y)]^2$$

for pictures of size $X \times Y$. The root mean square error (RMS) is simply $RMS = \sqrt{MSE}$. For measuring image quality, an MSE of zero means that two images match with perfect accuracy.
The peak signal-to-noise ratio, abbreviated PSNR, is defined as

$$PSNR = 10 \log \frac{m^2}{MSE}$$

where m is the maximum possible pixel value of an image[9]. When the two images are identical, the MSE will be zero, resulting in an infinite PSNR.

[9] e.g. 255 for 8-bit images.

Both MSE and PSNR are widely used, since they are intuitive and their computation is very easy and fast. But because they only perform a pixel-by-pixel comparison of images, they only have a limited, approximate relationship with the actual visual distortion. Therefore, they only poorly represent the way the human visual system perceives the difference between two images. Under certain conditions, the PSNR can even be improved by adding noise to the image. The reason for this is that distortions are often much more disturbing in relatively smooth areas, whereas distortions is textured regions are not taken into account by pixel-based metrics. Therefore, the perceived quality of images with the same PSNR can be very different.

3 Material Representations

The physically correct simulation of the light-matter interaction is infeasible in practice. Therefore, much effort has been spent on developing material representations that describe the reflection properties of a possibly wide range of materials with a few parameters only. Ideally, these representations are also computational inexpensive to be rendered in real-time.
This section begins with a short look at the rendering equation, which describes the interaction of light with a surface, including indirect lighting and reflection properties. In the next section, the different types of light interaction prevailing in the real world are discussed in detail. After this, the BRDF, the BSSRDF, the BTF, and the d-BRDF are explained.

3.1. The Rendering Equation

The radiance of an infinitesimal surface point is defined by the rendering equation

$$L_r(\lambda, \mathbf{x}, \omega_r) = L_e(\lambda, \mathbf{x}, \omega_r) + \int_{\Omega^+} \rho(\lambda, \mathbf{x}, \omega_r, \omega_i) L_i(\lambda, \mathbf{x}, \omega_i) \cos\Theta_i d\omega_i$$

where L_r is the outgoing radiance at surface point $\mathbf{x}$ into direction ω_r. The equation has to be solved for each wavelength λ separately to account for effects like chromatic dispersion or to simulate spectral rendering[10]. The emissive power L_e is only greater than zero if the point $\mathbf{x}$ lies on a light source. Otherwise, the incoming radiance depends on the integral of all incoming light directions ω_i over the positive hemisphere Ω^+. The term $L_i(\lambda, \mathbf{x}, \omega_i)$ represents the irradiance coming from direction ω_i. Note, that for the computation of L_i, the rendering equation has to be solved again for each incoming light direction ω_i. Due to this nesting, the rendering equation cannot be solved analytically. Finally, the reflectance function ρ represents the material at this particular surface point, i.e. it describes the way the surface reflects the light. The rendering equation was first introduced by Kajiya [Kaj86].
However, there are some properties not included in the rendering equation, such as phosphorescence[11] and subsurface scattering[12].

3.2. Light Matter Interaction

In general, when a light ray hits a surface, a complicated light-matter dynamic occurs. This interaction depends on the physical characteristics of the light as well as the physical composition and characteristics of the surface. Some of these processes are illustrated in Figure 2.4.

Basically, three types of interaction can occur:

- Reflection

[10] as required for correct atmospheric scattering, for example

[11] Phosphorescence is the time-shifted re-emission of the radiation that is absorbed by the material.

[12] For the simulation of subsurface scattering, the integral has to be computed not only over all directions of incoming light, but also over an area A.

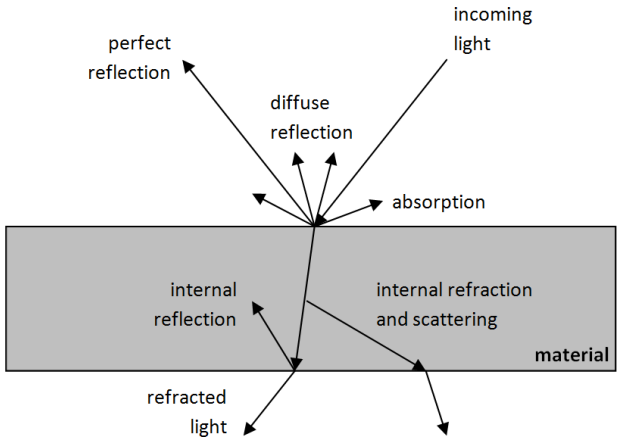

Figure 2.4.: Interaction of light with a material.

- Absorption
- Transmittance

As shown in Figure 2.4, the the portion of light that is not absorbed can either be reflected perfectly or diffusely into all directions of the positive hemisphere. It can as well penetrate the material, following the direction of the refracted ray according to the law of Snellius. For grazing angles, reflection occurs instead of refraction, a phenomenon described by the Fresnel equation. Both of these effects depend on the light's wavelength, i.e. the red, green, and blue potions of the light have different directions after the transformation. Finally, light can be reflected by small particles in the material, causing it to leave at an arbitrary surface position. This effect occurs for materials such as milk, wax or human skin.
Some very basic materials are shown in Figure 2.5: one of the simplest materials models only describes only diffuse reflection (top left), defined by the dot product between the surface normal and the light vector. The light intensity can be computed per vertex[13] or per pixel. Another well-known material is the Phong lighting model [Pho75], which adds a specular highlight to the diffuse reflection (top right). Both of these materials have the advantage that they can easily be rendered in real-time. This is not the case for perfect or diffuse reflection (bottom left) and refraction (bottom right), which are computationally more expensive.

These reflection types are just a small subset of the variety of effects that occur in the real world. Currently, no existing material representation is able to comprise all of them. Therefore, it is left to the graphics programmer to observe real-world materials and translate the light-matter interaction into source code. This process can be described in three elemental steps:

1. **Analysis**: Observation of real-world behaviors, knowledge of the actual physical processes.

[13] Faster in general.

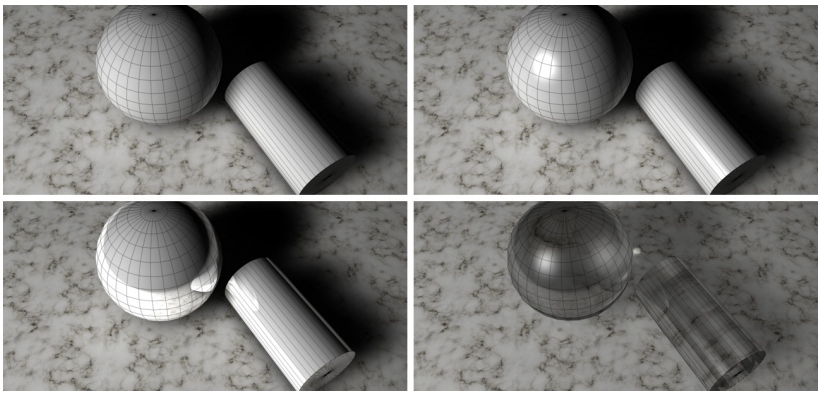

Figure 2.5.: Basic materials types: diffuse, specular, reflective, and refractive.

2. **Formalization**: The light-matter interaction has to be transformed into a set of mathematical rules and equations.

3. **Simulation**: The formalization has to be simulated by source code, which is then executed on the CPU or the GPU.

In the case of real-time graphics, the simulation must also be done highly efficient to meet performance constraints, whereas offline renderers (e.g. used in CG movies) may spend hours for the computation of a single image.

3.3. The BRDF

The BRDF [NRH+92] is a weighting function that describes how light is reflected when it makes contact with a surface, depending on the physical characteristics of light as well as the physical composition and characteristics of the matter. Consequently, a BRDF is a function of incoming light and outgoing view direction to a local orientation at the light interaction point. Additionally, when light interacts with a surface, different wavelengths of light may be absorbed, reflected and transmitted to varying degrees depending upon the physical properties of the material itself. This means that the BRDF is also a function of wavelength. Considering these aspects, a BRDF can be written as:

$$\rho(\omega_o, \omega_i, \mathbf{x}, \lambda)$$

where $\omega_o = (\Theta_o, \Phi_o)$ and $\omega_i = (\Theta_i, \Phi_i)$ are spherical angles and defined over the positive hemisphere of point $\mathbf{x}$. BRDFs can be classified into two classes: isotropic and anisotropic BRDFs. The latter class exhibits variant reflection properties when rotated around the normal. Furthermore, two important properties of the BRDF are reciprocity

$$\forall x, y : \rho(\omega_o, \omega_i) = \rho(\omega_i, \omega_o)$$

and conservation of energy

$$\forall x, y, \omega_i : \int_{\Omega_o} \rho(\omega_o, \omega_i) d\omega_o \leq 1$$

The sources of BRDF data are manifold: phenomenological models (e.g. Phong) have intuitive parameters and are the most used BRDF in real-time computers graphics. Physical models like Cook-Torrance [CT82] simulate the behavior based on a microfacette model.

3.4. The BSSRDF

One major flaw of the BRDF is the assumption that light entering a material leaves at the same position. This approximation is valid for metals, but fails for translucent materials, which exhibit significant transport below the surface, as shown in Figure 2.6.

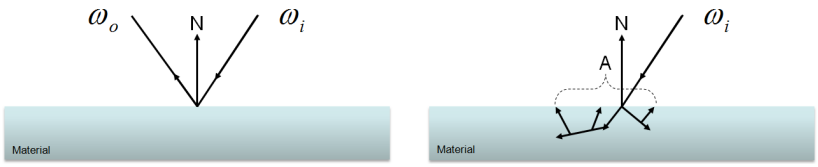

Figure 2.6.: Reflection properties of the BRDF (left) and the BSSRDF (right).

To simulate translucent materials such as milk, wax or skin, Jensen et al. [JMLH01] introduced the more general Bidirectional Subsurface Scattering Distribution Function or BSSRDF, which can describe light transport between any two rays that hit a surface. The outgoing radiance L_o at point x_o into direction ω_o of the BSSRDF is defined as

$$L_o(x_o, \omega_o) = \int_A \int_{2\pi} S(x_i, \omega_i, x_o, \omega_o) L_i(x_i, \omega_i)(n\omega_i) d\omega_i dA(x_i)$$

where S is the BSSRDF integrated over all surface positions x_i of the surface area A and all incoming lighting directions ω_i.

3.5. The BTF

The BRDF and the BSSRDF have several disadvantages: since they describe reflectance on a micro-scale only, they cannot capture the full complexity of inhomogeneous materials at meso-scale, such as self-occlusion and inter-reflections. Materials with spatially varying properties can be faithfully described by the Bidirectional Texture Function or BTF which was first introduced by Dana et al. [DvGNK99]. Like the BRDF, the BTF is a function of six dimensions:

$$L_o(u, v, \theta_i, \phi_i, \theta_o, \phi_o)$$

Here, θ_i, ϕ_i, θ_o, and ϕ_o are spherical angles and u, v is the position on the parameterized surface.

The acquisition of the BTF is a very simple process which can be performed using a standard off-the-shelf-camera and an image-processing software. On the contrary, the acquisition of BTFs requires a complex and controlled measurement environment. As BTF acquisition is physical measurement of real-world reflection, special attention has to paid to the device calibration and image registration. Rendering directly from this data via linear interpolation is impractical even on todays graphics hardware. Therefore, compression methods have to be applied to the data before rendering.

Figure 2.7 shows the relation of processing power and memory consumption in proportion to realism. For this purpose, the surface functions Phong Model, the BRDF, the BSSRDF, and the BTF are plotted.

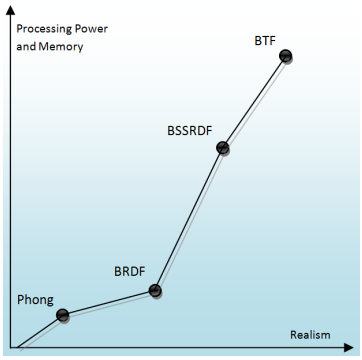

Figure 2.7.: The relation of realism to processing power and memory consumption.

3.6. The d-BRDF

The distribution-based BRDF (d-BRDF) introduced by Ashikhmin [Ash06] plays a very elemental role for the material representation that is presented later in this work. It is based on the assumption that the surface consists of a large number of small facettes as first proposed by Cook and Torrance [CT82].

$$\rho(\mathbf{k}_1, \mathbf{k}_2) = \frac{cp(\mathbf{h})F(\mathbf{kh})}{(k_1 n) + (k_2 n) - (k_1 n)(k_2 n)}$$

Here, $p(h)$ is the so called Normal Distribution Function (NDF), that simulates the distribution of microfacettes at microscale. The constant c is an RGB scaling factor, and h is the halfway vector. The denominator is a continuous function for simulation of masking and shadowing as shown in Figure 2.8. The Fresnel function can be computed using Schlick's approximation:

$$F(\mathbf{kh}) = r_0 + (1 - r_0)(1 - (\mathbf{kh}))^5$$

where r_0 is a user-defined value that controls the minimum reflectance. The most useful property of the d-BRDF is, that the NDF function can be stored as

a two-dimensional bitmap. This can be achieved by a mapping of the halfway-vector from three to two dimensions. The dimensional reduction can be done with spherical or paraboloid mapping.

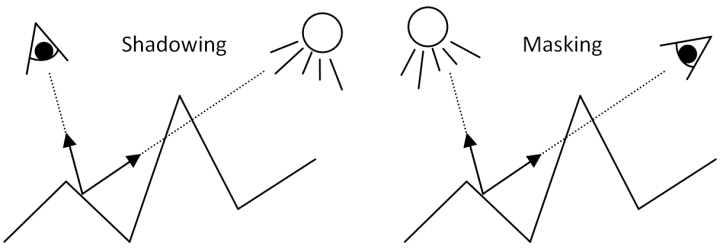

Figure 2.8.: Shadowing and masking in the d-BRDF representation.

4 Brief Survey of Simplification Algorithms

Currently, polygonal models dominate interactive computer graphics. Graphics cards are specialized hardware for the rendering and rasterization of polygonal models. While NURBS[14] or subdivision surfaces define an object analytically, polygonal models are discrete linear approximations of a surface. In most cases, an approximation with more polygons is visually more appealing than an approximation with just a few polygons. This is true especially for the silhouettes of three-dimensional objects, as shown in figure 2.9.

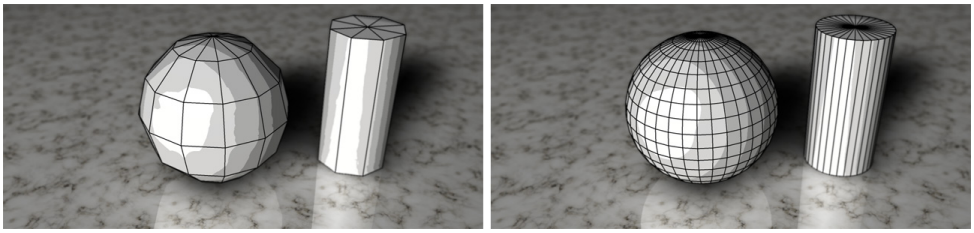

Figure 2.9.: Comparison of silhouette quality. Left: a sphere with 72 polygons and a cylinder with 24 polygons. Right: a sphere with 648 polygons and a cylinder with 96 polygons.

Often, a polygonal model is referred to as a mesh. In its general form, a mesh is identical to an undirected graph $G = (V, E)$, where V is the set of nodes and E is the set of edges, which are 2-element subsets of V. In a mesh, a node is called a vertex. Two connected vertices define an edge. Three vertices, connected to each other by three edges, define a triangle, which is the simplest polygon in Euclidean space. Note, that a model of more complex polygons can always be reduced to a set of triangles by triangulation algorithms. Sometimes, a polygon is also called a face.
Mesh simplification often requires collapsing an edge into a single vertex. This operation requires deleting the faces bordering the edge and updating the faces which shared the vertices at the end points of the edge. This step requires to discover adjacency relationships between components of the mesh, such as the faces and the vertices. While these operations certainly can be implemented on the simple mesh representation mentioned above, they will most likely be costly. Many will require a search through the entire list of faces or vertices, or possibly even both.
To implement these types of adjacency queries efficiently, more sophisticated boundary representations (b-reps) have been developed which explicitly model the vertices, edges, and faces of the mesh with additional adjacency information stored inside.
The half-edge data structure is a slightly more sophisticated b-rep which allows all of the queries listed above to be performed in constant time. In addition, even though adjacency information are included in the faces, vertices and edges, their

[14] Non-Uniform rational B-Spline

size remains fixed (no dynamic arrays are used) as well as reasonably compact. These properties make the half-edge data structure an excellent choice for many applications, however it is only capable of representing manifold surfaces, which in some cases can prove prohibitive. Mathematically defined, a manifold is a surface where every point is surrounded by a small area which has the topology of a disc. For the purpose of a polygon mesh, this means that every edge is bordered by exactly two faces; t-junctions, internal polygons, and breaks in the mesh are not allowed.
In the remainder of this section, a brief survey of simplification algorithms is given. A good survey is also found in [Lue01].

4.1. Vertex Clustering

Vertex clustering was first proposed by Rossignac and Borrel [dLK93]. First, a 3D grid is laid over the object. Next, an importance is assigned to each vertex, depending on curvature and size of adjacent triangles. Then all vertices inside a grid cell are collapsed to the vertex with the highest importance. When a triangle becomes degenerated, it is replaced by a line, merging them together when several lines connect the same two vertices. When a line is collapsed it is replaced by a point and again merged with points at the same position. In this algorithm the grid resolution determines the trade-off between quality and reduction rate.

4.2. Simplification Envelopes

Cohen et al. developed simplification envelopes [CVM+96] to guarantee fidelity bound while enforcing local and global topology preservation. The simplification envelopes consist of two offset surfaces at some distance ϵ from the original surface. Since these envelopes are not allowed to self intersect, ϵ is decreased at high curvature regions. By keeping the simplified surface inside these envelopes, the algorithm can guarantee a geometric deviation of at most ϵ. Additionally, surface self-intersections are prevented during simplification.

4.3. Quadric Error Metrics

The quadric error metric simplification algorithm [GH97] provides perhaps the best balance between speed, fidelity and robustness. The algorithm works by iteratively merging pairs of vertices which do not necessarily need to be connected by an edge. Candidate pairs include all edges plus all vertices closer than a user specified threshold t. The major contribution of this algorithm was the way to represent the error and calculate a new vertex position using a quadric. Another advantage besides speed and quality is that the algorithm also performs topological simplification and therefore does not require a manifold input mesh. Since the number of candidate pairs approaches $O(n^2)$, as t approaches the model size, Erikson and Manocha proposed an adaptive threshold selection scheme [EM98] to improve performance. A comparison between Vertex Clustering and Quadric Error Metrics is shown in Figure 2.10.

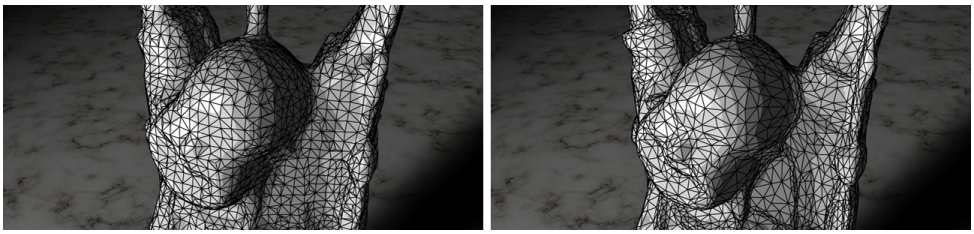

Figure 2.10.: Comparison between Vertex Clustering (left) and Edge Collapse (right) for the Buddha model with 27,303 triangles: Vertex Clustering simplifies without respect to curvature and spends too many triangles in flat areas (such as the forehead).

4.4. Appearance Preserving Simplification

Cohen, Olano and Manocha extended the simplification envelopes and introduced an appearance-based simplification algorithm [COM98] by using a fidelity error term, which is based on maximum screenspace deviation. This means that the simplification's appearance when rendered should must not deviate from the original appearance by more than a user-defined number of pixels. Attributes affecting the appearance are surface position, surface color and surface curvature.

4.5. Progressive Meshes

A progressive mesh is a representation of a polygonal model as sequence of collapse operations. It was introduced as the first dynamic simplification algorithm for manifold polygon meshes by Hoppe [Hop96]. The progressive mesh is composed of a simple base mesh created by a series of edge collapses and the sequence of vertex split operations necessary to reconstruct the original model. Although the original progressive mesh used only edge collapses, adding other collapse operations is possible. The constructed sequence encodes the simplification process from the model to the base mesh. Since the edge collapse and vertex split operation can be performed relatively fast, it is possible to compute them during runtime. Later Hoppe extended the progressive meshes to support view-dependent simplification [Hop97].

4.6. Geometric Error

The distance $d(p, S')$ between a point p and a surface S and another surface S' is defined as:

$$d(p, S') = \min_{\forall p' \in S'} d(p, p')$$

where $d(p, p')$ is the Euclidean distance between two points in E^3. The geometric distance - also called one-sided or single-sided Hausdorff distance - between two surfaces S and S' is then defined as

$$D(S, S') = \min_{\forall p \in S} d(p, S')$$

Note, that this distance is not symmetric in general, i.e. $D(S, S') \neq D(S', S)$. Therefore, the (symmetrical) Hausdorff distance is defines as

$$\mathcal{H}(S, S') = max(D(S, S'), D(S', S))$$

This value gives a more accurate measure of the distance between two surfaces by preventing the possible underestimation, which can occur if using only one-sided distances.

4.7. Disadvantages of the Geometrical Error

Quadric error metrics can be extended by taking the Hausdorff distance into account. This has the advantage, that silhouettes with very high visual quality can be preserved. Unfortunately, the same error bound must be used for all regions of the object, even for regions where a more aggressive simplification is possible. For this purpose, a perception-based error metric is introduced in part 3 of this thesis.

5 Programming the GPU

In the early nineties, most computer graphics hardware was hard-wired to the specific tasks of vertex and fragment processing. This means that algorithms to rasterize, shade and draw triangles were fixed within the hardware. Even though these hard-wired graphics algorithms could be configured by graphics applications in a variety of ways, reprogramming the hardware was not possible.
Fortunately, as the graphics hardware advanced, the vertex and fragment processing units in recent GPUs became programmable. Before programmable hardware was available, no programming language existed. Therefore, the only available way to address the capabilities of the GPU was low-level assembly language. A shading language makes it much easier to program GPUs. The following sections give an overview of the languages:

- High Level Shading Language (HLSL)
- Compute Unified Device Architecture (CUDA)
- Open Compute Language (OpenCL)

5.1. Shading Languages in General

A shading language specifically targets graphics hardware, as it is specialized to control the shape, appearance, and motion of objects. These languages are different from conventional programming languages, because they are based on a data-flow computational model. In such a model, computation occurs in response to data that flows through a sequence of processing steps, or stream. The most widely used shading languages are currently Cg (NVidia), HLSL (Microsoft) and GLSL (OpenGL ARB). All three languages have in common that they are compiled down to assembly code. Cg code can be compiled to fragment code for different platforms, HLSL compiles directly to DirectX and GLSL compiles natively.

5.2. CPU and GPU Architecture

In 2010, the consumer market for CPU is dominated by multicore processors containing two to four cores integrated onto multiple dies in a single chip package. The user benefits from this development, since several tasks can be performed concurrently, accelerating tasks like video processing or rendering.
At the same time, graphics hardware has become more flexible due to programmability. The GPU has evolved into a highly parallel, multithreaded, many-core processor with tremendous computational power and very high memory bandwidth. In contrast to a multi-purpose CPU, the GPU is specialized for compute-intensive, highly-parallel computation, such as computer graphics. Therefore, the GPU's architecture is devoted more to data processing rather than data caching and flow control.
The reason why the GPU operates so much faster than the CPU is, that CPUs are designed to get the maximum performance from a stream of instructions,

which operates on diverse data. This data can consist of integers, floats, pointers and strings in a random access manner. Though multiple instructions can be executed in parallel by a modern CPU, the problem is that there is a limit to the parallelism that is possible to get out of a sequential stream of instructions.
In contrast to the CPU, memory access in a GPU is extremely coherent: when a texel is read, a few cycles later the neighboring texel is read. When the memory is organized intelligently, performance comes close to the theoretical bandwidth. As a result, the GPU does not need an enormous cache. Table 2.1 shows that for sequential data, the GPU outperforms the CPU by more than three times.

Proc	Cache	Sequential	Rand
GPU	45	20	3
CPU	44	6	1

Table 2.1.: Comparison of GPU and CPU memory access in gbyte/sec

5.3. Limits of Parallelism

The amount of performance gained by the use of multi-core processors is strongly dependent on the algorithm and the implementation. Generally, the speedup is determined by the fraction of the software that can be parallelized to run on multiple cores simultaneously. This effect is described by Amdahl's law, which says that in the best case the speedup factors are near the number of cores, also called embarrassingly parallel.

5.4. HLSL

The High Level Shading Language or HLSL was developed by Microsoft for the use with the Microsoft Direct3D API. It is almost identical to the Cg shading language. A typical HLSL program consists of

- A **vertex shader** that performs per-vertex processing such as transformations, skinning, vertex displacement, and calculating per-vertex material attributes. As a minimum, a vertex shader must output vertex position in homogeneous clip space. Optionally, the vertex shader can output texture coordinates, vertex color, vertex lighting, fog factors, and so on.
- A **geometry shader** that performs per-primitive processing such as material selection and silhouette-edge detection, and can generate new primitives for point sprite expansion, fin generation, shadow volume extrusion, and single pass rendering to multiple faces of a cube texture.
- A **pixel shader** that performs per-pixel processing such as texture blending, lighting model computation, and per-pixel normal and/or environmental mapping. Pixel shaders work in concert with vertex shaders; the output of a vertex shader provides the inputs for a pixel shader. In Direct3D 9 some pixel operations (such as fog blending, stencil operations, and render-target blending) occur after the pixel shader is finished.

Since DirectX 11, the rendering pipeline has been extended by the following, hardware-accelerated tessellation features:

- **Hull Shader**: Receives patch control points, outputs patch control point after basis conversion
- **Tesselator**: Converts control points to topology (primitive assembly)
- **Domain Shader**: Receives final control points and creates domain points with displacement.

These three stages are located before the geometry shader. As shown in Figure 2.11, the new tesselation feature greatly improves the visual quality at relatively low computational cost. This is because not the full geometry has to pass the vertex shader.

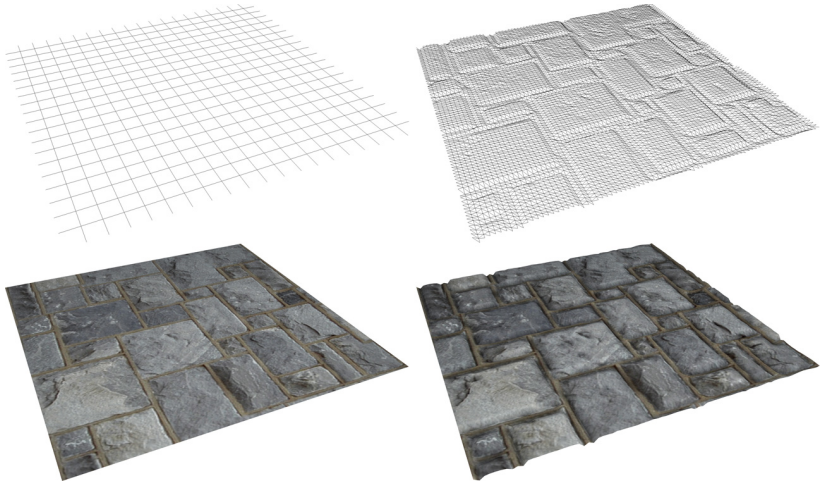

Figure 2.11.: Tesselation at runtime on DirectX11 hardware.

A typical HLSL program must contain at least one *technique*, which is addressed by the application. A technique must consist of at least one *pass*. If more than one pass is defined (*multi pass*), the passes are processed in sequential order. Optionally, the DirectX Standard Annotations and Semantics (DXSAS) can be used directly in the shading language, as shown in Listing 2.1. The main purpose of DXSAS is that shaders can be used in a standard way in a variety of applications, tools and game engines.

Listing 2.1: HLSL Basic Structure

```
technique DX9x
{
    // optional: DXSAS instructions
    pass p0
    {
        //...
    }
}
```

Listing 2.2 shows the definition of the vertex- and pixel shaders. In DirectX 9, they are generated using the *compile* keyword, followed by the shader model (SM). Note that in DirectX 9, the highest available model is SM 3.0. Renderstates such as enabling the depth buffer or backface culling can be set in the pass. The second part of the listing shows that these definitions heavily changed in DirectX 10. The compile keyword and the setting of the render states was replaced by specific commands. Also, a new keyword for the definition of the geometry shader is available.

Listing 2.2: Defining Vertex- and Pixel Shaders in D3D9 and D3D1x

```
technique D3D9x
{
    pass p0
    {
        ZEnable = true;
        Cullmode = none; // CW, CCW
        VertexShader = compile vs_3_0 VertexShader( );
        PixelShader = compile ps_3_0 PixelShader( );
    }
}

technique D3D10
{
    pass p0
    {
        SetDepthStencilState( EnableDepth, 0 );
        SetVertexShader( CompileShader( vs_4_0, VertexShader( ) ) );
        SetGeometryShader( NULL );
        SetPixelShader( CompileShader( ps_4_0, VertexShader( ) ) );
    }
}
```

5.5. CUDA

CUDA is the acronym for *Computing Unified Device Architecture* and was first presented by NVidia in early 2007. CUDA works with all NVidia GPUs from G8X onwards.

