An Interactive Shader for Natural Diffraction Gratings

Bachelorarbeit

der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von

Michael Single

2014

Leiter der Arbeit:
Prof. Dr. Matthias Zwicker
Institut für Informatik und angewandte Mathematik

Abstract

In nature color production is the result of physical interaction of light with a surface's nanostructure. In his pioneering work, Stam developed limited reflection models based on wave optics, capturing the effect of diffraction on very regular surface structures. We propose an adaption of his BRDF model such that it can handle complex natural gratings. On top of this, we describe a technique for interactively rendering diffraction effects, as a result of physical interaction of light with biological nanostructures such as snake skins. As input data, our method uses discrete height fields of natural gratings acquired by using atomic force microscopy (AFM). Based on Taylor Series approximation we leverages precomputation to achieve interactive rendering performance (about 5-15 fps). We demonstrate results of our approach using surface nanostructures of different snake species applied on a measured snake geometry. Lastly, we evaluate the qualtiy of our method by a comparision of the maxima for peak viewing angles using the data produced by our method against the maxima resulting by the grating equation.

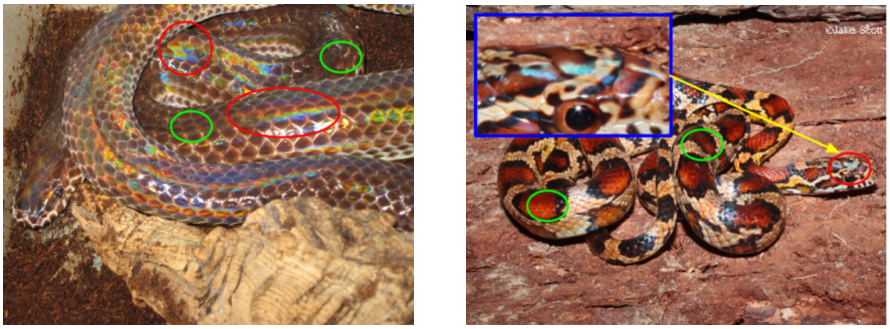
Contents

Chapter 1

Introduction

1.1 Motivation

As human beings, we visually perceive and experience our whole world in terms of colors, resulting from various physical phenomina involving interaction between light and matter. Particularly in nature, there are basically two main causes for color production. Firstly, due to pigmentation, which occurs since certain molecules in a biological structure selectively absorb or reflect specific wavelengths from an incident light source. And secondly because of structural colors which are the result of physical interaction of light with a nanostructure, exclusively relying on the structuring of the material and not any other property. A natural diffraction grating is a semitransparent layer of biological nano-structures which exhibits a certain degree of regularity to produce structural colors by diffracting an incident light source. One particular example for such biological color production are the colors we can see when having a closer look at the illuminated skin of snakes, as shown in figure 1.1.



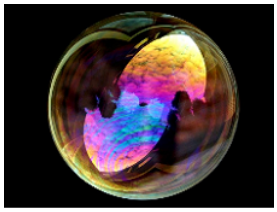
(a) Xenopeltis snake (b) Elaphe Guttata snake

Figure 1.1: Examples of pigmentation color (green circles) and structural color (red circles) on different snake species[1].

Some species like Xenopeltis express structural colors in form of iridescent patterns along their scales way strongerthan others like Elaphe species. The reason for this lies on the nanostructure

[1]image source of figure 1.1(a) `http://www.snakes-alive.co.uk/gallery_5.html` and figure 1.1(b) `http://www.the-livingrainforest.co.uk/living/view_price.php?id=464`

of their skins. There are a vast amount of additional reasons for producing structural colors in nature, such as thin film interference, intra-cellular photonic crystals or diffraction gratings. More detailed examples are shown in figure 1.2.



(a) Thin Film Interference soap bubble

(b) Multilayer Interference on abdomen of beetle

(c) Photonic crystals in Wings of butterfly

(d) Scattering of light on butterfly wing

(e) Artificial Diffraction Grating on CD

(f) Natural Diffraction Grating on snake skin

Figure 1.2: Examples[2] for structural colors on the wings and the abdomen of insects, liquids, synthetic structures, and on scales on the skin of reptiles.

As far back as in the 17th century, Robert Hooke was able to relate the cause of structural colors to the the microstructure of a material. During his examinations of peacock feathers he found that the colors could be made disappear by wetting the feathers and further observed that the feathers are made of tiny ridges. Building on top of the latest knowledge about interference, Newton related structural colors with wave interference. Recently, in the field of computer graphics, many researchers have developed models to render structural colors, but most of the currently available models are not able to perform interactive rendering or are oversimplified and thus cannot model accurately the effect of diffraction.

This thesis investigates this particular problem in detail and provides a solution for rendering structural colors due to diffraction on natural gratings.

[2]image source of figure:

- 1.2(a): `http://www.ualberta.ca/~pogosyan/teaching/PHYS_130/FALL_2010/lectures/lect33/lecture33.html`
- 1.2(b): `http://www.itp.uni-hannover.de/~zawischa/ITP/multibeam.html`
- 1.2(c): `http://upload.wikimedia.org/wikipedia/commons/a/a4/Parides_sesostris_MHNT_dos.jpg`
- 1.2(d): From paper [MT10], figure 6.
- 1.2(e): `http://cnx.org/content/m42496/latest/?collection=col11428/latest`
- 1.2(f): `http://www.snakes-alive.co.uk/gallery_5.html`

1.2 Goals

The purpose of this thesis is firstly, to simulate physically accurate structural colors caused by the effect of diffraction on various biological structures and secondly implement this simulation as a renderer with interactive behaviour. We mainly focus on structural colors generated by natural diffraction gratings. In particular the approach presented in this thesis applies to surfaces with quasiperiodic structures at the nanometer scale which can be represented as height fields stored in gray-scale images.

Natural gratings like this are found on the scale of reptiles, wings of butterflies or the bodies of various insects but we restrict ourself and focus on snake skins. The data of our discrete valued height fields, which are representing the surface of a measured snake skin was acquired using atomic force microscopy (AFM)[3]. Figure 1.3 shows a measured height field of a Xenopeltis snake, which stored in a grayscale image. The surface of its skin is composed of many finger like structures. Locally, these fingers seem to be very regularly aligned (red box). However, globally, we observe that the alignment of the fingers is irregularly curved (indicated by green curves) along the whole surface. This kind of global irregularity is why it makes it hard to model the structural complexity of natural gratings[4].

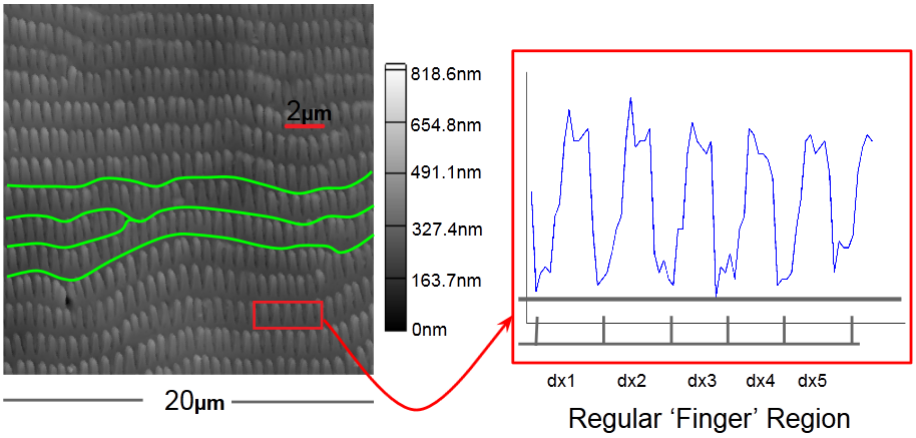


Figure 1.3: Height field of a Xenopeltis snake[5] skin taken by AFM and stored as a grayscale image. Locally, this natural grating consists of regularly aligned (red box) finger-like substructures, but globall we observe a curved alignment of these structures (green curves).

The renderer discussed in this thesis is based on the pioneering work of J. Stam about diffraction shaders [Sta99] in which he formulated a BRDF modelling the effect of diffraction. Nevertheless we have to adapt his BRDF model since his model assumes, that a given surface of a grating can either be formulated by an analytical function, and therefore has a closed form solution or it is simple enough to be modelled effectively by relying on statistical methods. However, we are dealing with natural diffraction gratings, represented as explicitly formulated height fields, which unfortunately are neither known analytically nor do they fit into simple statistical models. This

[3] All data is provided by the Laboratory of Artificial and Natural Evolition in Geneva. See their website:`www.lanevol.org`

[4] E.g. by relying on statistical methods, caputuring surface details by introducing an appropriate distribution function of the finger strucures.

[5] This image was provided by the LANE lab in Geneva

thesis thus proposes an extension of J. Stam's work for the complex case of explicitly defined, discrete and quasi-periodic height field structures.

In the following section a breif overview of previous work relevant and related to this work will be presented.

1.3 Previous work

The first scientific descriptions of structural colors was previousely by Hooke in 1665 in his book Micrographia[R.H12]. Hooke investigated feathers of peacocks using one of the first microscopes from his time and found out that the colors on the feather were canceled out whenever a drop of water moistened the feather. He proposed the speculation that a layer of thin plates and air were responsible for reflecting the light and thus he related the structure of the feather to colors. In Newton's book Opticks[I.N14] he described that the colors of the peacock feather are related to the thinness of the transparent part of the feathers. Around 1800 T.Young explains structural colors as a result of wave interference using his double-slit experiment[6][T.Y07], published in the journal Philosophical Transactions of the Royal Society.

In the field of computer graphics, J.Stam[Sta99] was the first one who was able to develop reflection models based on wave optics capturing the effect of diffraction due to nano-structure height fields. His model is an approximation of far field diffraction[7] effects relying on the Kirchhof integral[8]. For a certain class of surfaces which can be modelled as a height field he provides an analytical solution of the BRDF model. He assumes homogeneity of the structure and then the main idea of his model is the formulate a BRDF as the Fourier Transform applied on the the correlation function of the given height field. However, the height fields that Stam is dealing with are either extremely regular or can be considered as a superposition of randomly distributed bumps forming a periodic like structure relying on probabilistic distribution theory[9]. Both height field assumptions allow him to derive an analytical solution using statistical models. However, the height field we are dealing with are measured, complex, biological nano-structures and thus they do not exhibit regularity at a global scale as demonstrated in figure 1.3. It is not sufficient to superimpose one particular nano finger (considering it as a bump) for capturing the complexity of the measured structure since this poses a non-trivial problem of modeling the distribution of nano finger statistically. Therefore, we cannot directly use Stam's BRDF model when we want to perform interactive rendering for diffraction effects of natural gratings.

In 2012 Cuypers et all [CT12] proposed a wave based Bidirectional scattering distribution function (BSDF[10]) denoted as WBSDF. Using the rendering equation and Wigner Distribition Functions[11] (WDF) they related their WBSDF model to the incoming wavefront and hence, their model can be adapted such that it can be rendered by a Monte Carlo renderer. The advantage of their model over Stam's is that their models also captures near field diffraction effects. A disadvantage their model is computational expensive since the WDF of a two dimensional surface is a four dimensional function and therefore can hardly be used in order to perform interactive rendering.

[6] See http://en.wikipedia.org/wiki/Double-slit_experiment
[7] See http://en.wikipedia.org/wiki/Fraunhofer_diffraction
[8] See http://en.wikipedia.org/wiki/Kirchhoff_integral_theorem
[9] See http://en.wikipedia.org/wiki/Probability_distribution
[10] See http://en.wikipedia.org/wiki/Bidirectional_scattering_distribution_function
[11] See http://en.wikipedia.org/wiki/Wigner_distribution_function

Linday and Agu [CT12] proposed an approach in order to perform interactive rendering diffraction effect by precomputing and storing their BRDF model using spherical harmonics. Nonetheless, for complex natural gratings their BRDF may be insufficient accurate since their approach is using low order spherical harmonics.

1.4 Thesis Structure

The reminder of this thesis is organised as follows: Due to the fact that this thesis has a rather advanced mathematical complexity, chapter 2 introduces some important definitions about modelling light in computer graphics and some wave theory. These concept are required in order to be able to follow our later derivations. This is followed by a brief summary of J. Stam's Paper about diffraction shaders, since his BRDF formulation is the basis of our derivations.

In chapter 3 we adapt Stam's BRDF model step-wise in a way that we will end up with a representation which can be implemented as an interactive diffraction renderer when using natural diffraction gratings. We also propose an alternative formulation, the so called PQ approach in this chapter and discuss its short-comings.

Chapter 4 addresses the practical part of this thesis, the implementation of our diffraction model, explaining all precomputation steps and how rendering is preformed in our reference framework for this thesis.

Chapter 5 gives some further insight about diffraction by explaining the topic about diffraction grating in depth. Furthermore, within this chapter we evaluate the qualitative validity of our BRDF model applied on different surface gratings by computing their reflectance and comparing the results to the grating equation under similar conditions.

Chapter 6 presents our rendered results, first the so called BRDF maps for all our gratings and shading approaches under various shading parameters and then the actual renderings on a snake skin. And finally chapter 7 contains the conclusion of this thesis discussing what has been achieved in this thesis and all the drawbacks of the proposed method. It also contains a note about some of my personal experiences during this thesis.

Chapter 2

Theoretical Background

2.1 Basics in Modeling Light in Computer Graphics

2.1.1 Radiometry

One purpose of Computer Graphics is to simulate the interaction of light on a surface and how a real-world observer, such as a human eye, will perceive this. These visual sensations of an eye are modeled relying on a virtual camera which captures the emitted light from the surface. The physical basis to measure such reflected light depicts radiometry which is about measuring the electromagnetic radiation transfered from a soruce to a receiver.

Fundamentally, light is a form of energy propagation, consisting of a large collection of photons, whereat each photon can be considered as a quantum of light that has a position, direction of propagation and a wavelength λ. A photon travels at a certain speed $v = \frac{c}{n}$, that depends only the speed of light c and the refractive index n through which it prograte. Its frequency is defined by $f = \frac{v}{\lambda}$ and its carried amount of energy q, mearsured in the SI unit Joule, is given by $q = hf = \frac{hv}{\lambda n}$ where h is the Plank's constant. The total energy of a large collection of photons is hence $Q = \sum_i q_i$.

2.1.2 Spectral Energy

It is important to understand that the human eye is not equally sensitive to all wavelength of the spectrum of light and therefore responds differently to specific wavelengths. Remember that our goal is to model the human visual perception. This is why we consider the energy distribution of a light spectrum rather than considering the total energry of a photon collection since then we could weight the distribution according the human visual system. So the question we want to answer is: How is the energy distributed across wavelengths of light?

The idea is to make an energy histrogram from a given photon collection. For this we have to order all photons by their associated wavelength, discretize wavelength spectrum, count all photons which then will fall in same wavelength-interval, and then, finally, normalize each interval by the total energy Q. This will give us a histogram which tells us the spectral energy Q_λ for a given discrete λ interval and thus models the so called spectral energy distribution [1].

[1] Intensive quantities can be thought of as density functions that tell the density of an extensive quantity at an infinitesimal point.

2.1.3 Spectral Power

Rendering an image in Computer Graphics corresponds to capturing the color sensation of an illuminated, target scene at a certain point in time. As previousely seen, each color is associated by a wavelength and is directly related to a certain amount of enegry. In order to determine the color of a to-be-rendered pixel of an image, we have to get a sense of how much light (in terms of energy) passes through the area which the pixel corresponds to. One possibility is to consider the flow of energy $\Phi = \frac{\Delta Q}{\Delta t}$ transferred through this area over a small period of time. This allows us to measure the energy flow through a pixel during a certain amount of time.

In general, power is the estimated rate of energy production for light sources and corresponds to the flux. It is measured in the unit Watts, denoted by Q. Since power is a rate over time, it is well defined even when energy production is varying over time. As with Spectral Energy for rendering, we are really interested in the spectral power $\Phi_\lambda = \frac{\Phi}{\lambda}$, measured in Watts per nanometer.

2.1.4 Spectral Irradiance

Before we can tell how much light is reflected from a given point on a surface towards the viewing direction of an observer, we first have to know how much light arrives at this point. Since in general a point has no length, area or even volume associated, let us instead consider an infinitimal area ΔA around a such a point. Then, we can ask ourself how much light falls in such a small area. When further observing this process over a short period in time, this quantity is the spectral irradiance E as illustrated in figure 2.1. Summarized, this quantity tells us how much spectral power is incident on a surface per unit area and mathematically is equal:

$$E = \frac{\Phi_\lambda}{\Delta A} \tag{2.1}$$

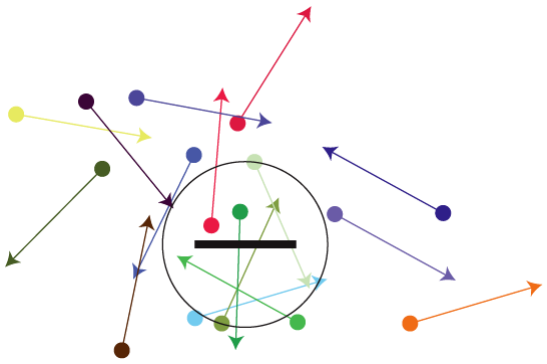
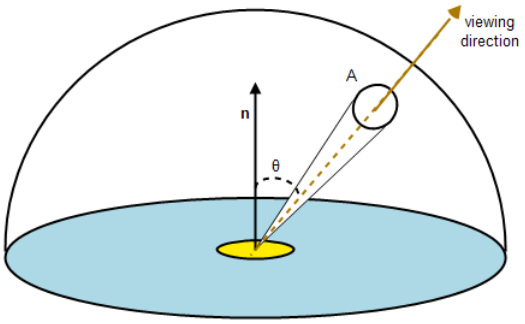


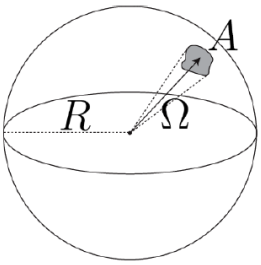
Figure 2.1: Irradiance is the summed up radiance over all directions

2.1.5 Spectral Radiance

When rendering an image we have to determine the color of each pixel of the image. Although irradiance tells us how much light is arriving at a point as illustrated in figure 2.1, it tells us little about the direction that light comes from. This relates to how the human eye perceives the brightness of an illuminated objects when looking at it in a certain direction.



(a) Radiances is the density of photons per area per solid angle



(b) Solid angle is the area of a surface patch on a sphere with radius R which is spanned by a set of directions

This concepted is described by the radiometric quantity radiance. Basically, this is a measure of light energy passing through or is emitted off from a small area around a point on a surface towards a given direction during a short period in time. More formally this is the spectral power emerging from an arbitrary point (an infinitimal area around this point) and falls within a given solid angle (see figure[2] 2.2(b))in specific direction (usually towards the observer) as shown in figure 2.2(a). Formally, this leads us to the following mathematical formalism:

$$L_\lambda(\omega) = \frac{d^2\Phi_\lambda}{dAd\Omega} \approx \frac{\Phi_\lambda}{\Omega A} \tag{2.2}$$

where L is the observed spectral radiance in the unit energy per unit area per solid angle, which is $Wm^{-}2sr^{-}1$ in direction ω which has an angle θ between the surface normal and ω, Θ is the total flux or power emitted, θ is the angle between the surface normal and the specified direction, A is the area of the surface and Ω is the solid angle in the unit steradian subtended by the observation or measurement.

It is useful to distinguish between radiance incident at a point on a surface and excitant from that point. Terms for these concepts sometimes used in the graphics literature are surface radiance L_r for the radiance *reflected* from a surface and field radiance L_i for the radiance *incident* at a surface.

2.1.6 BRDF

In order to render the colorization of an observed object, a natural question in computer graphics is what portion of the reflected, incident light a viewer will receive, when he looks at an illuminated object. Therefore for any given surfaces which is illuminated from a certain direction ω_i, we can ask ourself how much light is reflected off of any point on this surface towards a viewing direction ω_r. This is where the Bidirectional Reflectance Distribution Function (short: BRDF) comes into play, which is a radiometric quantity telling us how much light is reflected at an opaque surface. Mathematically speaking, the BRDF is the ratio of the reflected radiance pointing to the direction ω_r to the incident irradiance comming from the inverse direction of ω_i as illustrated in figure 2.2. Hence the BRDF is a four dimensional function defined by four angles θ_i, ϕ_i, θ_r and ϕ_r.

[2]Similar figute like used in computer graphics class 2012 in chapter colors

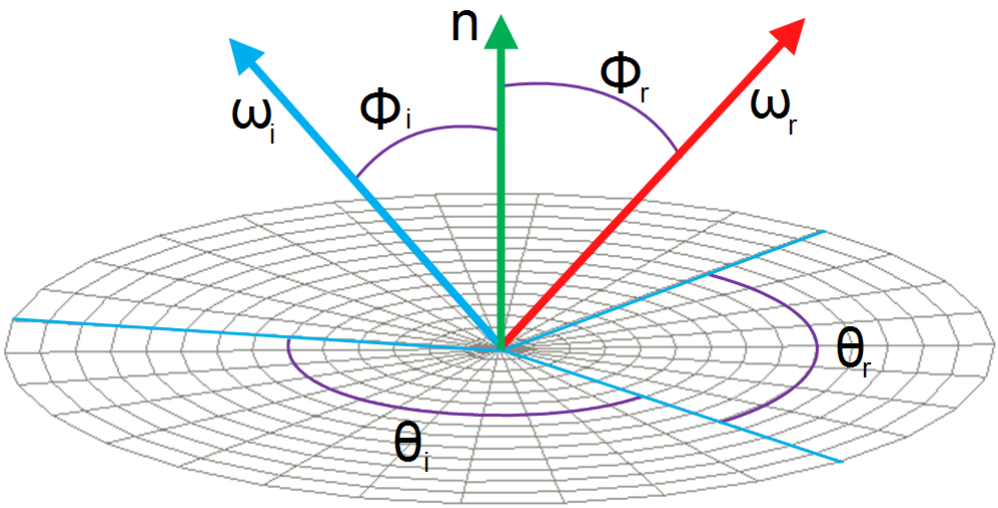


Figure 2.2: Illistration of the BRDF model, where ω_i is pointing to the light source and the existing direction is denoted by ω_r. Both direction unit direction vectors defined w.r.t to a surface normal **n** for every point on the surface.

Which formally is for any given wavelength λ equivalent to:

$$\begin{aligned} BRDF_\lambda(\omega_i, \omega_r) &= \frac{dL_r(\omega_r)}{dE_i(\omega_i)} \\ &= \frac{dL_r(\omega_r)}{L_i(\omega_i)cos(\theta_i)d\omega_i} \end{aligned} \tag{2.3}$$

Where L_r is the reflected spectral radiance, E_i is the spectral irradiance and θ_i is the angle between ω_i and the surface normal **n**.

2.1.7 Wavespectrum and Colors

In order to see how crucial the role of human vision plays, let us consider the following definition of color by *Wyszechkiu and Siles*[3] stating that *Color is the aspect of visual perception by which an observer may distinguish differences between two structure-free fields of view of the same size and shape such as may be caused by differences in the spectral composition of the radiant energy concerned in the observation.* Therefore, similarly like the humans' perceived sensation of smell and taste, color vision is just another individual sense of perception giving us the ability to distinguish different frequency distribution of light experienced as color.

[3]mentioned in Computer Graphics Fundamentals Book from the year 2000

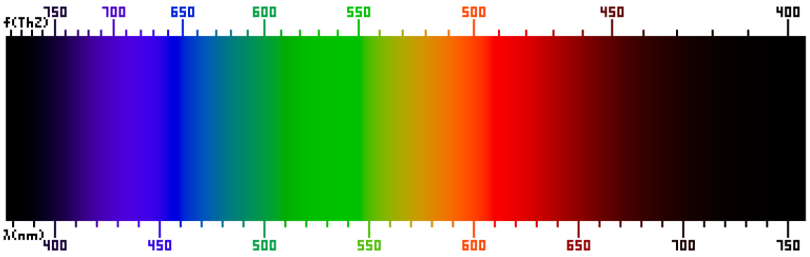


Figure 2.3: Frequency (top) and wavelenght (bottom) of colors of the visible light spectrum[4].

In general an eye consists of photoreceptor cells which are responsible for providing ability of color-perception. A schematic of an eye is illustratied in figure 2.4. Basically, there are two specialized types of photoreceptor cells, cone cells which are responsible for color vision and rod cells, which allow an eye to perceive different brigthness levels.

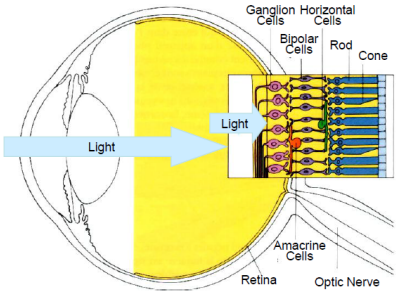


Figure 2.4: Schematic[5] of photoreceptor cells, cones and rods, in human eye

A human eye is made of three different types of cone cells, having their peak sensivity in sensing color at different wavelength ranges. More precisely, there are cone cells most sensitive to short wavelengths which are between $420nm$ and $440nm$, those which are most sensitive in the middle range between $530nm$ and $550nm$ and those which have their peak in the long range, from $560nm$ to $580nm$. In principle, any color sensation in human color perception as shown in figure 2.3 can therefore be described by just three parameters, corresponding to levels of stimulus of the three types of cone cells.

2.1.8 Colorspace

In order to render accurately images of how a human observer sees its world, a mathematical model of the human color perception is required. Remember that color sensation is due to a visual stimulus processed by cone cells in an eye. A human eye contains three different types of cone cells. Therefore, one possible approach is to describe each kind of these cone cells as a function of wavelength, returning a certain intensivity. In the early 1920, from a series of experiments the so called CIE XYZ color space was derived, describing response of cone cells of an average human individual, the so called standard observer. Basically, a statistically sufficiently large number of probands were exposed to different target light colors expressed by their wavelength. The task

[4]Similar figute like used in computer graphics class 2012 in chapter colors
[5]image of illustration has been taken from wikipedia

of each proband was to reproduce these target colors by mixing three given primary colors, red-, green- and blue-light. The strength of each primary color could be manually adjusted by setting their relative intensivity. Those adjustment weights have been measured, aggregated and averaged among all probands for each primary color. This model describes each color as a triple of three real valued numbers[6], the so called tristimulus values.

Pragmatically speaking, color spaces describes the range of colors a camera can see, a printer can print or a monitor can display. Thus, formally we can define it as a mapping a range of physically produced colors from mixed light to an objective description of color sensations registered in the eye of an observer in terms of tristimulus values.

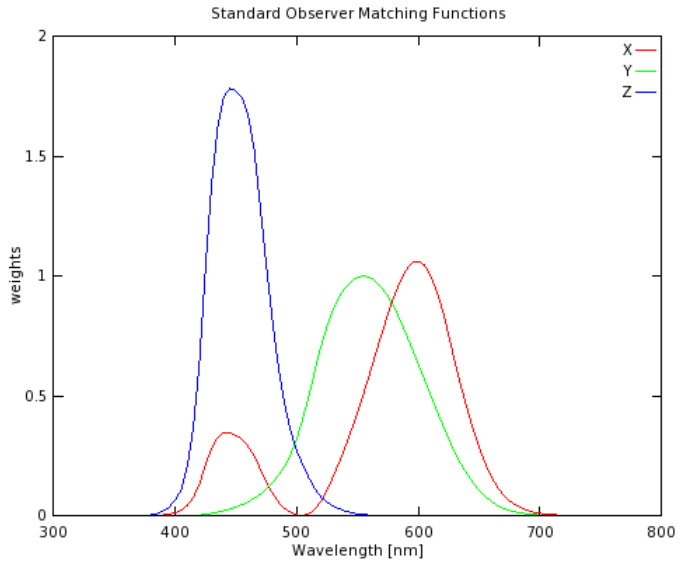


Figure 2.5: Plots of our color matching functions we used for rendering

Interpolating all measured tristimuli values gives us three basis functions, the CIE color matching functions $\overline{x}(\lambda), \overline{y}(\lambda), \overline{z}(\lambda)$. In figure 2.5 are the numerical description of the chromatic response of the observer. They can be thought of as the spectral sensitivity curves of three linear light detectors yielding the CIE Tristimulus values X, Y and Z.

The tristimulus values for a color with a spectral power distribution $I(\lambda)$, are given in terms of the standard observer by:

[6]note that there are negative color weights possible in the CIE XYZ colors space. This is why some human perceived color sensations could not be reconstructed using just an additive color model (adding three positively weighted primary values). Therefore, a probabant was also allowed to move one of the primary colors to the target color and instead was supposed to reproduce this new color mix using the two remaining primaries (subtractive model). The value of the selected, moved primary was then interpreted as beeing negative weighted in an additive color model.

$$X = \int_{\Lambda} I(\lambda)\,\overline{x}(\lambda)\,d\lambda$$
$$Y = \int_{\Lambda} I(\lambda)\,\overline{y}(\lambda)\,d\lambda$$
$$Z = \int_{\Lambda} I(\lambda)\,\overline{z}(\lambda)\,d\lambda \tag{2.4}$$

Where λ, is the wavelength of the equivalent monochromatic light spectrum $\Lambda = [380nm, 780nm]$. Note taht it is not possible to build a display that corresponds to the CIE XYZ colorspace. For this reasons it is necessary to design other color spaces, which are physical realizable, offers efficient encoding, are perceptual uniform and have an intuitive color specification. There are simple conversions between XYZ color space, to other color space described as linear transformations.