The CUDA-Toolkit and the CUDA-SDK offer an extended C-Compiler and several programming interfaces. The C-Compiler does not only understand conventional C-Code, but also some additional commands and so called *device code*. Written in the well-known C-syntax, device code can be transmitted and executed directly on the GPU. The C-compiler also understands so called *host-code*, which is executed on the CPU.

Functions that contain device-code always begin with the keyword __global__. These function are also called *kernel*. Recursion is not allowed in kernels. Inside a kernel function, only functions declared with the __device__ keyboard may be called. Such function calls are treated like macros due to inline-expansion. On the GPU, a kernel is executed in parallel as a *thread*.

A *thread* is a basic processing element of data. A *warp* is a group of 32 threads, which is the minimum size of the data processed in SIMD fashion. Warps cannot be manipulated directly, but have to be arranged in *blocks* that can contain 64 to 512 threads. Finally, these blocks are put together in *grids*, which are

automatically broke down by the CUDA runtime, depending on the underlying hardware: if the device has only few resources, grids are computed sequentially. If the hardware has a very large number of processing units, grids are processed in parallel, making CUDA scalable for future GPUs.

Listing 2.3: A CUDA kernel

```
30 __global__ void thresholdKernel( float* g_data, int width, float thres )
31 {
32     unsigned int x = blockIDx.x * blockIDx.x + threadIdx.x;
33     unsigned int y = blockIDx.y * blockIDx.y + threadIdx.y;
34
35     unsigned int index = y * width + x;
36
37     if( g_data[ index ] >= thres )
38     {
39         g_data[ index ] = 1.0f;
40     }
41     else
42     {
43         g_data[ index ] = 0.0f;
44     }
45 }
```

In Listing 2.3 computes a threshold operation on a two-dimensional array. The implicit variables in lines 3 and 4 are used to address the elements in the array. Another restriction is, that device-code can only access device-memory, that is memory on the graphics hardware. Host-code, instead, can allocate memory, release and change memory on the graphics hardware. Note that direct manipulation of the device memory, e.g. with device_array[0] = 0;, is not possible in the host-code.

The use of multiple blocks is required, since CUDA only allows 32 to 512 threads in a single block. All threads in a particular block share resources, such as memory or registers. Too many threads in a block result in a slow parallel execution or even parallelization is not possible. These and other limitations are listed in the *CUDA Programming Guide* and are hardware-independent. In the Guide, several levels of capabilities, called *compute capabilities*, are defined. They usually depend on the hardware that is used.

The most significant hardware enhancement is support for shared memory atomics. One simple application of that would be counting the number of characters in a string, as used for Huffmann encoding for example. Another new feature is the ability to run CUDA code as highly efficient SSE-based multithreaded C code on the CPU.

Listing 2.4: CUDA Host-Code

```
1  void executeThreshold( float* hostArray, int width, int height, float thres )
2  {
3      float* deviceArray;
4      int arraySize = width * height * sizeof( float );
5      cudaMalloc( ( void** )&deviceArray, arraySize );
6
7      // Copy: Host --> Device
8      cudaMemcpy( deviceArray, hostArray, arraySize, cudaMemcpyHostToDevice );
9
10     dim3 blocks( width / 16, height / 16, 1 );
```

```
11      dim3 threads( 16, 16, 1 );
12      thresholdKernel<<< blocks, threads >>>( deviceArray, width, thres );
13
14      // Copy: Device --> Host
15      cudaMemcpy( hostArray, deviceArray, arraySize, cudaMemcpyDeviceToHost );
16
17      cudaFree( deviceArray );
18  }
```

In line 6 of the example code memory is allocated on the graphics device. The C array passed as a parameter is then copied into this memory in line 9. The following lines describe the block-/thread layout: in this case it is assumed, that the width and height of the image are multiples of 16. When the kernel is called in line 13, 256 (16x16x1) threads work in concurrently in a block. After this, the content of the array if copied back to the memory of the host (line 16) and the memory on the graphics card is released (line 18). The kernel-call is *synchronous*, meaning that the execution of the host-code is paused as long as the kernel is working.

5.6. OpenCL

OpenCL, standing for *Open Computing Language*, was initially developed by Apple in 2008, but then submitted to the Khronos Group, an industry consortium that develops and maintains open standards, with Apple still holding trademark rights. Being supported by AMD, IBM, Intel and NVidia, OpenCL serves as a framework for programs running across heterogeneous platforms consisting of CPUs, GPUs and other types of processors.

OpenCL's main purpose is to use all computational resources in the system by supporting a data- and task-parallel compute model. OpenCL's execution model consists of a kernel, a program and an application queue. Kernels are executed across a global domain of *work-items*. Work-items are arranged into local *workgroups*, which share local memory and synchronization. Workitems must be independent and do not allow for a global synchronization.

In contrast to CUDA, which is restricted to NVidia hardware, OpenCL can be used by AMD cards as well as on other hardware.

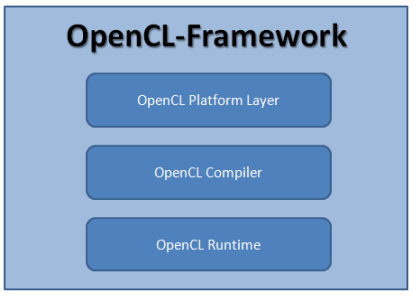

Figure 2.12.: Components of the OpenCL framework

The most important components of the OpenCL framework are the OpenCL Platform Layer, the OpenCL Compiler and the OpenCL Runtime Library 2.12.

The task of the Platform Layer is to handle different hardware with different capabilities. Once the hardware is recognized, the platform layer is able to create a so called *context*, which is explained in more detail later. The OpenCL Runtime is used to manage the context object and provides an API to create kernel-queues, memory- and program objects and kernels. The Runtime layer also manages the communication between host and device memory. The third and last component, the OpenCL Compiler, is responsible for the transformation of the shading language into executable assembly code.

5.6.1. Context and Command Queues

The context is created by the host and defines an environment to run kernels. No kernel can be executed without a proper context [Khr09], which consists of:

- *Devices*: The collection of OpenCL devices to be used by the host
- *Kernels*: The OpenCL functions that run on OpenCL devices.
- *Program Objects*: The program source and executable that implement the kernels.
- *Memory Objects*: A set of memory objects visible to the host and the OpenCL devices. Memory objects contain values that can be operated on by instances of a kernel.

The context is created and manipulated by the host using functions from the OpenCL API. The host creates a data structure called *command-queue* to coordinate execution of the kernels on the devices. The host places commands into the command-queue which are then scheduled onto the devices within the context. These include:

- *Kernel execution commands*: Execute a kernel on the processing elements of a device.
- *Memory commands*: Transfer data to, from, or between memory objects, or map and unmap memory objects from the host address space.
- *Synchronization commands*: Constrain the order of execution of commands.

The command-queue schedules commands for execution on a device.

- *In-order Execution*: Commands are launched in the order they appear in the command-queue and strictly in-order. In other words, a prior command on the queue completes before the following command begins. This serializes the execution order of commands in a queue.
- *Out-of-order Execution*: Commands are issued in order, but do not wait to complete before following commands execute. Any order constraints are enforced by the programmer through explicit synchronization commands.

Kernel execution and memory commands submitted to a queue generate event objects. These are used to control execution between commands and to coordinate execution between the host and devices.

Part II.

Realistic Materials

Chapter 3.

Previous Work

As shown in Figure 3.1, the main goal of this thesis is the real-time rendering of realistic materials. This means that the material representation has to be powerful enough to simulate even complex materials, but must also be simple enough to be computational in real-time. Furthermore, the representation should also be editable, which means that it must be possible for the user to change the appearance and behavior[1].

There are basically two fundamental classes of material representations: first are the analytical models as shown on the right side of Figure 3.1. Typically, the appearance of analytical models is controlled by only a few intuitive parameters. The Phong model is probably the most famous and popular analytical model, since its appearance can be controlled only by diffuse color, specular color and the size of the specular highlight. Unfortunately, analytical representations are restricted to a relatively small subset of materials and do not cover the diversity that is found in the real world.

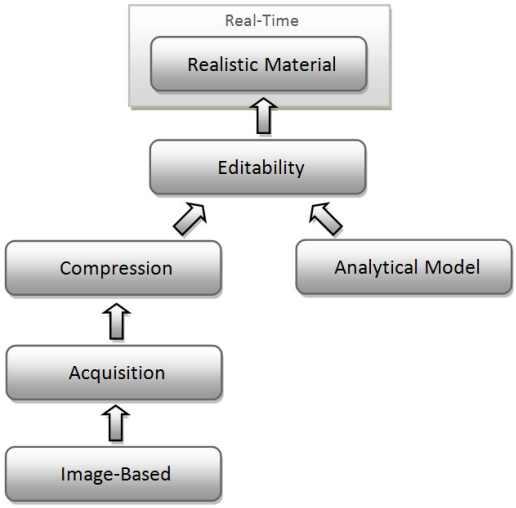

Figure 3.1.: Structure of the material representation chapter.

The second class are the image-based representations, which rely on materials that exist and that we observe in the real world. As shown on the left side

[1]By using standard image processing software, in the best case.

of Figure 3.1, there are two important issues in image-based representations: acquisition describes the technical and physical process in which the input images are obtained. Since this step usually results in several gigabytes of input data, compression is necessary to reduce the size of the data.

This part of the thesis covers acquisition, compression and rendering of realistic materials. Therefore, the previous work section covers a wide range of related work: first, the high dynamic range image compression is described, including surface parameterizations and statistical compression methods. This section is followed by two acquisition methods, covering image alignment for hand-taken photographs and the decomposition of the Bidirectional Texture Function. Finally, an efficient algorithm for the simulation of subsurface scattering is presented.

1 High Dynamic Range Image Compression

The efficiency of a compression method depends significantly on the way the data is represented. Therefore, the previous work about high dynamic range image compression is divided into two sections: first, different parameterizations of surfaces are presented. Then, an overview of the most relevant compression algorithms is given.

1.1. Surface Parameterizations

As the ability of the graphics hardware to display more and more triangles per frame is constantly increasing, the complexity of scenes and models used in real-time applications grows ever on and on. As a result, the manual design of realistic environments becomes very time-consuming and finally unfeasible. Possible solutions to this problem are procedural models or (semi-)automatic generated geometry.

Image-based rendering and modeling (IBRM) presents another solution to address this issue. Normally, input images are acquired using either 2D or 3D imaging or capture devices, before they are stored in a sample database [CBCG02][2]. The method proposed here is strongly correlated with the reflectance field introduced by Debevec et al. [DHT+00]. A non-local reflectance field is described as a set of light fields, denoted as

$$R_r(\mathbf{x}, \theta_r, \phi_r)$$

where R_r is the radiance exiting an object at surface point $\mathbf{x} = (u_r, v_r)$ into the direction θ_r, ϕ_r to the observer. A non-local reflectance field then is defined as

$$R(\mathbf{x}, \theta_r, \phi_r, \theta_i, \phi_i)$$

where θ_i, ϕ_i is the direction to the light source. An efficient method to render the sample database in real-time was introduced by Chen et al. [CBCG02]. They

[2] A good overview of different methods to cope with the representation of such a sample database is given in [MMS+05]

propose to partition a light field into

$$f_j^v(r,s,\theta,\phi) = \Lambda_j^v(r,s) f(r,s,\theta,\phi)$$

where f_j^v is a small area, the so called vertex light field (VLF), around the j-th vertex, sampled for a discrete set of outgoing directions (i.e. point of views). Due to small rotation angles, these sampled areas are highly coherent, an observation described by Nishino et al. [NSI99] before. After statistical compression and quantization, several VLFs are packed together and stored as texture map. Once loaded into the memory of a rendering device, they can be accessed extremely fast, making it possible to reconstruct the VLF in real-time. While this work originally focused on light fields only, it can also be used for local reflectance fields.

The conceptual difference between a BTF and a reflectance field is the same as between light fields and surface light fields: while reflectance fields are parameterized over screen space, BTFs are parameterized over the surface of an object and thus can be used in the same way as textures.

1.2. Statistical Compression

Statistical compression has been widely adopted in IBRM. Originally it has been introduced by Kautz and McCool for the compression of BRDFs [KM99, Kau99, Wyn01]. They proposed to restate image compression as a problem of matrix factorization. For this purpose, the 4D domain of a (possibly continuous) BRDF has to be translated into the discrete 2D domain of a matrix. Non-specular BRDFs can be fully reconstructed using normalized decomposition, essentially being a 1-term truncated PCA. More complex BRDFs contain diagonal structures, which do not allow to reconstruct the BRDF with a few terms only[3].

Kautz and McCool also showed that after quantization, the principal components can be stored as regular bitmaps allowing to use hardware-accelerated texture mapping for efficient reconstruction.

As the amount of data is especially huge for high dimensional representations like non-local reflectance fields and BTFs, several methods have been proposed to increase the covariance between pixels or texels. The pixels are either grouped in clusters of high covariance [MMK03] or a per-pixel local coordinate system is calculated such that the inter-pixel covariance is maximized [MSK06].

The principal component analysis has the disadvantage of evoking negative values that cannot be stored directly in a common texture on older graphics hardware. For this reason the homomorphic factorization was introduced by McCool [MAA01] and extended by Latta [LK02]. Instead of a partial sum, the matrix is decomposed as partial product containing only positive values. The homomorphic factorization has the disadvantage that it is non-progressive, meaning that for $N+1$ terms it has to be completely recomputed. Note, that the problem of negative values became more or less obsolete due to the programmability of modern rendering devices and their ability to store floating point textures.

[3] To a certain degree, the problem of diagonal structures can be addressed by the half-angle difference-angle Gram-Schmidt reparameterization as shown by Wynn [Wyn01].

2 Image Alignment

For the acquisition of materials and reflection properties, ordinary cameras mounted on special acquisition devices or gantries can be used. Currently, the dynamic range of most digital digital cameras is confined to 256 intensity levels. This is not enough for specular materials, where it is necessary to sample the input data in high dynamic range to preserve the original reflectance. For this purpose, a sequence of images with different exposures can be taken. If the images are taken manually without a tripod, they have to be aligned before they can be converted to a high dynamic range image.

For the creation of HDR images based on a sequence of different exposure levels, the algorithm of Debevec and Malik [DHT+00] is the most widely used. In their work, the dynamic range of an image is recovered by a weighted average of multiple input images, each one taken with a different exposure time. While the dynamic range of lambertian objects does not justify the use of HDR images, specular or reflecting objects provide a dynamic range that exceeds the capabilities of low dynamic range photographs. Although the weighting of the images has been improved by Robertson et al. [RBS99], the basic principle is still the same.

Initially, Debevec and Malik proposed using multiple exposures to recover the full dynamic range [DM97] of a scene. Their proposal arises from the observation that if one pixel has twice the value of another, it cannot be concluded that it received twice the irradiance. Instead, the irradiance of a pixel is determined by an unknown, nonlinear function called the response function or response curve. Once the response curve is reconstructed, it can be re-used for the same camera type. However, as the response curve is badly conditioned at extreme intensities, Debevec and Malik introduced a linear weighting function to assure that values near saturation have a lower contribution to the final image. The HDR image is then computed as a weighted average of each pixel, thus containing information captured by each of the images. Later, Robertson et al. introduced a smoother Gaussian weighting function [RBS99] that better fits the accuracy range of a CCD sensor.

The generation of HDR images from exposure sequences requires perfectly aligned pictures since the intensity is calculated on a per pixel basis. If this is not the case, the pictures need to be correctly aligned before HDR reconstruction. A fast and robust method for image alignment was developed by Ward [War03]. Based on a median cut in combination with an XOR fitness function, it is able to robustly remove translations between the images in the sequence. Grosch extended this method to rotations [Gro06] using a downhill simplex solver and exploiting graphics hardware to improve performance. The drawback of this method is that, despite its robustness to moving objects, only an alignment of static parts of the scene can be performed. The remaining artifacts, originating from object movement or parallax, are removed in a second, semi automatic phase at the cost of a reduced dynamic range.

The ghost removal algorithm proposed by Khan [KAR06] improves the weighting function by not only considering the probability that a pixel is correctly ex-

posed, but also the probability that the pixel is part of the background. For this purpose an estimation scheme, based on a d-variate Gaussian density function, is introduced. For each pixel in every input image a small neighborhood is considered to determine the contribution to the final image. The results of the proposed algorithm claim that no further local alignment is required.

Mere ghost removal, however, cannot replace the process of image alignment, because the dynamic range is removed from those areas where only a single image contributes to the final image. The differences between ghost removal and image alignment cannot always be seen at first glance, but become visible through the loss of highlights[4].

If not only the static background, but also moving foreground objects are to be aligned, a single linear transformation is not sufficient. A similar non-linear matching problem needs to be solved for state-of-the-art video codecs like MPEG-4 [WSBL03]. In this case, so called macroblocks from one image need to be found in a corresponding image. This matching problem is typically solved using optical flow techniques [BFBB94, BBF94].

In the context of temporal image processing, Kang [KUWS03] presents a method to create a sequence of HDR images from a stream of LDR input images. For this purpose, the input images are captured with alternating exposure times using a programmable capturing device. To assure that the number of output images is equal to the number of input images, neighboring images have to be aligned, combined and tone mapped. Subsequent images are registered by estimating an affine transform that maps one onto the other. A gradient-based optical flow is used to compute a dense motion field to form a local correlation.

The presented approach is comparable to [KUWS03] in the sense that always three neighbored images are registered. Instead of an affine transform macroblocks are used for the alignment. For motion estimation, a very accurate and – with respect to different luminance – robust methods is the cross-correlation [Lew95]. Another possibility to solve the non-linear alignment based on optical flow would be the use of dense gradient matching [CDR02, SDR04]. However, this technique only works well, if the luminance in both images is approximately identical. While this could be achieved by mapping an image to the dynamic range of the reference image, the method is not robust to noise and thus will fail at dark and bright regions of the reference image. However, these regions are the most important parts of the other images as they contain most of the additional information.

3 BTF Decomposition

As mentioned before, one of the first empirical but physically not plausible BRDF models was introduced by Phong [Pho75]. Later a physically correct model was introduced by Ward [War92]. Data-driven models are based on real-world measurements and construct a representation using orthogonal basis functions. Kautz and McCool [KM99] represent a BRDF as sum of separable functions

[4]This can be seen on the roof tiles in Figure 5.7.

where each of the functions can be stored as texture map and rendered in real-time.

However, the focus of this thesis lies on physically based BRDF models as introduced by Torrance and Sparrow [TS67]. Approaches of microfacette-based models have been generalized to arbitrary facet distribution by the Microfacette BRDF Generator [APS00] that was used to approximate measured BRDFs [NDM05]. Later, Ashikhmin proposed the d-BRDF model [Ash06] which has a simplified shadowing and masking term and was used for BRDF acquisition by Ghosh and Heidrich [GH07]. Instead of discrete functions, Lafortune et al. [LFTG97] fit measured BRDFs analytically using multiple cosine-lobes, resulting in a very compact representation.

BTFs were introduced by Dana et al. [DvGNK99] and a comprehensive overview of acquisition, synthesis and rendering is given by Müller et al. [MMS+05]. Much effort has been spent in the field of BTF compression, mostly based on statistical compression. Suykens et al. [SvBLD03] treat a BTF as a set of spatially varying apparent BRDFs and introduce the chained matrix factorization. Resulting factors can be stored as ordinary texture maps and rendered on consumer hardware. Müller et al. [MMK03] took a similar approach but used local principal component analysis to compress the apparent BRDFs, whereas Liu et al. [LHZ+04] perform singular value decomposition of the whole BTF dataset.

In addition to compression, some work has been conducted on measuring or extracting the meso-scale geometry of BTFs. Lensch et al.x [LKG+01] present a robust fitting algorithm of BRDFs to spatially varying materials, which does not include interreflections or self-shadowing.

Magda and Kriegman proposed to decompose a BTF into a set of semi-transparent layers [MK06] to encode the meso-structure. While this approach has the advantage that complex structures can be modeled, the micro-structure is stored volumetrically and thus can hardly be edited intuitively. Another drawback is the limited depth resolution resulting in rather strong shearing artifacts at grazing angles.

Wang et al. [WTS+05] proposed a method to directly capture the meso-geometry of a BTF and store it as 4D distance field, allowing for correctly shaped silhouettes. Müller et al. [MSK06] extract a normal and tangent map and use them to better align the ABRDFs with a data-driven local coordinate system. The Meso-structure can also be obtained from specularity maps [CGS06] or using the so-called Helmholtz stereopsis [ZBK02] where the structure and normals are estimated by exploiting the Helmholtz reciprocity. Pairs of light source and camera positions are chosen that guarantee that the ratio of the emitted radiance to the incident irradiance is the same for corresponding points in two images. The advantage is that no assumptions about the underlying BRDF have to be made, but the resulting depth maps show strong artifacts for uniform materials.

None of the methods discussed so far consider the aspect of editing BTF data. The non-parametric Inverse Shade Tree [LBAD+06] contains 1- or 2-dimensional material representation at leaf level. These low dimensional materials can be

edited, but the method is restricted to flat materials which can be described by a few BRDFs only. Pellacini et al. [PL07] present an approach for editing spatially- and temporally-varying BRDFs that adopts a stroke-based workflow. Efficient solvers are used to allow interactive refinement of this appearance-driven optimization, but there is no straight-forward extension to BTF editing. Kautz et al. [KBD07] proposed an out-of-core data management for editing raw data of an entire BTF. They developed sophisticated operators to edit shadows, specularity, meso-structure and a method to build a selection-tree in order to represent distinct materials. The main drawback of this approach is that if the meso-geometry is changed, affected features – e.g. shadows – do not change accordingly. While interactive editing of a BTF is possible and took about 10-15 minutes for the examples they used, the compression required for real-time rendering takes several hours. Müller et al. [MSK07] developed a technique for procedural editing of BTFs. They are able to alter or replace the original mesostructure using procedural models and generate a new BTF using constraint synthesis. Their main benefit is that the whole complexity of shadowing and light transport is preserved. Unfortunately, the system is restricted to previously acquired materials and the BTF has to be recompressed before rendering.

The idea of representing materials as combination of heightfield and BRDFs is not novel: One of the first approaches to model meso-scale geometry was bump mapping [Bli78] which simulates small-scale variation of surfaces by shifting normals into tangent and bitangent directions. Normal maps can also be directly created from meshes and stored per pixel [COM98] to improve the visual quality of a simplified mesh. Parallax mapping is an enhancement of bump mapping where per-pixel depth information is stored in a texture. The surface is displaced accordingly per pixel which provides more realism for bumpy surfaces and can further be improved by adding soft shadows in real-time [Tat06].

4 Subsurface Scattering

Fundamental work on translucent materials has been done by Wann Jensen et al. [JMLH01]. They state that a BRDF can only describe metallic materials, but not translucent materials where light does not necessarily enter and leave at the same position. Therefore only methods that consider subsurface scattering can capture the true appearance of translucent materials. For this purpose they propose to use a diffuse approximation for the BSSRDF, which is based on the observation that a light distribution in highly scattering media tends to become diffuse and isotropic, that is invariant with respect to rotations around the normal.

One of the first real-time approximations for subsurface scattering was presented by Green [Gre04]. The assumption is that the farther through the material light travels, the more it is scattered and absorbed. In order to measure the distance light has traveled through a material, the scene is rendered from the light's position, storing the distance from the light to a texture. In the second rendering pass this texture is used to obtain the distance from the light at a given point seen by the viewer. By subtracting this value from the distance from the light to this point, an estimate of the distance the light has traveled

through the object is obtained. One problem of this approach is that it deals with convex objects only and it does not consider refraction. Furthermore, only a single view-dependent light path is calculated per pixel and thus the scattering changes with the camera position.

Banterle and Chalmers [BC06] proposed a real-time technique for the rendering of translucent materials based on spherical harmonics. The technique consists of two passes, first the irradiance of the object is projected onto a spherical harmonics basis, then in the second pass the exitant radiance is evaluated. This technique supports also deformable objects with no precomputation time. For the evaluation of the scattering they use a simplified approximation of the diffuse dipole equation, which is enough to generate translucency effects, but not capable of accurately reproducing the appearance of real-world materials. In addition, the local geometry that has a significant contribution to the scattering is only roughly approximated by a thickness term similar to [Gre04].

For the simulation of realistic human skin, d'Eon and Luebke [dL07] developed a multilayer model consisting of a thin oil layer, the epidermis and dermis. The epidermis layer is a very rough surface, while the dermis holds many blood vessels which leads to the reddish color when light is scattered inside the skin. For the simulation of scattering, they combine a series of intermediate Gaussian convolution textures which are combined in the final render pass. This technique is called Texture Space Diffusion and handles only very local scattering. For the transmission through thin surface regions such as ears or fingers, they use the Translucent Shadow Maps (TSM) technique [DS03]. TSMs are an extension of ordinary shadow maps and additionally store normals and irradiance values. For each visible pixel, the according texel in light space in computed and its local and global neighborhoods are sampled. They use the dipole approximation for simulation of the light subsurface scattering. Reflectance Shadow Maps (RSMs) [DS05] extend the TSMs by additionally storing so called Flux values, because each pixel is a potential light source. This algorithm is based on the assumption that a shadow map contains all pixels that contribute to the indirect lighting of a visible pixel. The disadvantage of their approach is that a very high number of samples (up to 448) is required for accurate results.

To reduce the number of samples Ki and Oh [KO08] introduced a GPU-based light hierarchy for real-time approximation of environment or indirect illumination. They store virtual point lights in images and then build a hierarchy of these light sources into image pyramids using a clustering strategy. While the proposed static clustering works very well for distant light sources which occur in environmental or indirect lighting, the performance gain for subsurface scattering where most of the light comes from close virtual lights is rather low.

Hoberock and Jia [HJ07] represent a polygonal mesh with a set of approximating disks in order to compute high quality ambient occlusion in real-time. A disk is associated with each vertex of the mesh to approximate its adjacent polygons. The algorithm computes the occlusion at a vertex by summing shadow contributions from every other individual disk. A straightforward implementation has a complexity of $O(n^2)$, which is avoided by recursively aggregating disks that

have little variation in their contribution. The resulting hierarchical tree can be exploited for the computation of subsurface scattering by simply replacing the integration method for the energy transfer. The disadvantage of this technique is that the computation of the disks as well as the aggregation have to be computed offline and therefore it cannot be applied to dynamic objects.

Tong et al. [TWL+05] acquire material representations from physical samples in a way that allows arbitrary geometric models to be rendered with these materials. The key observation is that the subsurface scattering characteristics of quasi-homogeneous materials are locally non-uniform and require a local representation with respect to the surface mesostructure. Since diffuse scattering does not allow variation of the inner material properties, they propose to variate the portion of light entering and exiting the surface at a given point. This way the core subsurface scattering algorithms can still be used.

Chapter 4.

Compression using Log-Space

As mentioned in the previous sections, the BRDF is able to convincingly model macro-scale structures, but its limitations are easily apparent when rendering structures at meso- or micro-scale. At this level, certain reflectance properties like inter-reflection or subsurface scattering become visible. These phenomena profoundly occur in materials such as fabrics, fur, and skin.

As BRDFs vary over viewing- and lighting-directions only and cannot express spatial varying surface properties, reflectance fields, bidirectional texture functions and surface light fields were introduced to compensate for this inability. Reflectance fields and BTFs both are extensions of BRDFs as they have two additional dimensions to represent spatial variance. They both are 6D functions $f(u, v, \theta_i, \phi_i, \theta_o, \phi_o)$, i.e. they store an incoming and outgoing ray for each point on screen or on the surface. SLFs are defined as 4D functions $f(\mathbf{x}, \theta_o, \phi_o)$, but stored as a piecewise linear approximation for each vertex only to meet memory constraints.

Currently, it is not feasible to create analytical models that are able to represent a material at arbitrary scales. Therefore, image-based methods become increasingly important. All image-based methods have in common that they are acquired as a set of real-world input images. They can be divided into two classes, namely geometry-less and geometry-based methods. The latter ones additionally use a piecewise linear approximation of the object geometry. As low entropy and spatial proximity are strongly correlated, compression methods can be expected to work highly effective. In addition, geometry-based methods need fewer input images.

The input for the image-based methods is gathered by taking a large number of input images under varying light and view conditions. The acquisition of the input images is both very time- and space-consuming and the efficient compression of the acquired data is a challenging task. Common techniques either use statistical compression such as principal component analysis, homomorphic factorization, normalized decomposition, fixed bases like wavelets and spherical harmonics or vector quantization. More sophisticated algorithms combine several of these techniques, e.g. vector quantization and PCA.

The problem that arises is that none of the aforementioned techniques can be applied to these exponentially distributed values directly since all of them

minimize the RMS-error[1], whereas the domain of radiance values is non-linear. In addition, the radiance values are not directly visualized, but a tone mapping operator is applied to map them into the lower dynamic range of a given display device. Note, that even if this mapping was not necessary, the visual perception of the radiance is non-linear as well, such that a linear least squares approximation can never yield optimal visual quality.

The solution to these problems is to apply an invertible operator that maps the radiance values to a uniformly distributed domain before compression and remaps them into their original domain after decompression. The mapping needs to be chosen in such a way, that it is invertible and perceptually as linear as possible to achieve high visual quality. Working in a perceptually linear domain, any compression and quantization method can be applied that minimizes the RMS difference. After decompression, the inverse mapping translates the values back into their original domain thereby recovering the full HDR radiance values. Afterwards, an arbitrary tone mapping operator can be applied to the compressed data for visualization.

The novel compression method presented in the following sections allows for a high-quality compression and decompression of light fields in high dynamic range. A comparison with the traditional approach is shown in Figure 4.1.

Figure 4.1.: Sample dataset from Debevec's Lightstage Gallery. Notice the poor approximation quality of the linear PCA (left) compared to the original input image (middle) and the non-linear PCA (left). Size of the datasets from left to right: 96 MB, 1518 MB, 8 MB.

1 Compression

In this approach PCA was chosen to compress the input images, because its quality is superior to vector quantization (VQ). The first step of the algorithm is to rearrange the input data into a 2D matrix. Then, a non-linear transformation is applied to each element to achieve a uniform distribution of the radiance values. After this, the matrix is approximated with a linear least squares rank-k approximation using a truncated PCA. Finally, the principal components are quantized to 8 bits and stored as ordinary bitmaps. To further reduce the size of these

[1] i.e. they produce an optimal linear solution

images, hardware-accelerated texture compression is used. The reconstruction of the data is performed by accumulating the tensor products[2] of the principal components. After reconstruction, the inverse non-linear transformation is applied to the reconstructed RGB values in order to recover the original high dynamic range radiance values. To reduce the dynamic range to the displayable range [0..255], a simple tone mapping operator[3] is used.

2 Non-linear PCA

In order to apply matrix factorization to the sampled input images, the four-dimensional data has to be rearranged into a 2D matrix. This is accomplished by fixing two domains and letting the two other domains vary, and vice versa. In case of a reflectance field this means that while the lighting-angles θ_i and ϕ_i remain fixed, the components u_r, v_r of the surface location $\mathbf{x}$ may vary. Essentially, this unrolls the 2D images into 1D column vectors of the matrix $\mathbf{M}$:

$$\mathbf{M} = \begin{pmatrix} f(r_1, s_1, \theta_1, \phi_1) & \cdots & f(r_1, s_1, \theta_N, \phi_N) \\ \vdots & \ddots & \vdots \\ f(r_M, s_M, \theta_1, \phi_1) & \cdots & f(r_M, s_M, \theta_N, \phi_N) \end{pmatrix},$$

Here, M is the total number of red, green and values in each image and N the total number of input images. In the next step, these values have to be transformed into the linear domain.

If an arbitrary tone mapping operator is applied to the input matrix $\mathbf{M}$, its values will be more or less linearly distributed, depending on the quality of the chosen operator[4]. It is important to note that not every operator can be applied unless it is completely invertible. The latter condition is essential for the reconstruction of the full dynamic range. In this approach, a logarithmic operator was chosen to do this transformation, because HDR images are created by the following equation introduced by Debevec and Malik [DM97]:

$$E_i = e^{g(Z_i) - \ln \Delta t_j},$$

Here, E_i is the radiance of a single pixel, calculated by evaluating the irradiance Z_i and the exposure time Δt_j. To transform the images into a domain with uniformly distributed values, the logarithm has to be applied to all elements of the input matrix $\mathbf{M}$:

$$\ln(\mathbf{M}) = \begin{pmatrix} \ln(\delta + m_{11}) & \cdots & \ln(\delta + m_{1N}) \\ \vdots & \ddots & \vdots \\ \ln(\delta + m_{M1}) & \cdots & \ln(\delta + m_{MN}) \end{pmatrix} = \tilde{\mathbf{M}},$$

δ is a very small number to avoid the singularity of the logarithm and $\tilde{\mathbf{M}}$ is the transformed matrix containing linear distributed values.

[2] or outer products

[3] inverse gamma or square root

[4] i.e. whether it is local or global

The next step is a linear least-squares approximation of $\tilde{\mathbf{M}}$ using principal component analysis and keeping only those k eigenvectors with the largest corresponding eigenvalues:

$$\tilde{\mathbf{M}} \approx \sum_{i=1}^{k} \mathbf{u}_i^t \sigma_i \mathbf{v}_i = \mathbf{U}_k^t \mathbf{\Sigma}_k \mathbf{V}_k,$$

The values $\sigma_i \mathbf{u}_i$ and $\mathbf{v}_i$ are stored to save the memory for the singular values. Note that for the datasets, U_k^t has the dimension 253×253 and V_k the dimension $(1024 \times 1024 \times 3)^2$, so a progressive decomposition of the PCA should be computed instead of a computational expensive complete decomposition.

The fact that only the first few principal components are required can be exploited by first projecting the BTF into a Krylov subspace [Kry31] of low dimension using Lanczos iteration [Lan50], since calculating the first k components from a subspace with a dimension of $4k$ already yields a very accurate approximation. So instead of $\tilde{\mathbf{M}}$, this approach uses

$$\tilde{\mathbf{M}} \approx \tilde{\mathbf{M}}_{4k} = \sum_{i=1}^{4k} \mathbf{w}_i^t \mathbf{q}_i = \mathbf{W}_{4k}^t \mathbf{Q}_{4k}.$$

When generating the subspace, the first basis vector has to be guessed. Typically, a random unit vector is chosen for this purpose. In the context of eigenimages[5] however, choosing a white image as first basis vector leads to a more stable subspace construction, as all pixels are all positive and thus the second eigenimage becomes the average image. To find the first k principal components, the property is exploited that the eigenvalues of the orthogonalization matrix $\mathbf{H}$ constructed during the Lanczos iteration are those of $\tilde{\mathbf{M}}$ and the principal components can be found by unprojecting those of $\mathbf{H}$.

3 Quantization

An eigenimage created by the PCA is an array of floating point numbers and may contain negative values. The problem caused by this representation is that only floating point texture formats support negative values. At this point, another advantage of the transformation into the linear domain becomes obvious: Since the radiance values of the eigenimages are distributed linearly, they can efficiently be compressed by first quantizing them uniformly and then applying common (lossy) texture compression. Quantization is done in a two-step algorithm: For each eigenimage, first the minimum and maximum color values, $(r, g, b)_{min}$ and $(r, g, b)_{max}$ are determined. In a second, the interval $[(r, g, b)_{min}, (r, g, b)_{max}]$ and thus all color values are scaled into the interval $[(0, 0, 0), (255, 255, 255)]$ such that they can be stored in a common 8-bit per channel bitmap. The same scaling is applied to the weights with the exception that they are scalar values per light and/or view direction and thus stored in an 8 bit per pixel luminance image.

After quantization each pixel of the eigenimages can be stored with 24 bpp

[5] An eigenimage is a set of eigenvectors

instead of 48 bpp (half precision) or 96 bpp (float precision). To further reduce the required memory, these textures are compressed using DXT1 compression that stores 16 input pixels in a 64 bit array, consisting of two 16-bit RGB 5:6:5 color values and a 4×4 two bit lookup table. Other variations of DXT additionally consider the alpha channel, which is not needed for the eigenimages. After quantization and compression, the size of the eigenimages has been reduced to 4 bit per pixel, resulting in a compression ratio of 1:12 compared to a half precision float texture.

4 Reconstruction

To reconstruct the dataset from the first principal components, each step of the quantization has to be inverted in reverse order. Since DXT1 is supported by hardware, it can be loaded directly into the texture memory of the rendering device. After this, the color values of each eigenimage have to be rescaled from $[(0,0,0),(255,255,255)]$ back to $[(r,g,b)_{min},(r,g,b)_{max}]$. Since the fragment shader is used to perform the reconstruction in real-time, the scaling parameters have to remapped into a texture in order to make them accessible to the rendering device. Section 5 describes these steps in detail.

The next step is the reconstruction of each pixel according to the number of eigenimages used. This is done with the sum

$$\tilde{\mathbf{M}} \approx \sum_{i=1}^{k} \mathbf{u}_i^t \sigma_i \mathbf{v}_i,$$

which computes the approximation of the original reflectance field. For this purpose, the highly parallel architecture of the GPU can be exploited, making it possible to accumulate several pixel at the same time. After the reconstruction the image still lies in the linear domain and has to be remapped to the original high dynamic range non-linear domain. This is accomplished with the mapping

$$(\tilde{r},\tilde{g},\tilde{b}) to (r,g,b) = (e^{\tilde{r}} - \delta, e^{\tilde{g}} - \delta, e^{\tilde{b}} - \delta).$$

Finally, a tone mapping operator – either global or local – is applied to the reconstructed image for output on a low dynamic range display device.

5 Resampling of Weights using Cube-Maps

As stated in section 3.2, minimum and maximum offset values have to be stored for each eigenimage. In order to make these offset values accessible to the fragment shader, they have to be converted to texture data first. A radial basis function is chosen in order to perform all-frequency interpolation at any given sample position.

For this purpose the offset values have to be remapped onto the six planes of a cubemap. A radial function e^{-d^2} is used to map them onto the unit sphere, first. After this, a linear equation system has to be solved in order to remap these coordinates to the unit cube. Mipmap degradation then gives the desired approximation of the integral over all directions.

$$f_i(x) = \sum_{j=1}^{N} w_{i,j} e^{-\|p_j - x\|}$$

with

$$f_i(p_j) = u_{i,j}$$

6 Results

As database for evaluation of the non-linear PCA Debevec's Light Stage Data Gallery[6] was chosen. Although the presented technique was not tested on high dynamic range (surface) light fields or higher dimensional datasets like 6D reflectance fields or BTFs, the experimental results will directly transfer to these. Each dataset shows a static subject captured under 253 individual lighting directions covering the full sphere of illumination, resulting in a 4D reflection field. The input images are stored with 48 bpp at a resolution of one megapixel.

First, the RMS of the reconstructed input images is compared to the original signal after a simple global tone mapping operator (adaptive exposure followed by gamma 2.2) is applied. In Figure 4.2 you can see the RMS plots for each of the six datasets, indicated by a small scale sample in the lower left corner. As can be seen in any of these plots, the RMS error of the non-linear PCA falls much faster compared to the linear PCA. While this argument always holds for more than four terms, it can be observed that the results strongly differ for less than four terms. In contrast, in the dataset "knight standing" it can be seen that the linear PCA is better for less than four terms. Figure 4.2 also shows that close-up models such as "helmet side" and "helmet front" can be approximated with less terms than the other models.

The RMS plots do already indicate that the overall approximation quality of the non-linear PCA is superior to the linear PCA. In figure 4.3 a direct visual comparison is presented, showing the original input images in the middle column. In the left column are the reconstructed images from the linear PCA. Here, many dark areas are noticeable, originating from the fact that the linear PCA needs many terms for the approximation of the exponentially distributed radiance values, underrating the areas with lower radiance. In the right column images from the non-linear PCA are presented. In the "knight fighting" dataset it is observable that sharp highlights lose their details, though the overall impression is by far better than the linear PCA.

[6] http://gl.ict.usc.edu/Data/LightStage/

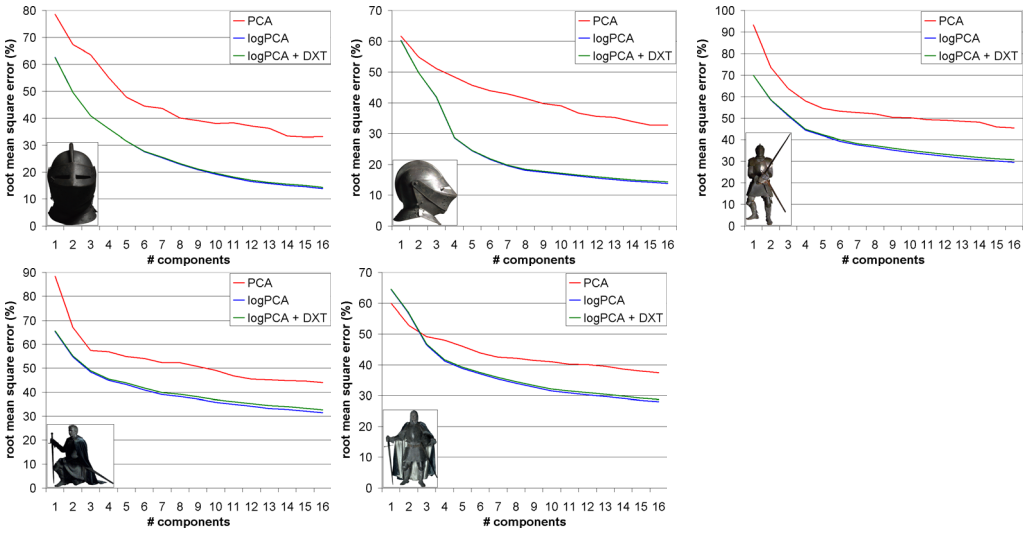

Figure 4.2.: RMS plots of each dataset. The according dataset is indicated by a small scale model in the lower left corner.

Figure 4.3.: Five datasets from Debevec's Lightstage Data Gallery. In the middle are the original, uncompressed input images. On the left side are the reconstructed images after linear PCA was applied. It is clearly to see that poor approximation leads to visual artifacts. On the right side are the images created with the non-linear PCA.

Chapter 5.

Measuring and Combining HDR Data

The dynamic range of an image is the perceived ratio between the darkest and the brightest pixel. This ratio easily exceeds the capabilities of a common photograph taken with a single exposure. Instead, the full dynamic range of a scene can be obtained by taking multiple photographs, each one with a different exposure time. This way it is guaranteed that shadowed areas as well as very bright areas are captured in full detail. After the actual acquisition, the next step prior to combining the images is the reconstruction of the response function of the camera that relates color values to light intensities. The discrete response function is a non-linear curve that can be different for each color channel and is stored as a simple lookup table containing 256 values. After this step, the radiance map is computed as a weighted sum of pixel intensities from the images with different exposure times.

For the recovery of the dynamic range it is crucial for all pixel locations to be exactly aligned. This constraint originates in the assumption that the irradiance of each pixel remains constant throughout the whole exposure sequence. For this purpose it is advisable to create exposure sequences using a tripod and in addition, a remote controlled release. Besides avoiding undesirable vibrations, the scene has to be static as well, which makes capturing of naturally moving objects, such as clouds, leaves or waves, difficult to impossible. Up to now, these constraints limit HDR photography to professionals or dedicated hobby photographers. This restriction can only be lifted by a tool that is able to align the images of an exposure sequence by removing any motion origination from camera or object movement.

To construct a robust freehand motion compensation algorithm, it is important to understand which effects may occur besides translation of the image:

- When photographs are taken by hand, rotation is inevitable, though very small angles of typically less than 20 degree can be expected.
- Even the slightest translation may cause severe parallax effects, e.g. when looking through a window or straight down a wall.
- In addition to the parallax, occlusion can also occur whenever the camera or an object moves.

- In contrast to traditional image matching, it cannot be assumed that finding matching patterns between different exposures is always possible due to shifted detail. Detailed areas in one image are possibly totally saturated or at noise level in the next or previous image of the sequence, making it impossible to find a direct matching. This can be interpreted as missing data problem.

Considering these problems, a two step algorithm is proposed. In the first step, the images are aligned as faithfully as possible. In the second step, the occlusion and missing data problems are handled by a robust HDR reconstruction method including a ghost removal step. This ends up in these two essential contributions:

- A hierarchical non-linear alignment algorithm that is robust with respect to missing data due to black level noise and saturation.
- A robust HDR reconstruction algorithm that removes the remaining artifacts originating from occlusion while preserving information that is not contained in the other images of the sequence.

The first step of the algorithm is the non-linear alignment of the images using macroblocks and maximizing the cross-correlation. The alignment is performed hierarchically to assure that matching blocks as well as non-matchable blocks are moved into a consistent direction. An in depth description of this process is given in section 1.

After the alignment, most pixel locations match exactly but occlusion and parallax effects inside a single leaf level macroblock cannot be solved, which can lead to ghosting artifacts. This is solved in the HDR reconstruction phase described in section 2 by calculating a per-pixel confidence value and marking non-plausible pixels so that they do not contribute to the final image.

1 Hierarchical Matching

For the non-linear matching, first an anchor image r is chosen by simply selecting the one containing the highest entropy[1] and then all other images are aligned to this one. One way to capture the input images freehand is to use the automatic exposure bracketing (AEB) function of the camera. The AEB function induces the camera to capture a sequence containing an optimally exposed, underexposed and overexposed image. It is expected that the optimal exposed image contains the highest entropy and thus, this image is set as anchor for every macroblock. Since capturing more than three images seems impractical for ad hoc purposes, the implementation is limited to three input images, though the full dynamic range of outdoor scenes might not be covered with three input images only. In addition, if more than three images are used, finding an optimal anchor image might be a difficult task, since optimal correlation can expected only for neighboring images.

[1] i.e. containing most variation and thus possible image information and contrast

The basic idea for the motion estimation is that for a given macroblock M in the anchor image, the displaced matching macroblock in image i will have the locally maximal cross correlation $C_i(M, \delta)$. Instead of normalizing both vectors before computing the cross correlation, a slightly different formulation is used that is basically just a scaled version with the same maximum[2]:

$$C_i(M,\delta) = \frac{\sum_{p\in M} \tilde{\mathbf{c}}_i(p+\delta)\cdot\tilde{\mathbf{c}}_r(p)}{N(M)\sqrt{\sum_{p\in M}\|\tilde{\mathbf{c}}_i(p+\delta)\|^2}}$$

with

$$\tilde{\mathbf{c}}_i(p+\delta) = \mathbf{c}_i(p+\delta) - \begin{pmatrix}1\\1\\1\end{pmatrix}\frac{\sum_{p\in M}\|\mathbf{c}_i(p+\delta)\|_1}{3N(M)}$$

$$\tilde{\mathbf{c}}_r(p) = \mathbf{c}_r(p) - \begin{pmatrix}1\\1\\1\end{pmatrix}\frac{\sum_{p\in M}\|\mathbf{c}_r(p)\|_1}{3N(M)},$$

where $\mathbf{c}_k(p)$ is the RGB color vector of image k at pixel p, and $N(M)$ is the number of pixels contained in the macroblock. To emphasize local contrast, the edges in all input images are enhanced using the following filter kernel F:

$$F = \begin{pmatrix} 0 & -\frac{1}{4} & 0 \\ -\frac{1}{4} & 2 & -\frac{1}{4} \\ 0 & -\frac{1}{4} & 0 \end{pmatrix},$$

where pixels at the boundaries are duplicated for consistent filtering.

Unfortunately, choosing a good macroblock size that works for every input sequence is not possible. If the size is too large, local movements cannot be compensated for, while too small macroblocks cannot always be matched reliably in the absence of local detail. Therefore, the image i is aligned in a hierarchical manner starting with a single centered square macroblock of size 2^n that is large enough to contain the whole image. After the best match for this root block is found, it is divided into four sub blocks and for each of them the best match is searched recursively while re-using the alignment of the previous level as initial guess.

Hierarchical guessing is prone to be less robust towards large motion of small objects. The problem is that while the confidence value of a large macroblock might by high, the motion of smaller objects cannot be estimated until a finer subdivision level is reached. In order to find the object, the algorithm then has to increase the search radius instead of actually reducing it in order to converge. Here, a trade-off between robustness of motion estimation and computing speed has to be found.

[2]The reason for omitting the normalization step is that the modified version is somewhat more robust to areas of uniform luminance and little contrast.

To prevent random displacement when the macroblock contains noise only, a block is not displaced if its cross correlation less than 10% of its parent block's cross correlation. If this value is lower for the current block, it gradually decreases from level to level. During recursive search, the maximum search radius is reduced by a factor of two at each subdivision until it reaches two at a block size of 16×16 pixel. This allows to also compensate for rotations of up to approximately 14 degree.

At each recursion step, the displacement is bilinearly interpolated between the four adjacent parent nodes. Then an iterative search is performed for each macroblock starting with its interpolated displacement and that of its up to eight neighbors. For each initial displacement, the next local maximum is found using a gradient descent method with a fixed step size of one pixel. The only exception is the root block, were the algorithm starts with a step size of four pixels in order to find the global maximum. The search from the current start displacement is stopped when either the local maximum is found, or the number of iterations reaches the search radius of the current level.

When the local maxima are found for each of the nine starting displacements, the final displacement for the current block is chosen to be that one with the highest cross correlation. If the maximum is reached for more than one different displacement, the closest one of them to the initial interpolated displacement is chosen to prevent displacement of macroblocks that do not contain local detail. Figure 5.1 shows the hierarchical alignment process for an input image where the clouds had moved with respect to the reference image.

Figure 5.2.: Difference of aligned image and reference image converted into same exposure. The darker the color, the higher is the difference.

After calculating the displacement vectors for the finest level macroblocks, the transformed image is assembled. For each macroblock, the corresponding pixels – without edge enhancement – are copied from the displaced position to their original position in the reference image. This way, the features of the current image become aligned with their corresponding features in the reference image. To prevent the well known blocking artifact of macroblock based motion compensation,

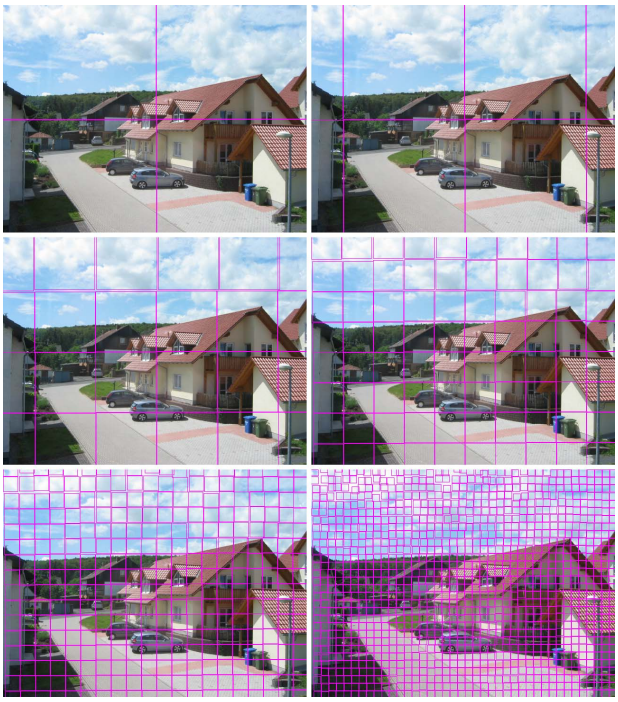

Figure 5.1.: Hierarchical alignment of an image where the clouds had moved upwards and right during the exposure sequence. The magenta squares depict the matching macroblocks.

the displacement is bilinearly interpolated per pixel for the transformation of the image. While this might introduce some distortions in badly aligned region, the overall image quality is greatly improved. Figure 5.2 shows the difference of the example image after alignment to the reference image mapped into the same exposure. Notice that most of the difference is due to highlights that are over saturated in the aligned image. The movement of the clouds and the slight foreground parallax have been corrected.

2 HDR Image Synthesis

The HDR reconstruction is based on a per-pixel weighted average of the measured logarithmic irradiance. Thus, the irradiance has to be reconstructed from each image as first step by calculating the camera response function f using the least squares solver proposed by Debevec and Malik [DM97]. The accumulated irradiance $E(p)$ at the current pixel p is then calculated from the color values c_i and the exposure times t_i of each image i for each color channel in the following way:

$$\log E(p) = \frac{\sum_i w_i(p) \log f(c_i(p)) - \log t_i}{\sum_i w_i(p)}$$

The key to both, faithful reconstruction and artifact removal now lies in an ap-

propriate weighting function. The first weighting function proposed by Debevec and Malik [DM97] was a simple piecewise linear function:

$$w_i(p) = \begin{cases} c_i(p) & : \quad c_i(p) \leq 127.5 \\ 255 - c_i(p) & : \quad c_i(p) > 127.5 \end{cases}$$

The drawbacks of this function are the non-zero derivatives at both ends and the relatively high weighting of extreme values. These introduce discontinuities and can reduce the dynamic range of the final image. While the first problem was solved with the Gaussian weighting function of Robertson et al. [RBS99] the range reduction is even more apparent. To solve both problems, a quadratic spline weighting function is proposed that has zero values and derivatives at both ends:

$$w_i(p) = \begin{cases} 8\left(\frac{c_i(p)}{255}\right)^2 & : \quad \frac{c_i(p)}{255} < \frac{1}{4} \\ 1 - 8\left(\frac{c_i(p)}{255} - \frac{1}{2}\right)^2 & : \quad \frac{1}{4} \leq \frac{c_i(p)}{255} \leq \frac{3}{4} \\ 8\left(1 - \frac{c_i(p)}{255}\right)^2 & : \quad \frac{c_i(p)}{255} > \frac{3}{4} \end{cases}$$

Due to the low weight for extreme values, the result might however become noisy if the pixel is almost black or white in all images of the exposure sequence. Therefore, the weight is set to one in the longest exposure image if the color value is below 127.5 and in the shortest exposure image if the value is above 127.5 since these contain the most accurate information. Figure 5.3 shows the resulting weight functions for a sequence of five images with two f-stops between each successive pair.

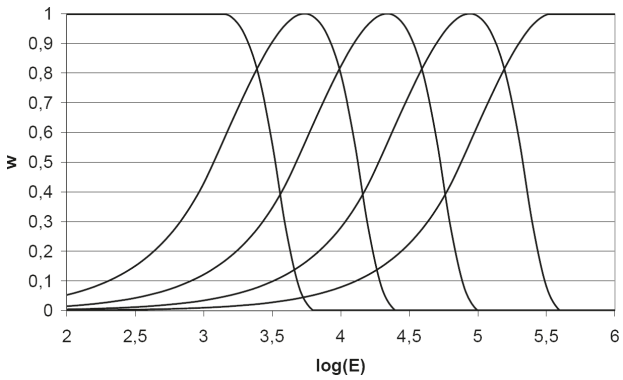

Figure 5.3.: Weight functions for a sequence of five images with an exposure stepping of two f-stops and a gamma-2.2 camera response curve.

3 Ghost Removal

While these intensity based weighting functions only account for the accuracy of the CCD sensor, Grosch [Gro06] additionally proposed to test the luminance and color value of a pixel against the corresponding one in an anchor image for

plausibility. If the difference between expected value and sampled value exceeds some error threshold, the pixel is treated as ghost and its weight is set to zero. This plausibility test however is prone to inaccuracies of the camera response function and thus the confidence map often needs to be edited by the user.

Instead, the following ghost removal technique is proposed: For each pixel in the all images, except the anchor image r, a confidence value, derived from the cross-correlation, is calculated. A small neighborhood N_p around the pixel p is extracted from each image i and the confidence $K_i(p)$ is calculated based on these:

$$K_i(p) = \frac{\sum_{j \in N_p} \mathbf{c}_i(j) \cdot \mathbf{c}_r(j)}{\sqrt{\sum_{j \in N_p} \|\mathbf{c}_i(j)\|^2} \sqrt{\sum_{j \in N_p} \|\mathbf{c}_r(j)\|^2}}$$

By considering the neighboring pixels instead of the camera response curve, the result is much more robust and less sensitive to noise[3]. If the confidence of a pixel falls below a threshold t_K, its weight is set to zero. The only exception is made for the shortest and longest exposure images: if the only information about a pixel is contained in one of these images – i.e. the color values are below the black noise level t_b or above saturation t_s in all other images – the confidence test is skipped and thus the weight is not altered. Altogether, this leads to the final weight:

$$w_i^*(p) = \begin{cases} 0 & : \quad K_i(p) < t_K, t_b < c_i(p) < t_s \\ w_i(p) & : \quad \text{else} \end{cases}$$

In the implementation, a neighborhood of 11×11 pixel is used that is centered at the current pixel and $t_K = 0.95$, $t_b = 0.05$, and $t_s = 0.95$ as threshold values.

Figure 5.4 shows the regions identified as non-plausible in the aligned example image from section 2. The non-matchable lens reflection, as well as some less accurately aligned features are removed.

[3]Note, that in contrast to the cross-correlation used for matching, the mean is not subtracted from the blocks since this would prevent detecting differences of the average luminance.

Figure 5.5.: Image sequences used for evaluation: The top row shows a sequence that can be aligned using linear transformations, while the middle sequence contains moving objects (the clouds) and the bottom one significant parallax effects and occlusions.

Figure 5.4.: Pixels marked as ghosts in an aligned image.

4 Results

Figure 5.5 shows the three exposure sequences used as test data to compare the presented approach with linear alignment. The first sequence is included to demonstrate the that the new method delivers as least as good results as linear alignment. Figure 5.6 shows a comparison of tone-mapped images generated with the two techniques. As tone mapping operator, the gradient domain HDR compression [FLW02] is used. The difference between the two images is almost invisible and the ghost removal is required for neither of them.

Figure 5.6.: Tone-mapped results generated from the first sequence the result using linear alignment (left) and the non-linear alignment (right). Both are generated without ghost removal.