2.1.9 Spectral Rendering

When rendering an image, most of the time we are using colors described in a certain RGB color space. However, a RGB colorspace results from a colorspace transformation of the tristimulus values, which themself are inherent to the human visual system. Therefore, many physically light phenomenon are poorly modeled when always relying on RGB colors for rendering. Using only RGB colors for rendering is alike we would assume that a given light source emits light of only one particular wavelength. But in reality this is barely the case. Spectral rendering is referring to use a certain wavelength spectrum, e.g. the human visible light spectrum, instead simply using the whole range of RGB values in order to render an illuminated scene. This captures the physical reality of specific light sources way more accurate. Keep in mind that, even when we make use of a spectral rendering approach, we have to convert the final spectra to RGB values, when we want to display an image on an actual display.

2.2 Wave Theory for Light and Diffraction

2.2.1 Basics in Wave Theory

In order prepare the reader for physical relevant concepts used during later derivations and reasonings within this thesis, I am going to provide a quick introducation to the fundamental basics of wave theory and related concepts. In physics a wave describes a disturbance that travels from one location to another through a certain medium. The disturbance temporarly displaces the particles in the medium from their rest position which results in an energy transport along the medium during wave propagation. Usually, when talking about waves we are actually refering to a complex valued function which is a solution to the so called wave equation which is modeling how the wave disturbance proceeds in space during time.

There are two types of waves, mechanical waves which deform their mediums during propagation like sound waves and electromagnetic waves consiting of periodic oscilations of an electromagnetic field such as light for example. Like simplified illustrated in figure 2.6, there are several properties someone can use and apply in order to compare and distinguish different waves:

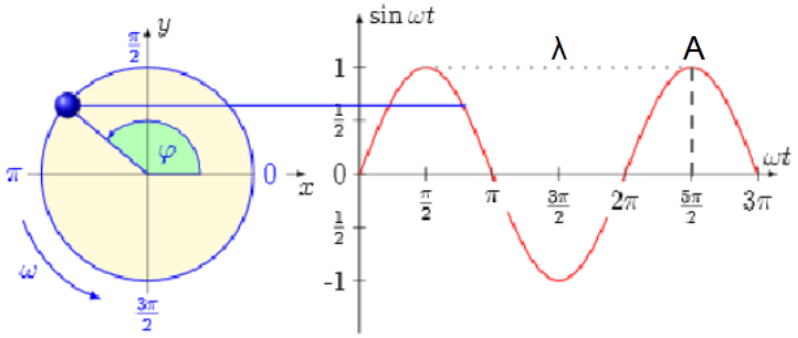


Figure 2.6: Simplified, one dimensionaly real valued wave function[7], giving an idea about some important wave properties. We denote the crest of a wave as the hightest point relative to the equilibrium line (zero height along time axis) and similarly the trough as the lowest point.

Wavelength: Is usually denoted by λ and is a measure for the spatial distance from one point to another until the shape of a wave repeats

Amplitude: Is denoted by A and there are two possible interprations: First, it is a measure of the height from the equilibrium point to the heighest point of a crest on the wave or the lowest point of a trough. This mean the amplitude can be positive or negative. However, usually, someone is just interested the absolute value of an amplitude, the magnitude of a wave. For light waves it is a relative measure of intensity or brightnes to other lught waves of the same wavelength. And secondly, it can be interpreted as a measure how much energy a wave carries wherate the greater the absoulte aplitute value, the bigger the amount of energy being carried.

Frequency: Is a measure of the number of waves which are passing through a particular point in the propagation medium during a certain time and is denoted by f.

Phase: Is denoted by ϕ. Describes either the offset of initial position of a wave or the relative displacement between or among waves having the same frequency. Two waves two waves with same frequency are denoted by being in phase if they have the same phase. This means they line up everywhere. As a remark, we denote by ω the angular frequency which is equal $2\pi f$.

A geometrical property of waves is their wavefront. This is either a surface or line along the path of wave propagation on which the disturbance at every point has the same phase. Three are basically three types of wavefronts: spherical-, cylindrical- and plane wavefront. If point in a isotropic medium is sending out waves in three dimensions, then the coresponding wavefronts are spheres, centered on the source point. Hence spherical wavefront is the result of a spherical wave, also denoted as a wavelet. Note that for electromagnetic waves, the phase is a poisition of a point in time on a wavefront cycle (motion of wave over a whole wavelength) whereat a complete cycle is defined as being equal 360 degrees.

2.2.2 Wave Interference

Next, after having seen that a wave is simply a traveling disturbance along a medium, having some special properties, someone could ask what happens when there are several waves travaling

[7]Image source: http://neutrino.ethz.ch/Vorlesung/FS2013/index.php/vorlesungsskript

on the same medium. Especially, we are interested how these waves will interact with each other. In physics the term interference denotes the interaction of waves when they encounter each other at a point along their propagation medium. At each point where two waves superpose, their total displacement at these points is the sum of the displacements of each indiviudal wave at those points. Then, the resulting wave is having a greater or lower amplitude than each seperate wave and this we can interprete the interference as the addition operator for waves. Two extreme scenarios are illustrated in figure 2.7. There are basically three variants of interferences which can occur, depending on how crest and troughs of the waves are matched up:

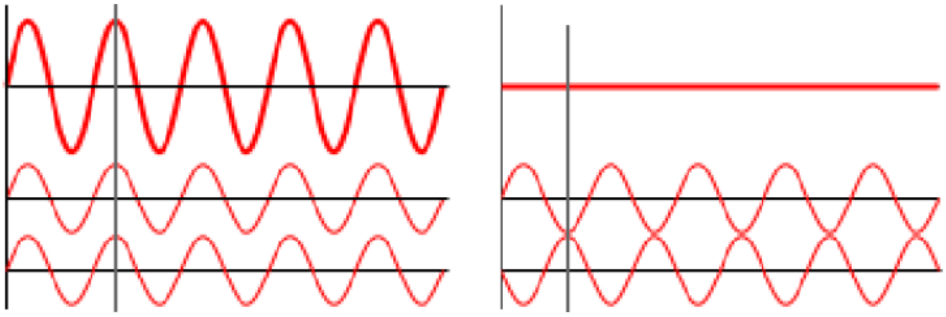


Figure 2.7: Interference scenarios[8] when two waves waves meet: On the left handside, there is constructive interference and on the right handside there is destructive interference illustrated.

- Either a crest of a wave meets a crest of another wave or similarly a trough meets a trough of another wave. This scenario is denoted as constructive interference and occurs at any location along the medium where the two interfering waves have a displacement in the same direction. This is equivalent like saying that the phase difference between the waves is a multiple of 2π. Then the resulting amplitude at that point is being much larger than the amplitude of an individual wave. For two waves with an equal amplitude interfering constructively, the resulting amplitude is twice as large as the amplitude of an individual wave.
- Either a crest of a wave meets a trough of another wave or vice versa. This scenario is denoted as destructive interference and occurs at any location along the medium where the two interfering waves have a displacement in the opposite direction. This is like saying that the phase difference between the waves is an odd multiple of π. Then the waves completely cancel each other out at any point they superimpose.
- If the phase difference between two waves is intermediate between the first two scenarios, then the magnitude of the displacement lies between the minimal and maximal values which we could get from constructive interference.

Keep in mind that when two or more waves interfere which each other, the resulting wave will have a different frequency. For a wave, having a different frequency also means having a different wavelength. Therefore, this directly implies that a light of a different color, than its source waves have, is emitted.

[8]Image source: `http://en.wikipedia.org/wiki/Interference_(wave_propagation)`

2.2.3 Wave Coherence

When considering waves which are traveling on a shared medium along the same direction, we could examine how their phase difference is changing over time. Formulating the changement of their relative phase as a function of time will provides us a quantitative measure of the synchronism of two waves, the so called wave coherence. In order to better understand this concept, let us consider a perfectly mathematical sine wave and second wave which is a phase-shifted replica of the first one. A property of mathematical waves is that they keep their shape over an infinity amount of moved wavelengths. In our scenario, both waves are traveling along the same direction on the same medium, like exemplarily illustrated in figure 2.8.

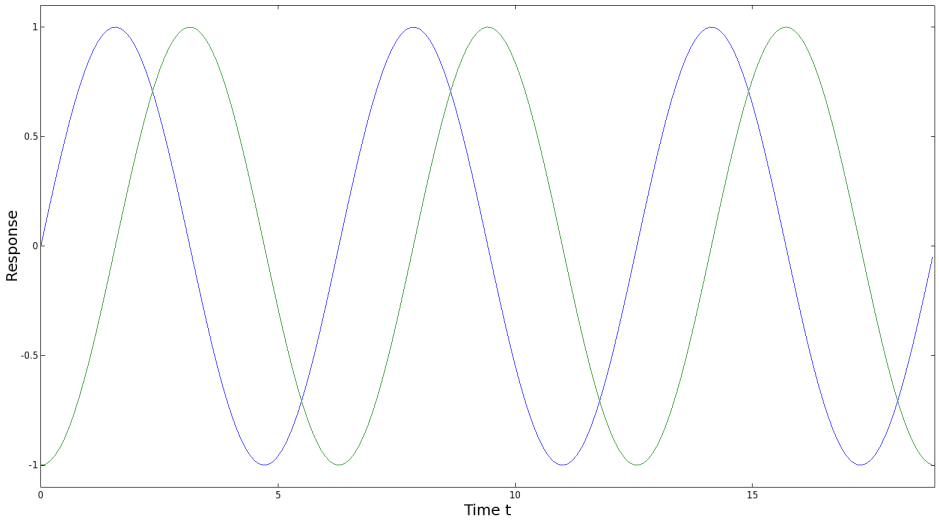


Figure 2.8: Two mathematical sine waves which are perfectly coherent which means that their phase difference is constant for every point in time.

Taking the difference between the two sine waves from the previous figure yields always a constant number. Therefore, those two waves are said to be coherent and hence perfectly synchronous over time. Notice that this scenario is completly artificial since in nature there are no mathematical sine waves. Rather, the phase difference is then a fuction of time $p(t)$. The more coherent two waves are, the slower this function will change over time. In fact, two waves are said to be coherent if they are either of the same frequency, temporally in phase or have the same amplitude at every point in time. Thus two waves are coherent if they are generated at the same time, having the same frequency, amplitude, and phase. Reversely, Waves are considered incoherent or also asynchronous if they have no stable phase difference. This means $p(t)$ is heavly varying over time. Coherence describes the effect of whether waves will tend to interfere with each other constructively or destructively at a certain point in time and space. Thus this is a property of waves that enables stationary interference. The more correlated two waves are, the higher their degree of coherence is. In physics coherence between waves is quantified by the corss-correlation function, which basically predicts the value of a second wave using the value of the first one. There are two basic coherence classifications:

- Spatial coherence is dealing with the question of what is the range of distance between two points in space in the extend of a wave for which there is occuring a significant effect of

interference when averaged over time. This is formally answered by considering the correlation between waves at different point in space. The range of distance is also denoted as the coherence area.

- Temporal coherence examines the ability of how well a wave will interfer with itself at different moments in time. Mathematically, this kind of coherence is computed by averaging the measured correlation between the value of the wave and the delayed version of itself at different pairs of time. The Coherence time denotes the time for which the propagating wave is coherent and we therefore can predict its phase using the correlation function. The distance a wave has traveled during the coherence time is denoted as the coherence length.

2.2.4 Huygen's Principle

Besides from a wave's phase and amplitude, also its propagation directly affects the interaction between different waves and how they could interfere with each other. This is why it makes sense to formulate a model which allow us to predict the position of a moving wavefront and how it moves in space. This is where Huygen's Principle comes into play. It states that any each point of a wavefront may be regarded as a point source that emits spherical wavelets in every direction. Within the same propagation medium, these wavelets traval at the same speed as their source wavefront. The position of the new wavefront results by superimposing all of these emitted wavelets. Geometrically, the surface that is tangential to the secondary waves can be used in order to determine the future position of the wavfront. Thesrefore, the new wavefront encloses all emitted wavelets. Figure 2.9 visualizes Huygen's Prinicple for a wavfront reflected off from a plane surface.

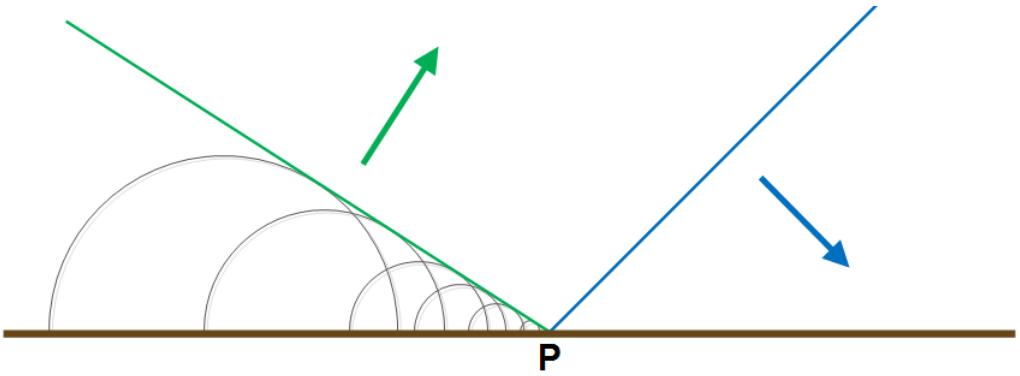


Figure 2.9: A moving wavefront (blue) encounters an obstacle (a surface in brown colors) and produces a new wavefront (green) as a result of superposition among all secondary wavelets.

2.2.5 Waves Diffraction

Revisting Hugen's Principle we know that each point on a wavefront can be considered as a source of a spherical wavelet which propagates in every direction. But what exactly happens when a wave's propagation direction is occluded by an object? What will be the outcome when applying Huygen's Principle for that case? An example scenario for this case is shown in figure 2.10.

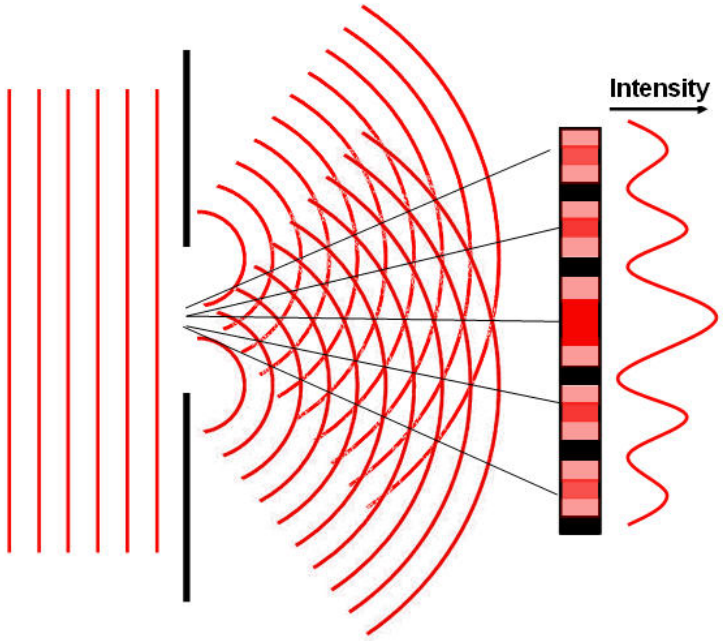
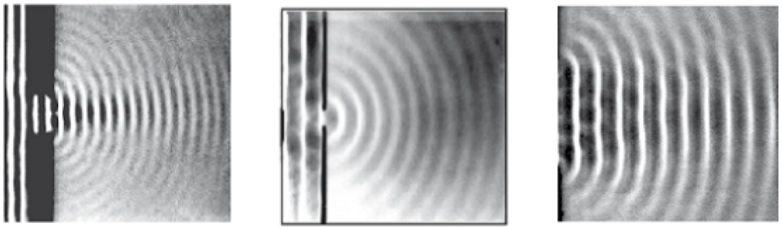


Figure 2.10: Illustation[9] of a diffraction scenario in which a plane wavefront passes through a surface with a certain width and how the wave will be bent, also showing the intensity of the resulting wave.

Whenever a propagating wavefront is partially occluded by an obstacle, the wave is not only moving in the direction along its propagation, but is also bent around the edges of the obstacle. In physics, this phenomenon is called diffraction. Waves are diffracted due to interference which occures among all wavelets when applying Huygen's Principle for the case when a wavfront hits an obstacle. Generally, the effect of diffraction is most expressed for waves whose wavelength is roughly similar in size to the dimension of the occluding object. Conversely, if the wavelength is hardly similar in size, then there is almost no wave diffraction perceivable at all. This relationship between the strength of wave diffraction and wavelength-obstacle-dimensions is conceptually illustrated in figure 2.11 when a wave is transmitted through a surface. A reflective example is provided in figure 2.9.

[9]Image source:`http://cronodon.com/images/Single_slit_diffraction_2b.jpg`



(a) W $\ll \lambda$ (b) W $\approx 2\lambda$ (c) W $\approx 6\lambda$

Figure 2.11: Illustration[10] of how the effect of diffraction changes when a wave with wavelength λ propagates through a slit of width equal W.

In everday's life, we can see the direct outcome of the effect of wave diffraction in form of structural colors. There are examples from nature such as the irridecent colors on various snake skins as well as artificial examples such as the colorful patterns notable when having a close look at an illuminated compact disc. All in common having having a surface made of very regular nanostructure which is diffracting an incident light. Such a nanostructure which exhibits a certain degree of regularity is also denoted as diffraction grating. More about this in section 5.2.

2.3 Stam's BRDF formulation

The theoretical foundation of this thesis is based on the pioneering work of J.Stam who derived in his paper about Diffraction Shader[Sta99] a BRDF which is modeling the effect of far field diffraction for various analytical anisotropic reflexion models, relying on the so called scalar wave theory of diffraction for which a wave is assumed to be a complex valued scalar.

It's noteworthy, that Stam's BRDF formulation does not take into account the polarization of the light. Fortunately, light sources like sunlight and light bulbs are unpolarized. The principal idea behind J. Stam's approach is illustrated in figure 2.12.

[10]Image taken from:`http://neutrino.ethz.ch/Vorlesung/FS2013/index.php/vorlesungsskript`, chapter 9, figure 9.14

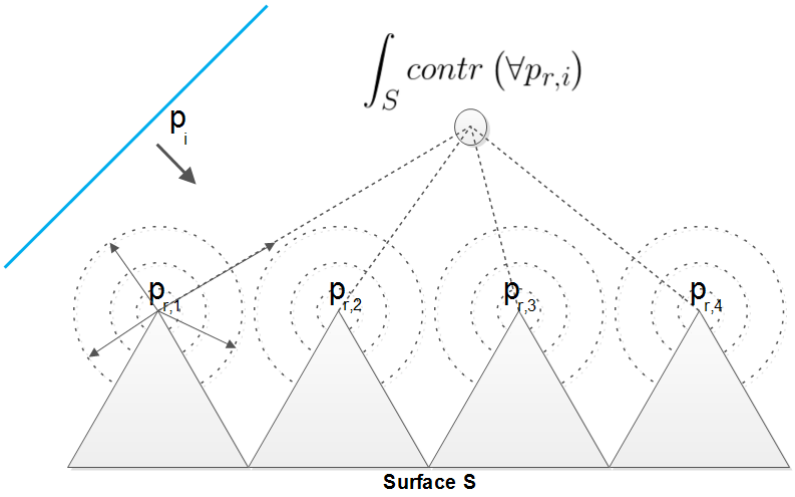


Figure 2.12: Illustration of idea behind Stam's approach: Integartion over all secondary sources according to Huygen's principle resulting from an incident wave will give us an identity for the total contribution at a certain point in space.

An incident wave p_i from a a light source encounters a surface, representing a diffraction grating. According to Huygen's Principle, at any point i on the grating, at which the incident wave meets the grating, a secondary, spherical wavelet $p_{r,i}$ will be emitted. A viewer, in the figure indicated by a gray circle, will perceive the superimposed contributation of all wavelets along the surface S (in the figure indicated by a integration symbol), which wil directly follow the laws of wave interference. Therefore the resulting color which an observer sees is the final radiance at that point which itself is affected by stationary interference of all emitting secondary sources due to Huygen's principle.

A further assumption in Stam's Paper is, that the emanated waves from the source are stationary, which implies the wave is a superposition of independent monochromatic waves. This further implies that each wave is associated to a definite wavelength λ. However, directional light sources, such as sunlight fulfills this fact and since we are using these kinds of light sources for our simulations, Stam's model can be used for our modeling purposes.

The main idea of his model is the formulate a BRDF as the Fourier Transform applied on the given height field, representing a surface like shown in figure $fig : geometricsetup$. The classes of surfaces his model is able to support either exhibit a very regular structure or may be considered as a superposition of bumps forming a periodic like structure. Therefore, the surfaces he is dealing with can either be modeled by probabilistic distributions or have a direct analytical representation. Both cases allow him to derive an analytical solution for his BRDF model.

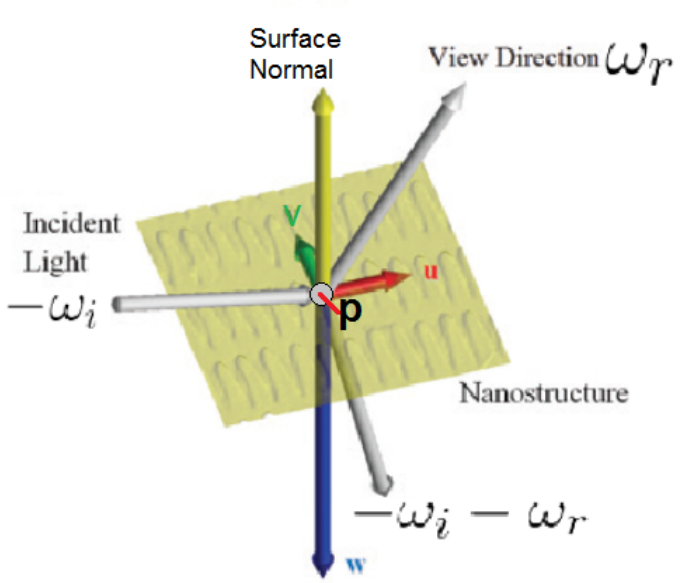


Figure 2.13: Illustration[11] of geometrical setup of Stam's approach where ω_i is a direction, pointing towards the light source, ω_r points towards the camera, n is the surface normal, (u, v, w) are the components of the vector $-\omega_i - \omega_r$.

The direction vector of the secondary wavelet can be computet by taking the difference between the incident and viewing direction like shown in equation 2.5:

$$(u, v, w) = -\omega_i - \omega_r \tag{2.5}$$

These coordinates will later be used in order to compute the total contributation of all secondary sources used in Stam's BRDF in equation 2.8. For simplification, let us introduce an auxillary function Φ defined in equation 2.6, which models the phase of a wave from the provided height field.

$$\Phi(x, y) = \frac{2\pi}{\lambda} wh(x, y) \tag{2.6}$$

Then, any secondary wavelet p which is emitted off from the given surface will be equal:

$$p(x, y) = e^{i\Phi(x,y)} \tag{2.7}$$

using the idea presented for figure 2.12 and performing all mathematical steps shown in the appendix B, will lead us to the final BRDF representation, modeling the total contribution of all secondary sources reflected off the the provided surface h:

$$BRDF_\lambda(\omega_i, \omega_r) = \frac{k^2 F^2 G}{4\pi^2 A w^2} \langle |P(ku, kv)|^2 \rangle \tag{2.8}$$

where F denotes the Fresnel coefficient and G is the so called geometry term[12] which is equal

[11] Modified image which originally has been taken from D.H. Dhillon's poster[DSD14].

[12] The geometric Terms expresses the correction factor to perform an integration over an area instead over a surface. For further information, please have a look at `http://en.wikipedia.org/wiki/Surface_integral`, and read the definition about *surface element*

to:

$$G = \frac{(1 + \omega_i \cdot \omega_r)^2}{cos(\theta_i)cos(\theta_r)} \tag{2.9}$$

One last word about the term Fourier Transform Stam uses in his derivation: Conventionally, following the definitions in Mathematics of the Fourier Tranformation, we are dealing with the inverse Fourier Transformation. However, especially in electrical engeneering, it is quite common to define this inverse Fourier Transformation by the Fourier Transformation. The reason behind this lies in the fact that we simply could substitude the minus sign like the following equation 2.10:

$$\begin{aligned} \mathcal{F}_{FT}\{f\}(w) &= \int_{\mathbb{R}^n} f(x)e^{-iwt}dt \\ &= \int_{\mathbb{R}^n} f(x)e^{i\hat{w}t}dt \\ &= \mathcal{F}_{FT}^{-1}\{f\}(w) \end{aligned} \tag{2.10}$$

where $\hat{w}$ is equal $-w$.

The height fields we are dealing with in this work are, however, natural gratings, containing a complex shaped nano-structure and hence far apart from being very regularly aligned. The reason why Stam's approach in its current form is not suitable for our purpose is twofold: First his approach does not capture the complexity of natural gratings accurately well enoug when relying on his his statistical approaches and secondly it is way too slow in order to be usable for interactive rendering since his BRDF needs an evaluation of a Fourier Transform for every directional changement.

In the follwing a brief comparission between Stam's[13] and our final approach using two different kinds of gratings, a synthetic, regularly aligned grating and a natural, complex structured grating. These gratings are shown in figure $fig : stameggratings$.

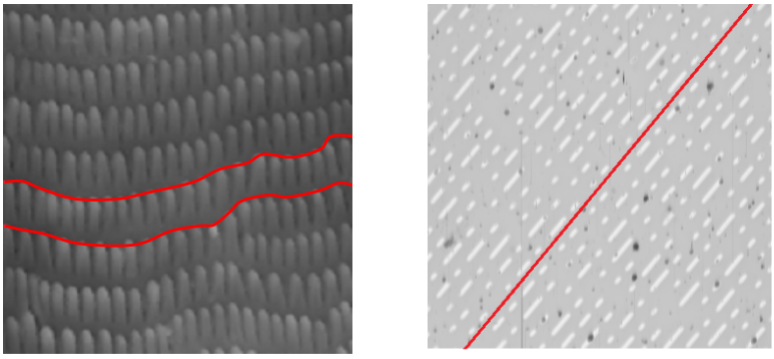


Figure 2.14: Alignment of nanostructures in diffraction gratings. On the left a complex, natural grating of the Elaphe snake species and on the right a synthetic, very regularly aligned grating of a CD.

[13] A reference implementation of Stam's Diffraction Shader[Sta99] is provided by Nvidia's GPU Gems at `http://http.developer.nvidia.com/GPUGems/gpugems_08.html`

Figure 2.15 shows an example of a case where Stam's approach performs well. Considering the red-line in the figure we notice that the nano-scaled structures of a compact disc are very regularly aligned along the surface. Tracks of a CD are uniformally spaced and bumps along a track are distributed according to the poisson distribution[14]. All angles of the diffraction pattern look the same as in the images produced by our apporach.

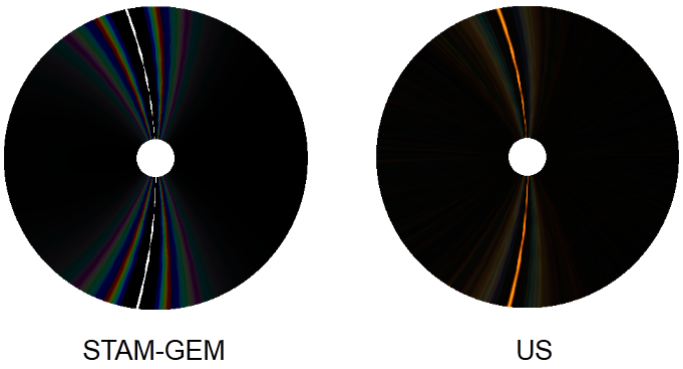


Figure 2.15: Comparission of our approach against a reference implementation of Stam's provided by Nvida Gem. For synthetic diffraction gratings, which have a very regular structure, Stam's apporach is doing well. All angles of the diffraction pattern look the same as in the images produced by our apporach.

Figure 2.16 shows an attempt to reproduce real structural colors on the skin of the Xenopeltis snake using our method and comparing it to Stam's approach. Even the results of Stam might look appropriate, there are some differences notable such missing colors close to speclar regions, or such as the colordistribution which is rather discrete in Stam's approach. Furthermore, Stam's approach has at some places color contribution, where it should not have.

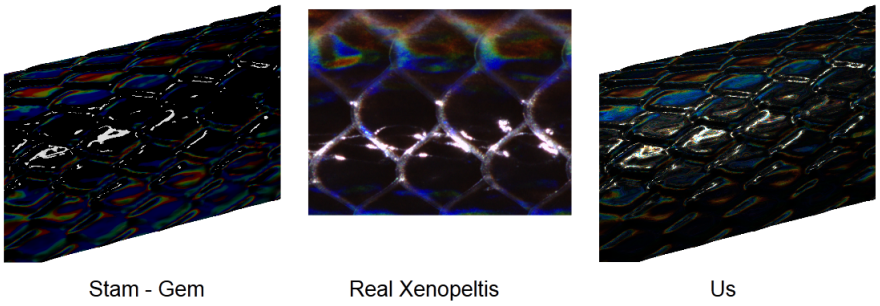


Figure 2.16: Comparission of our approach(on the right) against a reference implementation of Stam's(on the left) provided by Nvida Gem by trying to render a reproduction of a real Elaphe skin (center) under similar lightning and viewing conditions. For natural diffraction gratings, which have a rather complex structure, Stam's apporach is doing rather bad.