The second example demonstrates the superiority of the novel approach in the presence of moving objects. Figure 5.7 shows a tone-mapped version of the HDR image generated with the new approach in comparison with linear alignment. While the ghost removal is capable to remove most of the artifacts of the linear alignment technique, except for the still slightly blurry clouds (see magnifications below the images), the dynamic range of the image is degraded. The improved alignment of the new method does not only reduce the misalignment artifacts, but also exhibits a higher dynamic range of the final image after ghost removal, e.g. in the highlights of the roof tiles.

Figure 5.7.: Tone-mapped results generated from the second sequence using linear alignment (left) and the non-linear alignment (right). The top row is without and the bottom with ghost removal.

The final example is chosen to exploit the limitations of the presented tech-

nique. Due to the significant parallax and large occluded areas, an alignment is not possible for all parts of the image sequence. Thus the final HDR image (a tone-mapped version is shown in figure 5.8) contains many artifacts in these areas. While the ghost removal is capable of removing almost all artifacts – except for a region in the upper right part of the image that was completely white in two of the images and occluded in the third one – it degrades the dynamic range of the image. Nevertheless, the result with the new technique is significantly better than with linear alignment, where even the ghost removal is not able to resolve all ambiguities.

Figure 5.8.: Tone-mapped results generated from the third sequence the result using linear alignment (left) and the non-linear alignment (right). The top row is generated without and the bottom row with ghost removal.

Chapter 6.

An Editable BTF Representation: The G-BRDF

In general, surface appearance is either determined by reflectance (coarse-scale) or texture (fine-scale). When view and light directions vary, the equivalent descriptions are the BRDF and the BTF. As explained in the first chapter, the BTF can be understood as a simultaneous measurement of per-pixel BRDFs, called apparent BRDFs (ABRDFs) as they also contain effects of the underlying meso-scale geometry and represent the reflectance fields of single pixels. As such, a BTF accommodates self-shadowing, inter-reflection, masking, and parallax effects of a complex material without explicit representation of the small scale geometry. Currently, there are two major reasons speaking against BTFs as common modeling resource: First, acquisition systems are expensive and the measurement process lasts from several hours to days, and second, the size of a BTF is in the range of several gigabytes, so effective compression methods have to be applied before synthesis or rendering [MMS+05].

Recently much effort was spent on developing efficient compression methods for BRDFs and BTFs. With these at hand, sample databases containing gigabytes of input images can be reduced to a fraction of their original size, preserving most of the original visual appearance. Many of these methods follow a similar approach: They reduce high-dimensional input data to low-dimensional subspaces containing most of the variation. Resulting eigenvectors can be stored in ordinary texture maps and the BTF can be rendered in real-time.

Nevertheless, BTFs are still far from being used as common modeling resource like texture maps, one reason being that compressed data offers very little expressiveness to designers. Although user-guided modification and editing of raw BTFs has become possible, in existing systems a BTF needs to be compressed for real-time rendering which again requires several hours The solution to this problem would be a method that breaks a BTF down into comprehensible properties and then use these for rendering as well as for editing. This would not only enable designers to modify or combine BTFs to create novel materials, but also yield a very efficient compression, as some properties have little contribution to the final appearance and only the most visible ones are required to describe a BTF's appearance under varying light and view directions.

The reason why all the BTF information is needed, is that real-world surfaces comprise extremely complex light interaction which physically-based BRDFs de-

scribe at micro-scale without considering geometric variations at meso-scale. Since these are contained in almost every BTF, a BTF cannot be treated as spatially varying BRDF, because light interaction is closely related to geometry and thus not locally bound.

The contribution of this chapter is hence an intuitive representation for both meso- and micro-scale properties of a BTF as shown in Figure 10.9. By separating these two scales, an artist can use BRDF-editing techniques together with geometric editing to modify an existing BTF or to create a novel one from scratch. Whereas the combination of heightfield and BRDFs is not new to artists, the particular contribution of the presented approach is that an algorithm is proposed to extract the meso-scale structure and analytical BRDFs from measured BTFs. This means that physical properties of a real-world surface are described instead of a hand-generated texture. This does not only allow to efficiently compress a BTF and render it in real-time, but also to use measured data as basis for artistic editing of a real-world material.

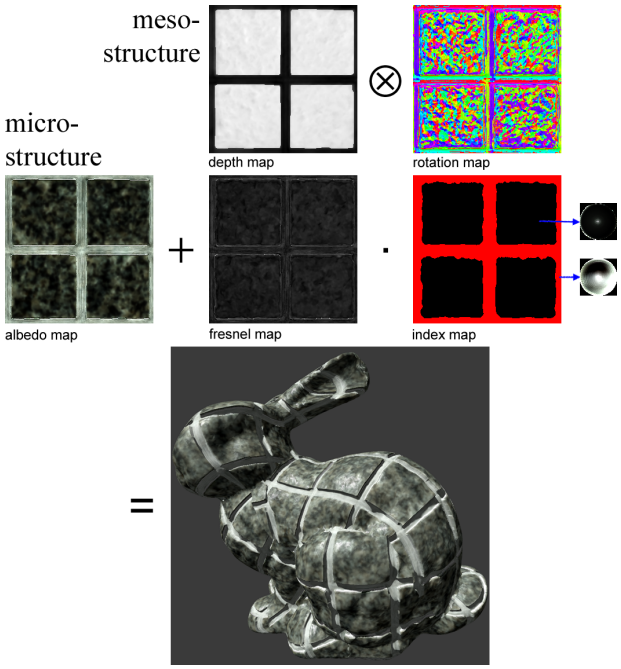

Figure 6.1.: Rendering of a BTF from its g-BRDF representation.

1 The g-BRDF Representation

To ensure both compression and editability, a BTF is best represented by a set of 2D images describing either light interaction or the geometric structure of the underlying surface. Therefore, two distinct classes are defined: light interaction maps and geometry maps.

The meso-scale geometry is represented by a depth map and two scalar values defining minimum and maximum depth offsets. In this form, the geometry can

easily be modified by just increasing or decreasing pixels or whole areas. For more complicated operations up to generating a complete depth map from scratch, a variety of existing tools that have been developed to create depth maps for 3D computer games can be used. These tools range from procedural texture generation to modeling systems that can output the depth map from a 3D model instead of a rendered image. An additional normal map is not required, since it can be derived from the depth map with the advantage that a designer does not need to take care of consistency between the two. The only remaining degree of freedom for the local coordinate system is a rotation about the normal which is encoded in the orientation (or tangent) map, which is required for anisotropic BRDFs. In the orientation map, each pixel stores the direction of the tangent vector encoded as hue of the HLS color model, although any other coding would be possible as well. While this second texture map might not seem very intuitive at first sight, designers quickly get used to it and are able to achieve the desired results. Figure 6.2 shows an example of depth and orientation maps.

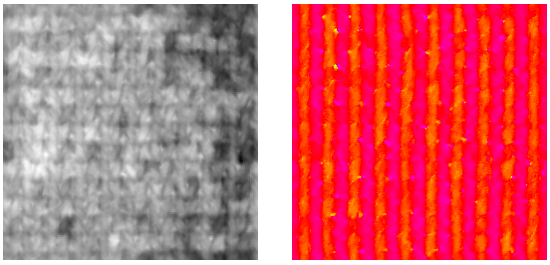

Figure 6.2.: Meso-structure maps: depth (left) and orientation (right).

While the meso-scale geometry is completely represented by these two images, the number of parameters required to describe the micro-structure is significantly higher in order to preserve the visual appearance. Since it is impossible to completely define the light interaction for each texel, both from a designers point of view, as well as considering the required storage amount, the BTF is divided into a set of a few distinct basis materials. Each basis material can be represented by an arbitrary BRDF model, where some low-dimensional parameters – like the diffuse color – are stored per texel to allow for more variation.

2 BRDF Model

So far, geometry has been separated from light interaction and thus each material can be represented by a BRDF $\rho(\mathbf{k_1}, \mathbf{k_2})$ that describes what fraction of light coming from a given direction $\mathbf{k_1}$ is reflected into another direction $\mathbf{k_2}$. The choice of an appropriate BRDF model is important because the complexity of the underlying model determines how easily each basis material can be edited. The more accurate (and thus more complex) a BRDF model is, the harder it becomes to achieve a desired appearance by changing its properties. Therefore, Ashikhmin's distribution-based BRDF [Ash06] is chosen, as it is an excellent trade-off between accuracy and intuition. There the BRDF is defined as:

$$\rho(\mathbf{k}_1, \mathbf{k}_2) = \frac{c_d}{\pi} + \frac{c_s p(\mathbf{h}) F(\mathbf{k}_1\mathbf{h})}{\mathbf{k}_1\mathbf{n} + \mathbf{k}_2\mathbf{n} - (\mathbf{k}_1\mathbf{n})(\mathbf{k}_2\mathbf{n})},$$

where the Normal Distribution Function $p(\mathbf{h})$ is an arbitrary anisotropic function that describes the normal distribution of the BRDF's underlying micro-structure and the half-vector $\mathbf{h}$ is the normalized average of $\mathbf{k_1}$ and $\mathbf{k_2}$. The NDF is stored in a texture using the parabolic maps parametrization [HS98] which is simple to calculate and has a relatively uniform sampling of the hemisphere, but any other parametrization could be used as well.

For the Fresnel term, Schlick's approximation

$$F(\mathbf{k}_1\mathbf{h}) \approx r_0 + (1 - r_0)(1 - \mathbf{k}_1\mathbf{h})^5,$$

is used, where the reflectance at normal incidence r_0 depends on the refraction index of the material.

Due to the complexity of the NDF, only a single one is stored per basis material. The other two parameters c_d and r_0 are low-dimensional and thus allowed to change per texel. This way, a compact editable representation is combined with the possibility to introduce per-texel variation by modulating the diffuse color and the specular reflection. Figure 6.3 shows an example of the complete set of images defining the micro-structure of a BTF.

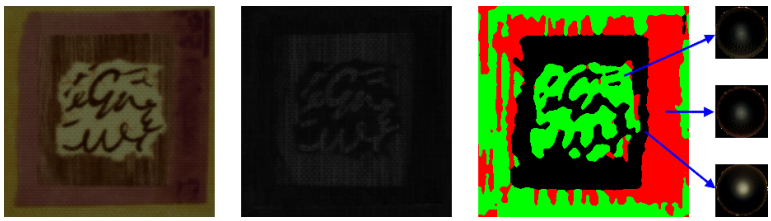

Figure 6.3.: Micro-structure maps (from left to right): Diffuse, Fresnel, cluster, and NDF maps.

3 Rendering

Although it is possible to use the proposed g-BRDF representation directly for rendering, a few modifications can improve the performance. Grouping of textures can save texture units and texture fetches, preprocessing is applied if possible in order to save shader instructions. The grouping and preprocessing requires a few milliseconds per BTF only and could even be integrated as additional shader pass operating on intermediate textures.

The surface normals are precomputed by discrete derivation of the depth map using the 2D Sobel-filter [GW77]. The x- and y-components of the normals are stored in the red and blue color channels of a 2D texture. The z-component is redundant as it can be calculated with $z = 1 - \sqrt{x - y}$. To rotate $\mathbf{k_1}$ and $\mathbf{k_2}$ into the local coordinate system, the sine and cosine of the tangent angles are

precomputed and stored in the remaining two components of the rotation texture. This saves their costly computation at run-time and avoids discontinuities between 0 and 2π during interpolation.

The cluster indices are originally stored in an indexed color map, which has the drawback of not being suited for bilinear interpolation. For this reason, every four materials are combined into a weight texture where each channel contains the weight for one material. The monochrome Fresnel map is combined with the diffuse map into a single RGBA texture and finally, all NDFs are aggregated into a 3D texture, where the third dimension is equal to the number of distinct materials. This does not only reduce the number of required texture units, but also the number of texture fetches since the interpolated probability between two adjacent NDFs can be obtained with a single texture fetch. Figure 6.4 shows an overview of the textures and the overall workflow of the rendering algorithm.

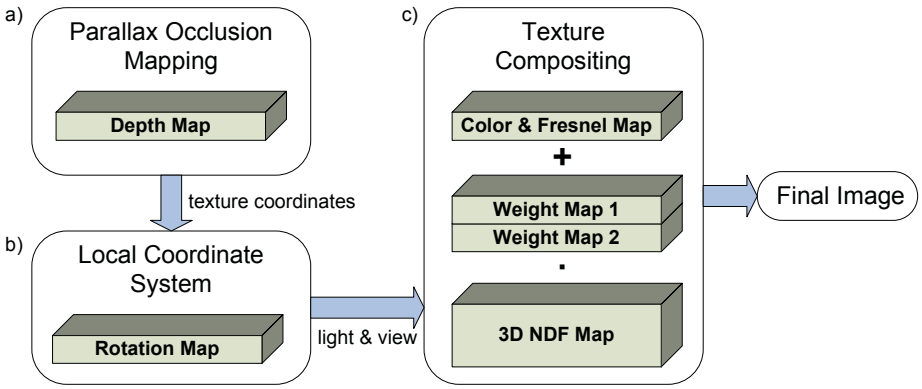

Figure 6.4.: Rendering workflow: a) the Parallax Occlusion Mapping determines (u, v) texture coordinates b) (u, v) are used to index the rotation map c) local light and view are used to calculate the BRDF.

The most time consuming part of the rendering algorithm is the mapping of screen pixels onto the meso-geometry. For this step, Parallax Occlusion Mapping (POM) [Tat06] is used, which essentially is a per-pixel ray-tracing algorithm that is able to resolve inter-penetrations and self-occlusion in real-time. In addition to the view ray, a shadow ray is traced toward the light source to generate self-shadowing effects. Finally, the BRDF can be evaluated using the local coordinate system (Figure 6.5).

4 BTF Decomposition

Since there is already a substantial amount of measured BTF data available, an algorithm to decompose these measured BTFs into a corresponding g-BRDF is desirable to use them as basis for editing and combine them with other g-BRDFs in a real-time application. This decomposition needs to be split into two phases since before each texel of the BTF can be approximated with the BRDF model,

```
g_BRDF(u,v,k1,k2) {
  x = trace(u,v,k2);
  Li = traceLight(x,k1);
  R = setupRotationMatrix(x);
  Lr = BRDF(x,Rk1,Rk2)*Li*(n · Rk1);
  return Lr;
}
```

Figure 6.5.: Basic structure of g-BRDF pixel shader.

the meso-structure has to be recovered in order to establish correct surface point correspondences.

4.1. Meso-structure reconstruction

A surface to be generated by a 3D reconstruction algorithm is typically defined to minimize some error functional. For a measured BTF, such functionals can be derived from the physical properties of BRDFs, namely Helmholtz reciprocity and lambertian law of cosines:

$$\begin{aligned} \rho(\mathbf{k}_1, \mathbf{k}_2) &= \rho(\mathbf{k}_2, \mathbf{k}_1) \\ L_r(\mathbf{k}_1, \mathbf{k}_2) &= \rho(\mathbf{k}_1, \mathbf{k}_2)(\mathbf{k}_1\mathbf{n})L_i, \end{aligned}$$

where L_i is the irradiance which is constant for all $\mathbf{k}_1$, $\mathbf{k}_2$. Note, that most BTFs that are stored as regular images in a database also do not contain linear luminance so these have to be converted into linear values with an inverse gamma correction before a reconstruction.

In addition to parallax effects, the meso-structure also influences the local surface normal. Therefore, can neither the BRDFs be directly extracted from the input images, that only contain the reflected radiance $L_r(\mathbf{k}_1, \mathbf{k}_2)$, nor can the reciprocity be used as error functional alone. These properties are also the basis for the Helmholtz stereopsis [ZBK02] where the depth is defined such that the difference to a reciprocal BRDF with unknown normal is minimized. While this works well for materials with texture – i.e. per pixel color variation – no depth values can be reconstructed for uniform materials. For such surfaces only global optimization methods taking into account the local neighborhood of the point can produce reasonable reconstructions.

Assuming that the meso-structure is locally planar, the cosine factor can be canceled out by matching the local neighborhood of a point with the normalized cross correlation of the log-scale radiance between $L_r(\mathbf{k}_1, \mathbf{k}_2)$ and $L_r(\mathbf{k}_2, \mathbf{k}_1)$. The similarity S at point $\mathbf{x}$ is then:

$$S(\mathbf{x}) = \sum_{\mathbf{k}_1, \mathbf{k}_2} \frac{\mathbf{u}(\mathbf{x}, \mathbf{k}_1, \mathbf{k}_2) \cdot \mathbf{u}(\mathbf{x}, \mathbf{k}_2, \mathbf{k}_1)}{\|\mathbf{u}(\mathbf{x}, \mathbf{k}_1, \mathbf{k}_2)\| \|\mathbf{u}(\mathbf{x}, \mathbf{k}_2, \mathbf{k}_1)\|},$$

where $\mathbf{u}(\mathbf{x}, \mathbf{k}_1, \mathbf{k}_2)$ is the log-mapped and mean-removed neighborhood of $\mathbf{x}$ projected into the input image $(\mathbf{k}_1, \mathbf{k}_2)$ written as a column vector:

$$\mathbf{u}_i(\mathbf{x}, \mathbf{k}_1, \mathbf{k}_2) = \log L(\mathbf{p}_i(\mathbf{x}), \mathbf{k}_1, \mathbf{k}_2) - \bar{L}_{\log}(\mathbf{x}, \mathbf{k}_1, \mathbf{k}_2),$$

with $\bar{L}_{\log}(\mathbf{x}, \mathbf{k}_1, \mathbf{k}_2)$ being the mean log-radiance of the neighborhood $\mathbf{p}_i(\mathbf{x})$ of $\mathbf{x}$ in input image $(\mathbf{k}_1, \mathbf{k}_2)$.

A consistent depth reconstruction for the whole BTF is obtained using graph-cut stereo [RC98]. For a set of discrete depth values, the similarity is calculated per texel and a 3-dimensional graph is constructed where each texel in every layer is connected to its eight neighbors and to itself in the next and previous layer. The weight of the edges corresponds to the sum of the similarity of both nodes, where vertical connections are weighted with a smoothness factor k. The minimal cut separating top and bottom layer then yields a reasonable approximation of the parallax and approximately solves the point correspondence problem.

The local normal at $\mathbf{x}$ now minimizes the difference between $\rho(\mathbf{k}_1, \mathbf{k}_2)$ and $\rho(\mathbf{k}_2, \mathbf{k}_1)$. Since radiance values are exponentially distributed, ρ is again transformed into log-space:

$$E(\mathbf{x}) = \sum_{\mathbf{k}_1, \mathbf{k}_2} v(\mathbf{x}, \mathbf{k}_1, \mathbf{k}_2) \left(\log \frac{\rho(\mathbf{x}, \mathbf{k}_1, \mathbf{k}_2)}{\rho(\mathbf{x}, \mathbf{k}_2, \mathbf{k}_1)} \right)^2,$$

with

$$\rho(\mathbf{x}, \mathbf{k}_i, \mathbf{k}_j) = \frac{L(\mathbf{x}, \mathbf{k}_i, \mathbf{k}_j)}{\mathbf{k}_i \mathbf{n}(\mathbf{x})},$$

where $v(\mathbf{x}, \mathbf{k}_1, \mathbf{k}_2)$ is one if $\mathbf{x}$ is visible from both $\mathbf{k}_1$ and $\mathbf{k}_2$, and zero otherwise. The optimal local normal $\mathbf{n}$ at $\mathbf{x}$ is found using Levenberg-Marquardt optimization [Lou04]. Convergence to a local minimum – a common problem when solving non-linear equation systems – cannot occur since the error functional has only a single minimum. Also, Levenberg-Marquardt has the fastest convergence-rate. Finally, depth values and normals are combined using the method of Nehab et al. [NRDR05]. Due to the normal reconstruction from the reciprocity, a global normal shift cannot occur and thus the normal correction is skipped and only optimize the depth values with the extracted normals. However, as the normal map is not an integral part of the representation, new normals are calculated from the depth map as described in Section 3 for the subsequent decomposition steps.

4.2. Estimation of the BRDF parameters

When the meso-structure is fixed, the micro-structure can be reconstructed per surface point by mapping the input images onto the generated height field and transforming the light and view directions into the local coordinate systems. Note that this transformation also includes the removal of occluded and shadowed samples since they are specifically handled within the Parallax Occlusion Mapping algorithm. Using the d-BRDF model with Schlick's Fresnel approximation, three parameters need to be determined: the diffuse color c_d, the NDF $p(\mathbf{h})$, and the reflectance at normal incidence r_0. As each of these can be calculated

analytically if the other two are known, an alternating least squares method is used, starting with c_d, to find all of them.

Solving Equation 2 for the diffuse color yields the following equation for a least square fit of the RGB-color c_d:

$$c_d = \frac{\sum_{\mathbf{k}_1,\mathbf{k}_2} \lambda_d(\mathbf{k}_1,\mathbf{k}_2)\delta_d(\mathbf{k}_1,\mathbf{k}_2)}{\sum_{\mathbf{k}_1,\mathbf{k}_2} \lambda_d(\mathbf{k}_1,\mathbf{k}_2)^2},$$

where

$$\begin{aligned}\delta_d(\mathbf{k}_1,\mathbf{k}_2) &= \frac{L_r(\mathbf{k}_1,\mathbf{k}_2)}{L_i} - \frac{c_s p(\mathbf{h}) F(\mathbf{kh})(\mathbf{k}_1\mathbf{n})}{\mathbf{k}_1\mathbf{n} + \mathbf{k}_2\mathbf{n} - (\mathbf{k}_1\mathbf{n})(\mathbf{k}_2\mathbf{n})} \\ \lambda_d(\mathbf{k}_1,\mathbf{k}_2) &= \frac{\mathbf{k}_1\mathbf{n}}{\pi},\end{aligned}$$

with the constraint that no color channel can be below zero or above one due to energy conservation. The normal distribution $p(\mathbf{h})$ is extracted similar to [NDM05] where the least squares fit is:

$$p(\mathbf{h}_i) = \frac{\sum_{\mathbf{k}_1+\mathbf{k}_2=\mathbf{h}_i} \lambda_p(\mathbf{k}_1,\mathbf{k}_2)\delta_p(\mathbf{k}_1,\mathbf{k}_2)}{\sum_{\mathbf{k}_1+\mathbf{k}_2=\mathbf{h}_i} \left(\lambda_p(\mathbf{k}_1,\mathbf{k}_2)\right)^2},$$

with

$$\begin{aligned}\delta_p(\mathbf{k}_1,\mathbf{k}_2) &= \frac{L_r(\mathbf{k}_1,\mathbf{k}_2)}{L_i} - \frac{c_d}{\pi} \\ \lambda_p(\mathbf{k}_1,\mathbf{k}_2) &= \frac{c_s F(\mathbf{kh})(\mathbf{k}_1\mathbf{n})}{\mathbf{k}_1\mathbf{n} + \mathbf{k}_2\mathbf{n} - (\mathbf{k}_1\mathbf{n})(\mathbf{k}_2\mathbf{n})}\end{aligned}$$

From the probability for discrete halfway vectors $\mathbf{h}_i$ the parabolic map over the upper hemisphere is calculated using the push-pull algorithm [GGSC96]. Then the specular intensity is extracted with:

$$c_s = \frac{\sum_{\mathbf{k}_1,\mathbf{k}_2} \lambda_s(\mathbf{k}_1,\mathbf{k}_2)\delta_s(\mathbf{k}_1,\mathbf{k}_2)}{\sum_{\mathbf{k}_1,\mathbf{k}_2} \lambda_s(\mathbf{k}_1,\mathbf{k}_2)^2},$$

where

$$\begin{aligned}\delta_s(\mathbf{k}_1,\mathbf{k}_2) &= \frac{L_r(\mathbf{k}_1,\mathbf{k}_2)}{L_i} - \frac{c_d}{\pi} \\ \lambda_s(\mathbf{k}_1,\mathbf{k}_2) &= \frac{p(\mathbf{h}) F(\mathbf{kh})(\mathbf{k}_1\mathbf{n})}{\mathbf{k}_1\mathbf{n} + \mathbf{k}_2\mathbf{n} - (\mathbf{k}_1\mathbf{n})(\mathbf{k}_2\mathbf{n})}\end{aligned}$$

Energy conservation is enforced for the NDF in three steps: First, the probability for any halfway vector cannot be larger than the reciprocal of it's differential area, second, the sum of all probabilities (i.e. the integral over the hemisphere)

cannot exceed one, and third, c_s is restricted to at most $1 - c_d$. Finally, the reflectance at normal incidence r_0 (Equation 2) is calculated based on the current estimates for c_d and $p(\mathbf{h})$:

$$r_0 = \frac{\sum_{\mathbf{k}_1,\mathbf{k}_2} \lambda_f(\mathbf{k}_1,\mathbf{k}_2)\delta_f(\mathbf{k}_1,\mathbf{k}_2)}{\sum_{\mathbf{k}_1,\mathbf{k}_2} \left(\lambda_f(\mathbf{k}_1,\mathbf{k}_2)\right)^2},$$

where

$$\begin{aligned}\delta_f(\mathbf{k}_1,\mathbf{k}_2) &= \frac{L_r(\mathbf{k}_1,\mathbf{k}_2)}{L_i} - \left(\frac{c_d}{\pi} + c_s p(\mathbf{h})(1-\mathbf{kh})^5\right)\\ \lambda_f(\mathbf{k}_1,\mathbf{k}_2) &= \left(1-(1-\mathbf{kh})^5\right) c_s p(\mathbf{h}(\mathbf{k}_1\mathbf{n}))\end{aligned}$$

Up to now, the BRDF approximation process minimizes the root mean square error, whereas radiance values are exponentially distributed and thus the logarithmic error needs to be minimized. This can be accomplished by weighting the contribution of each radiance value $L_r(\mathbf{k}_1,\mathbf{k}_2)$ with the derivative of the logarithmic mapping function during summation of the λ's and δ's, which simply is the reciprocal of the radiance.

4.3. Clustering

To combine the independent per-texel BRDFs into n basis materials, those texels need to be found that have similar microfacet distributions. As the local normal is fixed, the only remaining degree of freedom is the rotation about the normal. When the NDF is parameterized over spherical coordinates (θ, ϕ), the rotation becomes a shift in ϕ and the offset can efficiently be found utilizing the Fourier shift theorem: Let F and G be the Fourier transformations of the NDFs f and g. Then the maximum of the inversely transformed function h of $H = F\bar{G}$ lies at $(0, \phi_0)$ with ϕ_0 being the rotation between f and g. Based on this pairwise alignment, a k-means clustering of NDFs is performed by aligning each NDF to the cluster center before distance calculation and summation. The rotation about the normals to the cluster centers is then equivalent to the orientation map of the g-BRDF.

As the clustering operates on each texel independently, the boundaries between different base material tend to become noisy. To reduce this boundary noise, an additional relaxation labeling [Gen89] is performed after clustering. The probability of an NDF to belong to a cluster is defined as the reciprocal of the distance between the two and normalize the total probability of each NDF to one. Then the relaxation labeling is performed to pull the probabilities of each NDF into the direction of those of its neighbors. The required number of clusters is automatically determined by calculating the separation index $SI(n)$ for all possible number of clusters n up to a given maximum and choosing the one for which $SI(n)$ is minimal. The separation index is defined as the ratio of the root mean square difference between the per-texel NDFs and their corresponding cluster centers to the minimum difference between two cluster centers. Finally,

the BRDF parameters are estimated for the complete BTF as described above, where the NDF $p_i(\mathbf{h})$ is calculated per cluster i and $c_d(\mathbf{x})$ and $r_0(\mathbf{x})$ per texel.

5 Results

For the BTF decomposition several BTFs from a publicly available database [SSK03] have been used. These measured BTFs have the advantage that the sample set is dense and spatially registered, that is the directions of incoming and outgoing light as well as camera parameters are known for each image. Other measured BTFs, like the pioneering CUReT database [DvGNK99] are unfortunately very sparse and not spatially registered such that they cannot be used without prior resampling. Another drawback of the CUReT database is that it contains some graphical errors, caused by frame-grabber artifacts or reflections of the robot sample holder plate visible in the raw data. Similarly, the database of Koudelka et al. [KMBK03] lacks reciprocal image pairs and would also require resampling.

The decomposition process needs approximately 3 hours for a $256^2 \times 81^2$ BTF on an Intel Core 2 Duo running at 2.4 GHz and the runtime is linear in the number of pixels contained in the dataset. The best results have been achieved using the following constants for decomposition: irradiance $L_i = 3$, cross-correlation window size 5×5, graph-cut smoothness $k = 0.1$, and depth weight $\lambda = 0.2$. The RMS difference of the decomposed BTF to the original one is 6 to 12 times the just noticeable difference (JND) in the CIELab color space according to the ΔE_{00} difference formula. This is slightly higher than for the currently best compression algorithm [MMK03] with an error of 5 to 7 JND (see Table 6.1), but they need 17 megabytes texture memory while the g-BRDF requires 1.5 megabytes only.

For comparison of depth map reconstruction quality the algorithms of [RC98] and [ZBK02] as shown in Figure 6.6 were implemented. The NCC does not clearly separate between the two different height layers of the shown BTF. Helmholtz Stereopsis performs better, but still produces some artifacts and lack of detail without optimization.

material	g-BRDF (1.5 MB)	[MMK03] (17 MB)
corduroy	10.09 JND	5.98 JND
impalla	12.24 JND	6.70 JND
proposte	12.41 JND	7.06 JND
wallpaper	5.87 JND	5.02 JND
wool	8.94 JND	5.36 JND

Table 6.1.: Root mean square CIELab ΔE_{00} color difference in JND for different BTFs.

The resulting g-BRDF textures for the six $256^2 \times 81^2$ BTFs from the database are shown in Figure 6.7-6.9 along with renderings from the compressed representation. Approximately 90% of the total rendering time is required for the two trace functions of the parallax occlusion mapping [Tat06]. Therefore, the overall performance is similar to Parallax Occlusion Mapping rendering alone which

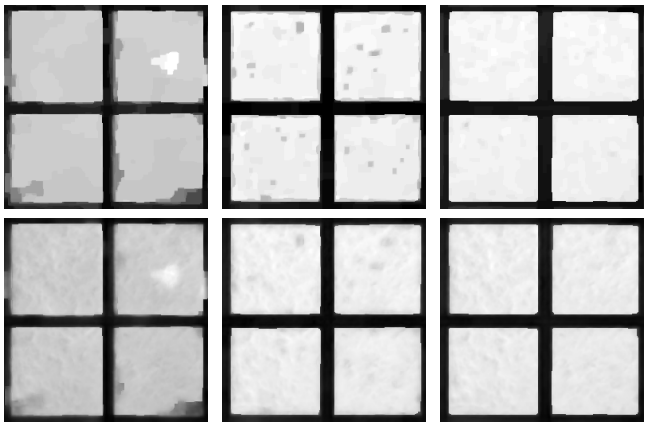

Figure 6.6.: Left to right: Depth Maps created with Normalized Cross Correlation, Helmholtz Stereopsis and the presented approach. Bottom row: Depth maps after global optimization.

achieves about 200 fps on current graphics hardware at full screen coverage on a 1280×1024 display. In addition, a visual comparison of the results with images generated from the uncompressed BTF data is given: on the right side of Figure 6.7-6.9 a closeup is shown (top) in comparison to using the original BTF data. Below, a difference image calculated in CIELab color space and exaggerated by a factor of two is shown to depict the visual difference.

The main difference to the captured BTF data lies in shadow regions, especially for the impalla BTF (Figure 6.7 bottom) due to the missing scattering. However, the image generated from the uncompressed impalla BTF exhibits significant blurring due to the naive image-based interpolation. For almost all BTFs, there is some chromatic difference – especially for corduroy (Figure 6.7 top) and wool Figure 6.9 bottom) – since microfacette models cannot model color changes when the halfway vector is constant. Altogether, the comparisons suggest that a more complex BRDF representation, that includes scattering, is required to accurately describe all effects of a BTF. Nevertheless, the rather simple d-BRDF model still produces convincing results.

To show how intuitively the g-BRDF representation can be edited, four different tasks were performed (Figure 6.10-6.13). In Figure 6.10 the orientation of the fibres of the corduroy material is inverted. For this purpose the logo is pasted into the orientation map. Then, the fibres below the logo are rotated by 180 degrees.

In Figure 6.11 the color of the wool is changed to red. For this the color channels of the NDFs and the diffuse texture were switched. To add some additional lint, the grazing angles of the NDF are selected and a light pink reflection is added.

In Figure 6.12 first the logo is copied into the diffuse texture. Then, the depthmap is changed so that the logo is engraved into the surface. Finally, a new BRDF for the paint of the logo is added. For this purpose, a synthetic specular

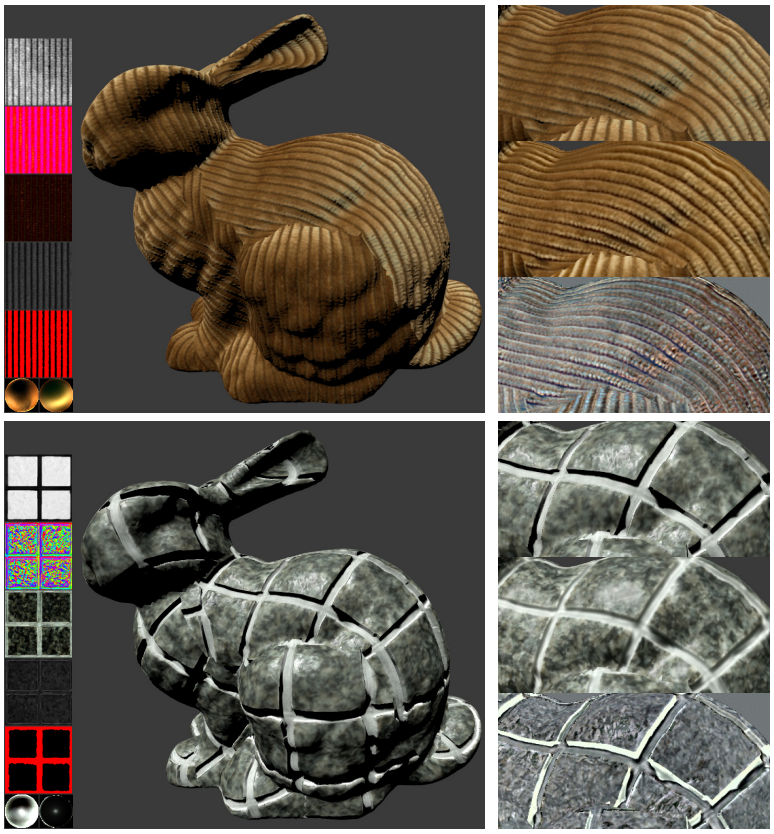

Figure 6.7.: Rendering of the corduroy (top) and impalla (bottom) BTFs from their g-BRDF representation shown on the insets. On the right a closeup is compared to renderings from the original BTF data together with the CIELab difference.

NDF is added. The index map is then altered accordingly in order to address the new NDF. Finally, color and Fresnel map are modified to change reflection behavior.

In the last example, a BTF was created from a single image. First the image was copied into the diffuse texture. Then, the NDF is created by adding some lint and a minor specular highlight. The rotation map is set to 45 degrees. The depthmap is created by grayscaling the diffuse texture.

Each of these tasks was performed within less than four minutes and the complete editing process can be seen in the accompanying video. A detailed list of the editing times is shown in Table 6.2.

6 Discussion and Limitations

The proposed texture maps for meso-geometry and light interaction can easily be edited with standard image operations. While previous approaches require special software to modify raw BTF input data, the proposed representation allows a designer to use his favorite image-editing tool and thus remain in a working

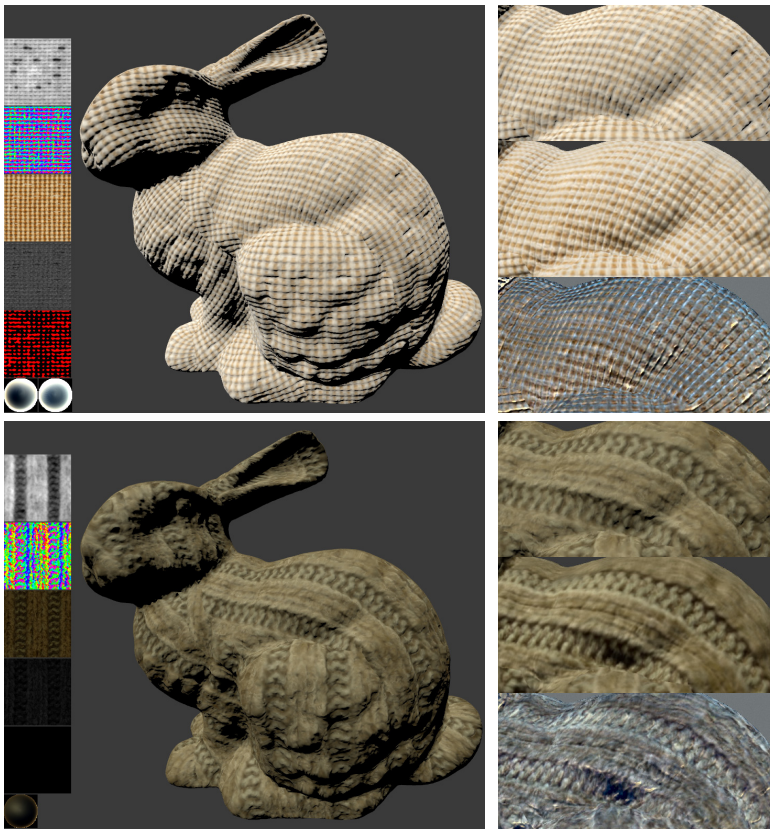

Figure 6.8.: Rendering of the proposte (top) and pulli (bottom) BTFs from their g-BRDF representation shown on the insets. On the right a closeup is compared to renderings from the original BTF data together with the CIELab difference.

task	time (mm:ss)
meso-structure editing	0:39
BRDF editing	1:32
adding synthetic material	2:52
creating a complete g-BRDF	3:51

Table 6.2.: Editing times for example tasks.

environment he is accommodated to. Another advantage is that every change of the maps directly affects the BTF's appearance and can be reviewed in a shader editor without latency, as no further compression is required. This way, a designer has full control over the final result at any time. In this approach it has also be shown that measured data can be seamlessly combined with user generated BRDFs and even the construction of a BTF from a single image is possible. Furthermore, the presented representation is able to represent a measured BTF with about 1.5 megabytes only.

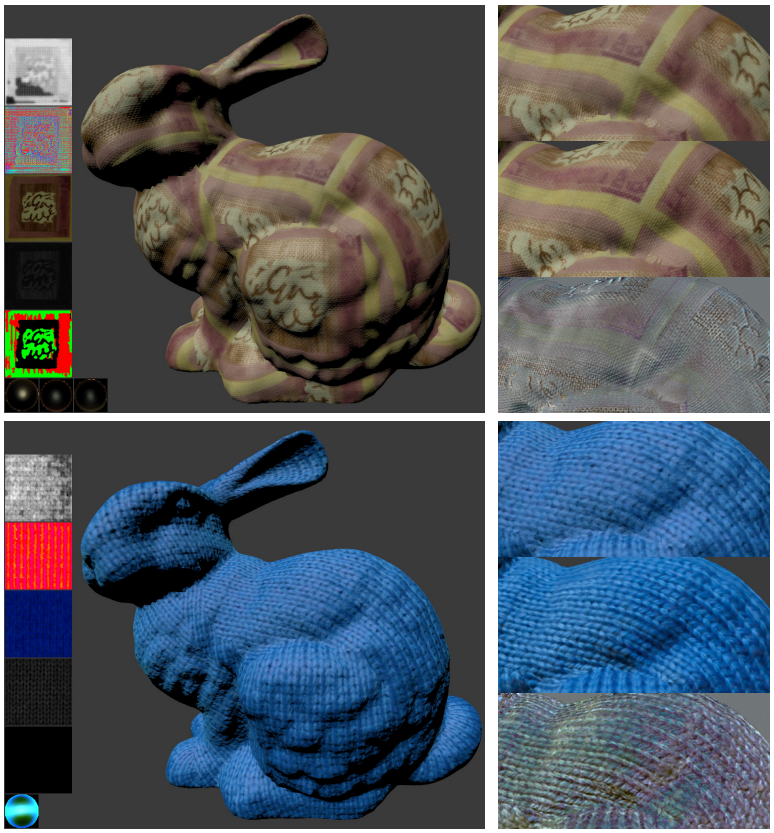

Figure 6.9.: Rendering of the wallpaper (top) and wool (bottom) BTFs from their g-BRDF representation shown on the insets. On the right a closeup is compared to renderings from the original BTF data together with the CIELab difference.

The main limitation of this approach clearly lies in the choice of the d-BRDF model which totally neglects subsurface-scattering and inter-reflections. With the new method, shadows are stored only implicitly and reconstructed using depth map and parallax occlusion mapping. The resulting harsh look could be compensated by simulating light transport on the surface, a task that has been done on meshes in real-time in [HJ07]. Another limitation is that only a single Normal Distribution Function was used for each material. This way much of the original variation and realism of a surface is lost. A solution to this would be to store a set of the most distinct NDFs for each material. Also, the use of a single depth map implies the assumption that the surface is opaque. This limitation could be lifted by using a multi-layer model or a volumetric representation similar to [MK06]. Such representations however significantly increase the complexity and thus hinder intuitive editing.

The realism could be further improved by using an appropriate BSSRDF model [JMLH01], but then the fitting of the parameters would become more difficult. An open question is however how scattering can be extracted from

Figure 6.10.: Editing the meso-structure of the corduroy BTF.

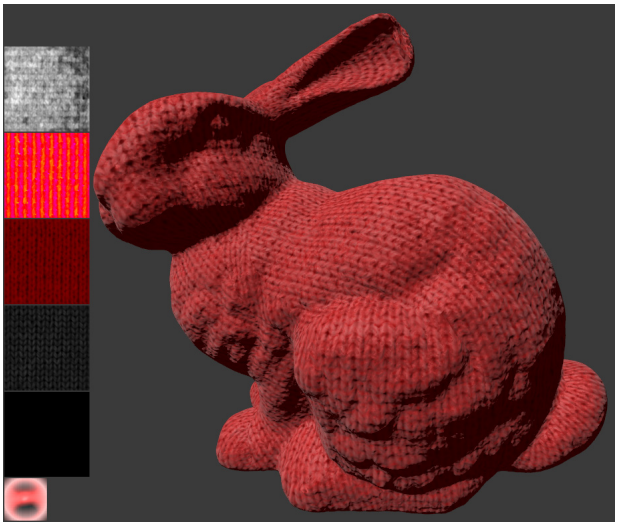

Figure 6.11.: Editing the BRDF of the wool BTF.

the input images as most acquisition systems do not even use back-lighting, although Lensch et al. [LGB+02] proposed one that uses laser beams to capture subsurface-scattering BTF data. Once its parameters are found, real-time rendering is possible using a screen-space approximation [BC06].

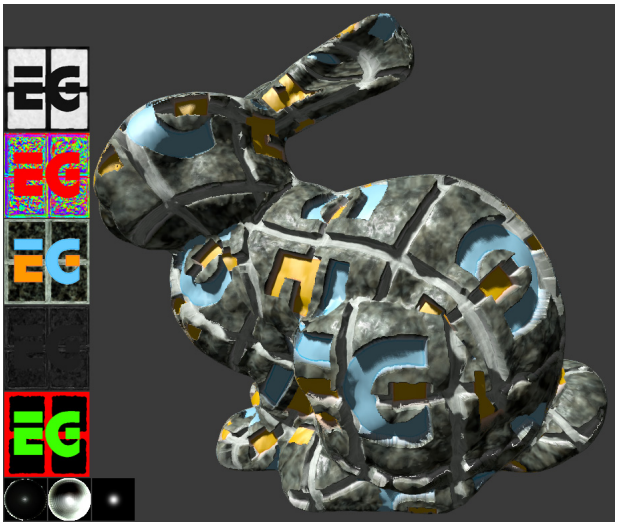

Figure 6.12.: Adding a synthetic material to the impalla BTF.

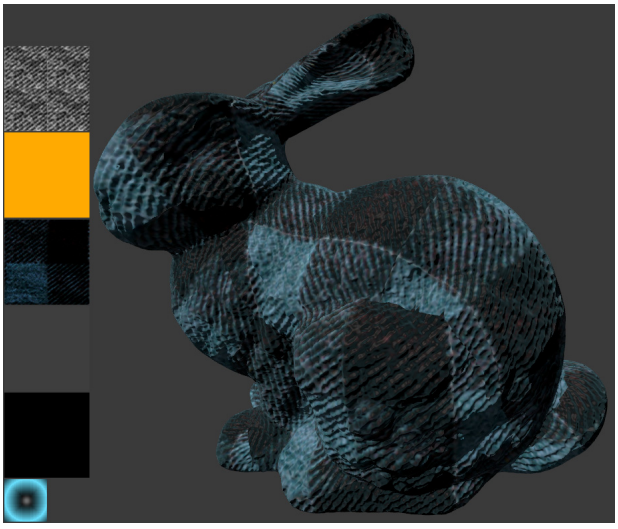

Figure 6.13.: Creating a BTF from a diffuse texture.

Chapter 7.

Realistic Subsurface-Scattering Post-Processing

Unfortunately, real global illumination and interactivity are usually incompatible. Solving the rendering equation using global illumination techniques requires minutes to hours to generate a single image. On the other side it has been observed that for many purposes, global illumination solutions do not need to be precise, but only plausible. Relatively new to the graphics community is the computation of global illumination features such as subsurface scattering, ambient occlusion or indirect lighting in screen space. Screen space algorithms are executed purely on the computer's GPU and implemented as pixel shaders. The basic idea is to generate arbitrary scene information in multiple, consecutive passes. The final scene is then rendered in a deferred pass using the previously generated render targets. Advantages of screen space algorithms are that they are independent of scene complexity, work with dynamic scenes and in the same consistent way for every pixel on the screen.

Screen space scattering algorithms are, generally spoken, an extension of shadow maps, which store in each pixel the distance of the closest surface from the light's position. This information is used in the rendering pass of the final image to determine whether a point lies in shadow by comparing its light source with the shadow map distance. This approach aims to improve the technique proposed by Dachsbacher et al. [DS03] by sampling the mipmap levels of the shadow maps adaptively in dependence of the underlying geometry. Points of low variation do only contribute little to the illumination integral, so a coarser sampling scheme needs to be applied. The base observation is that only a few representative points need to be sampled in very homogeneous regions. While this idea could also be applied to ambient occlusion, indirect lighting and the calculation of caustics, this approach focuses on subsurface scattering (see Figure 10.9).

Subsurface scattering summarizes all light reflection processes that occur underneath the optical boundary of a surface and is used whenever light enters a translucent material such as wax, milk or skin. In case of a completely transparent material, the light is reflected or refracted at a given surface point, enters the object and leaves at another point. If light is scattered inside the material, we distinguish between so called single scattering, where each non-absorbed photon is scattered once – leading to a deterministic path from the light source to the camera – and multiple scattering where each photon is scattered multiple times

which results in diffuse light exiting the surface. In the remainder of this chapter, the focus lies on the multiple scattering term since the contribution of single scattering is only significant for relatively opaque materials.

Figure 7.1.: Phlegmatic dragon with quasi-homogeneous marble rendered in real-time with the novel approach.

1 Diffuse Dipole Approximation

The well known BRDF is a simplification of the more general BSSRDF, because it assumes that light entering a material leaves at the same position. This approximation fails for all translucent materials, which exhibit significant light transport below the surface (see Figure 7.2).

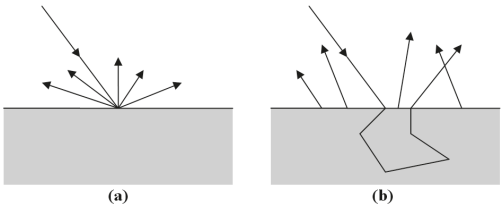

Figure 7.2.: Light leaves at the entering point in a BRDF (a), while it leaves distributed over an area in a BSSRDF (b)

Given a BSSRDF, the outgoing radiance is computed by integrating the hemisphere of incident radiance over incoming directions *and* an the surface area A:

$$L_o(\mathbf{x_{out}}, \omega_{\mathbf{out}}) = \int_A \int_\Omega S(\mathbf{x}_{in}, \omega_{in}, \mathbf{x}_{out}, \omega_{out}) \, L_i(\mathbf{x}_{in}, \omega_{in}) d\omega_{in} dA(\mathbf{x}_{in}), \tag{7.1}$$

where $S(\mathbf{x}_{in}, \omega_{in}, \mathbf{x}_{out}, \omega_{out})$ is the BSSRDF.

One of the most widely adapted ideas is that the exiting area of the multiple scattering can be approximated with a dipole point light source, called diffuse dipole approximation. The dipole method simply places two point light sources near the surface, with one light source beneath the surface and another negative light source located above the surface (see Figure 7.3). Note that this approximation delivers good results only for flat materials and shows artifacts for highly curved surface where the dipole source inside the material might appear too bright.

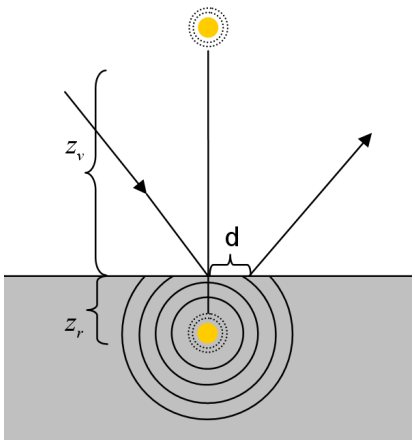

Figure 7.3.: Dipole approximation of subsurface scattering with one light source beneath the surface at distance z_r and one negative above at distance z_v

2 Translucent Shadow Maps

To approximate the integral from Equation 7.1, the shadow map is not only sampled at a single position, but over a larger area. The idea of Translucent Shadow Maps is to densely sample the local neighborhood and use a few additional distant samples in appropriate mipmap levels of the shadow map. Since it is based on the dipole approximation, TSMs simulate multiple scattering of light only and are therefore restricted to materials with high scattering, such as marble, milk or skin. In this case any relation between the directions of incident and exitant light is lost.

The first step is the evaluation whether the ray is scattered into the material or reflected due to the Fresnel term F_t. For an irradiance $I(\omega_i)$ from a point or parallel light this is:

$$E(\mathbf{x}_{in}) = F_t(\eta, \omega_{in}) \mid N(\mathbf{x}_{in}) \cdot \omega_{in} \mid I(\omega_{in}),$$

where η is the optical density of the material. The Fresnel term F_t can be approximated using Schlick's approximation.

In the second case light diffuses into the material, which is approximated with the subsurface diffusion function $R_d(\mathbf{x}_{in} - \mathbf{x}_{out}, \mathbf{n}_{in})$, where $\mathbf{x}_{in}, \mathbf{x}_{out} \in S$ are points on the surface. R_d describes the transport of light entering the object at $\mathbf{x}_{in}$ and exiting at $\mathbf{x}_{out}$ as is explained in [JMLH01]. This does not only depend on the distance between $\mathbf{x}_{in}$ and $\mathbf{x}_{out}$, but also on the surface normal $\mathbf{x}_{in}$ at $\mathbf{x}_{in}$:

$$B(\mathbf{x}_{out}) = \int_S E(\mathbf{x}_{in}) R_d(\mathbf{x}_{in} - \mathbf{x}_{out}, \mathbf{n}_{in}) d\mathbf{x}_{in}$$

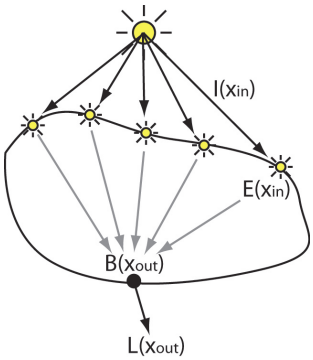

Figure 7.4.: The amount of light leaving at $\mathbf{x}_{out}$ is obtained by integrating the surface as seen from the light source

Note that this integral is only valid for convex objects, i.e. the path from $\mathbf{x}_{in}$ to $\mathbf{x}_{out}$ lies completely inside the object. When the light leaves the object, the Fresnel term has to be computed again because internal reflection can occur:

$$L(\mathbf{x}_{out}, \omega_{out}) = \frac{1}{\pi} F_t(\eta, \omega_{out}) B(\mathbf{x}_{out})$$

With TSMs, the integration process is separated into two passes: In the first pass Equation 2 is computed during the generation of the TSM. Additionally to the depth value, a TSM stores irradiance $E(\mathbf{x}_{in})$ and the surface normal $\mathbf{n}_{in}$ with every pixel. With this information, the integral in Equation 2 can be computed during rendering of the user's view as a filter of the TSM color values, with weights given by R_d. To speed up the summation, R_d is pre-calculated and stored in a texture map.

3 Adaptive Sampling

Though TSMs achieve an appealing simulation of local scattering, the low number of distant samples is not sufficient for highly scattering material and introduces banding and blurring artifacts. In contrast to this fixed sample kernel, the proposed technique adaptively approximates the integral from Equation 2 and prevents such artifacts by adding more samples if required. The precomputed large kernel used for RSMs on the other hand takes far too many samples in many situations and can thus not be as efficient as an adaptive technique.

To adaptively calculate the multi scattering integral, Equation 2 is approximated by partitioning the surface into regions where $R(\mathbf{x}_{in} - \mathbf{x}_{out}, \mathbf{n}_{in})$ is almost constant. This way, the light arriving at x_{out} below the surface becomes:

$$B(\mathbf{x}_{out}) = \sum_i R_d(\mathbf{x}_{out} - \mathbf{x}_i, \mathbf{n}_i) \int_{S_i} E(\mathbf{x}_{in}) d\mathbf{x}_{in},$$

where $\mathbf{x}_i$ and $\mathbf{n}_i$ are the average position and surface normal inside the region S_i.

Since an optimal partitioning is different for every $\mathbf{x}_{out}$, the shadow map is hierarchically subdivided using a quad-tree and integrate $E(\mathbf{x}_{in})$ for every node. This can be performed efficiently by summing up the incoming energy of each child node along with the position and surface normal. After generating the quad tree, the partitioning for every $\mathbf{x}_{out}$ can be calculated by traversing the quad-tree and subdividing a region if the approximation error supersedes a given threshold ε_{max}. Figure 7.5 shows a typical sampling scheme for subsurface scattering. As the contribution declines with the distance, the clustering can be performed more aggressively with increasing distance from $\mathbf{x}_{out}$.

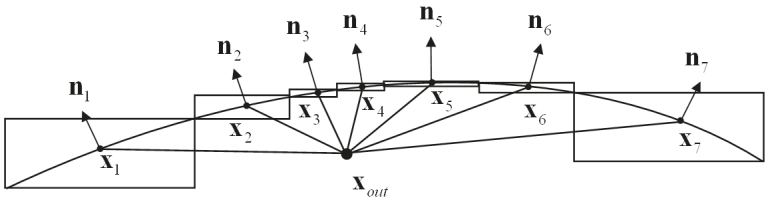

Figure 7.5.: Typical partitioning when approximating $B(\mathbf{x}_{out})$.

3.1. Approximation error estimation

Finding an optimal partitioning implies that an estimate for the approximation error for a given exitant point $\mathbf{x}_{out}$ and a lit surface patch S_i have been found. A straightforward approach would now be to simply set up an estimate for a given patch size δ_i of S_i with respect to $\mathbf{x}_{in} - \mathbf{x}_{out}$ and $\mathbf{n}_{in}$. This however would mean introducing a seven dimensional function to estimate the error.

Since the subsurface diffusion function $R_d(\mathbf{x}_{in} - \mathbf{x}_{out}, \mathbf{n}_{in})$ is isotropic, it's dimensionality can be reduced to two. This can be accomplished by first rotating $\mathbf{x}_{in} - \mathbf{x}_{out}$ into the local coordinate system of $\mathbf{n}_{in}$ which already reduces the dimensionality to three and then exploiting the fact that $R_d(\mathbf{x}_{in} - \mathbf{x}_{out}, \mathbf{n}_{in})$ is isotropic to arrive at a two-dimensional function. This means that it only depends on the distance d between $\mathbf{x}_{in}$ and $\mathbf{x}_{out}$ and the angle θ between surface normal and the distance vector.

Together with the maximum deviation δ_i from the average distance d_i, this now leads to a three dimensional function. The maximum difference – required for a conservative approximation – can now occur in one of the following configurations with only two surface points $\mathbf{x}_{in_1}$ and $\mathbf{x}_{in_2}$:

1. The points lie on the line between $\mathbf{x}_i$ and $\mathbf{x}_{out}$ on opposite sides of $\mathbf{x}_i$ at a distance of δ_i each.

2. If $\delta_i > d$, then also the same configuration is considered, but with a distance of d between the surface points and $\mathbf{x}_i$.

3. The points are orthogonal to the line $\mathbf{x}_i$, $\mathbf{x}_{out}$ at a distance of δ_i and orthogonal to $\mathbf{n}_i$.

4. The points lie in the direction of $\mathbf{n}_i$ and $-\mathbf{n}_i$ at a distance of δ_i each.

For two surface points in a quad tree node evaluating these configurations is sufficient since the dipole diffusion function is smooth and has only a single maximum. For more points this estimation is not conservative but for smooth surfaces it is a reasonable approximation of the upper error bound.

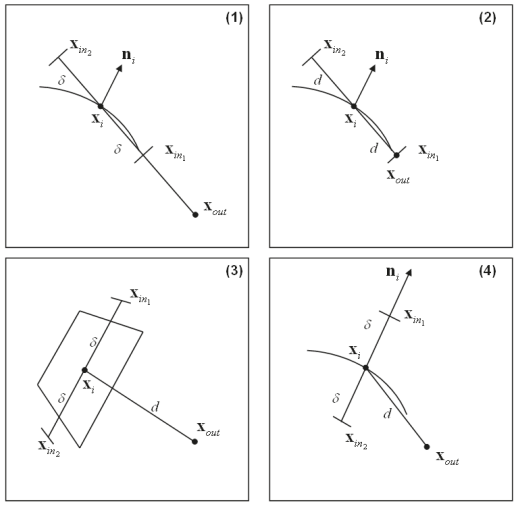

Figure 7.6.: Configurations evaluated to estimate the maximum approximation error.