In the next chapter we are going to adapt Stam's BRDF model such that it will be able to

[14]See `http://en.wikipedia.org/wiki/Poisson_distribution`

handle the kind of surfaces we are deling with and even will have a runtime complexity which allows to perform interactive rendering.

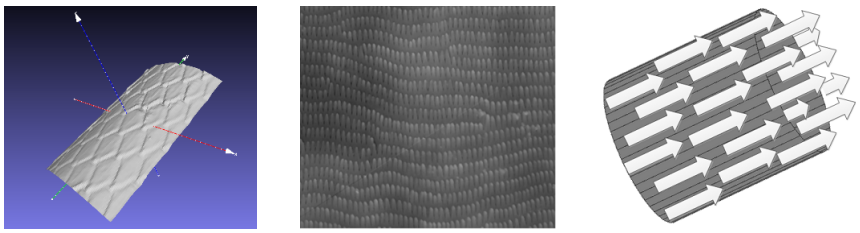
Chapter 3

Derivations

3.1 Problem Statement and Challenges

The goal of this thesis is to perform a physically accurate and interactive simulation of structural colors production like shown in figure 3.2, which we can see whenever a light source is diffracted on a natural grating. For this purpose we need to be provided by the following input data as shown in figure 3.1:

- A mesh representing a snake surface[1] with associated texture coordinates as shown in figure 3.1(a).
- A natural diffraction grating represented as a height field, its maximum height and its pixel-width-correspondence[2].
- A vectorfield which describes how fingers on a provided surface of the nanostructrue are aligned as shown in figure 3.1(c).



(a) Structure Geometry (b) Nanostructure Surface (c) Patch Orientation

Figure 3.1: Input for our simulation

We want to rely on the integral equation 2.8 derived by J. Stam in his paper [Sta99] about diffraction shaders. This equation formualtes a BRDF modeling the effect of diffraction under the assumption that a given grating can either be formulated as an analytical function or its structure

[1] Which is in our simulation an actual reconstruction of a real snake skin. These measurements are provided by the Laboratory of Artificial and Natural Evolition at Geneva.See their website:`www.lanevol.org`.

[2] Since the nanostructure is stored as a grayscale image, we need a scale telling us what length and height one pixel cooresponds to in this provided image.

is simple enough beeing modeled relying on statistical methods. These assumptions guarantee that 2.8 has an explicit solution. However, the complexisty of a biological nanostructures cannot sufficiently and accurately modeled simply using statistical methods. This is why interactive computation at high resolution becomes a hard task, since we cannot evaluate the given integral equation on the fly. Therefore, we have to adapt Stam's equation such that we are able to perform interactive rendering using explicitly provided height fields.

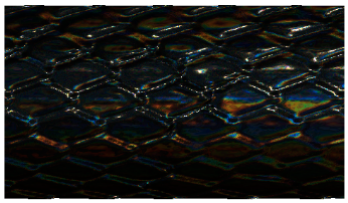


Figure 3.2: Output: Rendered Structural Colors

3.2 Approximate a FT by a DFT

3.2.1 Reproduce FT by DTFT

In the previous section, we have found an identity for the reflected spectral radiance $L_\lambda(\omega_r)$ when using Stam's BRDF for a given input height field. However, the derived expression in equation 3.13 requires to evaluate the Fourier Transform of our height field[3] for every direction. In this section we explain how to approximate the FT by the DTFT and apply it to our previous derivations. Figure 3.3 graphically shows how to obtain the DTFT from the FT for a one dimensional signal[4]

The first step is to uniformly discretize the given signal since computers are working finite, discrete arithmetic. We rely on the Nyquist–Shannon sampling theorem tells us how dense we have to sample a given signal $s(x)$ such that can be reconstructed its sampled version $\hat{s}[n]$[6]. In particular, a sampled version according to the Nyquist–Shannon sampling theorem will have the same Fourier Transform as its origianl singal. The sampling theorem states that if f_{max} denotes the highest frequency of $s(x)$, then, it has to be sampled by a rate of f_s with $2f_{max} \leq f_s$ in order to be reconstructable. By convention $T = \frac{1}{f_s}$ represent the interval length between two samples.

Next, we apply the Fourier Tranformation operator on the discretized singal $\hat{s}$ which gives us the following expression:

[3] actually it requires the computation of the inverse Fourier Transform of a transformed version of the given heightfield, the function p(x,y) defined in equation 2.7.

[4] For our case we are dealing with a two dimensional, spatial signal, the given height field. Nevertheless, without any constraints of generality, the explained approach applies to multi dimensional problems.

[5] Images of function plots taken from `http://en.wikipedia.org/wiki/Discrete_Fourier_transform` and are modified.

[6] n denotes the number of samples.

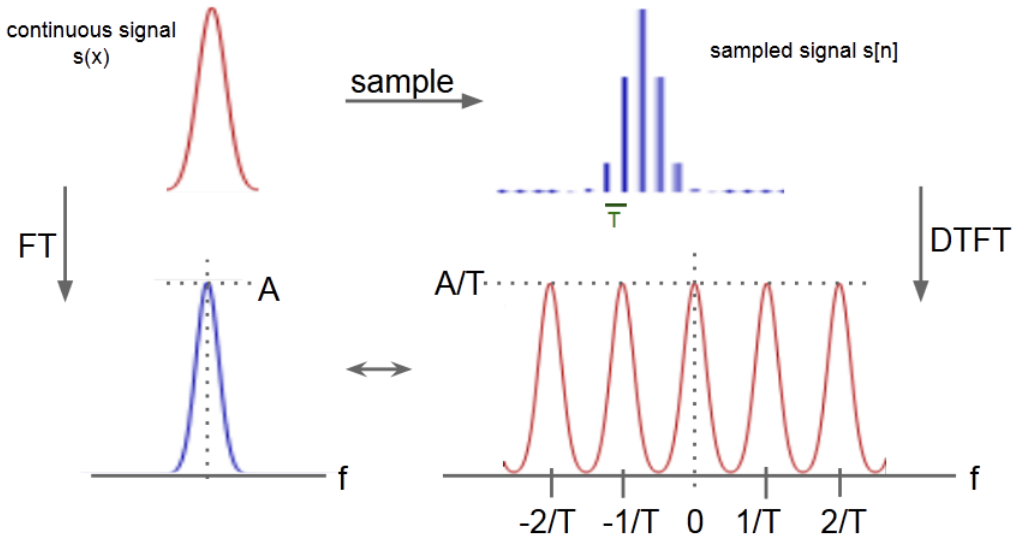


Figure 3.3: Illustration of how to approximate the analytical Fourier Transform (FT) [5] of a given continuous signal by a Discrete Time Fourier Transform (DTFT). The DTFT applied on a bandlimited, discretized signal yields a continuous, periodic response in frequency space.

$$\begin{aligned}\mathcal{F}_{FT}\{\hat{s}\}(w) &= \int_{\mathbb{R}} \hat{s}[n]e^{-iwx}dx \\ &= \int_{\mathbb{R}} mask(x)s(x)e^{-iwx}dx \\ &= T\sum_{x=-\infty}^{\infty} \hat{s}[x]e^{-iwx} \\ &= T\mathcal{F}_{DTFT}\{s\}(w) \end{aligned} \tag{3.1}$$

Equation 3.1 tells us that if $\hat{s}$ is sufficiently sampled, then its DTFT corresponds to the FT of $s(x)$. Notice that the resulting DTFT from the sampled signal has a height of $\frac{A}{T}$ where A is the height of the FT of s and thus is a scaled version of the FT.

For a given height field h, let us compute Stam's auxiliary function p defined as in equation 2.7. For the reminder of this thesis we introduce the following definition:

$$P_{dtft} \equiv \mathcal{F}_{DTFT}\{p\} \tag{3.2}$$

Therefore P_{dtft} denotes the DTFT of a transformed version of our height field h[7].

3.2.2 Spatial Coherence and Windowing

Before we can derive a final expression in order to approximate a FT by a DFT, we first have to revisit the concept of coherence introduced in section 2.2.3 of chapter 2.Previousely we have seen

[7] By transformed height field we mean $p(x,y) = e^{i\frac{2\pi}{\lambda}wh(x,y)}$ which we get, when plugging in h into equation 2.6 and this expression again plug in to equation 2.

that Stam's BRDf tells us what is the total contribution of all secondary sources which allows us to say what is the reflected spectral radiance at a certain point in space. This is related to stationary interference which itself depends on the coherence property of the emitted secondary wave sources. The ability for two points in space, t_1 and t_2, to interfere in the extend of a wave when being averages over time is the so called spatial coherence. The spatial distance between such two points over which there is significant intererence is limited by the quantity coherence area. For filtered sunlight on earth this is equal to $65\mu m$ [8].

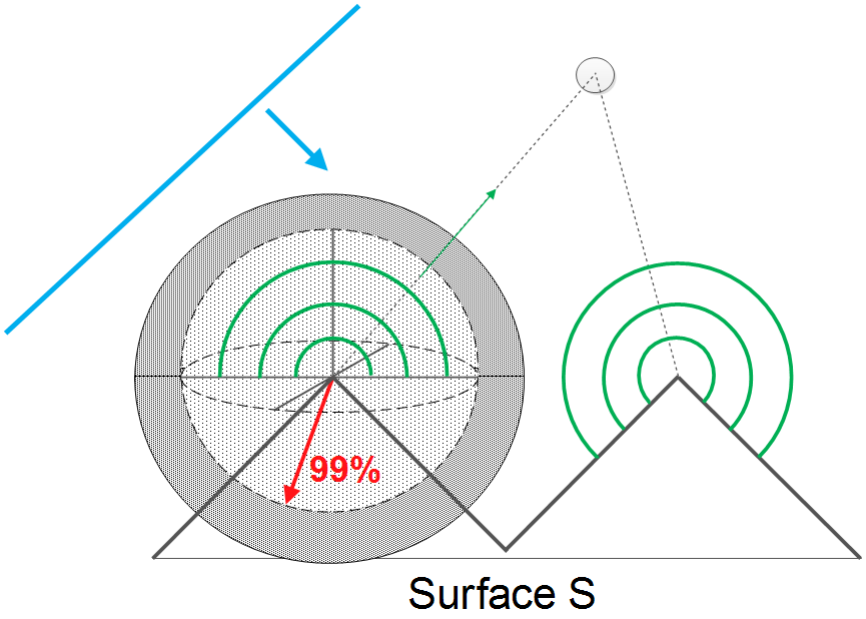


Figure 3.4: A plane wave encoungers a surface. According to Huygens principle, secondary wavelets are emitted of from this surface. The resulting wave at a certain point in space (here indicated by a gray circle) depends on the inteference among all waves encountering at this position. The amount of significant interference is directly affected by the spatial coherence property of all the wavelets.

Figure 3.4 illustrates the concept of spatial coherence. A wavefront (blue line) encounters a surface. Due to Hugen's Principle, secondary wavelets are emitted off from the surface. The reflected radiance at a certain point in space, e.g. at a viewer's eye position (denoted by the gray circle), is a result of interference among all wavelets at that point. This interference is directly affected by the spatial coherence property of all the emitted wavelets.

In physics spatial coherence is predicted by the cross correlation between t_1 and t_2 and usually modeled by by a Gaussian Random Process. For any such Gaussian Processes we can use a spatial gaussian window $g(x)$ which is equal:

$$g(x) = \frac{1}{\sqrt{2\pi} \cdot \sigma} \cdot e^{-\frac{x^2}{2\sigma^2}} \tag{3.3}$$

[8] A proof for this number can be looked up in the book Optical Coherence and Quantum Optics[LM95] on page 153 and 154.

We have chosen standard deviation σ_s of the window such that it fulfills the equation $4\sigma_s = 65\mu m$. This is equivalent like saying we want to predict about 99.99%[9] of the resulting spatial coherence interference effects in our model by a cross correlation function.

By applying the Fourier Transformation to the spatial window we get the corresponding window in frequency space will look like:

$$G(f) = e^{-\frac{f^2}{2\sigma_f^2}} \tag{3.4}$$

Notice that this frequency space window has a standard deviation σ_f equal to $\frac{1}{2\pi\sigma_s}$. Those two windows, the spatial- and the frequency space window, will be used in the next section in order to approximate the DTFT by the DFT by a windowing apporach.

3.2.3 Reproduce DTFT by DFT

In this section we explain how and under what assumptions the DTFT of a discretized signal[10] can be approximated by a DFT. The whole idea how to reproduce the DTFT by DFT is schematically illustrated in figure 3.5.

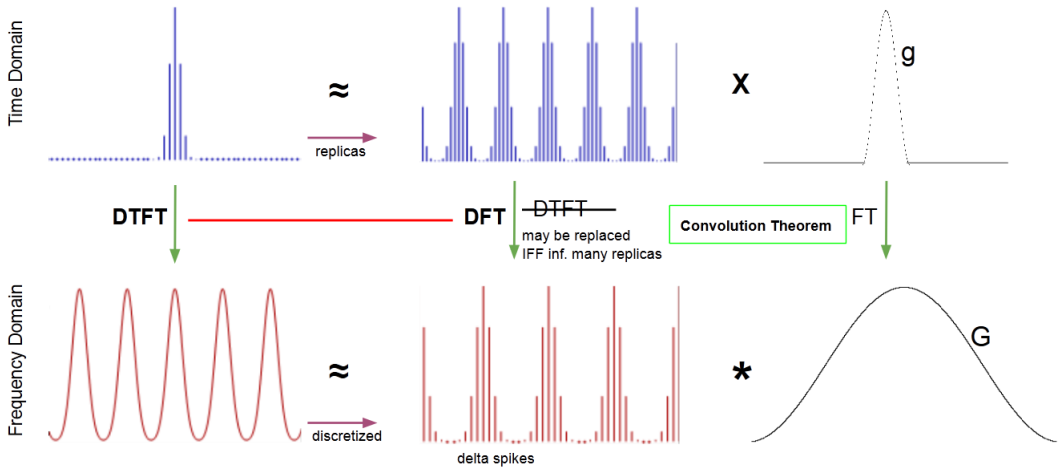


Figure 3.5: Illustration of how to approximate the DTFT [11] by the DFT relying on the Convolution Theorem, using a gaussian window function.

Given a spatial, bandlimited and discretized one dimensional signal $\hat{s}$. Our goal is to approximate this spatial signal in a way such that when taking the DTFT of this approximated signal, it will yield almost the same like taking the DTFT of the original sampled $\hat{s}$. For this purpose we will use the previouse introduced concept of gaussian windows and the so called Convolution Theorem which is a fundamental property of all Fourier Transformations.

[9] Standard deviation values from confidence intervals table of normal distribution provided by Wolfram MatheWorld `http://mathworld.wolfram.com/StandardDeviation.html`.

[10] E.g. a sampled signal like already presented in figure 3.3

[11] Images of function plots taken from `http://en.wikipedia.org/wiki/Discrete_Fourier_transform` and are modified. Note that the scales in the graphic are not appropriate.

The Convolution Theorem states that the Fourier Transformation of a product of two functions, f and g, is equal to convolving the Fourier Transformations of each individual function. Mathematically, this statement corresponds to equation 3.5:

$$\mathcal{F}\{f \cdot g\} = \mathcal{F}\{f\} * \mathcal{F}\{g\} \tag{3.5}$$

The principal issue is how to approximate our given signal $\hat{s}$. Therefore, let us consider another signal $\hat{s_N}$ which is the N times replicated version of $\hat{s}$ (blue signal at center top in figure).

Remeber that in general, the signal repsonse at a certain point in space is the result of interference among all signals meeting at that position. In our scenario, the source of those signals are emitted secondary wavelets. The interference strength between these points is related to their spatial coherence. Windowing the signals by a gaussian window g will capture a certain percentage of all interference effects. From the previous section 3.2.2 we know that we can use gaussian window like in equation 3.3 in order to approximate such spatial signals interference effects.

Using this insight, we can approximate $\hat{s}$ by taking the product of $\hat{s_N}$ with a gaussian window g. This fact is illustrated in the first row of figure 3.3. So what will the DTFT of this approximation yield? We already know that the DTFT of $\hat{s}$ is a continuous, periodic signal, since $\hat{s}$ is bandlimited. Thus, taking the DTFT of this found approximation should give us approximatively the same continuous, periodic signal.

This is where the convolution theorem comes into play: Applying the DTFT to the product of $\hat{s_N}$ and g is the same as convolving the DTFT of $\hat{s_N}$ by DTFT of g. From equation 3.4 we already know that the DTFT of g is just another gaussian, denoted by G. On the other hand the DTFT of $\hat{s_N}$ yields a continuous, periodic signal. The higher the value of N, the sharper the signal gets (denoted by delta spiked) and the closer it converges toward to the DFT. This is why the DFT is the limit of a DTFT applied on periodic and discrete signals. Therefore, for a large number of N we can replace the DTFT by the DFT operator when applied on $\hat{s_N}$.

Lastly, we see that the DTFT of $\hat{s}$ is approximitely the same like convolving a gaussian window by the DFT of $\hat{s_N}$. This also makes sense, since convolving a discrete, periodic signal (DFT of $\hat{s_N}$) by a continuous window function G yiels a continuous, periodic function.

In practise, we cannot compute the DTFT $A.3$ numerically due to finite computer arithmetic and hence working with the DFT is our only option. Furthermore, there are numerically fast algorithmes in order to compute the DFT values of a function, the Fast Fourier Transformation (FFT). The DFT $A.4$ of a discrete height field is equal to the DTFT of an infinitely periodic function consisting of replicas of the same height field. Not, let a spatial gaussian window g having a standard deviation for which $4\sigma_s$ is equal μm. Then, from before, it followis:

$$\mathcal{F}_{dtft}\{\mathbf{s}\} \equiv \mathcal{F}_{dft}\{\mathbf{s}\} * G(\sigma_f) \tag{3.6}$$

Therefore we can deduce the following expression from this:

$$\begin{aligned}
\mathcal{F}_{dtft}\{\mathbf{t}\}(u,v) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F_{dft}\{\mathbf{t}\}(w_u,w_v)\phi(u-w_u,v-w_v)dw_u dw_v \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \sum_i\sum_j F_{dft}\{\mathbf{t}\}(w_u,w_v) \\
&\quad \delta(w_u-w_i,w_v-w_j)\phi(u-w_u,v-w_v)dw_u dw_v \\
&= \sum_i\sum_j \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F_{dft}\{\mathbf{t}\}(w_u,w_v) \\
&\quad \delta(w_u-w_i,w_v-w_j)\phi(u-w_u,v-w_v)dw_u dw_v \\
&= \sum_i\sum_j F_{dft}\{\mathbf{t}\}(w_u,w_v)\phi(u-w_u,v-w_v) \qquad (3.7)
\end{aligned}$$

where

$$\phi(x,y) = \pi e^{-\frac{x^2+y^2}{2\sigma_f^2}} \qquad (3.8)$$

3.3 Adaption of Stam's BRDF discrete height fields

3.3.1 Rendering Equation

As already discussed in the theoretical background chapter, colors are associated to radiance. Since we are starting with Stam's BRDF[12] formulation but want to perform a simulation rendering structural colors, we have to reformulate this BRDF equation such that we will end up with an identity of the reflected spectral radiance. This is where the rendering equation comes into play. Lets assume we have given an incoming light source with solid angle ω_i and θ_i is its angle of incidence, ω_r is the solid angle for the reflected light. Further let λ denote the wavelength[13] and Ω is the hemisphere of integration for the incoming light. Then, we are able to formulate a $BRDF_\lambda$ by using its definition 2.3:

$$\begin{aligned}
& f_r(\omega_i,\omega_r) && = \frac{dL_r(\omega_r)}{L_i(\omega_i)cos(\theta_i)d\omega_i} \\
\Rightarrow\ & f_r(\omega_i,\omega_r)L_i(\omega_i)cos(\theta_i)d\omega_i && = dL_r(\omega_r) \\
\Rightarrow\ & \int_\Omega f_r(\omega_i,\omega_r)L_i(\omega_i)cos(\theta_i)d\omega_i && = \int_\Omega dL_r(\omega_r) \\
\Rightarrow\ & \int_\Omega f_r(\omega_i,\omega_r)L_i(\omega_i)cos(\theta_i)d\omega_i && = L_r(\omega_r) \qquad (3.9)
\end{aligned}$$

The last equation is the so called rendering equation . We assume that our incident light is a directional, unpolarized light source like sunlight and therefore its radiance is given as

$$L_\lambda(\omega) = I(\lambda)\delta(\omega-\omega_i) \qquad (3.10)$$

[12]Remember that a BRDF is the portion of a incident light source reflected off a given surface towards a specified viewing direction.

[13]Notice that, to keep our terms simple, we have droped all λ subscripts for spectral radiance quantites.

where $I(\lambda)$ is the intensity of the relative spectral power for the wavelength λ. By plugging the identity in equation 3.10 into our current rendering equation 3.9, we will get:

$$\begin{aligned} L_\lambda(w_r) &= \int_\Omega BRDF_\lambda(\omega_i, \omega_r) L_\lambda(\omega_i) cos(\theta_i) d\omega_i \\ &= BRDF_\lambda(\omega_i, \omega_r) I(\lambda) cos(\theta_i) \end{aligned} \tag{3.11}$$

where $L_\lambda(\omega_i)$ is the incident radiance and $L_\lambda(\omega_r)$ is the radiance reflected by the given surface. Note that the integral in equation 3.11 vanishes since $\delta(\omega - \omega_i)$ is only equal one if and only if $\omega = \omega_i$.

3.3.2 Reflected Radiance of Stam's BRDF

We are going to use Stam's main derivation (2.8) for the $BRDF(\omega_i, \omega_r)$ in 3.11 by applying the fact that the wavenumber is equal $k = \frac{2\pi}{\lambda}$:

$$\begin{aligned} BRDF(\omega_i, \omega_r) &= \frac{k^2 F^2 G}{4\pi^2 A w^2} \langle |P(ku, kv)|^2 \rangle \\ &= \frac{4\pi^2 F^2 G}{4\pi^2 A \lambda^2 w^2} \langle |P(ku, kv)|^2 \rangle \\ &= \frac{F^2 G}{A \lambda^2 w^2} \langle \left| P(\frac{2\pi u}{\lambda}, \frac{2\pi v}{\lambda}) \right|^2 \rangle \end{aligned} \tag{3.12}$$

Going back to equation 3.11 and plugging equation 3.12 into it, using the definition of equation 2.9 and the equation $D.2$ for ω we will get the following:

$$\begin{aligned} L_\lambda(\omega_r) &= \frac{F^2(1 + \omega_i \cdot \omega_r)^2}{A \lambda^2 cos(\theta_i) cos(\theta_r) \omega^2} \left\langle \left| P\left(\frac{2\pi u}{\lambda}, \frac{2\pi v}{\lambda}\right) \right|^2 \right\rangle cos(\theta_i) I(\lambda) \\ &= I(\lambda) \frac{F^2(1 + \omega_i \cdot \omega_r)^2}{\lambda^2 A \omega^2 cos(\theta_r)} \left\langle \left| P\left(\frac{2\pi u}{\lambda}, \frac{2\pi v}{\lambda}\right) \right|^2 \right\rangle \end{aligned} \tag{3.13}$$

Note that the Fresnel term F is actually a function of (w_i, w_r), but in order to keep the equations simple, we omitted its arguemts. So far we just plugged Stam's BRDF identity into the rendering equation and hence have not significantly deviated from his formulation. Keep in mind that P deontes the Fourier transform of the provided height field which depends on the viewing and incidence light direction. Thus this Fourier Transform has to be recomputed for every direction which will slow down the whole computation quite a lot[14]. One particular strategy to solve this issue is to approximate P by the Discrete Fourier Transform (DFT)[15] and seperate its computation such that terms for many directions can be precomputed and then later retrieved by look ups. The approximation of P happens in two steps: First we approximate the Fourier Transform by the Discrete Time Fourier Transform (DTFT) and then, afterwards, we approximate the DTFT by the DFT. For further about basics of signal processing and Fourier Transformations please consult the appendix A.

[14] Even a fast variant of computation the Fourier Transform has a runtime complexitiy of O(N log N) where N is the number of sample.

[15] See appendix A for further information about different kinds of fourier transformations.

Using the insight gained by equation 3.1 allows us to further simplify equation 3.13:

$$\begin{aligned} L_\lambda(\omega_r) &= I(\lambda)\frac{F^2(1+\omega_i\cdot\omega_r)^2}{\lambda^2 A w^2 cos(\theta_r)}\left\langle\left|P\left(\frac{2\pi u}{\lambda},\frac{2\pi v}{\lambda}\right)\right|^2\right\rangle \\ &= I(\lambda)\frac{F^2(1+\omega_i\cdot\omega_r)^2}{\lambda^2 A w^2 cos(\theta_r)}\left\langle\left|T^2 P_{dtft}\left(\frac{2\pi u}{\lambda},\frac{2\pi v}{\lambda}\right)\right|^2\right\rangle \end{aligned} \tag{3.14}$$

Where P_{dtft} is a substitude for $\mathcal{F}_{DTFT}\{s\}(w)$. Furthermore T the sampling distance for the discretization of $p(x,y)$ assuming equal and uniform sampling in both dimensions x and y.

3.3.3 Relative Reflectance

In this section we are going to explain how to scale our BRDF formulation such that all of its possible output values are mapped into the range $[0,1]$. Such a relative reflectance formulation will ease our life for later rendering purposes since usually color values are within the range $[0,1]$, too. Furthermore, this will allow us to properly blend the resulting illumination caused by diffraction with a texture map.

Let us examine what $L_\lambda(\omega_r)$ will be for a purely specular surface, for which $\omega_r = \omega_0 = \omega_i$ such that $\omega_0 = (0,0,1)$. For this specular reflection case, the correspinding radiance will be denoted as $L_\lambda^{spec}(\omega_0)$. When we know the expression for $L_\lambda^{spec}(\omega_0)$ we would be able to compute the relative reflected radiance for our problem 3.13 by simply taking the fraction between $L_\lambda(\omega_r)$ and $L_\lambda^{spec}(\omega_0)$ which is denoted by:

$$\rho_\lambda(\omega_i,\omega_r) = \frac{L_\lambda(\omega_r)}{L_\lambda^{spec}(\omega_0)} \tag{3.15}$$

Notice that the third component w from the vector in equation 2.5 is squared eqaul $(cos(\theta_i) + cos(\theta_r))^2$[16]. But first, let us derive the following expression:

$$\begin{aligned} L_\lambda^{spec}(\omega_0) &= I(\lambda)\frac{F(\omega_0,\omega_0)^2(1+\begin{pmatrix}0\\0\\1\end{pmatrix}\cdot\begin{pmatrix}0\\0\\1\end{pmatrix})^2}{\lambda^2 A(cos(0)+cos(0))^2 cos(0)}\langle\left|T_0^2 P_{dtft}(0,0)\right|^2\rangle \\ &= I(\lambda)\frac{F(\omega_0,\omega_0)^2(1+1)^2}{\lambda^2 A(1+1)^2 1}\left|T_0^2 N_{sample}\right|^2 \\ &= I(\lambda)\frac{F(\omega_0,\omega_0)^2}{\lambda^2 A}\left|T_0^2 N_{sample}\right|^2 \end{aligned} \tag{3.16}$$

Where $N_{samples}$ is the number of samples of the DTFT $A.3$. Thus, we can plug our last derived expression 3.16 into the definition for the relative reflectance radiance 3.15 in the direction w_r and will get:

[16] Consult section $D.2$ in the appendix

$$\begin{aligned}\rho_\lambda(\omega_i,\omega_r) &= \frac{L_\lambda(\omega_r)}{L_\lambda^{spec}(\omega_0)} \\ &= \frac{I(\lambda)\frac{F(\omega_i,\omega_r)^2(1+\omega_i\cdot\omega_r)^2}{\lambda^2 A(cos(\theta_i)+cos(\theta_r))^2 cos(\theta_r)}\langle\left|T_0^2 P_{dtft}(\frac{2\pi u}{\lambda},\frac{2\pi v}{\lambda})\right|^2\rangle}{I(\lambda)\frac{F(\omega_0,\omega_0)^2}{\lambda^2 A}\left|T_0^2 N_{sample}\right|^2} \\ &= \frac{F^2(\omega_i,\omega_r)(1+\omega_i\cdot\omega_r)^2}{F^2(\omega_0,\omega_0)(cos(\theta_i)+cos(\theta_r))^2 cos(\theta_r)}\langle\left|\frac{P_{dtft}(\frac{2\pi u}{\lambda},\frac{2\pi v}{\lambda})}{N_{samples}}\right|^2\rangle \end{aligned} \tag{3.17}$$

For simplification and better readability, let us introduce the following expression, the so called gain-factor:

$$C(\omega_i,\omega_r) = \frac{F^2(\omega_i,\omega_r)(1+\omega_i\cdot\omega_r)^2}{F^2(\omega_0,\omega_0)(cos(\theta_i)+cos(\theta_r))^2 cos(\theta_r) N_{samples}^2} \tag{3.18}$$

Using equation 3.18, we will get the following expression for the relative reflectance radiance from equation 3.17:

$$\rho_\lambda(\omega_i,\omega_r) = C(\omega_i,\omega_r)\langle\left|P_{dtft}(\frac{2\pi u}{\lambda},\frac{2\pi v}{\lambda})\right|^2\rangle \tag{3.19}$$

Using the previous definition for the relative reflectance radiance equation 3.15:

$$\rho_\lambda(\omega_i,\omega_r) = \frac{L_\lambda(\omega_r)}{L_\lambda^{spec}(\omega_0)} \tag{3.20}$$

Which we can rearrange to the expression:

$$L_\lambda(\omega_r) = \rho_\lambda(\omega_i,\omega_r) L_\lambda^{spec}(\omega_0) \tag{3.21}$$

Let us choose $L_\lambda^{spec}(w_0) = S(\lambda)$ such that is has the same profile as the relative spectral power distribution of CIE Standard Illuminant $D65$ discussed in 4.4.3. Furthermore, when integrating over λ for a specular surface, we should get CIE_{XYZ} values corresponding to the white point for $D65$. The corresponding tristimulus values using CIE colormatching functions 2.4 for the CIE_{XYZ} values look like:

$$\begin{aligned} X &= \int_\lambda L_\lambda(\omega_r)\overline{x}(\lambda)d\lambda \\ Y &= \int_\lambda L_\lambda(\omega_r)\overline{y}(\lambda)d\lambda \\ Z &= \int_\lambda L_\lambda(\omega_r)\overline{z}(\lambda)d\lambda \end{aligned} \tag{3.22}$$

where $\overline{x}$, $\overline{y}$, $\overline{z}$ are the color matching functions. Combining our last finding from equation 3.21 for $L_\lambda(\omega_r)$ with the definition of the tristimulus values from equation 3.22, allows us to derive a formula for computing the colors values using Stam's BRDF formula relying on the rendering equation 3.9. Without any loss of generality it suffices to derive an explicit expression for just one tristimulus term, for example Y, the luminance:

$$
\begin{aligned}
Y &= \int_\lambda L_\lambda(\omega_r)\overline{y}(\lambda)d\lambda \\
&= \int_\lambda \rho_\lambda(\omega_i,\omega_r)L_\lambda^{spec}(\omega_0)\overline{y}(\lambda)d\lambda \\
&= \int_\lambda \rho_\lambda(\omega_i,\omega_r)S(\lambda)\overline{y}(\lambda)d\lambda \\
&= \int_\lambda C(\omega_i,\omega_r)\langle\left|P_{dtft}(\frac{2\pi u}{\lambda},\frac{2\pi v}{\lambda})\right|^2\rangle S(\lambda)\overline{y}(\lambda)d\lambda \\
&= C(\omega_i,\omega_r)\int_\lambda \langle\left|P_{dtft}(\frac{2\pi u}{\lambda},\frac{2\pi v}{\lambda})\right|^2\rangle S(\lambda)\overline{y}(\lambda)d\lambda \\
&= C(\omega_i,\omega_r)\int_\lambda \langle\left|P_{dtft}(\frac{2\pi u}{\lambda},\frac{2\pi v}{\lambda})\right|^2\rangle S_y(\lambda)d\lambda
\end{aligned} \tag{3.23}
$$

Where we used the definition $S_y(\lambda)\overline{y}(\lambda)$ in the last step.