Figure 7.6 shows the four configurations described above. In any of these the approximation error ε is:

$$\varepsilon(d,\delta,\theta) = R_d(\mathbf{x}_i - \mathbf{x}_{out}, \mathbf{n}_i) - \frac{1}{2}(R_1 + R_2)$$
$$R_j = R_d(\mathbf{x}_{in_j} - \mathbf{x}_{out}, \mathbf{n}_i),$$

where the final value required for the traversal decision is the maximum approximation error of all configurations.

3.2. Screen space subsampling

As already noted by Dachsbacher and Stamminger [DS03], the subsurface light transport is of relatively low frequency. Translated into image space, this means

that the difference between neighboring pixels is low except at depth discontinuities, in regions of high surface normal variation, and near shadow boundaries. This observation allows to significantly speed up the calculation of B_{out} for each image pixel by first generating a low-resolution texture map – the subsurface light transport map – followed by a recursive upsampling where only for pixels near depth discontinuities, with high normal variation (i.e. the length of the filtered normal is less than a threshold δ_n), and shadow boundaries B_{out} is calculated using equation 3.

While depth discontinuities and normal variation are simple to extract from the mipmap levels of the depth and normal map of the rendered model (c.f. [DS05]), shadow boundaries are much harder to detect since their effect is not locally bound in image space. While it would be possible to mark shadow boundaries in a low resolution image and use this marking to determine if a pixel lies inside a shadow boundary, it is much more simple and efficient to compare the color of those pixels in the lower resolution subsurface light transport map from which the current one is interpolated. If these are sufficiently similar, the pixel either does not lie on a shadow boundary or the boundary is so smooth that the subsurface light transport can be safely interpolated. Note that in the second case, basing the decision on an exact determination of shadow boundaries would unnecessarily lead to a calculation of B_{out} for that pixel.

4 Implementation

The approach was implemented using Cg on an GeForce 8800 GTX. 16 bit floating point render targets have been used together with programmable vertex and fragment processing using the Shader Model 3.0 instruction set.

The algorithm consists of three main passes, where the model is rendered once in the first and once in the third pass. In the first pass all data necessary to calculate the subsurface light transport is generated and stored in textures as seen from the light source. The required data is the surface position, normal, and incident light. The three texture maps are generated in a single rendering pass using multi-render-targets.

In the next pass, the quad tree data required for the integration is constructed. In addition to mipmap levels for position, normal, and incident light, the maximum distance δ needs to be calculated which requires two additional mipmap pyramids to store the minimal and maximal points of the bounding boxes for clustered positions $\mathbf{x}_{in}$. To reduce the number of texture units, all four textures are stored in a single mipmap pyramid shown in figure 7.7.

To speed up rendering, $R_d(\mathbf{x}, \mathbf{n}_{in})$ are stored additionally, where $\mathbf{x} = \mathbf{x}_{in} - \mathbf{x}_{out}$ and $\varepsilon(d, \delta)$ in two texture maps. In contrast to [DS03] this approach stores $R_d(\mathbf{x}, \mathbf{n}_{in})$ in a 2D texture that can be recomputed directly on the GPU parameterized as $R_d(d, \mathbf{x} \cdot \mathbf{n}_{in})$. Since the parameters of both functions are within $[0, \infty]$, they are mapped into $[0, 1]$ with $t' = \frac{t}{t+1}$. Since both are low frequency, a resolution of 64 in each dimension is sufficient.

In the final rendering pass, B_{out} is calculated for each screen pixel and used to compute the total outgoing light at that point. As explained in section 3.2, first

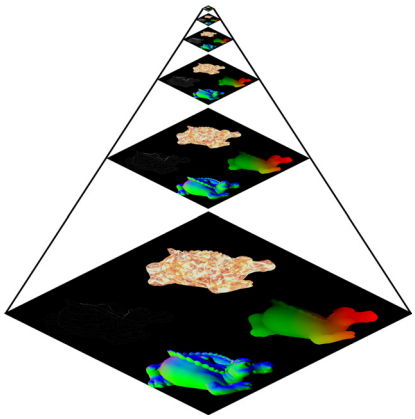

Figure 7.7.: Incident light quad tree stored in mipmap levels of a texture.

a low-resolution texture for B_{out} is computed. At this point, deferred shading is used to avoid rasterizing the model several times and to render fragments that are later overwritten. For the lower-resolution subsurface light transport textures, mipmaps are built from the generated textures. Afterward, the subsurface light transport textures are successively generated with increasing resolution, whereas B_{out} is computed for those pixels only which lie along a depth discontinuity, have a high normal variation, or whose four neighbors in the next lower level have too different colors (i.e. the difference of one of them to the average color is more than δ_B). For the normal variation the fact is exploited that when two different normals are averaged, their length equals the dot product between them. Thus a high normal variation occurs if the averaged normal's length is less than δ_n. Finally, the constructed subsurface light transport texture is combined with the other textures generated for the deferred shading to render the model onto the screen.

5 Results

To generate all images and measure the performance of the approach, $\varepsilon_{max} = 0.1$ is used as maximum approximation error, $\delta_n = 0.9$ as minimum filtered normal length, and $\delta_B = 0.2$ as maximum deviation of a pixel in the subsurface light transport texture for interpolation from the previous level. First the bi-adaptive sampling scheme is analyzed by evaluating the number of shadow map samples for each pixel and the mipmap level of the subsurface light transport texture from which B_{out} is taken. The results of this evaluation are shown in figure 7.8.

Table 7.1 lists the frame rates for various models at a screen resolution of 1280×1024. With the exception of the 7.2 million triangle David model all other examples render at real-time frame rates. Comparing to TSMs [DS03] about the same fill rate is achieved for simple models on the same graphics card (about 100 FPS for the bigguy model at a resolution of 512×512 using a TSM). For more complex models, the original implementation of the TSMs lacks behind since the model is rasterized 15 times which would nowadays not be necessary anymore.

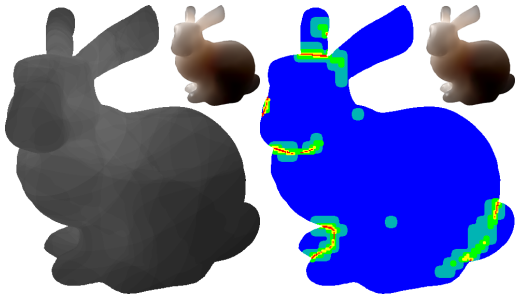

Figure 7.8.: Number of shadow map samples required to calculate B_{out} (left) and mipmap level from at which B_{out} was interpolated (right). In the left image black denotes zero samples and white 255, whereas in the right image, red is the full resolution and from yellow over green and cyan to blue are the lower-resolution mipmap levels.

Instead, only rasterizing it twice, as with the novel approach, would suffice.

model	#Triangles	FPS
bigguy	2,900	28.5
phlegmatic dragon	39,962	27.3
bunny	69,666	33.5
buddha	100,000	28.5
dragon	100,000	25.9
angel	474,048	24.6
david	7,227,031	9.6

Table 7.1.: Frame rate for various models at a resolution of 1280×1024.

Figure 7.9 shows a visual comparison of an image generated with the new approach and one using a TSM [DS03]. Notice that at the same performance, there are significantly less artifacts with the new technique. Especially the banding artifacts due to the mismatch between sample size and mipmap level and the continuously moving sampling positions are not present.

Figure 7.10 shows the different rendered models with materials measured by Wann Jensen et al. [JMLH01]. Notice that at the same performance as TSMs [DS03], there are significantly less artifacts. Especially the banding artifacts due to the mismatch between sample size and mipmap level and the continuously moving sampling positions are not present using the novel approach.

Figure 7.9.: Comparison of bunny model with marble BSSRDF rendered with TSM (left) and out approach (right).

Figure 7.10.: Images generated with the new technique (from left to right and top to bottom): bigguy (skin2), bunny (potato), buddha (ketchup), dragon (wholemilk), angel (skimmilk), and david (marble).

Part III.

Mesh Simplification

Chapter 8.

Advantages of Perception-based Simplification

Current graphics hardware is able to render scenes of striking realism in real-time: the ever growing processing power, memory size and bandwidth allows for the rendering of global illumination, realistic materials and smooth animations reserved to offline-renderers a few years ago. Three dimensional meshes have to keep pace with the render quality in terms of accuracy and amount of detail.

Modern 3D acquisition techniques are able to provide digitized objects with very high accuracy in the sub-millimeter range. This amount of detail often exceeds the ability of the graphics hardware to render the object in real-time. Therefore, real-time applications rely on simplified versions of the original mesh. The main goal of mesh simplification is to generate a low-polygon-count approximation that maintains the high fidelity of the original model. This involves preserving the model's features and overall appearance, such as surface position, curvature, color, and shading. Many simplification algorithms use error bounds on surface position only. This guarantees an upper bound on the maximum deviation of the object's silhouette, but other attributes are not preserved. The preservation of attributes – like texture coordinates or normals – is only possible when their deviations are mapped to a geometric difference.

The assumption that mesh quality improves with the number of primitives ignores the fact that visual fidelity is much more difficult to quantify. The important question concerning mesh simplification should not be geometric but perceptual: does the simplified model *look* like the original? This is motivated by the fact that certain regions of an object can be simplified in a more aggressive manner, resulting in a lower polygon count compared to geometric approaches.

In this part of the thesis a polygonal simplification algorithm directly based on the principles of the human visual system is described. In addition to the well known Hausdorff distance as geometric difference, a per-vertex perceptual metric is used that guarantees an upper bound on the visual error and preserves the visual appearance of the model. The basic idea is to measure the changes in contrast, curvature, and lighting at each vertex after a simplification step. Then the step is only applied if it is considered imperceptible. Deviations in image-space are measured using the contrast sensitivity function (CSF). In addition, the interaction of spatial frequencies and orientations to account for visual masking is included.

Chapter 9.

Previous Work

Since the presented approach integrates models for human visual perception into geometric simplification, a short overview of both fields is given and first approaches of their combination are analyzed.

1 Simplification

The principle of using multiple geometric representations of objects to improve performance was first introduced by Clark [Cla76]. Hoppe [Hop96] developed a geomorphing algorithm that smoothly interpolates between different levels of detail (LODs). These LODs are precomputed using a specialized simplification algorithm. Later he optimized the data structures and algorithms to further improve speed and memory usage [Hop98]. Hoppe also proposed an appearance-based quadric error metrics [Hop99] by comparing geometric correspondences. Fundamental work by Garland and Heckbert [GH97] introduced quadric error metrics for the geometric error created by edge removal. For the special case of multiresolution meshes with texture, Schilling and Klein [SK98a] proposed a simplification algorithm that does not strictly preserve texture coordinates unless the simplification affects the appearance. This is achieved by storing the color variation for each simplified vertex and simplifying more aggressively if it is not modified.

Cohen et al. [COM98] proposed to decouple geometry from appearance by sampling surface normal, curvature, and color attributes and storing them into texture and normal maps. These maps can be precomputed and are a common tool in interactive computer graphics. A similar approach based on deviation in texture space was presented by Schilling and Klein [SK98b]. Klein and Schilling also developed an illumination-based simplification algorithm [KSS98], where normal deviation is used for the accurate simplification of surfaces with specular highlights that are rasterized with Gouraud shading. They also proposed an algorithm for lighting-dependent refinement of multiresolution meshes based on normal cones [KS99]. Unfortunately, neither of these two approaches can be used for textured surfaces. Their ideas were later extended to textured models and arbitrary lighting by Guthe et al. [GBBK04]. Based on the local variation of an attribute in the unsimplified mesh, the attribute deviation is mapped to a geometric error. Although the results have a high visual quality, the number of required primitives is drastically increased. Luebke and Hallen [LH01] used a perceptual metric for the simplification operations by considering contrast and spatial fre-

quency. Unfortunately, this approach only works for Gouraud shaded surfaces. Williams et al. [WLC+03] improved this approach by accounting for textured and dynamically lit surfaces. The proposed algorithm is view-dependent, sensitive to silhouettes, texture content and illumination but cannot handle arbitrary materials. A more aggressive perceptual-based simplification technique was proposed by Luebke et al. [LHNW00]. They incorporate gaze-directed rendering to simplify even more aggressively in regions that are outside the center of the user's vision. Dumont et al. [DPF03] presented a decision framework based on efficient perceptual metrics for realistic rendering on commodity hardware. They address problems such as diffuse texture caching, environment map prioritization, and mesh simplification. Reddy was the first to attempt an LOD selection system completely guided by a model of the human visual system [Red97, Red01]. By analyzing the frequency content of objects and their LOD's from multiple viewpoints, the highest perceivable spatial frequency guides the LOD selection. While all of these approaches modify the LOD selection mechanism, much better results in terms of the number of required primitives could be achieved by integrating a visual model into the LOD creation, as proved by Luebke et al. [LH01] before.

2 Perception

Kelly [Kel75] was the first to develop an analytical model to describe the spatial frequency characteristics of retinal receptive fields. He showed that such a model can be used to describe the sensitivity of the visual system to sine-wave patterns. He also performed studies of the spatiotemporal sensitivity under conditions of stabilized vision [Kel79]. He found out that the shape of the CSF remains nearly constant for a wide range of velocities. Later Burbeck and Kelly [BK80] formulated an analytic contrast sensitivity function based on these experiments. These insights are elemental features of the metric that is proposed in this work. Based on this psychophysical model several measures for image and video difference were proposed. In our context the most important one is Daly's visible difference predictor (VDP) [Dal93] that imitates the image processing in the human eye and brain. A specialized model of visual masking was introduced by Ferweda et al. [FSPG97]. It describes how the presence of a visual pattern affects the detectability of another by predicting changes in contrast, spatial frequency and orientation of texture patterns. Based on Daly's VDP, an extension to high dynamic range images was introduced by Mantiuk et al. [MMS04]. The HDR VDP could be used for image-based simplification, similar to the approach of Lindstrom [LT00]. Mantiuk et al. [MDMS05] later extended their work by calibrating the HDR VDP using an advanced HDR display. A first approach to efficiently integrate human perception into the simplification pipeline was proposed by Qu and Meyer [QM08]. First, an importance map is generated based on pre-computed visual masking and then this map is used to locally increase the maximum simplification error. While the number of primitives can be reduced compared to traditional geometric simplification, there is neither a guarantee for the non-perceivability of the simplification nor for its efficiency with respect to the number of remaining primitives. Lavoue et al. [Lav09] [Lav09] show how

curvature, due to masking, can play an important role in the simplification of geometric models. Pan et al [YIB05] examine the factors that determine the quality of 3D images including geometry- and texture resolution, shading complexity, frame rate, and other psycho-visual factors. Corsini et al. [CDGEB07] address the problem of assessing distortions produces by watermarking 3D meshes by presenting a special visual metric. Yan et al. propose an algorithm that decouples the simplification process into shape analysis and edge contraction to extent collapses from immediately local to the more global [YSZ04]. Lee et al. introduce a low-level human vision model by integrating mesh saliency into the simplification process [LVJ05]. Their approach is based on a center-surround operator, which is present in many models of human vision.

Chapter 10.

Towards Perception-based Mesh Simplification

1 Overview

Figure 10.1 shows the overall concept of the presented approach: First, the original model is simplified using edge collapse operations. These operations are sorted into a priority queue. As stated before, the simplification can continue as long as the simplified model *looks* like the original. Unfortunately, three-dimensional input data can not be used directly for a visual comparison. For this reason, existing approaches rely on a per-pixel comparison of several pairs of rendered images. When rendering the two models, the color of each surface point depends on the position as well as the incoming and outgoing light direction. This results in a six dimensional domain: two dimensions for the position on the surface, and two pairs of spherical angles.

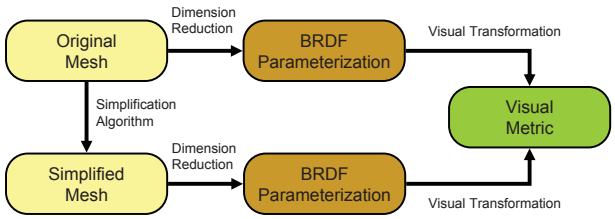

Figure 10.1.: Overview of the main concepts used in this paper.

In order to reduce this high dimensionality, a special BRDF parametrization is introduced which is explained in section 2. The basic idea is to compare the models on a per vertex basis. This reduces the comparison to a computation of the visual difference between two BRDFs. After the BRDF parametrization changes in reflection for a fixed light direction can be represented by two-dimensional maps. Using the proposed visual metric a perception-based comparison can be performed on these maps. The result of this computation is a distance at which the operation becomes perceivable.

The remainder of this paper is structured as follows: it starts with a brief summary of the simplification pipeline and the modifications that have been applied. After this, the BRDF parametrization is explained in detail. Then,

basic concepts of the visual metric – such as contrast sensitivity and masking – are explained. Finally, the results are presented and the algorithm is validated with a user study.

1.1. Simplification Pipeline

The LOD is typically modulated based on an object's distance from the viewpoint. Depending on the application's time and memory constraints, other criteria would be size, velocity, eccentricity, fixed frame rate or orientation. One common criterion is that geometric errors are only fully visible at grazing angles (due to the background). Ideally, the reduced visual quality of the model is unnoticeable because of the small difference of the object's appearance when it is far away or moving fast. Both, the acceptable perceptual quality and resource requirements depend on the particular computing environment: the acceptable perceptual quality is quite different for an automotive-selling application, or in a game environment in which the player may be willing to accept a level of artificiality.

One problem when using a geometric metric to compute discrete LODs is, that the entire object must be simplified uniformly. Unfortunately, errors at silhouettes tend to be more perceptible than other parts of the object because they produce a higher contrast. In order to preserve high quality silhouettes, the geometric deviation on the entire object has to be controlled. This however, ignores the fact that a much more aggressive simplification could be applied to areas that lie inside the object.

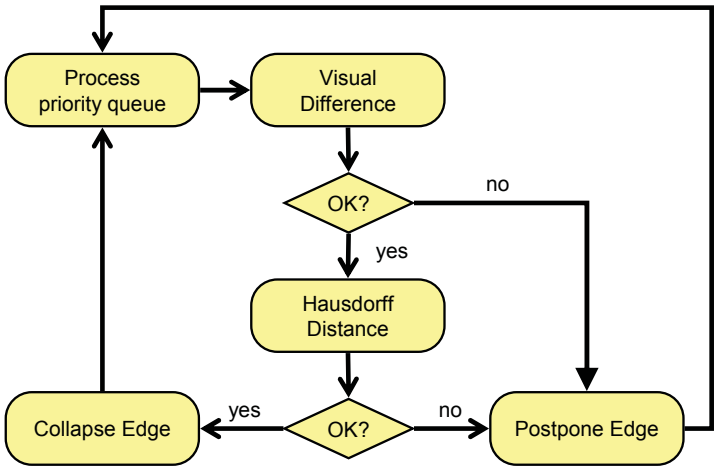

Figure 10.2.: Overall simplification pipeline.

As shown in Figure 10.2, the simplification pipeline is only slightly different from the common geometric approach. First, the edge collapses are sorted into a priority queue, depending on the error that this particular collapse induces: the collapse operation causing the smallest geometric error is the first element and the collapse operation causing the biggest error is the last. The error itself is

estimated by the quadric error metric as introduced by [Hop99]. The classic approach works by removing the edge inducing the smallest error and recomputing the error for all affected edges. This step is repeated until the smallest error is above a user-defined threshold.

This simple loop was modified by performing the visual error computation on the first element of the queue. If the collapse operation induces a visual error higher than the one estimated by the error quadric, the operation is postponed with the actual error. Otherwise, the Hausdorff distance is computed to preserve the silhouettes. If the geometric error is above the estimated one, the edge collaps llapsed
and th l along
with it o more
collaps
Figu l) con-
sisting error)
induce en the
view d strong
change

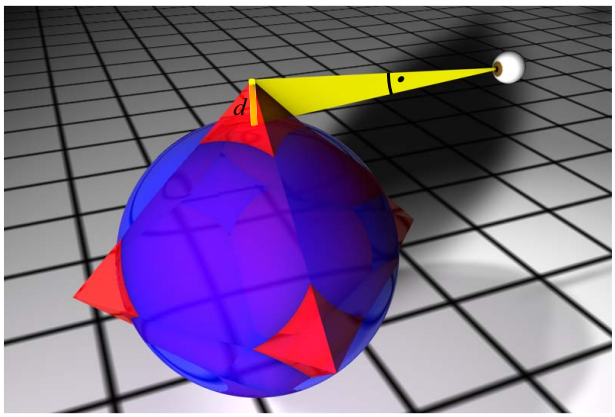

Figure 10.3.: Projection of geometric simplification error between original (blue) and simplified model (red).

The induced frequency with maximum amplitude is $\frac{1}{2\alpha}$ cpd, where $\alpha = 2 \arctan\left(\frac{d}{2D} \sin\theta_r\right)$ depends on the viewing distance D and the angle θ_r between V and N. If this frequency is visible, i.e. it's contrast sensitivity is not zero, the collapse can be perceived. As an upper bound of the split/collapse distance is required for the priority queue, the maximal projection of the geometric error is used, i.e. $\sin\theta_r = 1$. This way the maximal edge displacement frequency can be derived.

To prevent popping artifacts when switching between levels of detail, it is always necesarry to measure the difference between two subsequent levels. The most fine grained techniques are progressive meshes, where each simplification or refinement operation represents a switch between two levels. In this case

it is required to compare the mesh before and after the current simplification operation. While this prevents popping artifacts, the difference to the original model can become arbitrary. Therefore, the difference to the original model is computed additionally.

2 Perceptual Model

Each collapse operation inevitably results in a geometric deviation from the original surface. The geometric error allows one to simplify planar regions in a very aggressive manner, whereas silhouettes or curved areas can only be simplified to a certain degree, depending on viewing distance and direction. Additionally, each simplification step induces a change of curvature, lighting, and parametrization. The idea of our perceptual simplification is, that the degree of reduction depends on the contrast of a surface area. For this purpose we propose a perceptual model to determine whether a collapse operation induces a perceivable change in contrast or not. With the help of such a model, it can be estimated if this particular operation is visible from a given distance.

A straightforward approach is to measure contrast in image-space. If for example a surface has a strong, sharp highlight, it is nearly impossible to simplify that region without significantly altering the original appearance. On the other hand, if there is no contrast at all[1], it would be possible to replace even the most complex geometry by a simple surface patch. Finally, if an object's texture contains very high and random frequencies, it is also possible to simplify more aggressively due to a phenomenon called visual masking, which we address in Section 2.4.

2.1. Color Perception and Color Space Conversion

The first step to determine the visibility of a collapse operation is to mimic human color perception. Its main property is that it operates on contrast rather than absolute luminances. Another important issue is that we have a trichromatic perception of color signals: There are three different types of cones on the retina with peak sensitivities of 570nm, 540nm, and 440nm. They are referred to as long, middle and short (LMS) cones. Sensations of red and green as well as blue and yellow are encoded as color difference signals in separate visual pathways. This means that chromatic values are encoded in an opponent color space with the color channels white-black ($W - K$), red-green ($R - G$), and blue-yellow ($B - Y$).

The first step of the perceptual model is the conversion from the non-linear sRGB color space of the display into the energy absorbed by each of the three cone types in the human retina. This includes inverting the gamma-correction of sRGB and transforming from the device color space to the standard XYZ color space. From there, a transformation to the long, middle and short (LMS) wavelength color space can be performed as described by Hunt [Hun95]:

[1] e.g. in a shadowed area

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 0.7328 & 0.4296 & -0.1624 \\ -0.7036 & 1.6975 & 0.0061 \\ 0.0030 & 0.0136 & 0.9834 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Like Mantiuk et al. [MMS04] we do not model the photoreceptor response but transform the linear luminance values y (i.e. L/M/S) in the JND-scaled space. Their accurate approach is unfortunately too costly for our purposes. We thus approximate the transformation with

$$l = \psi^{-1}(y) \approx k_l \log(y + k_y),$$

where l is the response in JND-scaled space (i.e. L'/M'/S'), k_l and k_y are determined by a least squares fit to the threshold versus intensity function tvi, and ψ is the photoreceptor response. Under the assumption that the eye can adapt to a single pixel of luminance [Dal93], this function is:

$$tvi(\psi(l)) = \frac{\partial \psi(l)}{\partial l}$$

The resulting constants are $k_l = 100$ and $k_y = 1.02$ cd/m^2. Note that the controversial flattening of the receptor response above 10^8 cd/m^2 can be ignored since we can safely assume that no display will produce such a high luminance.

Based on perceptual experiments one can derive different opponent color spaces for the standard human observer. Since we are interested in pattern color separability, we use the color transformation defined by Poirson and Wandell [PW96]:

$$\begin{bmatrix} W-K \\ R-G \\ B-Y \end{bmatrix} = \begin{bmatrix} 0.990 & -0.106 & -0.094 \\ -0.669 & 0.742 & -0.027 \\ -0.212 & -0.354 & 0.911 \end{bmatrix} \begin{bmatrix} L' \\ M' \\ S' \end{bmatrix},$$

where the non-linear cone responses L', M', and S' are given in JND-scaled space.

2.2. Local Visual Difference

As shown in Figure 10.1, the visual metric compares the original model to the simplified one. For each edge collapse, we require the distance at which it becomes imperceivable for every combination of view and light direction. One possibility to evaluate pairs of input images would be to directly use the HDR VDP [MMS04]. Unfortunately, this is computationally too expensive, since it would result in a very high number of input image pairs.

To avoid the per-pixel comparison, we only compare at the vertices of the original and the simplified model. Since the number of vertices is usually much smaller than the number of pixels, a significant speedup can be expected. As a vertex defines an infinitesimal point on the surface, the frequencies can only originate from the curvature of the surface at this point. To capture all possible frequencies, each vertex needs to be rendered from all light and view di-

rections. These samples can be generated by rendering a hemisphere with its normal parallel to the view-direction. This is performed with different light elevations by varying the angle between light and view direction. The result is a four-dimensional reflectance field as shown in Figure 10.4 (right), where it is mapped into a two-dimensional image.

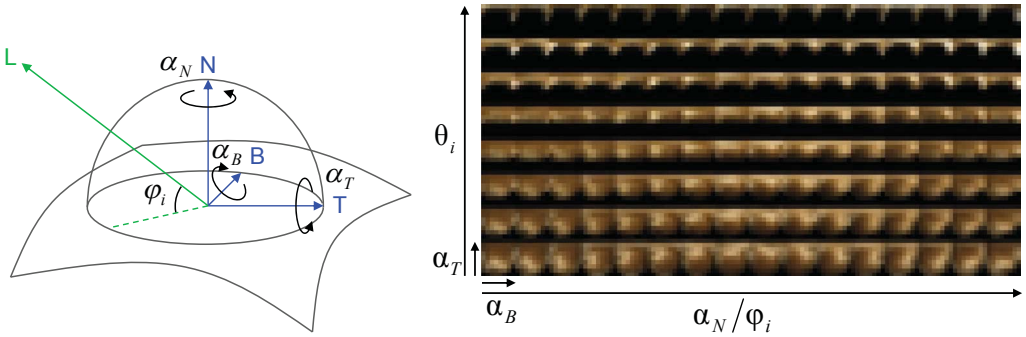

Figure 10.4.: Hemispherical view and light sampling (left) and resulting reflectance field (right).

When the curvatures of both the simplified and original local patch are identical, there are only two sources of differences. Either there is a rotation of the surface, or other shading parameters have changed. If we describe the rotation relative to the local coordinate system of the surface, we can obtain the angles α_t, α_b, and α_n for the rotation around tangent, bitangent, and normal respectively (see Figure 10.4 left). This is a similar parametrization to the well known rotation quaternions with the only difference that it consists of explicit rotation angles. The three rotation angles are computed from the quaternion q with

$$\alpha_{t/b/n} \quad = \quad \frac{q_{x/y/z}}{\|q_{xyz}\|} \cdot 2 \arcsin \|q_{xyz}\|.$$

Note that the absolute values of the angles are independent of the ordering of the two local coordinate systems as only the signs of the quaternion coordinates change when rotating in the opposite direction.

The rotations induce a phase shift ϕ:

$$\phi \quad = \quad \sqrt{\phi_t^2 + \phi_b^2 + \phi_n^2} \quad = \quad \sqrt{\frac{\alpha_t^2}{\omega_t^2} + \frac{\alpha_b^2}{\omega_b^2} + \frac{\alpha_n^2}{\omega_n^2}},$$

where ω_t, ω_b, and ω_n are the angular frequencies of the signal. The first two correspond to spherical oscillations and can thus be derived using a discrete spherical harmonics transformation [DH94]. Finally, ω_n corresponds to a discrete Fourier transformation along α_n. In total, we gain a signal amplitude for all combinations of some discrete values of ω_t, ω_b, and ω_n from these transformations. In addition to the angular phase shift, the difference in shading parameters can also lead to an additional phase shift for each frequency combination.

If we were only interested in the visual difference at the current vertex, we could simply compute the difference between the two shifted signals. Since we can assume that at some nearby point on the mesh at least the phase shift becomes zero again (e.g. at a not yet simplified vertex), we instead consider the maximum difference δ_{max} of the two signals with any phase shift up to the computed one. This yields the following maximal signal difference:

$$\begin{aligned} \delta_{max} &= \min(a_1, a_2) + |a_1 - a_2| \sqrt{2 - 2\cos\phi_{max}} \\ \phi_{max} &= \max(\phi + \phi_s, \pi), \end{aligned}$$

where a_i is the amplitude (L'M'S') of signal i and ϕ_s the phase shift between the two signals in their local coordinate systems.