3.4 Optimization using Taylor Series

Our final goal is to render structural colors resulting by the effect of wave diffraction. So far, we have derived an expression which can be used for rendering. Neverthesless, our current equation 3.23 used for computing structural colors, cannot directly be used for interactive rendering, since P_{dtft} had do be recomputed for every change in any direction[17].

In this section, we will address this issue and deliver an approximation for P_{dtft} defined in equation 3.2. This approximation will allow us to seperate P_{dtft} in a certain way such that some computional expsensive terms can be precomputed. The main idea is to formulate P_{dtft} as a series expansion relying on the definition of Taylor Series, like defined in equation $A.8$. Further, we will provide an error bound for our approximation approach for a given number of terms. Last, we will plug our finding extend our current BRDF formula from equation 3.23 by the findings derived within this section.

Let us consider $p(x,y) = e^{ikwh(x,y)}$ form Stam's Paper 2.3 where $h(x,y)$ is a given height field and $k = \frac{2\pi}{\lambda}$ denotes the wavenumber of wavelenght λ. For any complex number t the power series expansion of the exponential function is equal to:

$$
e^t = 1 + t + \frac{t^2}{2!} + \frac{t^3}{3!} + ... = \sum_{n=0}^{\infty}\frac{t^n}{n!} \tag{3.24}
$$

Now, when we use the exponent[18] of $p(x,y)$ as an input argument for equation 3.24 we get:

[17] According to changes in viewing- or incident light direction.
[18] This exponent is a complex valued function, equal to ikwh(x,y).

$$\begin{aligned} e^t &= e^{ikwh} \\ &= 1 + (ikwh) + \frac{1}{2!}(ikwh)^2 + \frac{1}{3!}(ikwh)^3 + ... \\ &= \sum_{n=0}^{\infty} \frac{(ikwh)^n}{n!}. \end{aligned} \tag{3.25}$$

where i is the imaginary unit for complex numbers. For simplification, in the reminder of this section we omitted the arguments of h. Equation 3.25 gives us an expression for an exponential series expansion for an for the exponent of $p(x,y)$. Please note that above's taylor series is convergent for any complex valued number. Therefore the equation 3.25 is equal to

$$p(x,y) = \sum_{n=0}^{\infty} \frac{(ikwh(x,y))^n}{n!} \tag{3.26}$$

and thus gives us a series representation of $p(x,y)$. Next, calculating the Fourier Transformation $\mathcal{F}$ of equation 3.26 gives us the identity:

$$\begin{aligned} \mathcal{F}\{p\} &\equiv \mathcal{F}\left\{ \sum_{n=0}^{\infty} \frac{(ikwh)^n}{n!}. \right\} \\ &\equiv \sum_{n=0}^{\infty} \mathcal{F}\left\{ \frac{(ikwh)^n}{n!} \right\} \\ &\equiv \sum_{n=0}^{\infty} \frac{(ikw)^n}{n!} \mathcal{F}\{h^n\} \end{aligned} \tag{3.27}$$

Where we have exploited the fact that the Fourier Transformation is a linear operator. Therefore, in equation 3.27, we have shown that the Fourier Transformation of a series is equal to the sum of the Fourier Transformation, applied on each individual series term. Reusing the identifier P[19] in order to determin the Fourier Transformation of Stams definition p from equation 2.7, equation 3.27 then correspond to:

$$P(\alpha,\beta) = \sum_{n=0}^{\infty} \frac{(ikw)^n}{n!} \mathcal{F}\{h^n\}(\alpha,\beta) \tag{3.28}$$

Up to now we have found a infinity series representation for P_{dtft}. Next we are going to look for an upper bound $N \in \mathbb{N}$ such that

$$\tilde{P}_N(\alpha,\beta) := \sum_{n=0}^{N} \frac{(ikwh)^n}{n!} \mathcal{F}\{h^n\}(\alpha,\beta) \approx P(\alpha,\beta) \tag{3.29}$$

$\tilde{P}_N$ is a good approximation of P, i.e. their absolute difference is small[20]. But first, the following two facts would have to be proven[21]:

1. Show that there exist such an $N \in \mathbb{N}$ s.t. the approximation of equation 3.29 holds true.

[19] This identifier P may be subscripted by *dtft* which will denote the DTFT variant of P.
[20] Mathematically speaking, this statement correspond to $||\tilde{P}_N - P|| \leq \epsilon$, where $\epsilon > 0$ is a small number.
[21] Please have a look in section C.1 in the appendix

2. Find a value for N s.t. this approximation is below a certain error bound, e.g. close to machine precision ϵ.

Assuming these facts are proven and there actually exists such an N, we can make use of the taylor series approximation from equation 3.29 and use it for approximating P_{dtft}. This idea allows us to adapt equation 3.23, which is used for computing the structural colors of our BRDF model, in nummerically fast way. E.g. the equation for the luminance is then be equal:

$$\begin{aligned} Y &= C(w_i, w_r) \int_\lambda \langle \left| P_{dtft}(\frac{2\pi u}{\lambda}, \frac{2\pi v}{\lambda}) \right|^2 \rangle S_y(\lambda) d\lambda \\ &= C(w_i, w_r) \int_\lambda \left| \sum_{n=0}^{N} \frac{(wk)^n}{n!} \mathcal{F}\{i^n h^n\}(\frac{2\pi u}{\lambda}, \frac{2\pi v}{\lambda}) \right|^2 S_y(\lambda) d\lambda \end{aligned} \tag{3.30}$$

Notice that equation 3.30 is contrainted by N and hence is an approximation of equation 3.23. Furthermore, it is possible to seperate out all the Fourier Terms in the summation and precompute them. This is why the approach in equation 3.30 is fast in order to compute structural color values according to our BRDF model.

3.5 Spectral Rendering

As the last step of our series of derivations, we plug all our findings together to one big equation. in order to compute the color for each pixel on our mesh in the CIE_{XYZ} colorspace. For any given heigh-field $h(x, y)$ representing a grating of a nano structure and for given direction vectors w_i and w_r as shown in figure 2.13 the resulting color caused by the effect of diffraction can be computed the following way:

$$DFT_n\{h\}(u, v) = F_{dft}\{i^n h^n\}(\frac{2\pi u}{\lambda}, \frac{2\pi v}{\lambda}) \tag{3.31}$$

$$W_n(u, v) = \sum_{(r,s) \in \mathcal{N}_1(u,v)} |DFT_n\{h\}(u - w_r, v - w_s)|^2 \phi(u - w_r, v - w_s) \tag{3.32}$$

where $\phi(x, y) = \pi e^{-\frac{x^2+y^2}{2\sigma_f^2}}$ is the Gaussian window from equation 3.2.3 and $\mathcal{N}_1(u, v)$ denotes the one-neighborhood around (u, v). Then $\forall(u, v, w)$ like defined in equation 2.5, our final expression for computing structural colors due to diffraction, using all our previous derivations, will be equal:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = C(\omega_i, \omega_r) \int_\Lambda \sum_{n=0}^{N} \frac{(2\pi w)^n}{\lambda^n n!} W_n(u, v) \begin{pmatrix} S_x(\lambda) \\ S_y(\lambda) \\ S_z(\lambda) \end{pmatrix} d\lambda \tag{3.33}$$

3.6 Alternative Approach

3.6.1 PQ factors

In this section we are presenting an alternative approach to the previous Gaussian window approach 3.2.3 in order to solve the issue working with $DTFT$ instead the DFT. We assume, that a given

surface S is covered by a number of replicas of a provided representative surface patch f. In a simplified, one dimensional scenario, mathematically speaking, f is assumed to be a repetative function, i.e. $\forall x \in \mathbb{R} : S(x) = S(x + nT)$, where T is its period and $n \in \mathbb{N}_0$. Thus, the surfaces can be written formally as:

$$S(x) = \sum_{n=0}^{N} f(x + nT) \tag{3.34}$$

What we are looking for is an identity for the Fourier Transform[22] of our surface S, required in order to simplify the (X, Y, Z) colors from 3.30:

$$\begin{aligned} \mathcal{F}\{S\}(w) &= \int f(x)e^{iwx}dx \\ &= \int_{-\infty}^{\infty} \sum_{n=0}^{N} f(x + nT)e^{iwx}dx \\ &= \sum_{n=0}^{N} \int_{-\infty}^{\infty} f(x + nT)e^{iwx}dx \end{aligned} \tag{3.35}$$

Next, apply the following substitution $x + nT = y$ which will lead us to:

$$\begin{aligned} x &= y - nT \\ dx &= dy \end{aligned} \tag{3.36}$$

Plugging this substitution back into equation 3.35 we will get:

$$\begin{aligned} \mathcal{F}\{S\}(w) &= \sum_{n=0}^{N} \int_{-\infty}^{\infty} f(x + nT)e^{iwx}dx \\ &= \sum_{n=0}^{N} \int_{-\infty}^{\infty} f(y)e^{iw(y-nT)}dy \\ &= \sum_{n=0}^{N} e^{-iwnT} \int_{-\infty}^{\infty} f(y)e^{iwy}dy \\ &= \sum_{n=0}^{N} e^{-iwnT} \mathcal{F}\{f\}(w) \\ &= \mathcal{F}\{f\}(w) \sum_{n=0}^{N} e^{-iwnT} \end{aligned} \tag{3.37}$$

We used the fact that the exponential term e^{-iwnT} is a constant factor when integrating along dy and the identity for the Fourier Transform of the function f. Next, let us examine the series $\sum_{n=0}^{N} e^{-iwnT}$ closer:

[22] Remember that we are using the definiton of Fourier Transform used in electrical engineering where $\mathcal{F}$ actually corresponds to the inverse Fourier Transform.

$$\begin{aligned}\sum_{n=0}^{N} e^{-uwnT} &= \sum_{n=0}^{N} (e^{-iwT})^n \\ &= \frac{1 - e^{iwT(N+1)}}{1 - e^{-iwT}}\end{aligned} \tag{3.38}$$

We recognize the geometric series identity for the left-hand-side of equation 3.38. Mainly relying on trigonometric identities, equation 3.37 can further simplified to:

$$\mathcal{F}\{S\}(w) = (p + iq)\mathcal{F}\{f\}(w) \tag{3.39}$$

where p and q are defined like:

$$\begin{aligned}p &= \frac{1}{2} + \frac{1}{2}\left(\frac{cos(wTN) - cos(wT(N+1))}{1 - cos(wT)}\right) \\ q &= \frac{sin(wT(N+1)) - sin(wTN) - sin(wT)}{2(1 - cos(wT))}\end{aligned} \tag{3.40}$$

Please notice, all derivation steps can be found in the appendix in section C.2.1.

Now lets consider our actual problem description. Given a patch of a nanoscaled sureface snake shed represented as a two dimensional heightfield $h(x, y)$. We once again assume that this provided patch is representing the whole surface S of our geometry by some number of replicas of itself. Therefore, $S(x, y) = \sum_{n=0}^{N} h(x + nT_1, y + mT_2)$, assuming the given height field has the dimensions T_1 by T_2. In order to derive an identity for the two dimensional Fourier transformation of S we can similarly proceed like we did to derive equation 3.39.

$$\mathcal{F}\{S\}(w_1, w_2) = (p + iq)\mathcal{F}_{DTFT}\{h\}(w_1, w_2) \tag{3.41}$$

Where all derivation steps can be found in the appendix in section C.2.2 and we have defined

$$\begin{aligned}p &:= (p_1 p_2 - q_1 q_2) \\ q &:= (p_1 p_2 + q_1 q_2)\end{aligned} \tag{3.42}$$

For the identity of equation 3.41 we made use of Green's integration rule which allowed us to split the double integral to the product of two single integrations. Also, we used the definition of the 2-dimensional inverse Fourier transform of the height field function. We applied a similar substitution like we did in 3.36, but this time twice, once for x_1 and once for x_2 separately. The last step in equation 3.41, substituting with p and q in equation C.18 will be useful later in the implementation. The insight should be, that the product of two complex numbers is again a complex number. We will have to compute the absolute value of $\mathcal{F}\{S\}(w_1, w_2)$ which will then be equal $(p^2 + q^2)^{\frac{1}{2}} \, |\mathcal{F}\{h\}(w_1, w_2)|$

3.6.2 Interpolation

In 3.6.1 we have derived an alternative approach when we are working with a periodic signal instead using the gaussian window approach from 3.2.3. Its main finding 3.41 that we can just integrate over one of its period instead iterating over the whole domain. Nevertheless, this main finding is using the inverse DTFT. Since we are using

We are interested in recovering an original analog signal $x(t)$ from its samples $x[t] =$

Therfore, for a given sequence of real numbers $x[n]$, representing a digital signal, its correspond continuous function is:

$$x(t) = \sum_{n=-\infty}^{\infty} x[n] sinc\left(\frac{t - nT}{T}\right) \tag{3.43}$$

which has the Fourier transformation $X(f)$ whose non-zero values are confined to the region $|f| \leq \frac{1}{2T} = B$. When $x[n]$ represents time samples at interval T of a continuous function, then the quantity $f_s = \frac{1}{T}$ is known as its sample rate and $\frac{f_s}{2}$ denotes the Nyquist frequency. The sampling Theorem states that when a function has a Bandlimit B less than the Nyquist frequency, then $x(t)$ is a perfect reconstruction of the original function.

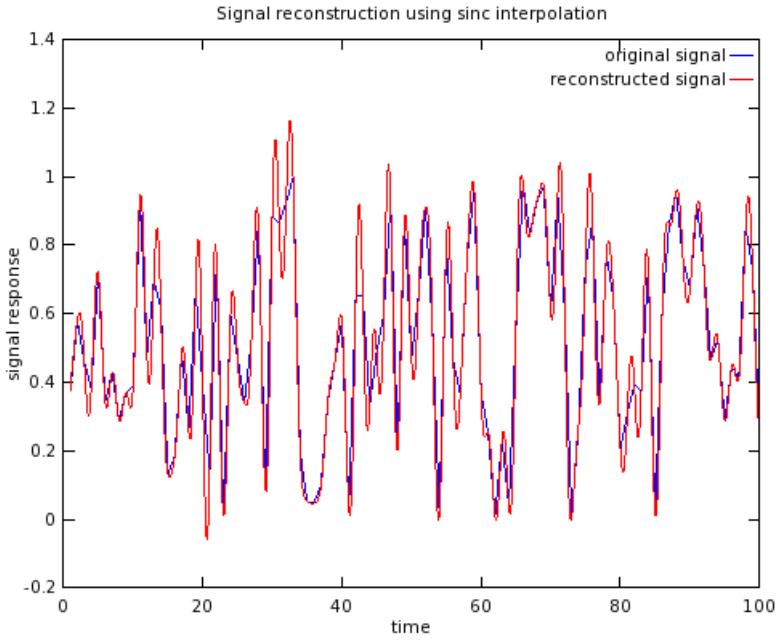


Figure 3.6: Comparission between a given random one dimensional input signal $s(t)$ and its sinc interpolation $\hat{s}(t)$. Notice that for the interpolation there were $N = 100$ samples from the original signal provided.

Chapter 4

Implementation

In computer graphics, we generally synthesize 2d images from a given 3d scene description[1]. This process is denoted as rendering. A usual computer graphics scene consist of a viewer's eye, modeled by a virtuel camera, light sources and geometries placed in the world[2], having some material properties[3] assigned to. In our implementation, scene geometries are modeled by triangular meshes for which each triangle is represented by a triple of vertices. Each vertex has a position, a surface normal and a tangent vector associated with.

The process of rendering basically involves a mapping of 3d sceene objects to a 2d image plane and the computation of each image pixel's color according to the provided lighting, viewing and material information of the given scene. These pixel colors are computed in several statges in so called shader programs, directly running on the Graphic Processing Unit (GPU) hardware device. In order to interact with a GPU, for our implementations, we rely on the programing interface of OpenGL[4], a cross-language, multiplattform API. In OpenGL, there are two fundamental shading pipeline stages, the vertex- and the fragment shading stage, each applied sequentially. Vertex shaders apply all transformations to the mesh vertices and pass this data to the fragment shaders. Fragment shaders receive linearly interpolated vertex data of a particular triangle. They are responsible to compute the color of his triangle.

In this chapter we explain in detail a technique for rendering structural colors due to diffraction effects on natural grainges, based on the model we have derived in the previouse chapter 3, summarized in section 3.5. For this purpose we implemented a reference framework which is based on a class project of the lecture *Computer Graphics* held by Mr. M. Zwicker which I attended in autum 2012[5].

For performing the rendering process, our implementation expects being provided by the fol-

[1] A usual computer graphics scene consist of a viewer's eye, modeled by a virtuel camera, light sources and geometries placed in the world, having some material properties assigned to.

[2] With the term world we are refering to a global coordinate system which is used in order to place all objects.

[3] Example material properties are: textures, surface colors, reflectance coefficients, refractive indices and so on.

[4] Official website:`http://www.opengl.org/`

[5] The code of underlying reference framework is written in Java and uses JOGL and GLSL[6] in order to comunicate with the GPU and can be found at `https://ilias.unibe.ch/`

[6] JOGL is a Java binding for OpenGL (official website `http://jogamp.org/jogl/www/`) and GLSL is OpenGL's high-level shading language. Further information can be found on wikipedia: `http://de.wikipedia.org/wiki/OpenGL_Shading_Language`

lowing input data[7]:

- the structure of snake skin of different species[8] represented as discrete valued height fields acquired using AFM and stored as grayscale images.
- real measured snake geometry represented as a triangle mesh.

The first processing stage of our implementation is to compute the Fourier Terms of the provided height fields like described in section 3.4. For this preprocessing purpose we use Matlab relying on its internal, numerically fast, libraries for computing Fourier Transformations[9]. The next stage is to read these precomputed Fourier Terms into our Java renderer. This program also builds our manually defined rendering scene. The last processing stage of our implementation is rendering of the iridescent colorpatterns due to light diffracted on snake skins. We implemented our diffraction model from chapter 3 as OpenGL shaders. Notice that all the necessary computations in order to simulate the effect of diffraction are performed within a fragment shader. This implies that we are modeling pixelwise the effect of diffraction and hence the overall rendering quality and runtime complexity depends on rendering window's resolution.

In the following sections of this chapter we are going to explain all render processing stages in detail. First, we discuss, how our precomputation process, using Matlab, actually works. Then, we introduce our Java Framework. It is followed by the main section of this chapter, the explanation how our OpenGL shaders are implemented. The last section discusses an optimization of our fragment shader such that it will have interactive runtime.

4.1 Precomputations in Matlab

Our first task is to precompute the two dimensional discrete Fourier Fransformations for a given input height field, representing a natural grating. For that purpose we have written a small Matlab [10] script conceptialized in algorithm 1. Our Matlab script reads a given image, which is representing a nano-scaled height field, and computes its two dimensional DFT (2dDFT) by using Matlab's internal Fast Fourier Transformation (FFT) function, denoted by $ifft2$[11]. Note that we only require one color channel of the input image, since the input image is representing an height field, encoded by just one color. Keep in mind that taking the Fourier transformation of an arbitrary function will result in a complex valued output which implies that we will get a complex value for frequency pairs of our input image. Therefore, for each input image we get as many output images, representing the 2dDFT, as the minimal number of taylor terms required for a well-enough approximation. In order to store our output images, we have to use two color channels instead of just one like it was for the given input image. Some example visualizations for the Fourier Tranformation are shown in figure 4.1. We store these intermediate results as binary files to offer floating point precision for the run-time computations to ensure higher precision.

[7] All data is provided by the Laboratory of Artificial and Natural Evolition in Geneva. See their website:`www.lanevol.org`

[8] We are using height field data for Elaphe and Xenopeltis snakes individuals like shown in figure 1.1

[9] Actually we use Matlab's inverse 2d Fast Fourier Transformation (FFT) implementation applied on different powers of quation 2.7. Further information can be read up in section 4.1

[10] Matlab is a interpreted scripting language which offers a huge collection of mathematical and numerically fast and stable algorithms.

[11] Remember, even we are talking about fourier transformations, in our actual computation, we have to compute the inverse fourier transformation. See paragraph 2.3 for further information. Furthermore our height fields are two dimensional and thus we have to compute a 2d inverse fourier transformation.

In our script every discrete frequency is normalized by its corresponding DFT extrema[12] in the range $[0,1]$ and the range extrema are stored seperately for each DFT term. The normalization is computed the following way:

$$\begin{aligned} f : [x_{min}, x_{max}] &\to [0,1] \\ x \mapsto f(x) &= \frac{x - x_{min}}{x_{max} - x_{min}} \end{aligned} \tag{4.1}$$

Where x_{min} and x_{max} denote the extreme values of a DFT term. Later, during the shading process of our implementation, we have to apply the inverse mapping. This is non-linear interpolation which is required in order to rescaled all frequency values in the DFT terms.

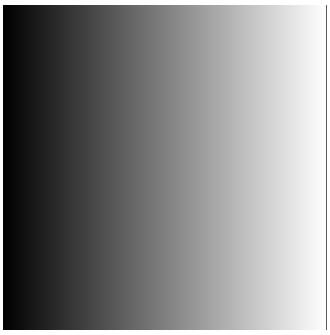
[12] We are talking about the i2dFFT of our height fields to the power of n. This is an N by N matrix (assuming the discrete height field was an N by N image), for which each component is a complex number. Hence, there is a a complex extrema as well as a imaginary extrema.

Algorithm 1 Precomputation: Pseudo code to generate Fourier terms

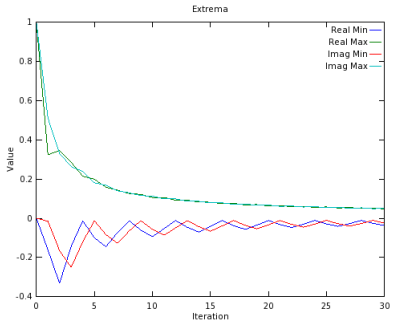
INPUT *heightfieldImg, maxH, dH, termCnt*
OUTPUT *DFT terms stored in Files*

```
% maxH:     A floating-point number specifying
%           the value of maximum height of the
%           height-field in MICRONS, where the
%           minimum-height is zero.
%
% dH:       A floating-point number specifying
%           the resolution (pixel-size) of the
%           'discrete' height-field in MICRONS.
%           It must be less than 0.1 MICRONS
%           to ensure proper response for
%           visible-range of light spectrum.
%
% termCnt: An integer specifying the number of
%           Taylor series terms to use.

function ComputeFFTImages(heightfieldImg, maxH, dh, termCnt)
dH = dh*1E-6;
% load patch into heightfieldImg
patchImg = heightfieldImg.*maxH;
% rotate patchImg by 90 degrees
for t = 0 : termCnt
  patchFFT = power(1j*patchImg, t);
  fftTerm{t+1} = fftshift(ifft2(patchFFT));

  % rescale terms as
  imOut(:,:,1)  = real(fftTerm{t+1});
  imOut(:,:,2)  = imag(fftTerm{t+1});
  imOut(:,:,3)  = 0.5;

  % rotate imOut by -90 degrees
  % find real and imaginary extrema of
  % write imOut, extrema, dH, into files.
end
```

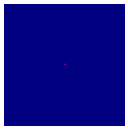
They key idea of algorithm 1 is to compute iteratively the Fourier Transformation for different powers of the provided height field. These DFT values are scaled by according to their extrema values. Another note about the command fftshift: It rearranges the output of the ifft2 by moving the zero frequency component to the centre of the image. This simplifies the computation of DFT terms lookup coordinates during rendering.



(a) Heightfield of a Blazed Grating

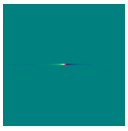


(b) Plot of extreme values for different powers of blazed grating



(c) ImRe0

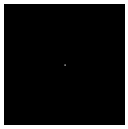
(d) ImRe1



(e) ImRe4

(f) ImRe10

(g) ImRe20



(h) Re0

(i) Re1

(j) Re4

(k) Re10

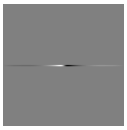
(l) Re20

(m) Im0

(n) Im1

(o) Im4

(p) Im10



(q) Im20

Figure 4.1: A visualization of DFT terms for a height field of a Blazed Grating.

In figure 4.1 we see examples of a visualization of Fourier Transformations generated by our Matlab script for a blazed grating[13] as an input height field image, shown in figure 4.1(a). Figure 4.1(b) shows plots of the extreme values of DFT terms for different powers of the blazed grating. We recognize that, the higher the power of the grating becomes, the closer the extreme values of the corresponding DFT terms get. The figure line from figure 4.1(c) until figure 4.1(g) show us example visualizations of DFT terms for different powers of our grating's height field. Remember that DFT terms are complex valued matrices of dimension as their height field has. In this visualization, all real part values are stored in the red- and the imaginary parts in the green color channel of an DFT image. The figure line from figure 4.1(h) till figure 4.1(l) show us the real part images from above's line corresponding figures. Similarly for the figur line from figure 4.1(m) until figure 4.1(q) showing the correspinding imaginary part DFT term images.

[13] A blazed grating is a height field consisting of ramps, periodically aligned on a given surface.

4.2 Java Renderer

This section explains the architecture of the rendering program which I implemented[14] and used for this project. The architecture of the program is divided into two parts: a rendering engine, the so called jrtr (java real time renderer) and an application program. Figure 4.2 outlines the architecture of the renderer.

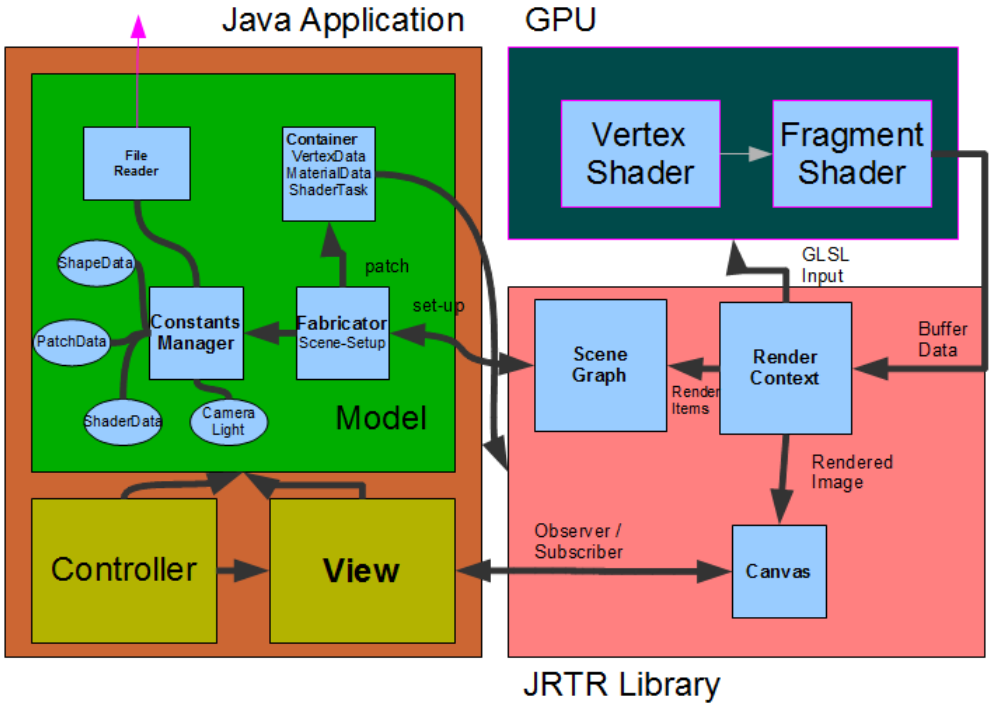


Figure 4.2: Schematical architecture of our Java renderer.

The application program relies on the MVC (Model-View-Controller) architecture pattern. The View just represents a canvas in which the rendered images are shown. The Controller implements the event listening functionalities for interactive rendering within the canvas. The Model of our application program consists of a Fabricator, a file reader and a constants manager. The main purpose of a Fabricator is to set up a rendering scene by accessing a constant manager containing many predefined scene constants. A scene consists of a camera, a light source, a frustum, shapes and their associated material constants. Such materials include a shape texture, precomputed DFT terms[15] for a given height field[16] like visualized in figure 4.1. A shapes is a geometrical object defined by a triangular mesh as shown in figure 4.3.

[14] This program is based on the code of a java real-time renderer, developed as a student project in the computer graphics class, held by M. Zwicker in autumn 2012.

[15] See section 4.1 for further information.

[16] and other height field constants such as the maximal height of its bumps or its pixel real-world width correspondance.

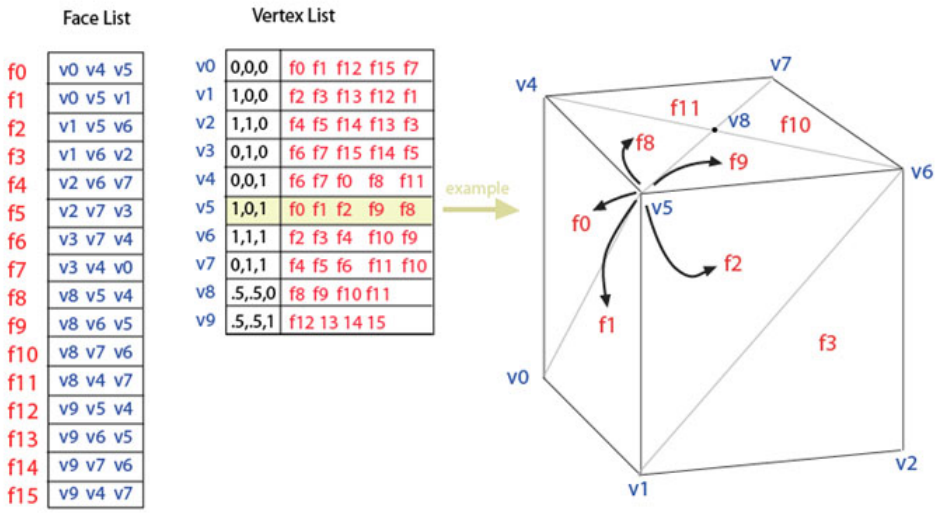


Figure 4.3: Representation[17] of a triangular mesh represents an object as a set of triangles and a set of vertices.

Such a mesh is represented as a data structure consisting of a list of vertices, each stored as a triplet of x, y, z positions and triangles, each defined by a triple of vertex-indices. Besides its position, a vertex can have further data assigned to, like a surface color, normals and texture coordinates. The whole scene is encapsulated in a scene graph data-structures, defined and managed within the rendering engine. A scene graph contains all scene geometries and their transformations in a tree like structured hierarchy.

All required configuration, in oder to communicate with the GPU through OpenGL, is performed in the jrtr rendering engine. Furthermore, jrtr's render context object, the wholeresource-management for various types of low-level buffers, which are used within the rendering pipeline by our GLSL shaders, takes place in the rendering place. More precisely, this means allocating memory for the buffers, assigning them with scene data and flushing them, when not used anymore. The whole shading process is performed in the GPU, stage-wise: The first stage is the vertex shader (see section 4.3.1) followed by the fragment shader (see section 4.3.2). The jrtr framework also offers the possibility to assign user-defined shaders written in GLSL.

4.3 GLSL Diffraction Shader

4.3.1 Vertex Shader

In our implementation we want to simulate the structural colors a viewer sees when light diffracted is on grating, e.g. on the skin of a snake. For this purpose, we reproduce a 2d image of a given 3d scene as seen from the perspective of a viewer for given lightning conditions. The color computation of an image is performed in the GPU shaders of the rendering pipeline. In OpenGL, there are two basic shading stages performed to rendern an image whereas the vertex shader is the first shading stage in the rendering pipeline.

[17] Modified image which originally has been taken from `http://en.wikipedia.org/wiki/Polygon_mesh`

As an input, a vertex shader receives one vertex of a mesh and other vertex data such as a vertex normals. It only can access this data and has no information about the neighborhood of a vertex or the topology of its corresponding shape. Since vertex positions of a shape are defined in a local coordinate system[18] and we want to render an image of the perspective of viewer, we have to transform the locally defined positions to a perspectively projected viewer space. Therefore, the main purpose[19] of a vertex shader is to tranform the position of vertices. Notice that a vertex shader can manipulate the position of a vertices, but cannot generate additional mesh vertices. Therfore, the output of any vertex shader is a transformed vertex position. Keep in mind that all vertex shader outputs will be used within the fragment shader. For an example, please have a look at our fragment shader 4.3.2.

In the following let us consider the whole transformation, applied in the vertex shader, in depth. Let p_{local} denote the position of a shape vertex, defined in a local coordinate system. Then the transformation from p_{local} into the perspective projected position as seen by a observer $p_{projective}$ looks like the following:

$$p_{projective} = P \cdot C^{-1} \cdot M \cdot p_{local} \tag{4.2}$$

where P, C^{-1} and M are transformation matrices[20], defined the following way:

Model matrix M: Each vertex position of a shape is initially defined in a local coordinate system. To make is feasible to place and transform shapes in a scene, a reference coordinate system, the so called world space, has to be introduced. Hence, for every shape a matrix M is associated, defining the transformation from its local coordinate system into the world space.

Camera matrix C: A camera models how the eye of a viewer sees an object, defined in world space like shown in figure 4.4. For calculating the transformation matrix C, a viewer's eye position and viewing direction, each defined in world space, are required. Therefore, C denotes a transformation from coordinates defined in camera space into the world space. Thus, in order to transform a position from world space to camera space, we have to use the inverse of C, denoted by C^{-1}.

Frustum P: The Matrix P defines a perspective projection onto image plance, i.e. for any given position in camera space, P determines the corresponding 2d image coordinate. Persepctive projections project along rays that converge in center of projection.

Since we are interested in modeling how a viewer sees structural colors on a given scene shape as shown in figure 4.4, modeling a viewer's eye by formulating the corresponding camera matrix C, is the most important component of the whole transformation series applied in the vertex shader. Hence, we next will have a closer look in how a camera matrix C actually can be computed.

[18] Defining the positions of a shape in a local coordinate system simplifies its modeling process and allows us to apply transformations to a shape.

[19] Furthermore, texture coordinates used for texture-lookup within the fragment shader and per vertex lightning can be computed.

[20] These tranformation matrices are linear transformations expressed in homogenous coordinates.

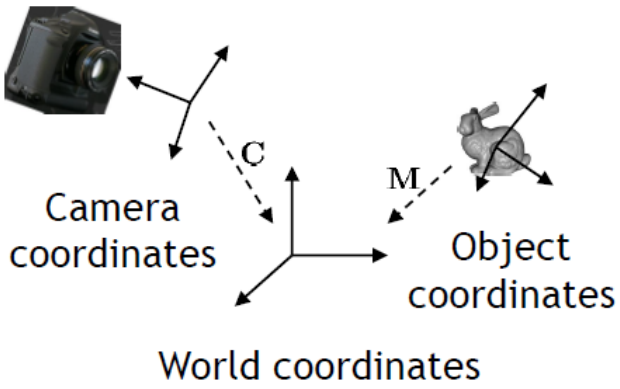


Figure 4.4: Illustration[21] of the Camera coordinate system where its origin defines the center of projection of camera.

The camera matrix C is constructed from its center of projection e, the position d where the cameras looks at and a direction vector up, defining what is the direction in camera space pointing upwards. These components, e, d and up, are defined in world coordinates. Figure 4.5 illustrates geometrical setup required in order to construct C.

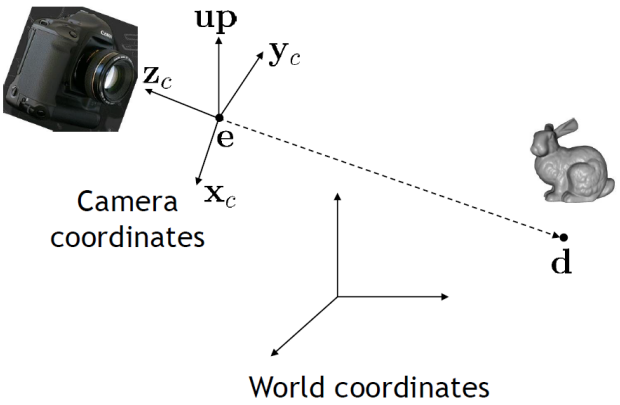


Figure 4.5: Illustration[22] of involved components in order to construct the camera matrix C. The eye-vector e denotes the positon of the camera in space, d is the position the camera looks at, and up denotes the cameras height. The camera space is spanned by the vectors helper vectors x_c, y_c and z_c. Notice that objects we look at are in front of us, and thus have negative z values

The mathematical representation of these vectors, x_c, y_c and z_c, spanning the camera space, introduced in figure 4.5, looks like the the following:

[21]This image has been taken from the lecture slides of computer graphics class 2012 which can be found on ilias.
[22]This image has been taken from the lecture slides of computer graphics class 2012 which can be found on ilias.

$$
\begin{aligned}
z_c &= \frac{e-d}{||e-d||} \\
x_c &= \frac{up \times z_c}{||up \times z_c||} \\
y_c &= z_c \times x_c
\end{aligned} \tag{4.3}
$$

As we can see, x_c, y_c and z_c are independent unit vectors. Therefore, they span a 3d space, the so called camera matrix. In order to express a coordinate in camera space, we have to project it onto these unit vectos. Using a homogenous coordinates representation, this a projection onto these unit vectors can be formulated by the transformation matrix C:

$$
C = \begin{bmatrix} x_c & y_c & z_c & e \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.4}
$$

In our vertex shader, besides transforming the vertex positions like described in equation 4.2, for every vertex, we also compute the direction vectors ω_i and ω_r described like in figure 2.13. Those direction vectors are transformed onto the tangent space, a local coordinate system spanned by a vertex's normal, tangent and binormal vector. For further information and more insight about the the tangent space, please bave a look at the appendix in the section $D.3$. The algorithmic idea of our vertex shader, stating all its computational steps, is conceptualized in algorithm 2.

Algorithm 2 Vertex diffraction shader pseudo code

Input: *Mesh* with vertex *normals* and *tangents*
Space tranformations $\{M, C^{-1}, P\}$
Light direction *lightDirection*
Output: Incident light and viewer direction $\omega_i,\ \omega_r$
Transformed position p_{per}

Procedures: *normalize*(), *span*(), *projectVectorOnTo*()

1: **Foreach** *VertexPosition position* $\in$ *Mesh* **do**
2: $vec3\ N = normalize(M * vec4(normal, 0.0).xyz)$
3: $vec3\ T = normalize(M * vec4(tangent, 0.0).xyz)$
4: $vec3\ B = normalize(cross(N, T))$
5: $TangentSpace = span(N, T, B)$
6: $viewerDir = ((cop_w - position)).xyz$
7: $lightDir = normalize(lightDirection)$
8: $\omega_i = projectVectorOnTo(lightDir, TangentSpace)$
9: $\omega_r = projectVectorOnTo(viewerDir, TangentSpace)$
10: $normalize(\omega_i);\ normalize(\omega_r)$
11: $p_{per} = P \cdot C^{-1} \cdot M \cdot p_{obj}$
12: **end for**

As input, our vertex shader algorithme 2 takes a mesh with of a given scene shape. Each of this vertexc should have a normal and a tangent assigned to. Furthermore, the direction of the scene light is required. For our implementation we always used directional light sources. An example of an directional light source is given in figure 4.6.

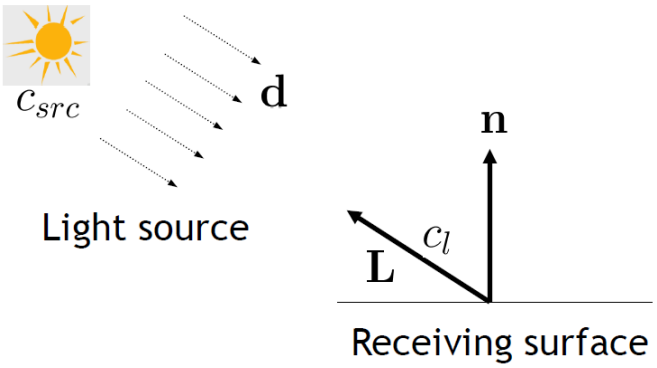


Figure 4.6: Illustration[23] of our light source setup. For a directional light source, all light rays are in parallel.

Last, in order to transform the positions of our mesh like described in equation 4.2, we also have to pass these transformation matrices[24]. For simplification purposes, we introduced the following helper procedures used in the vertex shading algorithm:

normalize(): Computes the normalized version of a given input vector.

span(): Assembles a matrix from a given set of vectors. This matrix spanes a vector space.

projectVectorOnTo(): Takes two arguments, a vector and a matrix. The first argument is projected onto each column of a given matrix. And returns a vector living the space spaned by the given argument matrix.

The output of our vertex shader is on the one side the transformed vertex postion and on the other side the incident light ω_i and viewing direction ω_r both transformed into the tangent space. The output of the vertex shader is used as the input of the fragment shader, discussed in the next section.

4.3.2 Fragment Shader

The purpose of a fragment shader is to render per fragment. A fragment is spanned by three vertices of a given mesh. For each pixel within a fragment in the fragment shader, the output of from its spanning vertices computed in the vertex shaders 2 is trilinearly interpolated depending on the pixel's position within the fragment. Furthermore, there can be additional input be assigned which is not directly interpolated from the output of vertex shader programs. In our fragment shader 3 this will be: all the references to the image buffers, containing the Fourier images computed in Matlab 4.1, the number steps for the taylor approximation (in our shader 30), the minimal and maximal wavelength, scaling factors, a reference to a lookup table containing the CIE_{XYZ} color weights. Basically the whole computation within our fragment shader relies on the direction of light and the viewing direction. Our shader performs a numerical integration for our final derived expression in equation 3.33 using the trapezoidal-rule with uniform discretization of the wavelength spectrum at $5nm$ step sizes. This implies we are compressing sampled frequencies to the region near to the origin of their frequency domain due to the fact we are dividing the (u,v) by the

[23]This image has been taken from the lecture slides of computer graphics class 2012 which can be found on ilias.
[24]When speaking about transformation matrices, we are refering to the model, camera and frustum matrix.

wavelength and this implies that the (u, v) space is sampeled non-linearly.

The Gaussian window approach derived in section 3.2.3 is performed for each discrete λ value using a window large enough to span $4\sigma_f$ in both dimensions. For precomputing DFT tables we generally use nanostructure height fields that span at least $65\mu m^2$ and are sampled with resolution of at least 100nm. This ensures that the spectral response encompasses all the wavelengths in the visible spectrum, i.e. from 380nm to 780nm.

Algorithm 3 Fragment diffraction shader pseudo code

Input: Precomputed DFT Terms
Scene Geometry
ω_i and ω_r
Output: Structural Color of a pixel

Procedures: the LIST

1: **Foreach** *Pixel* $p \in$ *Fragment* **do**
2: **INIT** $BRDF_{XYZ}, BRDF_{RGB}$ **TO** $vec4(0.0)$
3: $(u, v, w) = -\omega_i - \omega_r$
4: **for** $(\lambda = \lambda_{min}; \lambda \leq \lambda_{max}; \lambda = \lambda + \lambda_{step})$ **do**
5: $xyzWeights = ColorWeights(\lambda)$
6: $lookupCoord = lookupCoord(u, v, \lambda)$
7: **INIT** P **TO** $vec2(0.0)$
8: $k = \frac{2\pi}{\lambda}$
9: **for** $(n = 0$ **TO** $T)$ **do**
10: $taylorScaleF = \frac{(kw)^n}{n!}$
11: **INIT** F_{fft} **TO** $vec2(0.0)$
12: $anchorX = int(floor(center.x + lookupCoord.x * fftImWidth)$
13: $anchorY = int(floor(center.y + lookupCoord.y * fftImHeight)$
14: **for** $(i = (anchorX - winW)$ **TO** $(anchorX + winW))$ **do**
15: **for** $(j = (anchorY - winW)$ **TO** $(anchorY + winW))$ **do**
16: $dist = distVecFromOriginTo(i, j)$
17: $pos = localLookUp(i, j, n)$
18: $fftVal = rescaledFourierValueAt(pos)$
19: $fftVal *= gaussWeightOf(dist)$
20: $F_{fft} += fftVal$
21: **end for**
22: **end for**
23: $P += taylorScaleF * F_{fft}$
24: **end for**
25: $xyzPixelColor += dot(vec3(|P|^2), xyzWeights)$
26: **end for**
27: $BRDF_{XYZ} = xyzPixelColor * C(\omega_i, \omega_r) * shadowF$
28: $BRDF_{RGB}.xyz = D_{65} * M_{XYZ-RGB} * BRDF_{XYZ}.xyz$
29: $BRDF_{RGB} = gammaCorrect(BRDF_{RGB})$
30: **end for**

From line 4 to 26:
This loop performs uniform sampling along wavelength-space. $ColorWeights(\lambda)$ computes the color weight for the current wavelength λ by linear interpolation between the color weight for $\lceil\lambda\rceil$ and $\lfloor\lambda\rfloor$ which are stored in a external weights-table (assuming this table contains wavelengths in 1nm steps). At line 6: $lookupCoord(u, v, \lambda)$ the coordinates for the texture lookup are computed - See 4.7. Line 25 sums up the diffraction color contribution for the current wavelength in iteration λ.

From line 9 to 24:
This loop performs the Taylor series approximation using T terms. Basically, the spectral response is approximated for our current (u, v, λ). Furthermore, neighborhood boundaries for the gaussian-window sampling are computed, denoted as anchorX and anchorY.

From line 14 to 22:
In this inner most loop, the convolution of the gaussian window with the DFT of the patch is performed. $gaussWeightOf(dist)$ computes the weights in equation (3.8) from the distance between the current pixel's coordinates and the current neighbor's position in texture space. Local lookup coordinates for the current fourier coefficient $fftVal$ value are computed at line 17 and computed like described in 4.9. The actual texture lookup is performed at line 18 using those local coordinates. Inside $rescaledFourierValueAt$ the values $fftVal$ is rescaled by its extrema, i,e. $(fftVal * Max + Min)$ is computed, since $fftVal$ is normalized 4.1. The current $fftVal$ values in iteration is scaled by the current gaussian weight and then summed to the final neighborhood FFT contribution at line 20.

After line 26:
At line 27 the gain factor $C(\omega_i, \omega_r)$ 3.18 is multiplied by the current computed pixel color like formulated in 3.19. The gain factor contains the geometric term $refeq : geometricterm$ and the Fresnel term F. W approximate F by the Schlick approximation D.1, using an reactive index at 1.5 since this is close to the measured value from snake sheds. Our BRDF values are scaled by s shadowing function as described in (SEE REFERENCES - PAPER), since most of the grooves in the snake skin nano-structures would from a V-cavity along the plane for a wave front with their top-edges at almost the same height.

Last, we transform our colors from the CIE_XYZ colorspace to the CIE_RGB space using the CIE Standard Illuminant D65, followed by a gamma correction. See 4.4.3 for further insight.

4.4 Technical details

4.4.1 Texture lookup

In a GLSL shader the texture coordinates are normalized which means that the size of the texture maps to the coordinates on the range $[0, 1]$ in each dimension. By convention the the bottom left corner of an image has the coordinates $(0, 0)$, whereas the top right corner has the value $(1, 1)$ assigned.

Given a nano-scaled surface patch P with a resolution A by A microns stored as an N by N pixel image I. Then one pixel in any direction corresponds to $dH = \frac{A}{N}\mu m$. In Matlab we compute a series of n output images $\{I_{out_1}, ..., I_{out_n}\}$ from I, which we will use for the lookup in our shader - See figure 4.7. For the lookup we use scaled and shifted (u, v) coordinates from 2.5.

Since the zero frequency component of output images was shifted towards the centre of each image, we have to shift u, v to the center of the current N by N pixel image by a bias b. Mathematically, the bias is a constant value is computed the following:

$$b = (N\%2 == 0) \quad ? \quad \frac{N}{2} : \frac{N-1}{2} \tag{4.5}$$

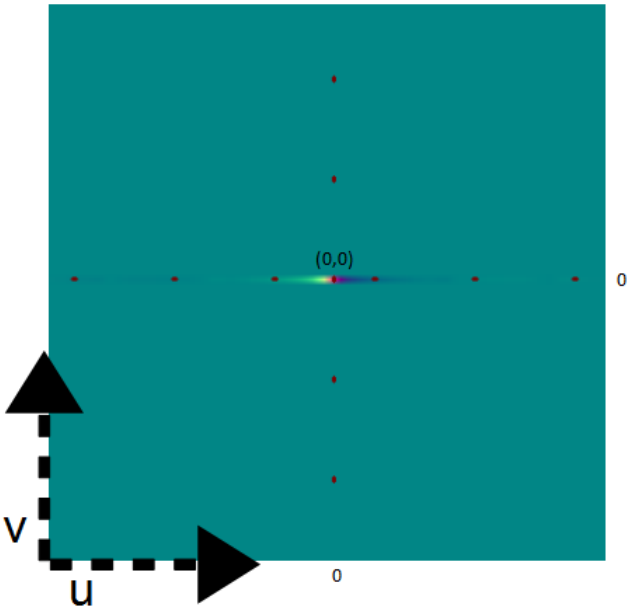


Figure 4.7: (u, v) lookup image

For the scaling we have to think a little further: lets consider a T periodic signal in time, i.e. $x(t) = x(t + nT)$ for any integer n. After applying the DFT, we have its discrete spectrum $X[n]$ with frequency interval $w0 = 2pi/T$ and time interval $t0$. Let $k = \frac{2\pi}{\lambda}$ denote the wavenumber for the current wavelength λ. Then the signal is both periodic with time period T and discrete with time interval t_0 then its spectrum should be both discrete with frequency interval w_0 and periodic with frequency period $\Omega = \frac{2\pi}{t_0}$. This gives us the idea how to discretize the spectrum: Let us consider our Patch P assuming it is distributed as a periodic function on our surface. Then, its frequency interval along the x direction is $w_0 = \frac{2\pi}{T} = \frac{2\pi}{N*dH}$. Thus only wave numbers that are integer multiples of w_0 after a multiplication with u must be considered, i.e. ku is integer multiple of w_0. Hence the lookup for the u-direction will look like:

$$\frac{ku}{w_0} = \frac{kuNdH}{2\pi} \tag{4.6}$$

$$= \frac{uNdH}{\lambda} \tag{4.7}$$

Using those findings 4.5, 4.7, the final (u, v) texture lookup-coorinates for the current wavelength λ in iteration, will then look like:

$$(u_{lookup}, v_{lookup}) = \left(\frac{uNdH}{\lambda} + b, \frac{vNdH}{\lambda} + b \right) \tag{4.8}$$

Note for the windowing approach we are visiting a one pixel neighborhood for each pixel p. This is like a base change with (u_{lookup}, v_{lookup}) as new coordinate system origin. The lookup coordinates for the neighbor-pixel (i, j) are:

$$(u_{lookup}, v_{lookup}) = (i, j) - (u_{lookup}, v_{lookup}) \tag{4.9}$$

4.4.2 Texture Blending

The final rendered color for each pixel is a weighted average of different color components, such as the diffraction color, the texture color and the diffuse color. In our shader the diffraction color is weighted by a constant $w_{diffuse}$. the texture color is once scales by a binary weight determined by the absolute value of the Fresnel Term F and once by $1 - w_{diffuse}$.

Algorithm 4 Texture Blending

$\alpha = (abs(F) > 1)?1 : 0$
$c_{out} = (1 - w_{diffuse}) * c_{diffraction} + (1 - \alpha) * c_{texture} + w_{diffuse} * c_{texture})$

4.4.3 Color Transformation

In our shader we access a table which contains precomputed CIE's color matching functions values from $\lambda_{min} = 380nm$ to $\lambda_{max} = 780nm$ in $5nm$ steps. Such a function value table can be found at downloaded at cvrl.ioo.ucl.ac.uk for example. We compute the (X, Y, Z) CIE_{XYZ} color values as described in section 2.1.8.

We can transform the color values into CIE_{RGB} by performing the following linear transformation:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = M \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{4.10}$$

where one possible transformation is:

$$M = \begin{bmatrix} 0.41847 & -0.15866 & -0.082835 \\ -0.091169 & 0.25243 & 0.015708 \\ 0.00092090 & -0.0025498 & 0.17860 \end{bmatrix} \tag{4.11}$$

There are some other color space transformation. The shader uses the CIE Standard Illuminant D65 which is intended to represent average daylight. Using D65 the whole colorspace transformation will look like:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = M \cdot \begin{bmatrix} X \cdot D65.x \\ Y \cdot D65.y \\ Z \cdot D65.z \end{bmatrix} \tag{4.12}$$

Last we perfrom gamma correction on each pixel's (R, G, B) value. Gamma correction is a non linear transformation which controls the overall brightness of an image.

4.5 Discussion

The fragment shader algorithm described in 3 performs the gaussian window approach by sampling over the whole wavelength spectrum in uniform step sizes. This algorithm is valid but also slow since we iterate for each pixel over the whole lambda spectrum. Furthermore, for any pixel, we iterate over its 1 neighborhood. Considering the loop for the taylor approximation as well, we will have a run-time complexity of $O(\#spectrtumIter \cdot \#taylorIter \cdot neighborhoodRadius^2)$. Hence, Instead sampling over the whole wavelength spectrum, we could instead integrate over just a few required lambdas which are elicited like the following: Lets consider (u, v, w) defined as 2.5. Let d be the spacing between two slits of a grating. For any $L(\lambda) \neq 0$ it follows $\lambda_n^u = \frac{du}{n}$ and $\lambda_n^v = \frac{du}{n}$. For $n = 0$ there it follows $(u, v) = (0, 0)$. If $u, v > 0$

$$N_{min}^u = \frac{du}{\lambda_{max}} \leq n_u \leq \frac{du}{\lambda_{min}} = N_{min}^u$$
$$N_{min}^v = \frac{dv}{\lambda_{max}} \leq n_v \leq \frac{dv}{\lambda_{min}} = N_{min}^v$$

If $u, v < 0$

$$N_{min}^u = \frac{du}{\lambda_{min}} \leq n_u \leq \frac{du}{\lambda_{min}} = N_{max}^u$$
$$N_{min}^v = \frac{dv}{\lambda_{min}} \leq n_v \leq \frac{dv}{\lambda_{min}} = N_{max}^v$$

By transforming those equation to $(\lambda_{min}^u, \lambda_{min}^u)$, $(\lambda_{min}^v, \lambda_{min}^v)$ respectively for any (u, v, w) for each pixel we can reduce the total number of required iterations in our shader.

Another variant is the PQ approach described in chapter 2 3.6.1. Depending on the interpolation method, there are two possible variants we can think of as described in 3.6.2. Either we try to interpolate linearly or use sinc interpolation. The first variant does not require to iterate over a pixel's neighborhood, it is also faster than the gaussian window approach. One could think of a combination of those tho optimization approaches. Keep in mind, both of these approaches are further approximation. The quality of the rendered images will suffer using those two approaches. The second variant, using the sinc function interpolation is well understood in the field of signal processing and will give us reliable results. The drawback of this approach is that we again have to iterate over a neighborhood within the fragment shader which will slow down the whole shading. The following algorithm describes the modification of the fragment shader 3 in oder to use sinc interpolation for the PQ approach 3.6.1.

Algorithm 5 Sinc interpolation for PQ approach

Foreach *Pixel* $p \in$ *Image* I **do**
  $w_p = \sum_{(i,j) \in \mathcal{N}_1(p)} sinc(\Delta_{p,(i,j)} \cdot \pi + \epsilon) \cdot I(i, j)$
  $c_p = w_p \cdot (p^2 + q^2)^{\frac{1}{2}}$
  $render(c_p)$
end for

In a fragment shader we compute for each pixel p in the current fragment its reconstructed function value $f(p)$ stores in w_p. w_p is the reconstructed signal value at $f(p)$ by the sinc function as described in 3.6.2. We calculate the distance $\Delta_{p,(i,j)}$ between the current pixel p and each of its neighbor pixels $(i, j) \in \mathcal{N}_1(p)$ in its one-neighborhood. Multiplying this distance by π gives us

the an angle used for the sinc function interpilation. We add a small integer ϵ in order to avoid division by zeros side-effects.