2.3. Distance Computation

To compute the perceived difference between the two signals, we also need to know their frequency in cycles per degree of visual angle. For this purpose, we first require the spatial frequency f_s which is then projected onto the retina depending on the viewing conditions. The mapping of the angular frequencies to local spatial frequencies depends on the curvatures of the surface. The torsial frequency f_n can be calculated directly from ω_n using the mean torsial curvature κ_n (i.e. the mean absolute derivative of the torsion τ) with

$$f_n = \kappa_n \omega_n = \sqrt{\left(\frac{\partial \tau}{\partial u}\right)^2 + \left(\frac{\partial \tau}{\partial v}\right)^2}\, \omega_n.$$

Note that in physical terms, the torsion τ corresponds to the angular momentum of an idealized top pointing along the tangent of the curve. For surfaces the top direction corresponds to the normal and the curve moves in tangent or bitangent direction. For the tangential frequencies the mapping is not that simple. The tangential frequencies f_t and f_b can only be defined in relation to each other. This is due to the fact that a combined rotation of light and view direction around the surface normal has the same effect as a rotation of the surface itself. Since f_t and f_b are orthogonal, they depend on the minimum and maximum tangential frequencies f_{min} and f_{max} as follows:

$$\begin{aligned} f_t &= \sqrt{(f_{min}\sin\beta)^2 + (f_{max}\cos\beta)^2} \\ f_b &= \sqrt{(f_{min}\cos\beta)^2 + (f_{max}\sin\beta)^2}, \end{aligned}$$

where β is the angle between the current light direction and the direction of minimal curvature. The minimum and maximum frequencies are then:

$$\begin{aligned} f_{min} &= \kappa_{min} \min(\omega_t, \omega_b) \\ f_{max} &= \kappa_{max} \max(\omega_t, \omega_b), \end{aligned}$$

where κ_{min} and κ_{max} are the minimum and maximum principal curvatures of the local surface patch. The total spatial surface frequency f of the coefficient is

now simply the L^2-norm of the three frequencies:

$$f = \sqrt{f_t^2 + f_b^2 + f_n^2}.$$

Exploiting the fact that $\cos^2\beta + \sin^2\beta = 1$ we can write:

$$f = \sqrt{f_{min}^2 + f_{max}^2 + f_n^2}.$$

The mapping of spatial surface frequencies onto the retina then depends on the view direction and distance. At distance D and observed under the angle θ_r between view direction and surface normal, the retinal frequency f_s is:

$$\frac{D}{\cos\theta_r} f \geq f_s(D, \theta_r) \geq D\, f.$$

Given the signals amplitude difference and frequency, we can compute if the average human observer will perceive a difference between them. For this purpose we use the spatio-temporal contrast sensitivity functions $C_{a/c}(f_s, f_t)$ defined by Burbeck and Kelly [BK80]. To prevent visible popping when the level of detail changes, we must use the maximum sensitivity for each spatial frequency with respect to any temporal frequency:

$$C_{a/c}(f_s) := \max_{\forall f_t} C_{a/c}(f_s, f_t)$$

This is simple for the chromatic contrast sensitivity C_c since the highest sensitivity is always at 0 Hz [Kel79]. For the achromatic contrast sensitivity C_a finding the maximum is too costly to be evaluated for each coefficient. We therefore approximate C_a using a sum of three Gaussians. For the sake of simplicity the chromatic sensitivity is also approximated using a sum of two Gaussians:

$$C_{a/c}(f_s) \approx \sum_i w_i e^{\frac{-f_s^2}{\sigma_i^2}}$$

The spreads σ_i and weights w_i for a least squares fit are listed in Table 10.1 and Figure 10.5 shows the approximated sensitivity functions.

	i	w_i	σ_i
C_a	1	1.98488	9.45161°
	2	1.06781	7.07609°
	3	−1.18094	3.14760°
C_c	1	0.27519	7.00112°
	2	0.24116	3.78524°

Table 10.1.: Spread and weight of Gaussian functions used to model the contrast sensitivity functions.

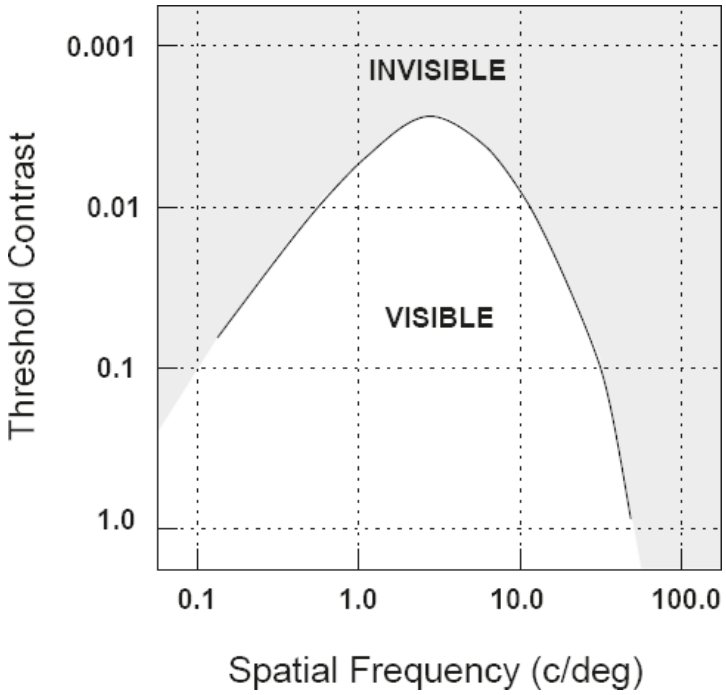

Figure 10.5.: Chromatic and achromatic contrast sensitivity: the reduction of the fall-off towards zero is necessary to prevent popping.

For both the contrast difference δ_{max} and the detection threshold t_d we need to know the mapping of the spatial frequency into the field of view. For this projection the viewing distance D is required. To determine the split distance, we need to find that D_{max} above which the difference cannot be perceived. As this distance becomes smaller with increasing frequency, only the lower bound of the spatial frequency f_s after projection needs to be considered. D_{max} itself can then be computed using Newton iteration to find the distance at which $\delta = t_d$ holds. In the achromatic case there may be two distances satisfying this condition, D_{min} and D_{max}, so one can either store both or D_{max} only. In the first case, the edge could also be collapsed if the distance D is below D_{min}. Since for each operation three vertices are considered and the split range of the two previous vertices also needs to be considered, this can however only marginally reduce the number of triangles.

2.4. Visual Masking

Visual masking is a perceptual phenomenon that was studied in detail by Campbell and Gubisch [CG66]. It occurs because the stimuli in the photoreceptors cells are not independent of each other but interact in various ways. As a result, a stimulus that is visible by itself cannot be detected due to the presence of another. The reason is that each ganglion cell on the retina receives its input from a local region called receptive field. The receptive field is not restricted to cells in the retina, but also occurs in many later stages of the visual pathways. Without

masking the detection threshold t_d is always one. Masking can elevate t_d for a given vertex, allowing for a more aggressive simplification beyond the bounds of the just noticeable difference. Values smaller than one are also possible for t_d due to the so-called facilitation, which occurs for very weak stimuli with similar characteristics.

We consider two types of masking in our approach: local masking, which occurs in the vicinity of a single vertex, and global masking, that depends on the entire model. A cortex transformation [MMS04], which performs a decomposition into spatial and orientational channels, is applied for the global masking. This is done by applying directional filters to each color channel (orientation processing). The transformation cannot be transferred to the local case, because here the signals are already localized. This limits the local masking to the frequency domain for which we have already performed a transformation using the spherical harmonics and Fourier transformations.

For the global masking the object is rendered from different view and lighting positions distributed over the unit sphere. This step results in a set of input images I_k for incoming and outgoing directions $(\omega_{i,k}, \omega_{o,k})$, which means that I_k is rendered from view direction $\omega_{i,k}$ with light direction $\omega_{o,k}$. For each I_k the color space conversion to JND-scaled space is applied. In contrast to local masking, where the cortex transform is calculated for the parametrization of a single vertex, it is now applied to the entire image as in [MMS04]. Then the visibility of each vertex is checked, since the difference at a vertex can only be masked when it is visible. If a vertex is visible, the maximum local amplitude of all receptive field directions is calculated for each frequency band f. Finally, the maximum amplitude over all images is computed and stored as global masking $a_g(f)$ for the vertex. This is correct under the assumption that when the masking is maximal, the signal difference is maximal as well. This assumption is reasonable for all real world objects and materials. To finally compute t_d we use the masking function of Daly with masking amplitude a_m and frequency f, where the masking amplitude is a combination of local and global masking with:

$$a_m \quad = \quad max(a_g(f), min(a_1, a_2))$$

We again need to use the maximum of the signals as the highest masker contrast determines the level of masking [Dal93].

3 Implementation

The first step when calculating the validity range of a collapse operation is to generate the hemispherical samples for a given vertex and the closest interpolated point on the reference model. For this purpose the same shader was used as to render the model in the interactive application. To generate the samples, precomputed meshes with appropriate normal and tangent vectors were used. Note that only meshes for α_n and ϕ_i are required as for different θ_i the meshes can be reused.

The following spherical harmonics transformation is underdetermined, since only the upper hemisphere is sampled. If the lower half is set to zero, frequency

ringing occurs at the boundary. To prevent this, the underdetermined linear equation system was solved that is set up by the inverse transformation under the constraint that

$$\sum_i \frac{|c_i|^2}{f_i^2} \quad \rightarrow \quad \min,$$

where c_i is the i-th coefficient and f_i its frequency. Note that it is not necesarry to solve this least squares problem for each transformation but simply calculate the pseudo-inverse of the constraint matrix once. Another possibility would be to use the hemi-spherical harmonics proposed by Gautron et al. [GKPB04]. However, these have the drawback that the phase shift cannot be computed directly from the coefficients.

The transformations that need to be applied to the generated samples are entirely implemented using CUDA. Such a parallel implementation is possible since each transformation is essentially a sparse matrix vector multiplication. In our implementation this general formulation is used even for the Fourier transformation because the dimension is rather small and thus the transformations are not the performance bottleneck. The most demanding part of the distance computation is the comparison of the transformed samples due to the iterative search for the distance at which the difference becomes just noticeable. In the presented approach, the distance computation is performed in parallel on each transformed coefficient. While this optimally exploits parallelism since the code executed for each coefficient is identical and only the data differs, an inter thread communication could be used to terminate those with smaller distance than the currently computed lower bound.

3.1. Out-of-core Simplification

For the processing of very large polygonal models an out-of-core implementation was developed. To optimize the input data for caching, the vertices and triangles of the model are sorted using spectral mesh sequencing. Note that other types of preprocessing such as cache-oblivious layouts [eYLPM05] would be possible as well.

The priority queue containing all collapse operations is stored on disk. Each element in the priority queue consists of a priority value and an index. The algorithm starts by loading the first n elements of the queue into memory. These elements are then sorted by the index of the target vertex v. Due to the preprocessing step this results in a cache local access pattern. The simplification error is then computed for this sorted list. Note that to compute the error, the mesh datastructure has to be loaded as well to find the vertex neighbors. File accessing is done using memory map files, which are provided by the operating system to optimize disk IO. If a vertex passes all tests it is stored in another list. After checking all vertices of the current block, the accepted operations are sorted by their perceivability distance. Then they are applied in that order unless they have become invalid due to the collapse of a neighbor vertex.

4 Results

The test system was a PC with an Intel Core2 Duo 3.33 GHz CPU, 2 GByte of main memory, and a GeForce GTX 295. As maximum display luminance of the final interactive rendering application, 1000 cd/m^2 was used, but this has little influence on the simplification result and no effect on the computation time. The hemispherical sampling has a resolution of $16 \times 16 \times 32 \times 16$ $(\alpha_b, \alpha_t, \alpha_n/\phi_i, \theta_i)$ which is sufficient for high frequency materials.

For each simplification operation three vertices are checked. Each of these tests requires approximately 0.9 ms which sums up to 2.7 ms per operation. With one third of the quadric error as estimate for the Hausdorff distance, almost every second checked operation is postponed during simplification. This results in a ratio of 2 : 1 for perception tests versus performed operations. The Hausdorff test on the other hand needs 4 ms on average, but is passed by nearly every operation. In total, approximately 100 collapses were computed per second which is a throughput of about 1 million faces in 90 minutes. This performance is still acceptable, although the fastest simplification algorithms that do not guarantee a maximum error are almost two orders of magnitude faster.

As simplification is a preprocessing step that is performed once per model only, the number of primitives required for a specific quality is far more important. In this context the algorithm was compared to the attribute preserving simplification of Guthe et al. [GBBK04]. To map view distance to simplification error, the same scaling was used as for the Hausdorff distance ($\frac{1}{30}^\circ$ fov). The resulting number of triangles for different simplification errors are shown in Table 10.2 for simplification errors of 0.1% and 1% of the bounding box diagonal. Using the presented approach saves roughly half of the primitives compared to the attribute preserving simplification.

	simpl. error	our algorithm	attrib. pres.
phlegmatic	0.1%	276,742	440,646
dragon	1%	61,208	133,528
Stanford	0.1%	439,446	723,732
dragon	1%	14,664	36,034
Happy	0.1%	652,808	834,450
Buddha	1%	53,054	61,678
XYZRGB	0.1%	1,203,295	3,357,094
manuscript	1%	156,437	263,793
XYZRGB	0.1%	1,570,734	3,776,114
dragon	1%	266,145	424,131

Table 10.2.: Number of triangles for models simplified to a specified error.

To validate our perceptual measure, several simplifications of the three smaller models with varying perceptual error threshold were produced. They were presented to a group of 32 (11 female, 21 male) subjects that had to decide if they can perceive a difference to the non-simplified reference model or not. The

subject can interactively view the model that is lit by a single directional light source that can also be rotated. For rendering we created progressive meshes of the three input models and used view-dependent refinement [Hop98] to adapt them for the current viewing conditions. For each model, two tests have been performed, one with increasing and one with decreasing error, until the subject noticed or stopped to notice the visual difference. To remove any subject that was only guessing, also a comparison of the original model with itself was added. A calibrated display was used with a maximum luminance of 500cd/m^2 and a viewing distance of 1 m (1.9× the image width). The surrounding illuminance was approximately 500 lux such that the ratio of display to surrounding luminance conforms with ITU-R BT.500-11. Figure 10.6 shows the experimental setup.

Figure 10.6.: Setup for perceptual experiment. In this example, the model on the left is simplified up to four times the just noticeable difference and the original model is shown on the right.

Table 10.3 shows the results of this study that are as expected as approximately half of the subjects started to notice the difference at 1 JND for most models. Only the results for the phlegmatic dragon differ. Here the estimated JND is only half of the real one, leading to the assumption that a more aggressive simplification would have been possible. This might be due to the rather sparse sampling when calculating the global masking. For both of the models, the real just noticeable difference approximately lies between 0.8 and 1.2 of the estimated JND. This validates the presented model as it complies with the formal definition of the just noticeable difference.

Figure 10.7 shows a comparison of the meshes generated by the two simplification algorithms. The presented approach especially reduces the number of triangles in smooth regions that contain high frequency noise, e.g. at the belly of the Buddha and scales of the XYZRGB dragon.

Finally, Figure 10.8 shows a comparison of the visual quality using QSlim [Hop99] and attribute preservation [GBBK04] with the same number of triangles as the novel approach. While the quadric error metric does not preserve smooth variation of attributes (especially texture coordinates), the attribute preservation

	phlegmatic dr.	Stanford dr.	Happy Buddha
	total (increasing/decreasing)		
$\frac{1}{4}$ JND	11% (23%/ 0%)	0% (0%/ 0%)	22% (33%/12%)
$\frac{1}{2}$ JND	17% (28%/ 7%)	6% (8%/ 3%)	41% (52%/30%)
1 JND	37% (49%/24%)	34% (38%/ 30%)	63% (63%/63%)
2 JND	50% (65%/34%)	95% (96%/ 94%)	73% (63%/84%)
4 JND	63% (72%/54%)	100% (100%/100%)	86% (88%/84%)

Table 10.3.: Percentage of subjects that perceived a visual difference with respect to perceptual error threshold of the presented model.

needs too many triangles resulting in a higher geometric error.

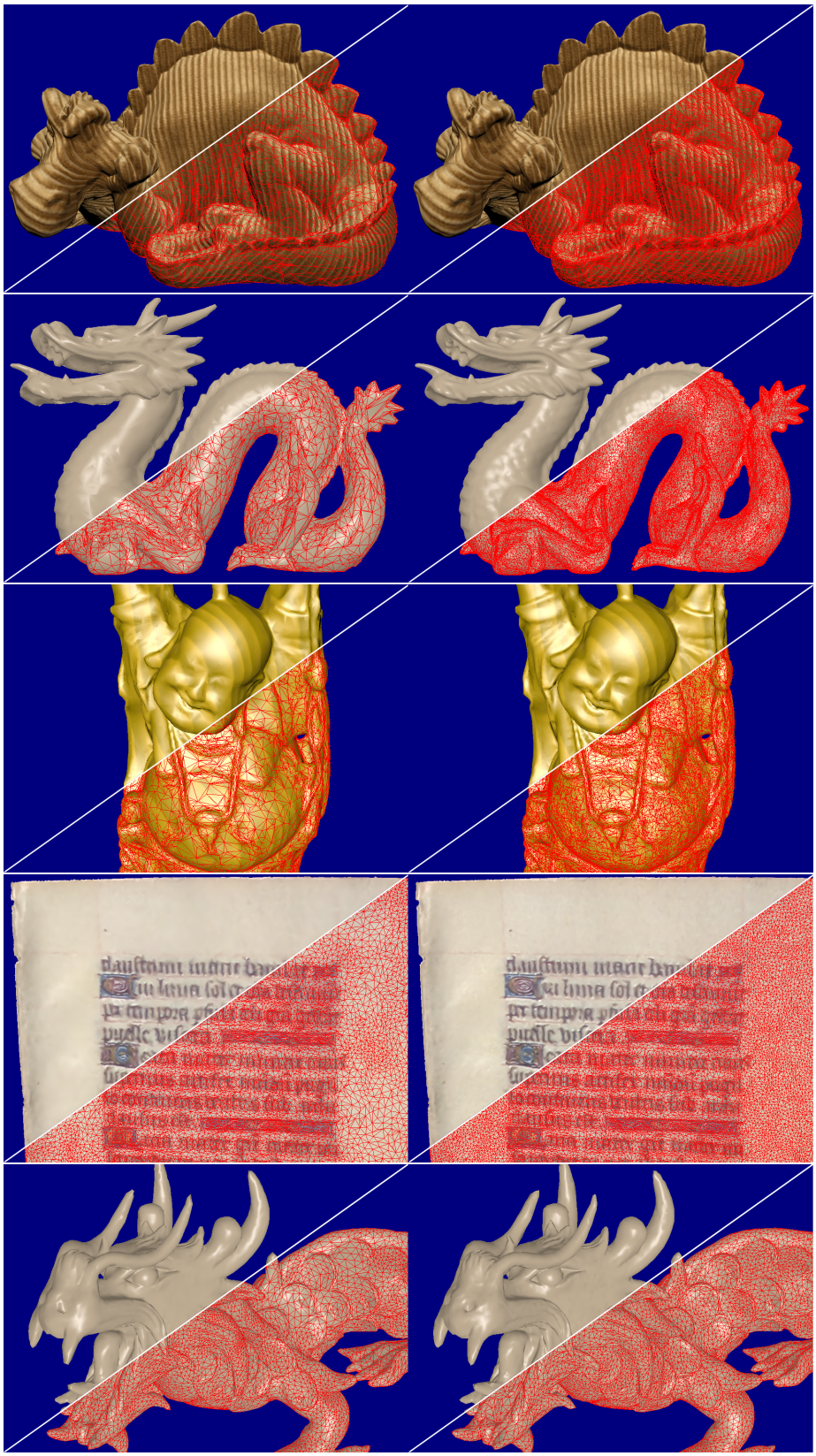

Figure 10.7.: Simplified models at 0.2% simplification error using the presented algorithm (left) compared to attribute preservation(right).

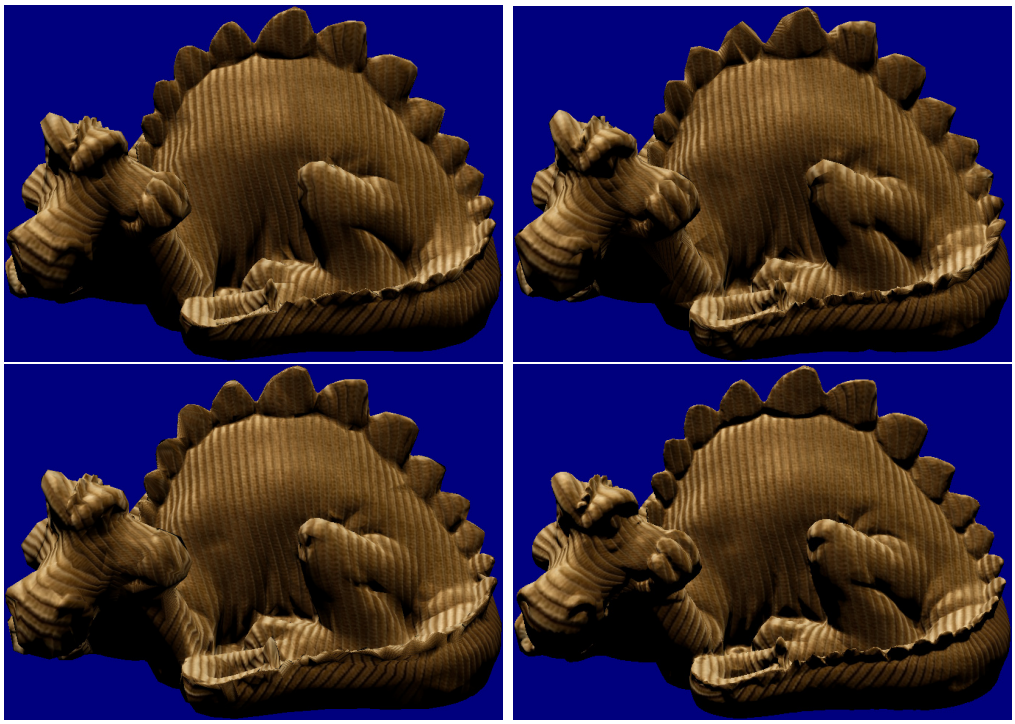

Figure 10.8.: Simplified models at the same number of triangles using QSlim (top left), attribute preservation (top right), and our approach (bottom left) compared to the original model (bottom right).

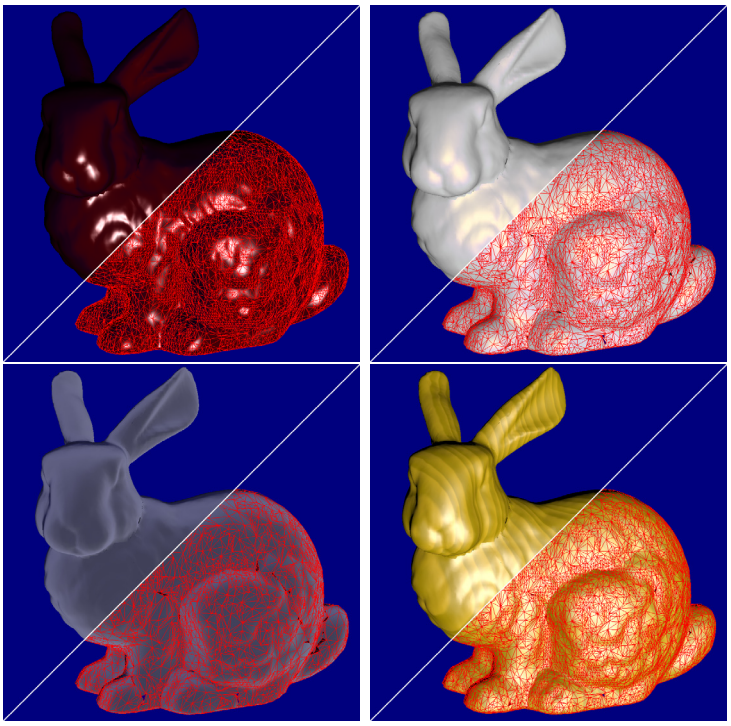

Figure 10.9.: Stanford bunny with different materials simplified up to an error threshold of 0.2%. Note how the number of triangles (32902, 24198, 17669, and 27447) depends on frequency content.

Part IV.

Conclusion and Future Work

Chapter 11.

Conclusion

In section 4 a compression method to preserve the full dynamic range of a reflectance field was presented. After reconstruction, any post production effect such as tone mapping or bloom can be applied, making it possible to use the data in any dynamic environment. The prevailing problem is that all of the compression techniques PCA, wavelets, spherical harmonics or VQ produce an optimal linear solution, whereas HDR radiance values are non-linear distributed. This work proposed to solve this problem by applying a non-linear transformation to the radiance data. It was shown that this transformation does not only allow to apply a compression step, but also a quantization step. In the examples a simple method to quantize the principle components to common RGB data format was presented. After this, lossy hardware-accelerated DXT1 compression was applied. After these steps one pixel takes 4 bits instead of 48, resulting in an compression ratio of 1:12. Stored as conventional texture maps it is possible to reconstruct the reflectance field in real-time using the fragment shader. The main limitation of the method lies in the chosen simple PCA compression. If the covariance between pixels is low, the image quality is low.

In section 5 a method was presented to generate HDR images from picture sequences that are taken without a tripod or automatic exposure sequence and can even contain moving objects (e.g. clouds, foliage, and people). Motion is identified using a hierarchical macroblock matching based on cross-correlation. This does not only compensate for moving objects, but also for movements of the camera that lead to parallax effects. While the current implementation has a runtime of less than 15 seconds for a sequence of three pictures and a resolution of about one mega pixel on a 2.4 GHz CPU, an implementation on current graphics hardware will be able to align high resolution pictures within a few seconds. A limitation of the method is that it cannot compensate for large occluded areas that were visible in the reference image and thus solely relies on the ghost removal technique during the second step in such cases. If no data is available in the other images (i.e. the region is completely black or white) the irradiance cannot be reconstructed. Another limitation is that the technique can only compensate small rotations (up to about 14 degree) about the z-axis of the camera. Thus a possible extension would be to allow rotation the the macroblocks during the alignment phase.

In section 6 a novel BTF representation was proposed that is suited for both editing and rendering at the same time. This is achieved by splitting the BTF

into texture maps describing meso-scale geometry and textures describing light interaction. The main advantage of this representation is that changing the textures directly affects the BTF rendering since no further compression is required. For designers, these textures are comprehensive and intuitive and do not require deep knowledge of the underlying physics. A shader based on parallax occlusion mapping to render g-BRDFs in real-time on current graphics hardware was presented. In addition to generating a BTF from scratch, the proposed decomposition process partitions a measured BTF into its g-BRDF representation and thus allows editing of real-world materials. While the quality of the compressed BTF is comparable to state-of-the-art statistical compression algorithm, the g-BRDF requires significantly less texture memory.

In section 7 the concept of shadow maps and TSMs was extended by introducing a novel hierarchical approach for view-dependent subsampling of the shadow map in dependence of the underlying projected geometry. For this purpose, multiple mipmap-levels of variance maps are created. Starting from the coarsest level, finer samples are only taken if the variance exceeds a certain threshold. This hierarchical approach allows to compute the whole scattering integral and can therefore be considered a global approach. The technique could be improved by adding a single scattering term. Furthermore, it is restricted to convex objects and cannot simulate hollowed objects or objects that consist of multiple materials, which could be solved using depth peeling. Another valuable addition would be the validity feature as introduced by Ki and Oh [KO08] in order to improve rendering speed with a quick validity check before estimating the approximation error.

In section 10 a perceptual-based simplification algorithm was presented that can be applied to arbitrary materials. In contrast to existing approaches, where a perceptual metric is applied per pixel, the presented metric is evaluated for each vertex. Since the number of visible vertices is usually much smaller than the number of pixels, a significant speedup was achieved. The simplification performance of 100 collapses per second is at least an order of magnitude faster than any algorithm based on per-pixel comparisons. To compare two vertices, a special BRDF parameterization was introduced that maps incoming and outgoing lighting directions into a two-dimensional map. The parameterization accounts for anisotropic materials as well. The main limitation of the method is that only the difference at vertices is checked which can lead to unnoticed deviations on textured surfaces. However, this did not occur for the models and materials that were tested. This limitation is most critical for models with few vertices and textures containing very high frequencies. For such materials, a per pixel metric has to be applied.

Chapter 12.

Future Work

For the simulation of subsurface scattering, more flexible algorithms have to be developed: They need to work dynamic meshes, for distant and very close-up views, and for any kind of material. Also, they must not depend on heavy precomputation, or precomputation must be very fast. Also, note that for hollow and concave objects, the diffuse dipole approximation[1] is not plausible - a more robust mathematical model has to be developed for such meshes. One possibility for a more precise calculation of scattering would be the use of form factors, although they are highly dependend on visiblity.

The methods for the acquisition of realistic materials must become faster and more affordable. Current systems are very expensive and have to be controlled by experts. Acquisition can last from several hours to days, resulting in data sets of several gigabytes. I would think of a device that can be placed on a table, with a small slide for the material sample and one or two gantries with the camera and the light source. Ideally would be a uniform sampling of incoming and outgoing light directions, including lighting samples from the back to measure subsurface scattering.

For the acquisition of high dynamic range images, cameras with faster exposure control have to be developed to avoid ghost artifacts and unwanted motion. Alternatively, CCD sensor chips capable of HDR have to be affordable for the consumer market. Current HDR chips are reserved to professionals and still have a very limited dynamic range, used in cameras of overdimensional size.

A possible improvement of our g-BRDF representation would be to include even more effects that are lost during the decomposition of the material properties, for example more precise self-shadowing or realistic interreflections. Also, an additional parameterization for realistic subsurface scattering could be included. A future implementation would possibly exploit the tesselation features of latest graphics hardware: the height map would then be directly used in the rendering pipeline to create a high number of additional triangles directly on the GPU. This would greatly improve the quality of the material[2]

[1] used in methods such as TSM

[2] This would also make the parallax occlusion mapping obsolete.

Glossary

A

AEB Automatic Exposure Bracketing provided by most digital cameras, p. 58.

B

b-reps Boundary Representation of a mesh (e.g. half-edge datastructure), p. 23.

BRDF Bidirectional Reflectance Distribution Function.

BSSRDF Bidirectional Surface Scattering Reflectance Distribution Function.

BTF Bidirectional Texture Function.

C

CG Computer-Generated or C for Graphics (context).

CRT Cathod Ray Tube.

CUDA Compute Unified Device Architecture.

D

display resolution The physical ability of the display device to show distinct pixels., p. 8.

dynamic range Relation of the brightest to the darkest pixel on the screen., p. 9.

F

frames per second Temporal resolution is measured in frames per seconds or Hertz., p. 9.

G

GPU Graphics Processing Unit.

H

HDR VDP High Dynamic Range Visual Difference Predictor.

HDRI High Dynamic Range Imaging.

High Definition High Definition Resolution is 1920x1080 picture elements, p. 8.

high dynamic range Describes the synthesis of images done in larger dynamic range., p. 9.

HLSL High Level Shading Language.

human visual system Human Visual System, p. 12.

I

IBRM Image-based Rendering and Modeling, p. 38.

L

LCD Liquid Crystal Display, p. 8.

N

NTFS National Television System Committee, p. 8.

O

OpenCL Open Computing Language.

P

PAL Phase Alternating Line, p. 8.

PCA Principal Component Analysis.

S

screen resolution The resolution of the framebuffer in the memory of the graphics device., p. 8.

SVD Singular Value Decomposition.

T

tone-mapping operator Tone-Mapping Operator, p. 10.

V

VLF Vertex Light Field.

W

WQSXGA Wide Quad Super Extended Graphics Array, p. 8.

WQXGA Wide Quad Extended Graphics Array, p. 8.

Bibliography

[AG06] Michael Ashikhmin and Jay Goyal. A reality check for tone-mapping operators. *ACM Trans. Appl. Percept.*, 3(4):399–411, 2006.

[APS00] Michael Ashikmin, Simon Premože, and Peter Shirley. A microfacet-based brdf generator. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 65–74, 2000.

[Ash02] Michael Ashikhmin. A tone mapping algorithm for high contrast images. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*, pages 145–156, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

[Ash06] Michael Ashikhmin. Distribution-based brdfs, 2006. Stony Brook University.

[Bay87] A.D. Baylor. Photoreceptor signals and vision. *Investigative Ophthalmology & Visual Science*, 28:34–49, 1987.

[BBF94] J. L. Barron, S. S. Beauchemin, , and D. J. Fleet. On optical flow. *6th Int. Conf. on Artificial Intelligence and Information-Control Systems of Robots (AIICSR)*, pages 3–14, 1994.

[BC06] Francesco Banterle and Alan Chalmers. A fast translucency appearance model for real-time applications. In *SCCG 2006 - Spring Conference on Computer Graphics*, 2006.

[BFBB94] J. L. Barron, D. J. Fleet, S. S. Beauchemin, and T. A. Burkitt. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, 1994.

[Bis95] C.M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, 1995.

[BK80] C. A. Burbeck and D. H. Kelly. Spatiotemporal characteristics of visual mechanisms: Excitatory-inhibitory model. *Journal of the Optical Society of America*, 70(9):1121–1126, 1980.

[Bli78] James F. Blinn. Simulation of wrinkled surfaces. In *SIGGRAPH '78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, pages 286–292, 1978.

[Bra95] D. H. Brainard. Colorimetry. In *Handbook of Optics: Fundamentals, Techniques, and Design*, New York, NY, USA, 1995. ACM.

[CBCG02] Wei-Chao Chen, Jean-Yves Bouguet, Michael H. Chu, and Radek Grzeszczuk. Light field mapping: Efficient representation and hardware rendering of surface light fields. *ACM Transactions on Graphics*, 21(1):447–456, 2002.

[CDGEB07] Massimiliano Corsini, Elisa Drelie Gelasca, Touradj Ebrahimi, and Mauro Barni. Watermarked 3d mesh quality assessment. *IEEE Transaction on Multimedia*, 9(2):247–256, February 2007.

[CDR02] U. Clarenz, M. Droske, , and M. Rumpf. Towards fast non–rigid registration. In *Proc. of the AMS*, 2002.

[CG66] F. Campbell and R. Gubisch. Optical quality of the human eye. *Journal of Physiology 186*, pages 558–578, 1966.

[CGS06] Tongbo Chen, Michael Goesele, and Hans-Peter Seidel. Mesostructure from specularity. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1825–1832, 2006.

[Cla76] James H. Clark. Hierarchical geometric models for visible surface algorithms. *Commun. ACM*, 19(10):547–554, 1976.

[COM98] Jonathan Cohen, Marc Olano, and Dinesh Manocha. Appearance-preserving simplification. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 115–122, New York, NY, USA, 1998. ACM.

[CT82] R. L. Cook and K. E. Torrance. A reflectance model for computer graphics. *ACM Trans. Graph.*, 1(1):7–24, 1982.

[CVM+96] Jonathan Cohen, Amitabh Varshney, Dinesh Manocha, Greg Turk, Hans Weber, Pankaj Agarwal, Frederick Brooks, and William Wright. Simplification envelopes. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 119–128, New York, NY, USA, 1996. ACM.

[Dal93] S. Daly. The visible difference predictor: An algorithm for the assessment of image fidelity. *Digital Image and Human Vision*, pages 179–206, 1993.

[DH94] James R. Driscoll and Dennis M. Healy, Jr. Computing fourier transforms and convolutions on the 2-sphere. *Adv. Appl. Math.*, 15(2):202–250, 1994.

[DHT+00] Paul Debevec, Tim Hawkins, Chris Tchou, Haarm-Pieter Duiker, Westley Sarokin, and Mark Sagar. Acquiring the reflectance field of a human face. *ACM Transactions on Graphics*, 19(1):145–156, 2000.

[dL07] E. d'Eon and D. Luebke. Advanced techniques for realistic real-time skin rendering. GPU Gems III, 2007.

[dLK93] Shou de Lin and Craig A. Knoblock. Jarek rossignacand paul borrel. multi-resolution 3d approximations for rendering complex scenes. In *Modeling in Computer Graphics: Methods and Applications*, pages 455–465, 1993.

[DM97] Paul E. Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 369–378, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[DPF03] Reynald Dumont, Fabio Pellacini, and James A. Ferwerda. Perceptually-driven decision theory for interactive realistic rendering. *ACM Trans. Graph.*, 22(2):152–181, 2003.

[DS03] Carsten Dachsbacher and Marc Stamminger. Translucent shadow maps. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering*, pages 197–201, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[DS05] Carsten Dachsbacher and Marc Stamminger. Reflective shadow maps. In *I3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 203–231, New York, NY, USA, 2005. ACM.

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

[EM98] Carl Erikson and Dinesh Manocha. Gaps: General and automatic polygonal simplification. In *In Proc. of ACM Symposium on Interactive 3D Graphics*, pages 79–88, 1998.

[eYLPM05] Sung eui Yoon, Peter Lindstrom, Valerio Pascucci, and Dinesh Manocha. Cache-oblivious mesh layouts. *ACM Trans. Graph*, 24:886–893, 2005.

[FLW02] Raanan Fattal, Dani Lischinski, and Michael Werman. Gradient domain high dynamic range compression. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 249–256, New York, NY, USA, 2002. ACM.

[FPSG96] James A. Ferwerda, Sumanta N. Pattanaik, Peter Shirley, and Donald P. Greenberg. A model of visual adaptation for realistic image synthesis. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 249–258, New York, NY, USA, 1996. ACM.

[FSDH07] Matthew Fisher, Peter Schröder, Mathieu Desbrun, and Hugues Hoppe. Design of tangent vector fields. *ACM Trans. Graph.*, 26(3):56, 2007.

[FSPG97] James A. Ferwerda, Peter Shirley, Sumanta N. Pattanaik, and Donald P. Greenberg. A model of visual masking for computer graphics. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 143–152, 1997.

[GBBK04] Michael Guthe, Pavel Borodin, Ákos Balázs, and Reinhard Klein. Real-time appearance preserving out-of-core rendering with shadows. In A. Keller and H. W. Jensen, editors, *Rendering Techniques 2004 (Proceedings of Eurographics Symposium on Rendering)*, pages 69–79 + 409. Eurographics Association, June 2004.

[Gen89] Carme Torrasi Genís. Relaxation and neural learning: points of convergence and divergence. *J. Parallel Distrib. Comput.*, 6(2):217–244, 1989.

[GGSC96] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 43–54, 1996.

[GH97] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[GH07] Abhijeet Ghosh and Wolfgang Heidrich. The d-brdf model as a basis for brdf acquisition. In *SIGGRAPH '07: ACM SIGGRAPH 2007 posters*, page 40, 2007.

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

[Gra98] F. Sebastin Grassia. Practical parameterization of rotations using the exponential map. *J. Graph. Tools*, 3(3):29–48, 1998.

[Gre04] S. Green. Real-time approximations to subsurface scattering. GPU Gems, 2004.

[Gro06] T. Grosch. Fast and robust high dynamic range image generation with camera and object movement. In *Vision, Modeling and Visualization, RWTH Aachen*, pages 277–284, 2006.

[GW77] R. C. Gonzales and P. Wintz. *Digital Image Processing*. Addison-Wesley, Reading, MA, 1977.

[HJ07] J. Hoberock and Y. Jia. High quality ambient occlusion. GPU Gems III, 2007.

[Hop96] Hugues Hoppe. Progressive meshes. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 99–108, New York, NY, USA, 1996. ACM.

[Hop97] Hugues Hoppe. View-dependent refinement of progressive meshes. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 189–198, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[Hop98] Hugues Hoppe. Efficient implementation of progressive meshes. *Computers & Graphics*, 22(1):27–36, 1998.

[Hop99] Hugues Hoppe. New quadric metric for simplifying meshes with appearance attributes. In *VISUALIZATION '99: Proceedings of the 10th IEEE Visualization 1999 Conference (VIS '99)*, Washington, DC, USA, 1999. IEEE Computer Society.

[HS98] Wolfgang Heidrich and Hans-Peter Seidel. View-independent environment maps. In *HWWS '98: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 39–44, 1998.

[Hun95] R. W. G. Hunt. *The Reproduction of Colour*. Wiley InterScience, 1995.

[JM03] Rafal Jaroszkiewicz and Michael D. Mccool. Fast extraction of brdfs and material maps from images. In *In Graphics Interface*, pages 1–10, 2003.

[JMLH01] Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy, and Pat Hanrahan. A practical model for subsurface light transport. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 511–518, New York, NY, USA, 2001. ACM.

[Kaj86] James T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150, 1986.

[KAR06] E. A. Khan, A. O. Akyuez, , and E. Reinhard. Ghost removal in high dynamic range images. In *In Proc. of the International Conference on Image Processing (ICIP 2006)*, pages 2005–2008, 2006.

[Kau99] J. Kautz. Hardware rendering with bidirectional reflectances. Technical Report TR-99-02, Dept. Comp. Sci., U. of Waterloo, 1999.

[KBD07] Jan Kautz, Solomon Boulos, and Frédo Durand. Interactive editing and modeling of bidirectional texture functions. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, pages 53–62, 2007.

[Kel75] D. H. Kelly. Perceptually optimized 3d graphics. *Vision Research*, 15:665–672, 1975.

[Kel79] D. H. Kelly. Motion and vision. ii. stabilized spatio-temporal threshold surface. *J. Opt. Soc. Am.*, 69(10):1340–1349, 1979.

[Khr09] KhronosGroup. The opencl specification. Programming Guide Version 1.0 Rev 48, Khronos Group, 2009.

[KM99] Jan Kautz and Michael D. McCool. Interactive rendering with arbitrary brdfs using separable approximations. In *SIGGRAPH '99: ACM SIGGRAPH 99 Conference abstracts and applications*, page 253, 1999.

[KMBK03] M. L. Koudelka, S. Magda, P. N. Belhumeur, and D. J. Kriegman. Acquisition, compression and synthesis of bidirectional texture functions. In *3rd International Workshop on Texture Analysis and Synthesis (Texture 2003)*, 2003.

[KO08] Hyunwoo Ki and Kyoungsu Oh. A gpu-based light hierarchy for real-time approximate illumination. *The Visual Computer*, 24(7-9):649–658, July 2008.

[Kry31] Aleksej Nikolajevič Krylov. O čislennom rešenii uravneniâ, kotorym v tehničeskih voprosah opredelâûtsâ častoty malyh kolebanij material'nyh sistem. *Izvestiâ Akademii Nauk SSSR*, VII(4):491–539, 1931.

[KS99] Reinhard Klein and A. Schilling. Efficient rendering of multiresolution meshes with guaranteed image quality. *The Visual Computer*, 15(9):443–452, 1999.

[KSS98] Reinhard Klein, A. Schilling, and W. Straßer. Illumination dependent refinement of multiresolution meshes. In F. E. Wolter and N. M. Patrikalakis, editors, *Computer Graphics International*, pages 680–687. IEEE Comput. Soc., 1998.

[KSS+04] Jan Kautz, Mirko Sattler, Ralf Sarlette, Reinhard Klein, and Hans-Peter Seidel. Decoupling brdfs from surface mesostructures. In *GI '04: Proceedings of Graphics Interface 2004*, pages 177–182, 2004.

[KUWS03] Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. High dynamic range video. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 319–325, New York, NY, USA, 2003. ACM.

[Lan50] Cornelius Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, 45(4):255–282, 1950.

[Lav09] Guillaume Lavoué. A local roughness measure for 3d meshes and its application to visual masking. *ACM Trans. Appl. Percept.*, 5(4):1–23, 2009.

[LBAD+06] Jason Lawrence, Aner Ben-Artzi, Christopher DeCoro, Wojciech Matusik, Hanspeter Pfister, Ravi Ramamoorthi, and Szymon Rusinkiewicz. Inverse shade trees for non-parametric material representation and editing. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 735–745, 2006.

[Lew95] J. P. Lewis. Fast normalized cross-correlation. In *Vision Interface*, pages 120–123. Canadian Image Processing and Pattern Recognition Society, 1995.

[LFTG97] Eric P. F. Lafortune, Sing-Choong Foo, Kenneth E. Torrance, and Donald P. Greenberg. Non-linear approximation of reflectance functions. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 117–126, 1997.

[LGB+02] Hendrik P. A. Lensch, Michael Goesele, Philippe Bekaert, Jan Kautz, Marcus A. Magnor, Jochen Lang, and Hans-Peter Seidel. Interactive rendering of translucent objects. In *PG '02: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, page 214, 2002.

[LH01] David P. Luebke and Benjamin Hallen. Perceptually-driven simplification for interactive rendering. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 223–234, London, UK, 2001. Springer-Verlag.

[LH06] Sylvain Lefebvre and Hugues Hoppe. Appearance-space texture synthesis. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 541–548, 2006.

[LHNW00] David Luebke, Benjamin Hallen, Dale Newfield, and Benjamin Watson. Perceptually driven simplification using gaze-directed rendering. Technical Report CS-2000-04, University of Virginia, 2000.

[LHZ+04] Xinguo Liu, Yaohua Hu, Jingdan Zhang, Xin Tong, Baining Guo, and Heung-Yeung Shum. Synthesis and rendering of bidirectional texture functions on arbitrary surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 10(3):278–289, 2004.

[LK02] Lutz Latta and Andreas Kolb. Homomorphic factorization of brdf-based lighting computation. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 509–516, New York, NY, USA, 2002. ACM.

[LKG+01] Hendrik P. A. Lensch, Jan Kautz, Michael Goesele, Wolfgang Heidrich, and Hans-Peter Seidel. Image-based reconstruction of spatially varying materials. In *In Rendering Techniques '01 (Proceedings of Eurographics Rendering Workshop*, pages 104–115, 2001.

[LKG+03] Hendrik P. A. Lensch, Jan Kautz, Michael Goesele, Wolfgang Heidrich, and Hans-Peter Seidel. Image-based reconstruction of spatial appearance and geometric detail. *ACM Trans. Graph.*, 22(2):234–257, 2003.

[LM99] Thomas Leung and Jitendra Malik. Recognizing surfaces using three-dimensional textons. In *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, page 1010, Washington, DC, USA, 1999. IEEE Computer Society.

[Lou04] M. I. A. Lourakis. levmar: Levenberg-Marquardt nonlinear least squares algorithms in C/C++. `http://www.ics.forth.gr/~lourakis/levmar/`, 2004.

[LT00] Peter Lindstrom and Greg Turk. Image-driven simplification. *ACM Trans. Graph.*, 19(3):204–241, 2000.

[Lue01] David P. Luebke. A developer's survey of polygonal simplification algorithms. *IEEE Comput. Graph. Appl.*, 21(3):24–35, 2001.

[LVJ05] C.H. Lee, A. Varshney, and D.W. Jacobs. Mesh saliency. In *ACM SIGGRAPH 2005 Papers*, page 666. ACM, 2005.

[LYS01] Xinguo Liu, Yizhou Yu, and Heung-Yeung Shum. Synthesizing bidirectional texture functions for real-world surfaces. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 97–106. ACM, 2001.

[MAA01] Michael D. McCool, Jason Ang, and Anis Ahmad. Homomorphic factorization of BRDFs for high-performance rendering. *ACM Transactions on Graphics*, 20(1):185–194, 2001.

[MDMS05] Rafał Mantiuk, Scott Daly, Karol Myszkowski, and Hans-Peter Seidel. Predicting visible differences in high dynamic range images -

model and its calibration. In Bernice E. Rogowitz, Thrasyvoulos N. Pappas, and Scott J. Daly, editors, *Human Vision and Electronic Imaging X, IS&T/SPIE's 17th Annual Symposium on Electronic Imaging (2005)*, volume 5666, pages 204–214, 2005.

[MK06] Sebastian Magda and David Kriegman. Reconstruction of volumetric surface textures for real-time rendering. In *Proceedings of the Eurographics Symposium on Rendering*, pages 19–29, 2006.

[MMK03] G. Müller, J. Meseth, and R. Klein. Compression and real-time rendering of measured btfs using local pca. In *Vision, Modeling and Visualisation 2003*, pages 271–280, November 2003.

[MMS04] Rafał Mantiuk, Karol Myszkowski, and Hans-Peter Seidel. Visible difference predicator for high dynamic range images. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, pages 2763–2769, 2004.

[MMS+05] G. Müller, J. Meseth, M. Sattler, R. Sarlette, and R. Klein. Acquisition, synthesis and rendering of bidirectional texture functions. *Computer Graphics Forum*, 24(1):83–109, 2005.

[MSK06] Gero Müller, Ralf Sarlette, and Reinhard Klein. Data-driven local coordinate systems for image-based rendering. *Computer Graphics Forum*, 25(3), September 2006.

[MSK07] Gero Müller, Ralf Sarlette, and Reinhard Klein. Procedural editing of bidirectional texture functions. In J. Kautz and S. Pattanaik, editors, *Eurographics Symposium on Rendering 2007*. The Eurographics Association, June 2007.

[NDM05] Addy Ngan, Frédo Durand, and Wojciech Matusik. Experimental analysis of brdf models. In *Proceedings of the Eurographics Symposium on Rendering*, pages 117–226, 2005.

[NIK91] Shree K. Nayar, Katsushi Ikeuchi, and Takeo Kanade. Surface reflection: Physical and geometrical perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(7):611–634, 1991.

[NRDR05] Diego Nehab, Szymon Rusinkiewicz, James Davis, and Ravi Ramamoorthi. Efficiently combining positions and normals for precise 3d geometry. *ACM Trans. Graph.*, 24(3):536–543, 2005.

[NRH+92] F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, and T. Limperis. Geometrical considerations and nomenclature for reflectance. *NBS Monograph*, pages 94–145, 1992.

[NRH04] Ren Ng, Ravi Ramamoorthi, and Pat Hanrahan. Triple product wavelet integrals for all-frequency relighting. *ACM Trans. Graph.*, 23(3):477–487, 2004.

[NSI99] Ko Nishino, Yoichi Sato, and Katsushi Ikeuchi. Appearance compression and synthesis based on 3d model for mixed reality. In *ICCV (1)*, pages 38–45, 1999.

[NVI09] NVIDIA. *CUDA Programming Guide*, 2009.

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

[PL07] Fabio Pellacini and Jason Lawrence. Appwand: editing measured materials using appearance-driven optimization. *ACM Trans. Graph.*, 26(3):54, 2007.

[PW96] A. B. Poirson and B. A. Wandell. Pattern-color separable pathways predict sensitivity to simple colored patterns. *Vision Research*, 36(4):515–526, 1996.

[QM08] Lijun Qu and Gary W. Meyer. Perceptually guided polygon reduction. *IEEE Transactions on Visualization and Computer Graphics*, 14(5):1015–1029, 2008.

[RBS99] Mark A. Robertson, Sean Borman, and Robert L. Stevenson. Dynamic range improvement through multiple exposures. In *ICIP (3)*, pages 159–163, 1999.

[RC98] Sébastien Roy and Ingemar J. Cox. A maximum-flow formulation of the n-camera stereo correspondence problem. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, page 492, 1998.

[Red97] Martin Reddy. *Perceptually Modulated Level of Detail for Virtual Environments*. PhD thesis, University of Edinburgh, 1997.

[Red01] Martin Reddy. Perceptually optimized 3d graphics. *IEEE Comput. Graph. Appl.*, 21(5):68–75, 2001.

[RSSF02] Erik Reinhard, Michael Stark, Peter Shirley, and James Ferwerda. Photographic tone reproduction for digital images. *ACM Trans. Graph.*, 21(3):267–276, 2002.

[SDR04] R. Strzodka, M. Droske, and M. Rumpf. Image registration by a regularized gradient flow. a streaming implementation in dx9 graphics hardware. *Computing*, 73(4):373–389, 2004.

[SK98a] A. Schilling and Reinhard Klein. Rendering of multiresolution models with texture. *Computer and Graphics*, 22(6):667–674, December 1998.

[SK98b] A. Schilling and Reinhard Klein. Texture dependent refinement of multiresolution meshes. Technical Report WSI–98–2, Wilhelm-Schickard-Institut für Informatik, Graphisch-Interaktive Systeme (WSI/GRIS), Universität Tübingen, 1998.

[SO95] Gerda J. F. Smets and Kees J. Overbeeke. Trade-off between resolution and interactivity in spatial task performance. *IEEE Comput. Graph. Appl.*, 15(5):46–51, 1995.

[SSK03] Mirko Sattler, Ralf Sarlette, and Reinhard Klein. Efficient and realistic visualization of cloth. In *EGSR '03: Proceedings of the 14th Eurographics Symposium on Rendering*, pages 167–177. Eurographics Association, 2003.

[SSZG95] Greg Spencer, Peter Shirley, Kurt Zimmerman, and Donald P. Greenberg. Physically-based glare effects for digital images. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 325–334, New York, NY, USA, 1995. ACM.

[SvBLD03] F. Suykens, K. vom Berge, A. Lagae, and P. Dutre. Interactive rendering with bidirectional texture functions. *Computer Graphics Forum*, 22(3):463–472, 2003.

[Tat06] Natalya Tatarchuk. Dynamic parallax occlusion mapping with approximate soft shadows. In *I3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 63–69, 2006.

[TS67] K. E. Torrance and E. M. Sparrow. Theory for off-specular reflection from roughened surfaces. *J. Opt. Soc. Am.*, 57(9):1105–1114, 1967.

[TWL+05] Xin Tong, Jiaping Wang, Stephen Lin, Baining Guo, and Heung-Yeung Shum. Modeling and rendering of quasi-homogeneous materials. *ACM Trans. Graph.*, 24(3):1054–1061, 2005.

[War92] Gregory J. Ward. Measuring and modeling anisotropic reflection. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 265–272, 1992.

[War03] Greg Ward. Fast, robust image registration for compositing high dynamic range photographs from hand-held exposures. *journal of graphics, gpu, and game tools*, 8(2):17–30, 2003.

[Win05] Stefan Winkler. Digital video quality. In *Digital Video Quality - Vision Models and Metrics*. Wiley, J, 2005.

[WLC+03] Nathaniel Williams, David Luebke, Jonathan D. Cohen, Michael Kelley, and Brenden Schubert. Perceptually guided simplification of lit, textured meshes. In *I3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics*, pages 113–121. ACM, 2003.

[WSBL03] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h.264/avc video coding standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7):560–576, August 2003.

[WTS+05] Jiaping Wang, Xin Tong, John Snyder, Yanyun Chen, Baining Guo, and Heung-Yeung Shum. Capturing and rendering geometry details for btf-mapped surfaces. *The Visual Computer*, 21(8-10):559–568, 2005.

[Wyn01] Chris Wynn. Real-time brdf-based lighting using cube-maps. NVidia Corporation, 2001.

[YIB05] Pan Y., Cheng I., and Basu. Quality metric for approximating subjective evaluation of 3-d objects. *IEEE Transactions on Multimedia.*, 7(2):269–279, 2005.

[YSZ04] J. Yan, P. Shi, and D. Zhang. Mesh simplification with hierarchical shape analysis and iterative edge contraction. *IEEE Transactions on visualization and computer graphics*, pages 142–151, 2004.

[ZBK02] Todd Zickler, Peter N. Belhumeur, , and David J. Kriegman. Helmholtz stereopsis: Exploiting reciprocity for surface reconstruction. *International Journal of Computer Vision*, 49(2/3):215–227, 2002.

Lebenslauf

Persönliche Daten

Name	Dipl. Inf. Nicolas Menzel
Anschrift	Ortsstr. 12 64646 Heppenheim Telefon: 06252/69988500
Geburtsdatum und -ort	20.11.1979 in Bielefeld
Familienstand	verheiratet

Studium

2007 bis 2010	**Doktorand** (Philipps-Universität Marburg)
2000 bis 2007	**Studium der Informatik** (Philipps-Universität Marburg) Studienschwerpunkte: Computergrafik, Rechnernetze, rechnergestützte Beweissysteme Diplomarbeit: „*High Dynamic Range Surface Light Fields*“ (Note 1,0) Diplom Gesamtnote **"sehr gut"**

Schulische Ausbildung

1992 - 1999	Helmholtz Gymnasium Bielefeld, Abschluss: Abitur
1986 - 1992	Fröbel Grundschule Bielefeld

Zivildienst

1999-2000	Lebenshilfswerk Marburg (LHM) Betreuung behinderter Erwachsener in einem Wohnheim

Publikationen

Jahr	***Titel***	
2010	*Parallel View-Dependent Out-of-Core Progressive Meshes*	Evgenij Derzapf, Michael Guthe, Nicolas Menzel
2010	*Parallel View-Dependent Refinement of Compact Progressive Meshes (Co-Author)*	Evgenij Derzapf, Michael Guthe, Nicolas Menzel
2009	*g-BRDFs: An Intuitive and Editable BRDF Representation*	Nicolas Menzel, Michael Guthe
2008	*A Bi-Adaptive Sampling Scheme for Real-Time Subsurface Scattering*	Nicolas Menzel, Michael Guthe
2007	*High Dynamic Range Preserving Compression of Light Fields and Reflectance Fields*	Nicolas Menzel, Michael Guthe
2007	*Freehand HDR Photography with Motion Compensation*	Nicolas Menzel, Michael Guthe

Eidesstattliche Erklärung

Die Dissertation habe ich selbständig angefertigt. Alle Hilfsmittel und Hilfen habe ich angegeben, insbesondere habe ich die wörtlich oder dem Sinne nach anderen Veröffentlichungen entnommenen Stellen kenntlich gemacht.

Einer Doktorprüfung habe ich mich bisher nicht unterzogen. Die Dissertation hat bisher weder in der gegenwärtigen noch in einer anderen Fassung weder dem Fachbereich Mathematik und Informatik der Philipps-Universität Marburg noch einer anderen Fakultät oder einem anderen Fachbereich oder einem ihrer bzw. seiner Mitglieder vorgelegen.

_____

Ort, Datum

_____

Unterschrift