Chapter 5

Evaluation and data acquisition

5.1 Data Acquisition

Our goal is to perform physically accurate simulations of diffraction effects on natural gratings. As for every simulation, its outcome highly depends on the provided input data we also require measurements[1] of real natural gratings. For that purpose, samples of skins of Xenopeltis and Elaphe snake species were fixed on a glass plate and then, by using an Atomic Force Microscope (AFM), their surface topography was measured and stored as grayscale images, indicating the depth. In general, an ADM is is a microscope that uses a tiny probe mounted on a cantilever to scan the surface of an object. The probe is extremely close to the surface, but does not touch it. As the probe traverses the surface, attractive and repulsive forces arising between it and the atoms on the surface induce forces on the probe that bend the cantilever. The amount of bending is measured and recorded, providing a map of the atoms on the surface. Atomic force microscopes is a very high-resolution type of scanning probe microscopy, with demonstrated resolution on the order of fractions of a nanometer, more than 1000 times better than the optical diffraction limit.

5.2 Diffraction Gratings

In order to evaluate the quality of our simulations, it is important to understand what a diffraction grating actually is. An idealised diffraction grating like in figure 5.2 is made of a large number of parallel, evenly spaced slits in an opaque medium. In general, if the spacing between slits is wider than the wavelength of the incoming light, then the better we can observe how the light is diffracted on the grating. Simply speaking, each slit in the grating acts as a point light source from which light spreads and propagates in all directions. According to Huygen's Principle the outgoing light may have a different outgoing angle as it hat initially. Figure 5.1 illustrates this behaviour for a monochromatic light source passing through a grating and shows that the the outgoing angle will be different from the incident angle. Hence, the diffracted light is composed of the sum of interfering wave components emanating from each slit in the grating.

[1] All measured data has been provided by the Laboratory of Artificial and Natural Evolition at Genava - Website:`www.lanevol.org`

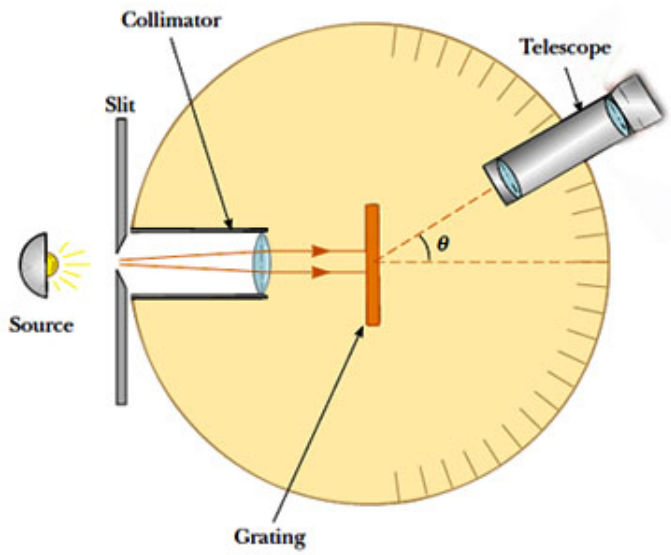


Figure 5.1: Spectometer: When a beam of monochromatic light passes through a grating placed in a spectrometer, images of the sources can be seen through the telescope at different angles.

Suppose that a monochromatic light source is directed at the grating, parallel to its axis as shown in figure 5.1. Let the distance between successive slits be equal the value d.

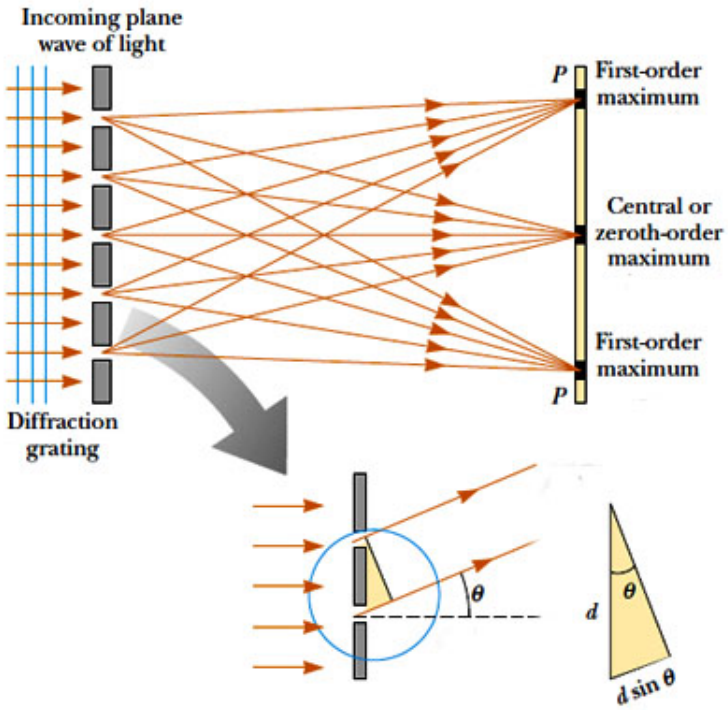


Figure 5.2: Light directed to parallel to grating:

The observable diffraction pattern is the result of interference effects among outgoing wavelets according to Huygen's Principle. The path difference between waves from any two adjacent slits can be derived by drawing a perpendicular line between the parallel waves. Applying some trigonometry, this path difference is $dsin(\theta)$. If the path difference equals one wavelength or a multiple of the wave's wavelength, the emerging, reflected waves from all slits will be in phase and a bright line will be observed at that point. Therefore, the condition for maxima in the interference pattern at the angle θ is:

$$dsin(\theta) = m\lambda \tag{5.1}$$

where $m \in \mathbb{N}_0$ is the order of diffraction. Because d is very small for a diffraction grating, a beam of monochromatic light passing through a diffraction grating is split into very narrow bright fringes at large angles θ.

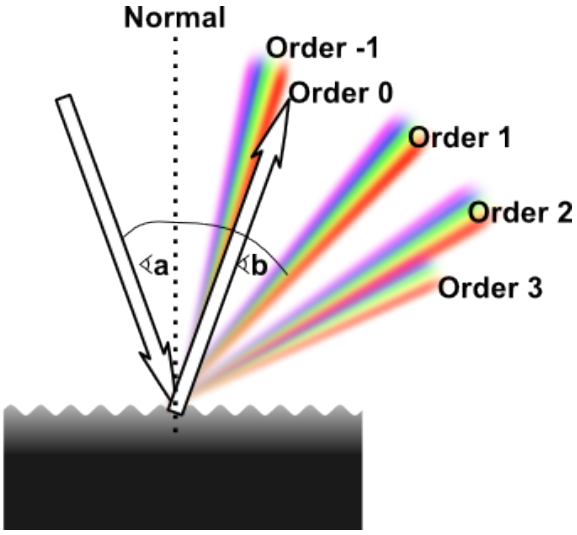


Figure 5.3: Different Orders of diffraction

When a narrow beam of white light is directed at a diffraction grating along its axis, instead of a monochromatic bright fringe, a set of colored spectra are observed on both sides of the central white band as shown in figure 5.3.

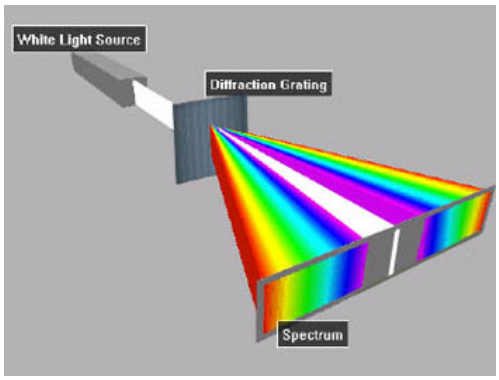


Figure 5.4: White Light beam causes coloured diffraction spectra

Since the angle θ increases with wavelength λ, red light, which has the longest wavelength, is diffracted through the largest angle. Similarly violet light has the shortest wavelength and is therefore diffracted the least. This relationship between angle and wavelength is illustrated in figure 5.4. Thus, white light is split into its component colors from violet to red light. The spectrum is repeated in the different orders of diffraction, emphasizing certain colors differently, depending on their order of diffraction like shown in figure 5.3. Note that only the zero order spectrum is pure white. Figure 5.5 shows the relative intensity resulting when a beam of light hits a diffraction grating for different number of periods. From the graph we recognise that the more slits a grating has, the sharper more slopes the function of intensity gets. This is similar like saying that, the more periods a grating has, the sharper the diffracted color spectrum gets like shown in figure 5.6.

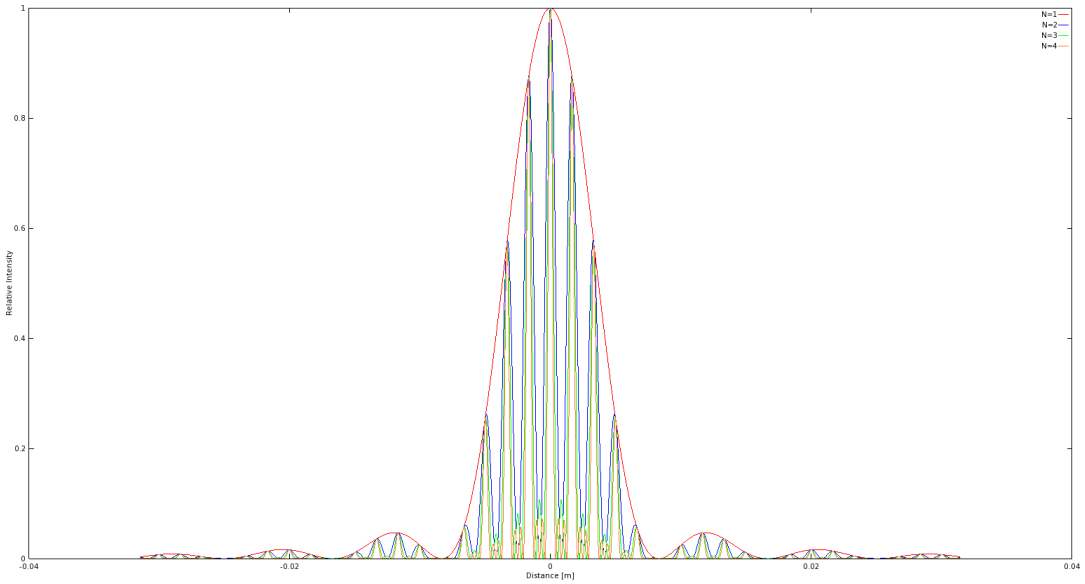
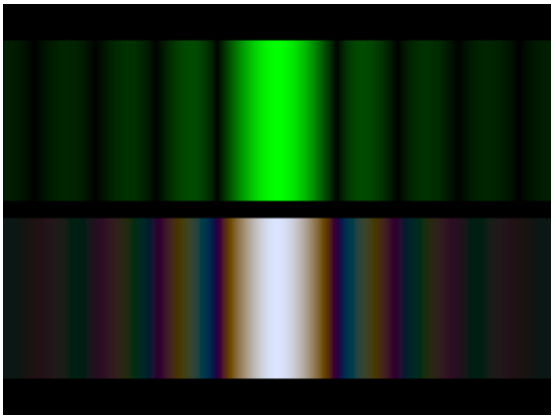
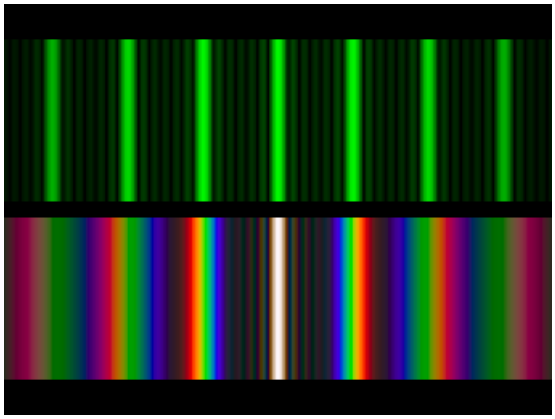


Figure 5.5: Relative intensities of a diffracted beam of light at wavelength $\lambda = 500nm$ on a grating for different number of periods N width slit width of 30 microns and slit separation of 0.15 mm each. The viewer is 0.5m apart from the grating.



(a) one slit



(b) seven slits

Figure 5.6: Difference of diffraction pattern between a monochromatic (top) and a white (bottom) light spectra for different number of slits.

5.3 Evaluation

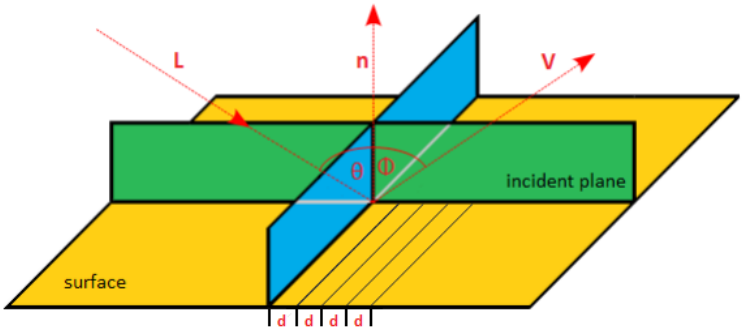


Figure 5.7: Experimental setup for evaluation: A light beam with direction L hits the surface, representing a grating pattern with periodicity d, at the incident plane relative to the surface normal n at angle θ and emerges at angle ϕ with direction V.

The physical reliability of our BRDF models has been verified by applying those on various patches which are a synthetic blazed grating, an Elaphe and a Xenopeltis snake shed sample patch. We compared the resulting response against the response resulting by the grating equation, which models the relationship between the grating spacing and the angles of the incident and diffracted beams of light. Figure 5.7 illustrates the geometrical setup for our evaluation approach: A monochromatic beam of light with wavelength λ hits a surface with periodicity d at an angle θ relative to the normal n along its incident plane. The beam emerges from the surface at the angle ϕ.

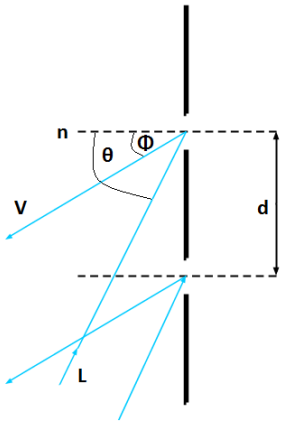


Figure 5.8: Reflecting grating: When the incident light direction is not parallel to its axis at the grating, there is another $sin(\phi)$ involved. See also the grating equation 5.2.

The maximum in intensity is given by the grating equation derived from the equation 5.1 following figure 5.8:

$$sin(\theta) = sin(\phi) + \frac{m\lambda}{d} \tag{5.2}$$

In our evaluation we are interested in the first order diffraction, i.e. m equals one which. We further assume that the incident light direction ω_i is given. In contrast the direction of the reflected wave ω_r is not given. In Mathematics, a three dimensional direction vector is fully defined by two two angles, i.e. it can be represented by spherical coordinates with radius $r = 1$. By convention, we denote those two vectors by θ and ϕ like in figure 5.7. Hence, θ_i, ϕ_i and ϕ_r are given constants whereas θ_i is a free parameter for our evaluation simulation. Therefore, we are going to compare the maxima for peak viewing angles corresponding to each wavelength using data produced by our method against the maxima resulting by the grating equation 5.2.

5.3.1 Precomputation

For evaluation purposes we have implemented our BRDF models in java. We once again use our geometrical setup as illustrated in figure 2.13 where θ_i, ϕ_i and ϕ_r are provided as input values and θ_i is a free parameter. Within our evaluation we have set them to $\theta_i = 75$ $\phi_i = 0$ $\phi_r = 180$ degrees. The wavelength space Λ and the range Θ of our free parameter θ_i are discretized in equidistant steps whereas their step sizes are given as input arguments for our Java program:

$$\Lambda = \{\lambda | \lambda = \lambda_{min} + k \cdot \lambda_{step}, \quad k \in \{0, .., C-1\}\} \tag{5.3}$$

where $\lambda_{step} = \frac{\lambda_{max} - \lambda_{min}}{C-1}$ and C is the discretisation level of the lambda space. We similarly discretise the angle space by predefining an minimal and maximal angle boundary and $ceil(angMax - angMin)/angInc$ is the number of angles. Our Java BRDF model implementations are applied on the grid $[\Lambda, \Theta]$ and will store their spectral response in a matrix

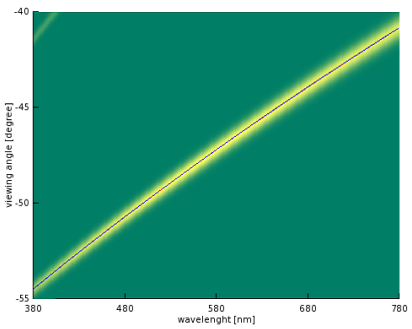
$$R = \{response(\lambda_i, \theta_r^j) | i \in Index(\Lambda), \quad j \in Index(\Theta)\} \tag{5.4}$$

We will plot this matrix and compare its graph against the grating equation for similar condition like in stated in algorithm 6.

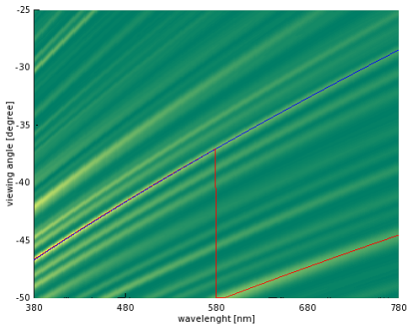
Algorithm 6 Vertex diffraction shader

load matrix R 5.4
$\lambda_{count} = |\Lambda|$
$\lambda_{inc} = \frac{\lambda_{max} - \lambda_{min}}{\lambda_{count}}$
$\lambda = \lambda_{min} + \lambda_{inc} \cdot (-1 + [1 : \lambda_{count}])$
$[maxCmaxI] = max(R)$
$viewAngForMax = angMin + angInc \cdot (maxI - 1)$
$thetaV = asin\left(\frac{\lambda}{d} - \sin\left(\frac{\theta_I \pi}{180}\right)\right) \cdot \frac{180}{\pi}$
$plot(\lambda, viewAngForMax)$ ▷ graph resulting by our brdf model
$plot(lambda, thetaV)$ ▷ graph resulting by grating equation

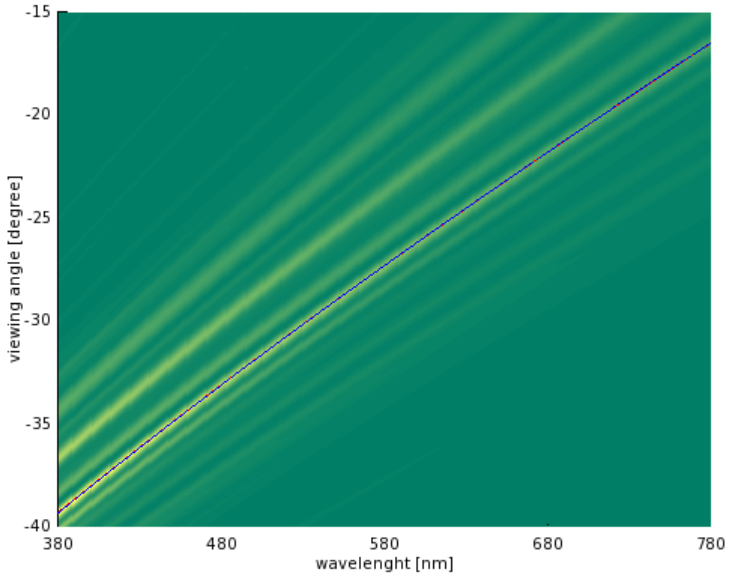
5.3.2 Evaluation graphs



(a) Blaze grating



(b) Xenopeltis grating



(c) Elaphe grating

Figure 5.9: Reflectance obtained by using the shading approach described in algorithm 3 simulating a BRDF which models the effect of diffraction at different viewing angles over the spectrum of visible light.

In this section we discuss the quality of our BRDF models applied to different surface structures. For that purpose we compare the resulting relative reflectance computed as described in section 5.3.1 for each of our BRDF models to the idealized grating equation 5.2.

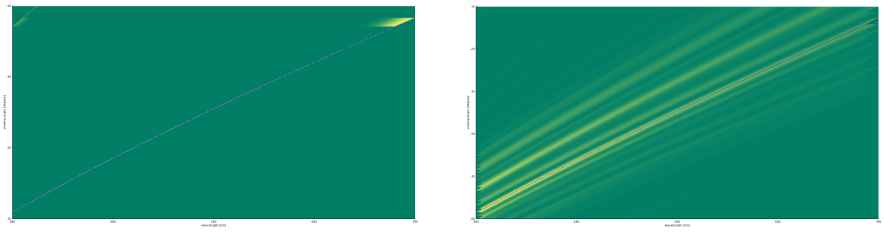
Patch	Mean[mm]	Variance[mm]
Blazed grating 6.2(a)	2500.34	0.16
Elaphe grating 6.2(b)	1144.28	0.15
Xenopeltis grating 6.2(c)	1552.27	0.45

Table 5.1: Statistics of periodicity d of our used gratings 6.2 estimated by using the grating equation 5.2. This table was provided by Mr. D.Singh.

Figure 5.9 shows the reflectance graphs resulting by the shading approach of sampling the whole lambda space descriped in algorithm 3. This evaluation has been applied to different idealized periodic structures, namely to the Blaze- 5.9(a), Elaphe- 5.9(c) and Xenopeltis grating 5.9(b), using an illumination angle $\theta_i = 75$ degrees. Note that higher response values are plotted in yellow and lower values in green. For each of the graphs we determine the viewing angles with peak reflectance for various wavelengths and then plot this peak viewing angles against their wavelength as solid red curves. The blue curve represents diffraction angles for an idealized periodic structure with a certain periodicity d according to the grating equation 5.2. The corresponding periodicity for every grating structure is estimated using the precomputed response data using again the grating equation and are tabulated in table 5.1.

The red and blue curve are closely overlapping in our figures 5.9(a) and 5.9(c). For Blaze and Elaphe there is only diffraction along only along one direction perceivable. Since the Blazed grating is synthetic we use its exact periodicity to plot the blue curve instead of estimating it. The Xenopeltis grating is evaluated just along the direction for the finger like structures. For Xenopeltis it is interesting to see that the red curve for the peak viewing angle toggles between two ridges corresponding to two different periodicities. this happens because there are multiple sub regions of the nanostructure with slightly different orientations and periodicity. Each sub region carves out a different yellowish ridge. depending on the viewing angle, reflectance due to one such subregion can be higher than from the others.

Figure 5.10 shows the evaluation plots for the (N_{min}, N_{max}) shading approach which integrates over a reduced wavelength spectrum applied to the Blaze- 5.10(b) and the Elaphe-grating 5.10(b). This optimization approach is mentioned within the discussion section of the implementation chapter 4.5 as a run-time complexity enhancement of the whole lambda space sampling approach 5.9. The response curve again closely matches the corresponding grating equation curve for both evaluation graphs and also look similar to the corresponding evaluation plots when integrating over the whole lambda space shown previously in figure . Therefore we may assume this optimization to be valid.

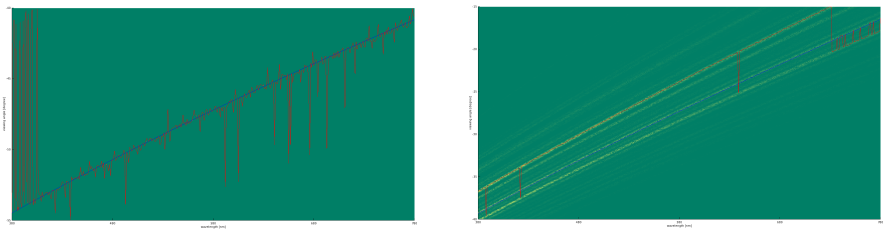


(a) Blaze grating (b) Elaphe grating

Figure 5.10: Reflectance obtained using $N_{min}N_{max}$ optimization apporach

Last let us consider the evaluation graphs of the PQ approach 5 in figure 5.11. The PQ approach

assumes the given grating being periodically distributed on a shape's surface. For this approach we have plotted evaluation graphs of the Blaze- 5.11(a) and Elaphe grating 5.11(b). For both graphs their response curves have some similarities but also some differences compared to their corresponding grating equation curve. We could say that the response curve of the blaze grating is weakly oscillating around the grating equation curve (blue) but basically following it even there are some outliers. The response curve of the Elpahe grating is not following its corresponding first order grating equation curve rather another response curve for the PQ approach. This could be due to the assumption of the PQ approach that a given patch must be periodically distributed along the surface which is actually not that case. Nevertheless, the red curve fits one of the response curves.



(a) Blaze grating (b) Elaphe grating

Figure 5.11: Reflectance obtained using PQ optimization apporach

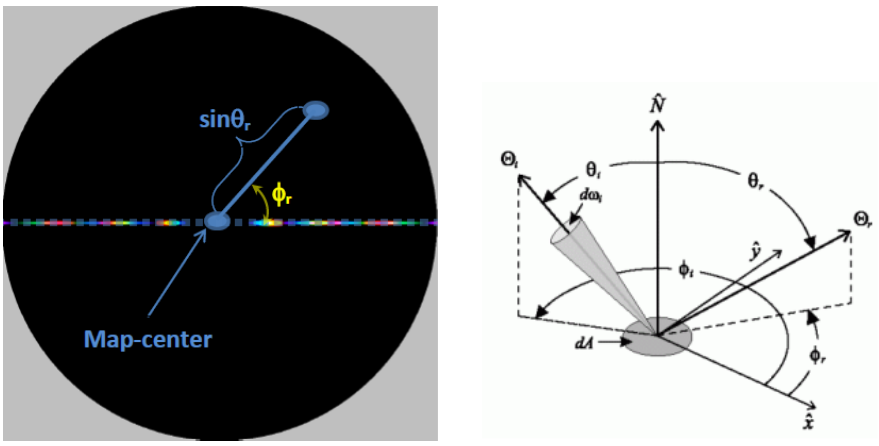
path resolutions

Chapter 6

Results

6.1 BRDF maps

In this chapter we examine the rendered output resulting by the implementation of our BRDF models applied on different input patches such as Blaze grating or Elaphe 1.1(b) and Xenopeltis 1.1(a) snake nano-scaled surface sheds. We going to discuss and compare both, their BRDF maps 6.1 and the corresponding renderings on a snake geometry 6.2 for various input parameters. Last we also show a real experimental image showing the effect of diffraction for similar parameters like we have. A BRDF map shows a shader's output for all possible viewing directions for a given, fixed, incident light direction. We assume that each viewing direction is in spherical coordinates form $D.2$ (θ_v, ϕ_v) and is represented in the map at point $(x, y) = (sin(\theta_v)cos(\phi_v), sin(\theta_v)sin(\phi_v))$ with its origin at the map-center. The light direction for normal incidence (θ_i, ϕ_i) has been fixed to $(0, \pi)$ for our rendered results.

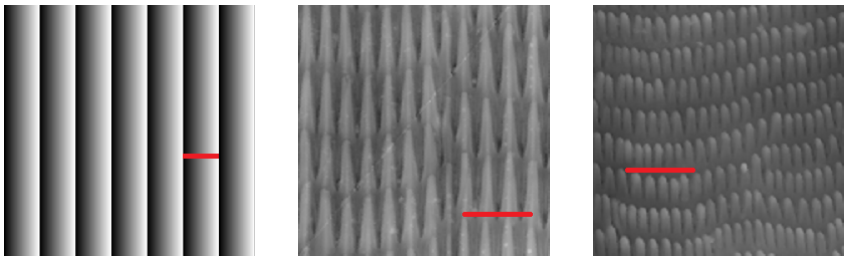


(a) BRDF map schema (b) Light reflection geometrical setting

Figure 6.1: BRDF maps for different patches: $\Theta = (\theta, \phi)$ is the direction of light propagation

In this chapter we examine the rendered output resulting by the implementation of our BRDF models applied on different input patches 6.2 such as Blaze grating or Elaphe 1.1(b) and Xenopeltis 1.1(a) snake nano-scaled surface sheds. We going to discuss and compare both, their BRDF maps

6.1 and the corresponding renderings on a snake geometry 6.2 for various input parameters. Last we also show a real experimental image showing the effect of diffraction for similar parameters like we have. A BRDF map shows a shader's output for all possible viewing directions for a given, fixed, incident light direction. We assume that each viewing direction is in spherical coordinates form $D.2$ (θ_v, ϕ_v) and is represented in the map at point $(x, y) = (sin(\theta_v)cos(\phi_v), sin(\theta_v)sin(\phi_v))$ with its origin at the map-center. The light direction for normal incidence (θ_i, ϕ_i) has been fixed to $(0, \pi)$ for our rendered results.



(a) Blaze grating with scale of 2.500 μm

(b) Elaphe patch with scale of 3.270 μm

(c) Xenopeltis patch with scale of 3.210 μm

Figure 6.2: Cutouts of our nano-scaled surface gratings used for rendering within our shader with a scale indicator (red line) for each patch. Note that for rendering we use larger patches.

Figure 6.3 shows the BRDF maps of the full lambda space sampling approach 4.3.2 applied on different nanoscale surface gratings as shown in figure 6.2. In Subfigure 6.2(a) we see the BRDF map for the Blazed grating, showing high relative brightness for its first order diffraction which means that for the Blazed gratings most of the diffracted spectral energy lies in its first order. Note that for Blazed grating their first-order diffracted light returns along the same path as the incident light. Higher diffraction orders are still perceivable (second and third order diffraction) but with a much lower relative brightness. The asymmetry of the pattern is due to the asymmetric geometry of the grating 6.2(a).

The finger-like structures contained in the Elaphe surface grating 6.2(b) are quite regularly aligned and hence diffraction occurs along the horizontal axis for the BRDF map as shown in figure 6.3(b). The reason for not seeing any strong diffraction color contribution along other directions in the BRDF map is due to the fact that these 'nano-fingers' overlap across layers and thus do not exhibit any definite periodicity along finger direction.

For Xenopeltis surface grating 6.2(c), we observe diffraction along many different, almost in vertical directions in the BRDF map 6.3(c) since the layers of the finger-like structures do not overlap and are shifted significantly along their length but still exhibit some local consistency. A similar argument holds true for diffraction across locally periodic finger patches with slightly different orientations.

(a) Blazed grating (b) Elaphe grating (c) Xenopeltis grating

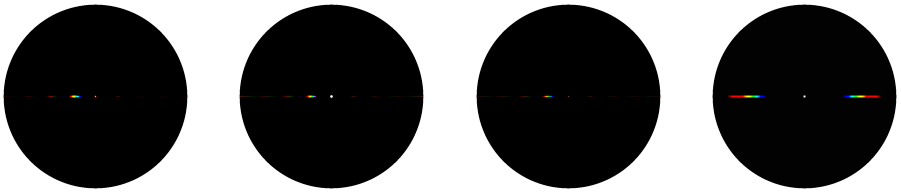
Figure 6.3: BRDF maps for different patches

TODO: CHANGE PQ, N MIN MAX IMAGE SINCE SINC INTERPOL IS MISSING Figure 6.4 shows a the BRDF maps of all our BRDF models applied on the Blaze grating. The most left figure $6.4(a)$ is showing the ground truth result for the Blazed grating used in order to compare with our other rendering approaches.

Figure $6.4(b)$ shows the BRDF map for the N_{min}, N_{max} approach which we assume to be close to the ground truth since its corresponding evaluation plots $5.10(a)$ already were closely matching. Nevertheless there is a small difference notable: since in the actual technical implementation of the N_{min}, N_{max} shading approach treats the center of the BRDF map as a separate case, i.e. everything around a small ϵ-circumference has white color assigned we note a white circular spot around the map center. Except this fact, the BRDF map resembles the ground truth.

Figure $6.4(c)$ shows the BRDF map for the PQ approach which relies on sinc-interpolation. As expected, after having considered the corresponding evaluation plots shown in figure$5.11(a)$, the PQ BRDF map and the ground truth are visual alike. Compared to the evaluation plots, the PQ BRDF maps even persuade more. One difference we notice is that the first order is a little spread. This effect is would be strengthened when using linear interpolation instead of sinc-interpolation.

Last, let us consider figure $6.4(d)$ which shows the BRDF map produced by using the implementation of Nvidia Gem's implementation (ADD REFERENCE) of Stam's BRDF model, when constrainting the y-axisis of the BRDF map. This model does not consider much more than the spacing d of a given grating. It also always produces highly symmetric results. It also does not render different orders of diffraction rather than the zero and first order.

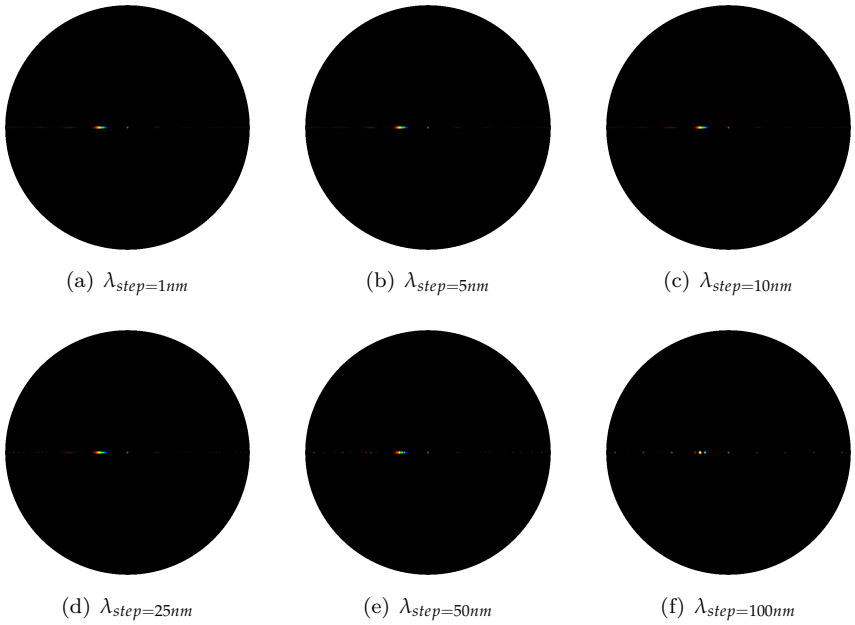


(a) FLSS (b) NMMS (c) PQ with Sinc-Interpolation (d) Nvidia Gem's Approach

Figure 6.4: BRDF maps for Blazed grating comparing our different rendering approaches

Figure 6.5 and figure 6.6 show the BRDF maps for different wavelength step sizes used in the fragment shader for the full lambda space sampling approach applied on the Blaze grating and the Elpahe snake shed, respectively. Within our fragment shaders the most outer loop iterate over the range $[380nm, 780nm]$ for fixed step sizes λ_{step} which basically constitutes the integration over the

wavelength spectrum. Therefore, having bigger step sizes implies having fewer step sizes which will fasten up the overall runtime of a shader but will also introduce artifacts and therefore lower the overall shading qualtity. For Elaphe surface grating, artifacts are perceptional arising when $\lambda_{step} \leq 10nm$. Results produced by using $5nm$ step sizes do not differ from those produced by using $\lambda_{step} = 1nm$ which allows us to set λ_{step} equal 5nm. For a Blazed grating we may chose even bigger step sized without losing any rendering quality.

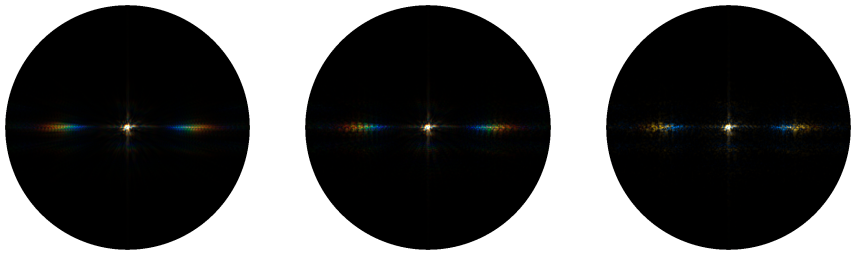


(a) $\lambda_{step=1nm}$ (b) $\lambda_{step=5nm}$ (c) $\lambda_{step=10nm}$

(d) $\lambda_{step=25nm}$ (e) $\lambda_{step=50nm}$ (f) $\lambda_{step=100nm}$

Figure 6.5: Blazed grating at $2.5\mu m$: Different λ step sizes

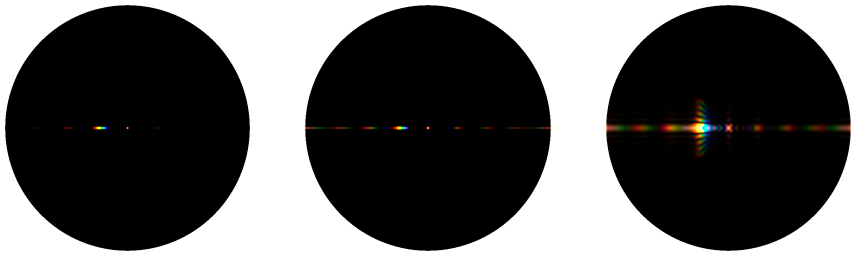
(a) $\lambda_{step=1nm}$ (b) $\lambda_{step=5nm}$ (c) $\lambda_{step=10nm}$



(d) $\lambda_{step=25nm}$ (e) $\lambda_{step=50nm}$ (f) $\lambda_{step=100nm}$

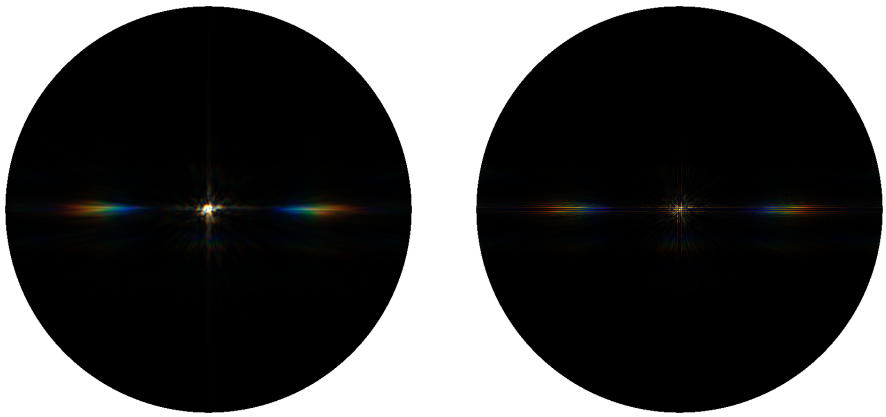
Figure 6.6: Elaphe grating at $65\mu m$: Different λ step sizes

The Figures 6.7, 6.9, 6.8 show a comparison of the BRDF maps produced by full lambda space sampling approach (on the left) and the PQ shading approach (on the right) applied on all our patches. For Blazed grating, as already mentioned, we notice that both approaches resemble each other. We also notice that for PQ map, the first order diffraction color contribution is spread. For Elaphe and Xenopeltis grating we notice similar shaped BRDF patterns, even when the angle of light varies, but nevertheless, they also contain some artifacts. This also holds true when we linearly interpolate instead relying on sinc-interpolation.



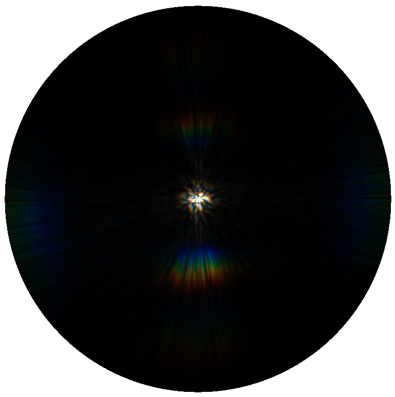
(a) Full Lambda Sampling: Blazed grating (b) Full Lambda Sampling brightened: Blazed grating (c) PQ Approach: Blazed grating

Figure 6.7: Blazed grating: PQ approach vs full lambda space sampling

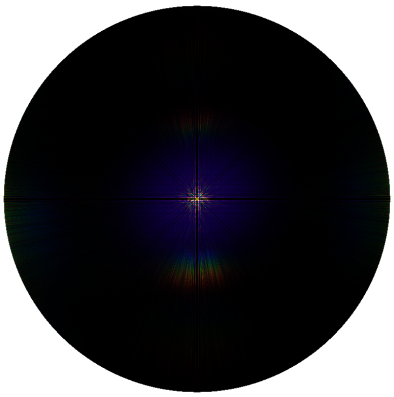


(a) Full Lambda Sampling: Elaphe grating (b) PQ Approach: Elaphe grating

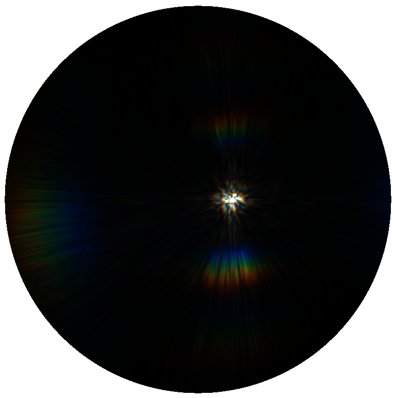
Figure 6.8: Elaphe grating: PQ approach vs full lambda space sampling



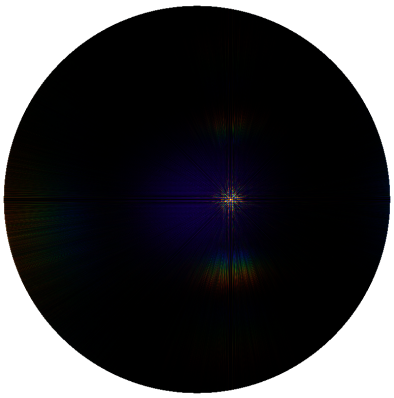
(a) Full Lambda Sampling: Xeno grating $\theta_i = 0 degree$



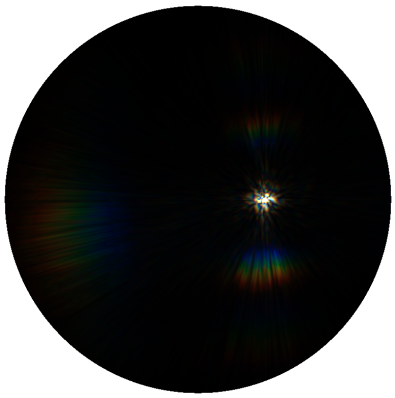
(b) PQ Approach: Xeno grating $\theta_i = 0 degree$



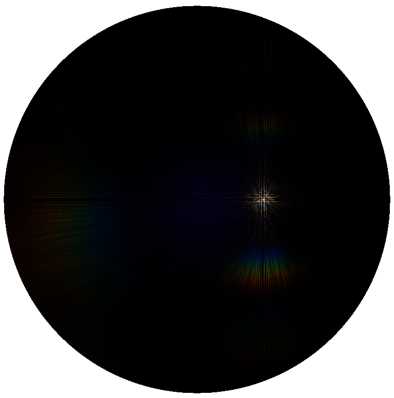
(c) Full Lambda Sampling: Xeno grating $\theta_i = 10 degree$



(d) PQ Approach: Xeno grating $\theta_i = 10 degree$



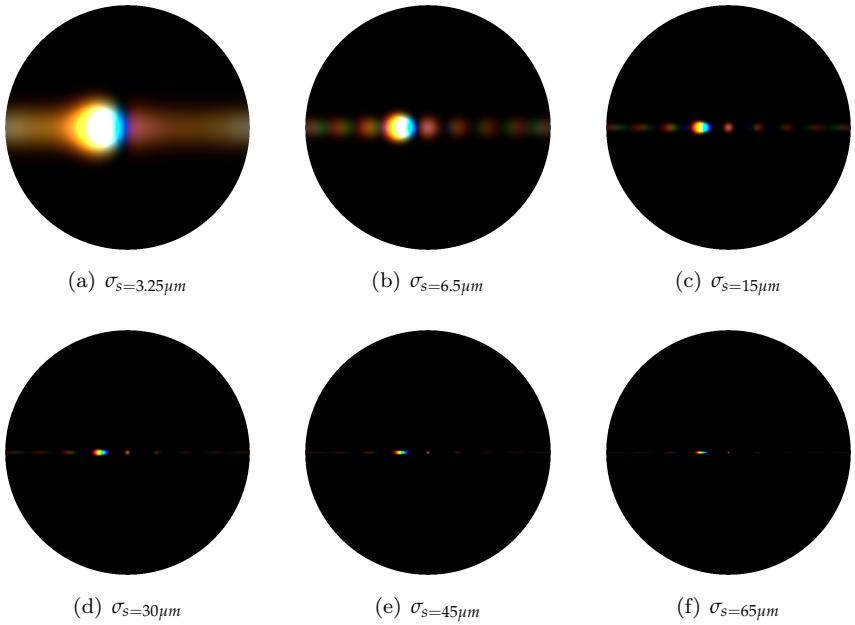
(e) Full Lambda Sampling: Xeno grating $\theta_i = 20 degree$



(f) PQ Approach: Xeno grating $\theta_i = 20 degree$

Figure 6.9: Xeno grating: PQ approach vs full lambda space sampling

Figure 6.10 shows BRDF map for the full lambda sampling approach applied on the Blazed grating, varying the value for the spacial variance σ_s. This is similar to consider the output of different coherence lengths for our incident light. The lower the coherence length, the fewer interacting grating periods produce produce blurred diffraction bands for different λ which overlap to produce poorly resolved colors.

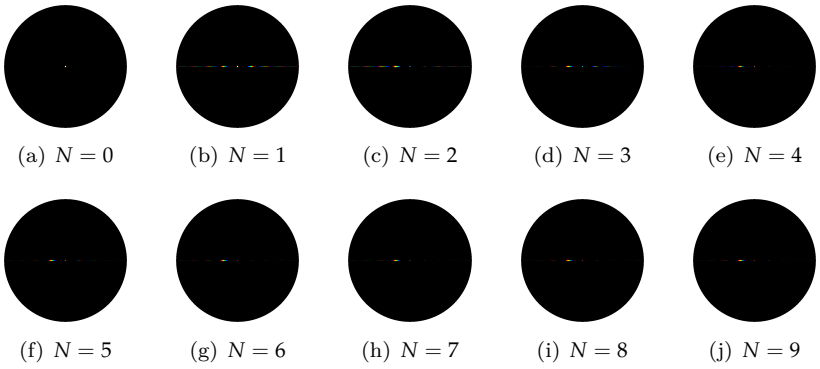


(a) $\sigma_{s=3.25\mu m}$ (b) $\sigma_{s=6.5\mu m}$ (c) $\sigma_{s=15\mu m}$

(d) $\sigma_{s=30\mu m}$ (e) $\sigma_{s=45\mu m}$ (f) $\sigma_{s=65\mu m}$

Figure 6.10: Blazed grating at $2.5\mu m$: Different σ_s sizes

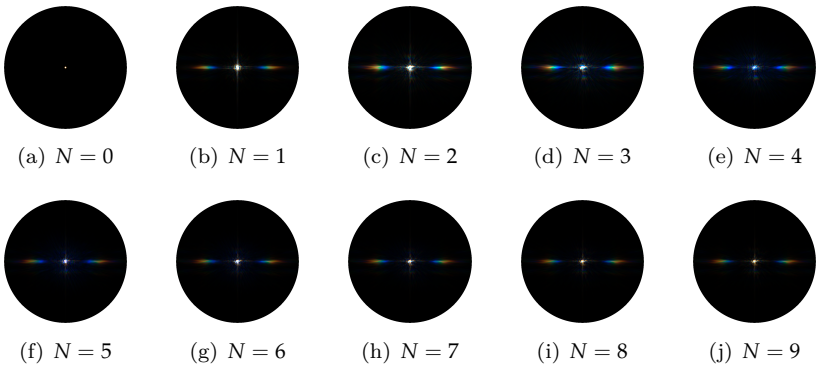
The figures 6.11 and 6.12 show the BRDF maps of the full lambda space approach using different values for N used for the taylor series approximation used within our fragment shaders. For both input patches we clearly visually observe the convergence of the taylor series for higher values for N. We visually observe convergence of the Taylor series for all our from a certain value of N. In section 3.4 we already provided a proof for the convergence for a specific geometrical set up which could be applied easily to another setup. The higher the value of N the less the BRDF map changes. For the Blazed grating, we do not see much changing patterns for $N \geq 7$ and for the Elaphe surface grating $N \geq 9$. In general it is rather hard to say for what value of N we have visual convergence since in algorithm 1 we compute the inverse FFT of $power(1j * patchImg, t)$ which is equivalent to $iFFT(patchImg)^n \cdot i^n$. Since we multiply this by i^n there are four possible series where each converges within its own convergence radius.



(a) $N = 0$ (b) $N = 1$ (c) $N = 2$ (d) $N = 3$ (e) $N = 4$

(f) $N = 5$ (g) $N = 6$ (h) $N = 7$ (i) $N = 8$ (j) $N = 9$

Figure 6.11: Blazed grating at $2.5\mu m$: N Taylor Iterations

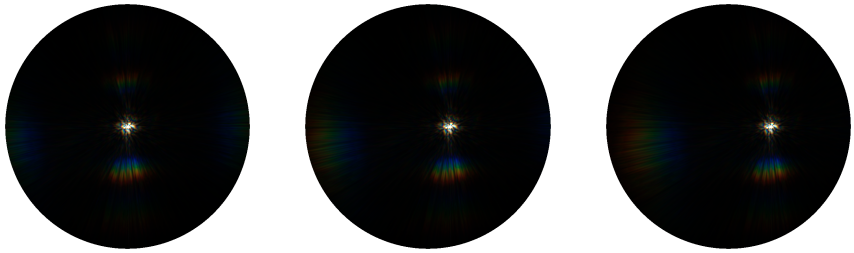


(a) $N = 0$ (b) $N = 1$ (c) $N = 2$ (d) $N = 3$ (e) $N = 4$

(f) $N = 5$ (g) $N = 6$ (h) $N = 7$ (i) $N = 8$ (j) $N = 9$

Figure 6.12: Elaphe grating at $65\mu m$: N Taylor Iterations

Figure 6.13 shows the BRDF maps of the full lambda shading approach applied on the Xenopeltis snake shed, using different θ_i incident angles. When slightly moving the incident angle θ_i, we can observe how the brdf map changes. For higher values of θ_i we start seeing diffraction color contribution on the right side of the BRDF map.



(a) Xeno grating $\theta_i = 0$ (b) Xeno grating $\theta_i = 10$ (c) Xeno grating $\theta_i = 20$

Figure 6.13: BRDF maps for Xeno grating: different θ_i angles

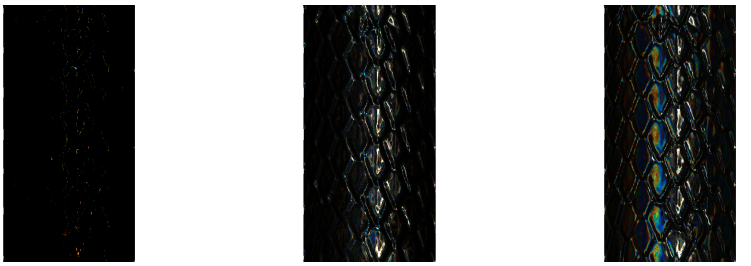
6.2 Snake surface geometries

In this section we are going to present our actual renderings simulating the effect of diffraction caused when a directional light source encounters different nano-scaled surfaces on a given curved snake mesh. We will see that diffraction colors change dramatically with changes in light direction, surface normals and viewing direction, which s typical for diffraction colors observed in nature. For rendering we are going to rely on our full wavelength space sampling approach. Unfortunately, this approach is rather slow and can barely be considered as being interactive performing. Nevertheless, we have introduced some optimizations in oder to become interactive in rendering, such as the N_{min}, N_{max} approach, we are going to use this slow approach since this resembles the ground truth and therefore is the most accurate among all our presented approaches. As mentioned we are going to render diffraction on a given snake mesh. Note that we actually just have one particular mesh, for all our renderings we are going to use the same snake mesh which has been produced by 3d scanning an Elpahe snake species, consisting of 11696 vertices and 22950 faces. The reason for that is that it was hard to get a Xenopeltis snake ready for being 3d scanned. In addition, the micro-geometry is highly similar among snake species, it is the geometry of the nano-structures that are highly different among species and that cause the snake to be or not be iridescent. So, even Xenopeltis would not give you very different geometry than Elaphe. Table 6.1 lists the system specifications of the machine I used in order to produce the rendered images.

Processor	Intel i7 CPU 970 @ 3.20 GHz (12 CPUs)
Memory	12288 MB RAM
Graphics Card	GeForce GTX 770

Table 6.1: Hardware specifications of the machine which produced rendered results. Statistics are provided using the tool NVIDIA Geforce Experience.

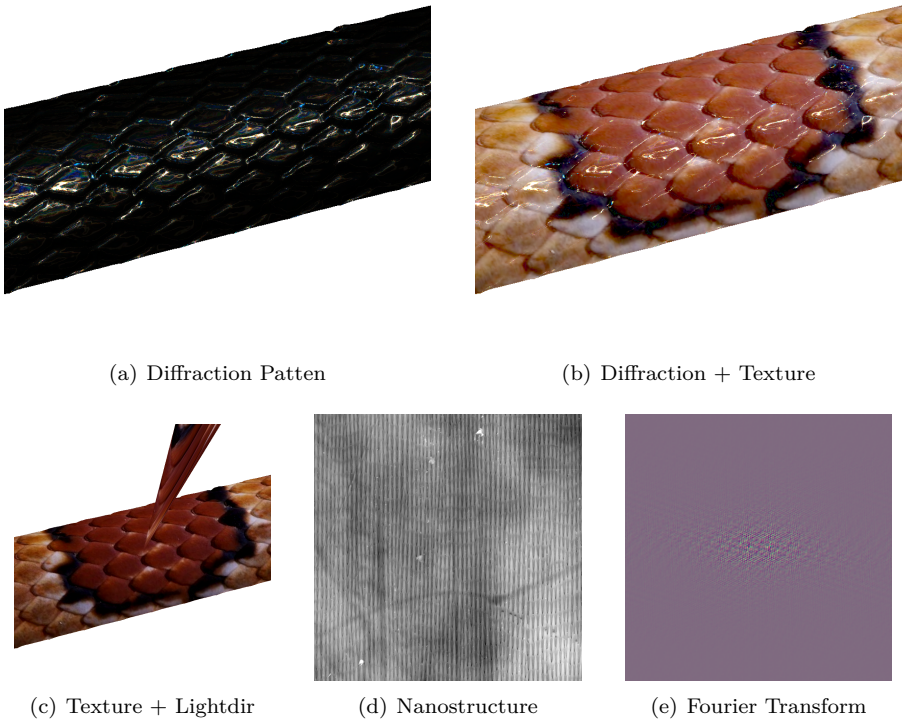
Figure 6.14 shows renderings produced by the full lambda sampling approach applied on a snake shaped mesh for different given input patches. Due do the fact that a Blazed grating has its maximum intensity for a certain direction and the geometry of the snake mesh is curved which means non-flat, we can expect rather less diffraction color contribution like shown in figure 6.14(b). Differently for our other two gratings, Elaphe and Xenopeltis. For both renderings, we can see color contribution despite the effect of diffraction whereas we see much less colorful patterns for Elpahe 6.14(b) than for Xenopeltis 6.14(c). This also corresponds to the reality, considering the figure 1.1 as a reference. The nano-scaled surface structure of the species Elapse as shwon in figure 6.2(b) does not look that regular under the electron scanning microscope. This is why it is much less iridescent than the other specie. Xeno has a brownish body with no pattern that makes the iridescence more spectacular than on Ellaphe.



(a) Blazed grating (b) Elaphe grating (c) Xeno grating

Figure 6.14: Diffraction of different snake skin gratings rendered on a snake geometry

Figure 6.15 contains a summary-collection of subfigures for rendering the effect of diffraction produced by the full lambda shading approach with all its required components, applied on our snake mesh, using the Elaphe nano-scaled surface structure. Subfigure 6.15(b) shows the final diffraction color-contribution result with texture-blending. We only see little diffraction color contribution in this subfigure which resembles quite well to the reality as shown in figure 1.1(b). In subfigure 6.15(d) we see the light cone in order to show the direction of the light source besides the rendered results. Subfigure 6.15(e) is a sample Fourier image of Elpahe's nanosclae surface structure 6.15(d).

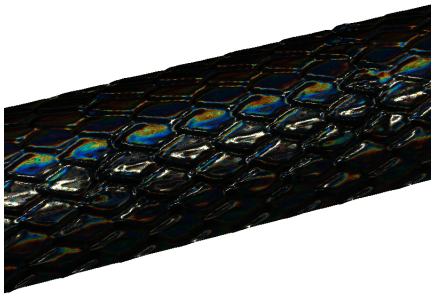


(a) Diffraction Patten (b) Diffraction + Texture

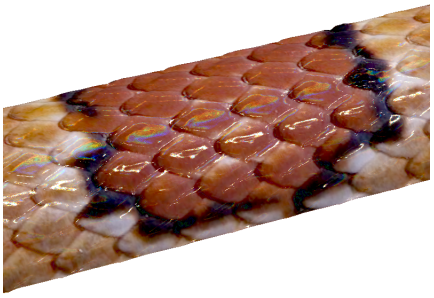
(c) Texture + Lightdir (d) Nanostructure (e) Fourier Transform

Figure 6.15: Diffraction for Elaphe snake skin

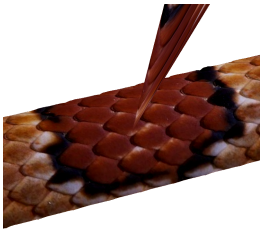
Like in the previous figure this figure 6.16 also shows a summary-collection of subfigures for the effect of diffraction with all its involved components but this time for the Xenopeltis snake surface. For texture blending we use the same texture like we used for Elaphe. For Xenopeltis see quite a lot color contribution due the phenomenon of diffraction like shown in figure 6.16(b). Comparing this to a real image 1.1(a) we notice much resemblance regarding the reflectance strength and colorful pattern.



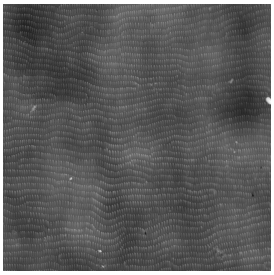
(a) Diffraction Patten



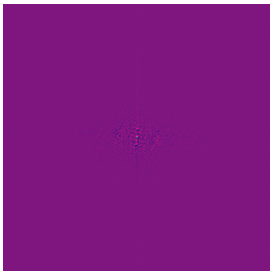
(b) Diffraction + Texture



(c) Texture + Lightdir



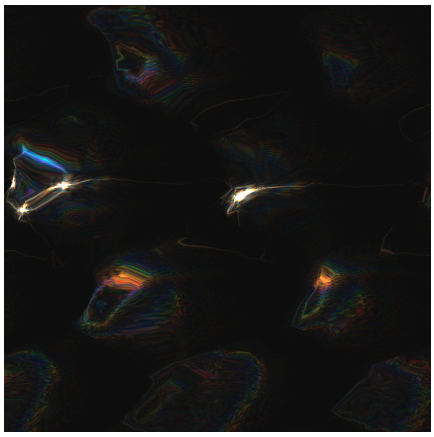
(d) Nanostructure



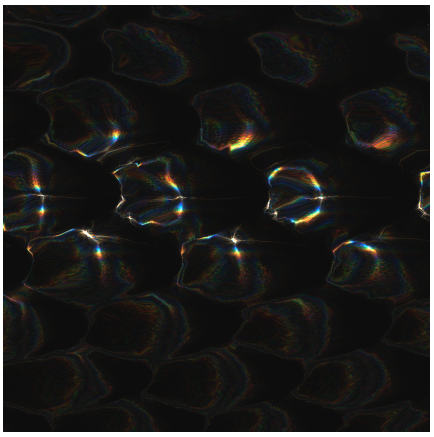
(e) Fourier Transform

Figure 6.16: Diffraction for Xeno snake skin

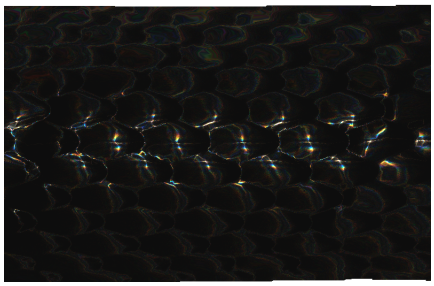
Figure 6.17 shows the diffraction pattern of a Elaphe snake shed for different zoom levels for fixed incident light and viewing direction using the full lambda sampling approach. From those different close up perspectives it would appear the complexity of the colorful diffraction pattern.



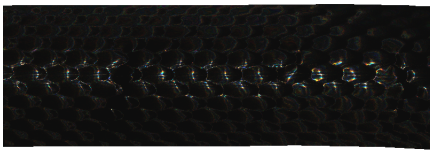
(a) $zoom = 0.1$



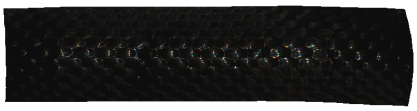
(b) $zoom = 0.2$



(c) $zoom = 0.5$



(d) $zoom = 1.0$

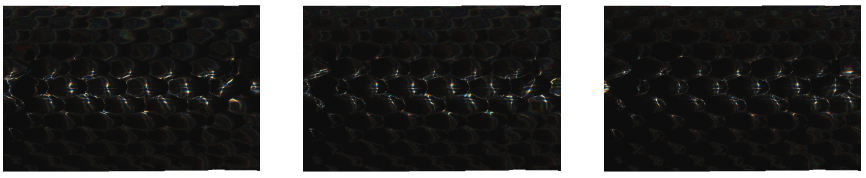


(e) $zoom = 1.5$

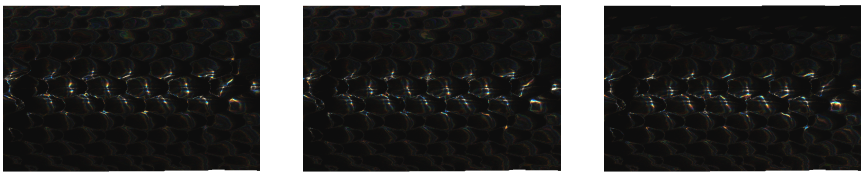
(f) $zoom = 2.0$

Figure 6.17: Diffraction on Elaphe snake skin grating: Different camera zoom levels

Figure 6.18 shows how the diffraction pattern changes when slightly moving the incident light direction. Which gives us an impression what kind of complex, perspective-dependent pattern the phenomenon of diffraction may cause.



(a) $(-3.3130, 0.0, -0.9999)$ (b) $(-0.1989, 0.0, -0.9799)$ (c) $(-0.3897, 0.0, -0.9208)$



(d) $(0.0995, 0.0993, -0.9900)$ (e) $(0.0995, 0.2940, -0.9505)$ (f) $(0.0995, 0.4770, -0.8731)$

Figure 6.18: Diffraction on Elaphe snake skin grating: Different light directions

Figure 6.19 shows a photo of an experimental setup for demonstrating the effect of diffraction using a Elpahe snake grating. The exact parameters for the experimental setup are unknown. Nevertheless this image gives us an impression of how close our model is to the reality comparing it with our simulated results since we notice similar diffraction patterns for our simulated results using an Elaphe snake shed.

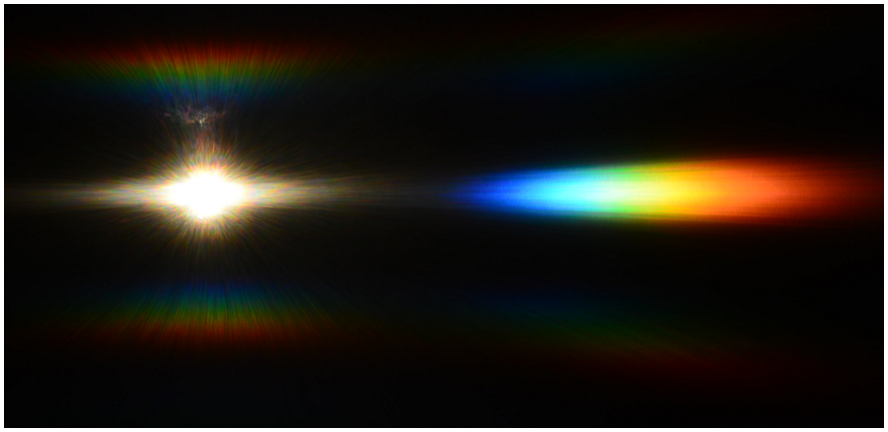


Figure 6.19: Diffraction Elaphe: experimental setup

Chapter 7

Conclusion

7.1 Review

The goal of this thesis was to simulate the effect of diffraction caused when a directional light source encounters an anisotropic surface, when provided explicitly by a small patch of this nano-scaled surface. For this purpose we developed a BRDF model which is based on J. Stam's main derivation described within his paper about diffraction shaders.

Rendering the effect of diffraction for explicitly provided surfaces, described as a height-field, was previously not possible and may therefore be considered as a kind of novelty.

Our formulation allows us to precompute many computational expensive terms, such as different powers of the two dimensional inverse Fourier transformation of the given input patch, and then combine them within our shader relying on Taylor series. This Taylor series approximation allows us to enhance the overall runtime complexity of our shading approach quite a lot, compared to Stam's formulation. Furthermore, we also introduced some other numerical approximations for our BRDF formulations, such as the N_{min}, N_{max} shading approach, which will tweak the overall runtime complexity of the whole shading.

We also formulated a complete different approach, the PQ shading approach, which is performing Sinc-Interpolation, assuming the given patch is periodically distributed on the surface of a given geometry.

We have implemented a Java renderer which is using GLSL shaders, implementing our BRDF models. We evaluated the quality of all our shading approaches for our given different provided surface gratings. Last we rendered BRDF maps and the effect of diffraction on a snake mesh for Blazed, Elaphe, Xenopeltis grating for various input parameters and discussed all results in depth.

We some approximation techniques we achieved all goals of this thesis. Even our shader is rather slow, we can denote it as being interactive, when using the N_{min}, N_{max} shading approach for our used hardware specifications. Note that when we are sampling the whole wavelength space, which is the most accurate variant among all our shading approaches, we barely can be denoted as being interactive at all. Despite this variant of shading is based in a gaussian windowing approach which is basically weighting each pixel by its neighborhood and also integrates over each wavelength of our spectrum, it has such a high runtime complexity. Nevertheless, this kind of shading produces quite reliable results, considering its evaluation plots and its actual renderings. The PQ shading approach is rather far apart from reality since it assumes the given patch being a representative of the whole surface due to fact being distributed periodically along the surface in each direction. It also is a rather slow shading approach since Sinc-Interpolation is iterating over the whole neighborhood of each pixel.

A non-physical and also non-mathematical sound hack is to combine the PQ approach with the N_{min}, N_{max} approach but not using Sinc-Interpolation. This would be like linearly interpolating in the Fourier domain, using the PQ factors, using no window and also a reduced spectrum. This approach sometimes produces good looking results but they are quite non-reliable.

There is a further optimization possible described in *PAPER* by further simplifying our BRDF formulation. These simplifications allow to precompute further computational expensive terms.

7.2 Personal Experiences

I always knew that eventually I will write a thesis in the field of computer graphics. After having taking the computer graphics class held by Mr. Zwicker, I was sure that i will definitely write my thesis at the computer Graphics Group at the University of Bern. Since I already acquired a Minor and am very interested in the field of Mathematics, I asked for getting a topic involving also some Mathematics. I have to admit that, after having read the Paper of J. Stam about Diffraction shaders, in the very beginning of my thesis, I thought this topic would exceed my knowledge in Mathematics and Physics. And It actually did but i did not give up. Despite of this thesis I could acquire new and even deepen already existent knowledge in concepts of Physics such as waves theory, wave-interference, diffraction of waves and diffraction gratings. Regarding Mathematics I learned quite a lot about the different kind of Fourier transforms and about their differences, about windowing approaches and about BRDF models and formulations and the numerics regarding those topics. It was a satisfactory experience to use and apply all the knowledge which I have acquired during the time as a bachelor student. Also this work gave me the opportunity to program quite a lot what I really liked and allowed me to strengthen my programming skills in java and OpenGL's shading language GLSL. Altogether it was a rewarding experience for me to write a bachelor thesis at the Computer Graphics group.

7.3 Acknowledgment

First I would like to thank Mr. Zwicker for giving me the opportunity to write a bachelor thesis at the computer Graphics Group at the University of Bern.

Foremost, I would like to express my sincere gratitude to my advisor Mr. Daljit Singh for the continuous support of my study, for his patience, motivation, enthusiasm, and knowledge. His guidance and active support helped me quite a lot deriving the BRDF formulation, during developing and evaluating the shaders and writing this thesis.

Last but not the least, I would like to thank my mother and brother, Manuela and Patrik Single and my close friend Radischa Iyadurai for supporting me mentally and spiritually throughout during this thesis.

Appendix A

Signal Processing Basics

A signal is a function that conveys information about the behavior or attributes of some phenomenon. In the physical world, any quantity exhibiting variation in time or variation in space (such as an image) is potentially a signal that might provide information on the status of a physical system, or convey a message between observers.

The Fourier Transform is an important image processing tool which is used to decompose an image into its sine and cosine components. The output of the transformation represents the image in the Fourier or frequency domain, while the input image is the spatial domain equivalent. In the Fourier domain image, each point represents a particular frequency contained in the spatial domain image.

A.1 Fourier Transformation

The Fourier-Transform is a mathematical tool which allows to transform a given function or rather a given signal from defined over a time- (or spatial-) domain into its corresponding frequency-domain.

Let f an measurable function over $\mathbb{R}^n$. Then, the continuous Fourier Transformation(**FT**), denoted as $\mathcal{F}\{f\}$ of f, ignoring all constant factors in the formula, is defined as:

$$\mathcal{F}_{FT}\{f\}(w) = \int_{\mathbb{R}^n} f(x)e^{-iwt}dt \tag{A.1}$$

whereas its inverse transform is defined like the following which allows us to obtain back the original signal:

$$\mathcal{F}^{-1}_{FT}\{f\}(w) = \int_{\mathbb{R}} \mathcal{F}\{w\}e^{iwt}dt \tag{A.2}$$

Usual w is identified by the angular frequency which is equal $w = \frac{2\pi}{T} = 2\pi v_f$. In this connection, T is the period of the resulting spectrum and v_f is its corresponding frequency.

By using Fourier Analysis, which is the approach to approximate any function by sums of simpler trigonometric functions, we gain the so called Discrete Time Fourier Transform (in short **DTFT**). The DTFT operates on a discrete function. Usually, such an input function is often created by digitally sampling a continuous function. The DTFT itself is operation on a discretized signal on a continuous, periodic frequency domain and looks like the following:

$$\mathcal{F}_{DTFT}\{f\}(w) = \sum_{-\infty}^{\infty} f(x)e^{-iwk} \tag{A.3}$$

Note that the DTFT is not practically suitable for digital signal processing since there a signal can be measured only in a finite number of points. Thus, we can further discretize the frequency domain and will get then the Discrete Fourier Transformation (in short **DFT**) of the input signal:

$$\mathcal{F}_{DFT}\{f\}(w) = \sum_{n=0}^{N-1} f(x)e^{-iw_n k} \tag{A.4}$$

Where the angular frequency w_n is defined like the following $w_n = \frac{2\pi n}{N}$ and N is the number of samples within an equidistant period sampling.

Any continuous function $f(t)$ can be expressed as a series of sines and cosines. This representation is called the Fourier Series (denoted by FS) of $f(t)$.

$$f(t) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty} a_n cos(nt) + \sum_{n=1}^{\infty} b_n cos(nt) \tag{A.5}$$

where

$$\begin{aligned} a_0 &= \int_{-\pi}^{\pi} f(t)dt \\ a_n &= \frac{1}{\pi}\int_{-\pi}^{\pi} f(t)cos(nt)dt \\ b_n &= \frac{1}{\pi}\int_{-\pi}^{\pi} f(t)sin(nt)dt \end{aligned} \tag{A.6}$$

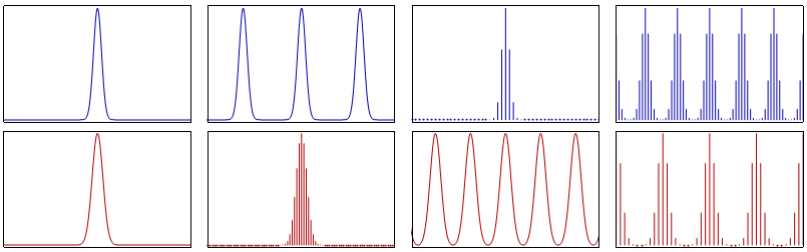


Figure A.1: Relationship[1] between the continuous Fourier transform and the discrete Fourier transform: Left column: A continuous function (top) and its Fourier transform $A.1$ (bottom). Center-left column: Periodic summation of the original function (top). Fourier transform (bottom) is zero except at discrete points. The inverse transform is a sum of sinusoids called Fourier series $A.5$. Center-right column: Original function is discretized (multiplied by a Dirac comb) (top). Its Fourier transform (bottom) is a periodic summation (DTFT) of the original transform. Right column: The DFT $A.4$ (bottom) computes discrete samples of the continuous DTFT $A.3$. The inverse DFT (top) is a periodic summation of the original samples.

[1] image of illustration has been taken from wikipedia

Spetail signal $f(t)$ is	Operator	Transformed frequency signal $\hat{f}(\omega)$ is
continuous and periodic in t	FS $A.5$	only discrete in ω
only continuous in t	FT $A.1$	only continuous in ω
only discrete in t	DTFT $A.3$	continuous and periodic in ω
discrete and periodic in t	DFT $A.4$	discrete and periodic in ω

Table A.1: Fourier operator to apply for a given spatial input signal and the properties of its resulting output signal in frequency space

A.2 Convolution

The convolution $f * g$ of two functions $f, g\colon \mathbb{R}^n \to \mathbb{C}$ is defined as:

$$(f * g)(t) = \int_{\mathbb{R}^n} f(t)g(t-x)dx \tag{A.7}$$

Note that the Fourier transform of the convolution of two functions is the product of their Fourier transforms. This is equivalent to the fact that Convolution in spatial domain is equivalent to multiplication in frequency domain. Therefore, the inverse Fourier transform of the product of two Fourier transforms is the convolution of the two inverse Fourier transforms. Last an illustration of the relationships between the previous presented Fourier transformations and different given input signals. First an concrete example shown in Figure $A.1$. Table $A.1$ tells what Fourier transformation operator has to be applied to which kind of input signal and what properties its resulting Fourier transform will have.

A.3 Taylor Series

Taylor series is a representation of a function as an infinite sum of terms that are calculated from the values of the function's derivatives at a single point.

The Taylor series $\mathcal{T}$ of a real or complex-valued function $f(x)$ that is infinitely differentiable at a real or complex number a is the power series:

$$\mathcal{T}(f;a)(x) = \sum_{n=0}^{\infty} \frac{f^n(a)}{n!}(x-a)^n \tag{A.8}$$

Appendix B

Summary of Stam's Derivations

In his paper about Diffraction Shader, J. Stam derives a BRDF which is modeling the effect of diffraction for various analytical anisotropic reflexion models relying on the so called scalar wave theory of diffraction for which a wave is assumed to be a complex valued scalar. It's noteworthy, that Stam's BRDF formulation does not take into account the polarization of the light. Fortunately, light sources like sunlight and light bulbs are unpolarized.

A further assumption in Stam's Paper is, the emanated waves from the source are stationary, which implies the wave is a superposition of independent monochromatic waves. This implies that each wave is associated to a definite wavelength lambda. However, sunlight once again fulfills this fact.

In our simulations we will always assume we have given a directional light source, i.e. sunlight. Hence, Stam's model can be used for our derivations.

For his derivations Stam uses the Kirchhoff integral[1], which is relating the reflected field to the incoming field. This equation is a formalization of Huygen's well-known principle that states that if one knows the wavefront at a given moment, the wave at a later time can be deduced by considering each point on the first wave as the source of a new disturbance. Mathematically speaking, once the field $\psi_1 = e^{ik\mathbf{x}\cdot\mathbf{ss}}$ on the surface is known, the field ψ_2 everywhere else away from the surface can be computed. More precisely, we want to compute the wave ψ_2 equal to the reflection of an incoming planar monochromatic wave $\psi_1 = e^{ik\omega_i * x}$ traveling in the direction ω_i from a surface S to the light source. Formally, this can be written as:

$$\psi_2(\omega_i, \omega_r) = \frac{ike^{iKR}}{4\pi R}(F(-\omega_i - \omega_r) - (-\omega_i + \omega_r)) \cdot I_1(\omega_i, \omega_r) \tag{B.1}$$

with

$$I_1(\omega_i, \omega_r) = \int_S \hat{\mathbf{n}} e^{ik(-\omega_i - \omega_r)\cdot\mathbf{s}} d\mathbf{s} \tag{B.2}$$

In applied optics, when dealing with scattered waves, one does use differential scattering cross-section rather than defining a BRDF which has the following identity:

$$\sigma^0 = 4\pi \lim_{R\to\infty} R^2 \frac{\langle|\psi_2|^2\rangle}{\langle|\psi_1|^2\rangle} \tag{B.3}$$

where R is the distance from the center of the patch to the receiving point x_p, $\hat{\mathbf{n}}$ is the normal of the surface at s and the vectors:

[1] See `http://en.wikipedia.org/wiki/Kirchhoff_integral_theorem` for further information.

The relationship between the BRDF and the scattering cross section can be shown to be equal to

$$BRDF = \frac{1}{4\pi}\frac{1}{A}\frac{\sigma^0}{cos(\theta_i)cos(\theta_r)} \tag{B.4}$$

where θ_i and θ_r are the angles of incident and reflected directions on the surface with the surface normal n. See 2.13.

The components of vector resulting by the difference between these direction vectors: In order to simplify the calculations involved in his vectorized integral equations, Stam considers the components of vector

$$(u, v, w) = -\omega_i - \omega_r \tag{B.5}$$

explicitly and introduces the equation:

$$I(ku, kv) = \int_S \hat{\mathbf{n}} e^{ik(u,v,w)\cdot \mathbf{s}} d\mathbf{s} \tag{B.6}$$

which is a first simplification of $B.2$. Note that the scalar w is the third component of 2.5 and can be written as $w = -(cos(\theta_i) + cos(\theta_r))$ using spherical coordinates. The scalar $k = \frac{2\pi}{\lambda}$ represent the wavenumber.

During his derivations, Stam provides a analytical representation for the Kirchhoff integral assuming that each surface point $s(x, y)$ can be parameterized by $(x, y, h(x, y))$ where h is the height at the position (x, y) on the given (x, y) surface plane. Using the tangent plane approximation for the parameterized surface and plugging it into $B.6$ he will end up with:

$$\mathbf{I}(ku, kv) = \int\int (-h_x(x, y), -h_y(x, y), 1) e^{ikwh(x,y)} e^{ik(ux+vy)} dxdy \tag{B.7}$$

For further simplification Stam formulates auxillary function which depends on the provided height field:

$$p(x, y) = e^{iwkh(x,y)} \tag{B.8}$$

which will allow him to further simplify his equation $B.7$ to:

$$\mathbf{I}(ku, kv) = \int\int \frac{1}{ikw}(-p_x, -p_y, ikwp) dxdy \tag{B.9}$$

where he used that $(-h_x(x, y), -h_y(x, y), 1)e^{kwh(x,y)}$ is equal to $\frac{(-p_x, -p_y, ikwp)}{ikw}$ using the definition of the partial derivatives applied to the function 2.7.

Let $P(x, y)$ denote the Fourier Transform (FT) of $p(x, y)$. Then, the differentiation with respect to x respectively to y in the Fourier domain is equivalent to a multiplication of the Fourier transform by $-iku$ or $-ikv$ respectively. This leads him to the following simplification for $B.7$:

$$\mathbf{I}(ku, kv) = \frac{1}{w} P(ku, kv) \cdot (u, v, w) \tag{B.10}$$

Let us consider the term $g = (F(-\omega_i - \omega_r) - (-\omega_i + \omega_r))$, which is a scalar factor of $B.1$. The dot product with g and $(-\omega_i - \omega_r)$ is equal $2F(1 + \omega_i \cdot \omega_r)$. Putting this finding and the identity $B.10$ into $B.1$ he will end up with:

$$\psi_2(\omega_i, \omega_r) = \frac{ike^{iKR}}{4\pi R}\frac{2F(1 + \omega_i \cdot \omega_r)}{w} P(ku, kv) \tag{B.11}$$

By using the identity *B.4*, this will lead us to his main finding:

$$BRDF_\lambda(\omega_i, \omega_r) = \frac{k^2 F^2 G}{4\pi^2 A w^2} \langle |P(ku, kv)|^2 \rangle \tag{B.12}$$

where G is the so called geometry term which is equal:

$$G = \frac{(1 + \omega_i \cdot \omega_r)^2}{cos(\theta_i) cos(\theta_r)} \tag{B.13}$$

Appendix C

Derivation Steps in Detail

C.1 Taylor Series Approximation

For an $N \in \mathbb{N}$ such that

$$\sum_{n=0}^{N} \frac{(ikwh)^n}{n!} \mathcal{F}\{h^n\}(\alpha, \beta) \approx P(\alpha, \beta) \tag{C.1}$$

we have to prove:

1. Show that there exist such an $N \in \mathbb{N}$s.t the approximation holds true.
2. Find a value for B s.t. this approximation is below a certain error bound, for example machine precision ϵ.

C.1.1 Proof Sketch of 1.

By the **ratio test** (see [1]) It is possible to show that the series $\sum_{n=0}^{N} \frac{(ikwh)^n}{n!} \mathcal{F}\{h^n\}(\alpha, \beta)$ converges absolutely:

Proof: Consider $\sum_{k=0}^{\infty} \frac{y^n}{n!}$ where $a_k = \frac{y^k}{k!}$. By applying the definition of the ratio test for this series it follows:

$$\forall y : limsup_{k\to\infty} |\frac{a_{k+1}}{a_k}| = limsup_{k\to\infty} \frac{y}{k+1} = 0 \tag{C.2}$$

Thus this series converges absolutely, no matter what value we will pick for y.

C.1.2 Part 2: Find such an N

Let $f(x) = e^x$. We can formulate its Taylor-Series, stated above. Let $P_n(x)$denote the n-th Taylor polynom,

$$P_n(x) = \sum_{k=0}^{n} \frac{f^{(k)}(a)}{k!}(x-a)^k \tag{C.3}$$

where a is our developing point (here a is equal zero).

We can define the error of the n-th Taylor polynom to be $E_n(x) = f(x) - P_n(x)$. the error of the n-th Taylor polynom is difference between the value of the function and the Taylor polynomial This directly implies $|E_n(x)| = |f(x) - P_n(x)|$. By using the Lagrangian Error Bound it follows:

$$|E_n(x)| \leq \frac{M}{(n+1)!}|x-a|^{n+1} \tag{C.4}$$

with $a = 0$, where **M** is some value satisfying $|f^{(n+1)}(x)| \leq M$ on the interval $I = [a, x]$. Since we are interested in an upper bound of the error and since **a** is known, we can reformulate the interval as $I = [0, x_{max}]$, where

$$x_{max} = \|i\| k_{max} w_{max} h_{max} \tag{C.5}$$

We are interested in computing an error bound for $e^{ikwh(x,y)}$. Assuming the following parameters and facts used within Stam's Paper:

- Height of bump: 0.15micro meters
- Width of a bump: 0.5micro meters
- Length of a bump: 1micro meters
- $k = \frac{2\pi}{\lambda}$ is the wavenumber, $\lambda \in [\lambda_{min,}\lambda_{max}]$ and thus $k_{max} = \frac{2\pi}{\lambda_{min}}$. Since $(u, v, w) = -\omega_i - \omega_r$ and both are unit direction vectors, each component can have a value in range [-2, 2].
- for simplification, assume$[\lambda_{min,}\lambda_{max}] = [400nm, 700nm]$.

We get:

$$\begin{aligned} x_{max} &= \|i\| * k_{max} * w_{max} * h_{max} \\ &= k_{max} * w_{max} * h_{max} \\ &= 2 * (\frac{2\pi}{4 * 10^{-7} m}) * 1.5 * 10^{-7} \\ &= 1.5\pi \end{aligned} \tag{C.6}$$

and it follows for our interval $I = [0, 1.5\pi]$.

Next we are going to find the value for M. Since the exponential function is monotonically growing (on the interval I) and the derivative of the **exp** function is the exponential function itself, we can find such an M:

$$\begin{aligned} M &= e^{x_{max}} \\ &= exp(1.5\pi) \end{aligned}$$

and $|f^{(n+1)}(x)| \leq M$ holds. With

$$\begin{aligned} |E_n(x_{max})| &\leq \frac{M}{(n+1)!}|x_{max} - a|^{n+1} \\ &= \frac{exp(1.5\pi) * (1.5\pi)^{n+1}}{(n+1)!} \end{aligned} \tag{C.7}$$

we now can find a value of n for a given bound, i.e. we can find an value of $N \in \mathbb{N}$ s.t. $\frac{exp(1.5\pi)*(1.5\pi)^{N+1}}{(N+1)!} \leq \epsilon$. With Octave/Matlab we can see:

- if N=20 then $\epsilon \approx 2.9950 * 10^{-4}$
- if N=25 then $\epsilon \approx 8.8150 * 10^{-8}$
- if N=30 then $\epsilon \approx 1.0050 * 10^{-11}$

With this approach we have that $\sum_{n=0}^{25} \frac{(ikwh)^n}{n!} \mathcal{F}\{h^n\}(\alpha, \beta)$ is an approximation of $P(u, v)$ with error $\epsilon \approx 8.8150 * 10^{-8}$. This means we can precompute 25 Fourier Transformations in order to approximate P(u,v) having an error $\epsilon \approx 8.8150 * 10^{-8}$.

C.2 PQ approach

C.2.1 One dimensional case

Since our series is bounded, we can simplify the right-hand-side of equation 3.38.

Note that e^{-ix} is a complex number. Every complex number can be written in its polar form, i.e.

$$e^{-ix} = cos(x) + isin(x) \tag{C.8}$$

Using the following trigonometric identities

$$\begin{aligned} cos(-x) &= cos(x) \\ sin(-x) &= -sin(x) \end{aligned} \tag{C.9}$$

combined with C.8 we can simplify the series 3.38 even further to:

$$\frac{1 - e^{iwT(N+1)}}{1 - e^{-iwT}} = \frac{1 - cos(wT(N+1)) + isin(wT(N+1))}{1 - cos(wT) + isin(wT)} \tag{C.10}$$

Equation C.10 is still a complex number, denoted as $(p + iq)$. Generally, every complex number can be written as a fraction of two complex numbers. This implies that the complex number $(p + iq)$ can be written as $(p + iq) = \frac{(a+ib)}{(c+id)}$ for any $(a + ib), (c + id) \neq 0$. Let us use the following substitutions:

$$\begin{aligned} a &:= 1 - cos(wT(N+1)) & b &= sin(wT(N+1)) \\ c &= 1 - cos(wT) & d &= sin(wT) \end{aligned} \tag{C.11}$$

Hence, using C.11, it follows

$$\frac{1 - e^{iwT(N+1)}}{1 - e^{-iwT}} = \frac{(a + ib)}{(c + id)} \tag{C.12}$$

By rearranging the terms, it follows $(a + ib) = (c + id)(p + iq)$ and by multiplying its right hand-side out we get the following system of equations:

$$\begin{aligned} (cp - dq) &= a \\ (dp + cq) &= b \end{aligned} \tag{C.13}$$

After multiplying the first equation of C.13 by c and the second by d and then adding them together, we get using the law of distributivity new identities for p and q:

$$\begin{aligned} p &= \frac{(ac+bd)}{c^2+d^2} \\ q &= \frac{(bc+ad)}{c^2+d^2} \end{aligned} \tag{C.14}$$

Using some trigonometric identities and putting our substitution from C.11 for a, b, c, d back into the current representation C.14 of p and q we will get:

$$\begin{aligned} p &= \frac{1}{2} + \frac{1}{2}\left(\frac{cos(wTN) - cos(wT(N+1))}{1-cos(wT)}\right) \\ q &= \frac{sin(wT(N+1)) - sin(wTN) - sin(wT)}{2(1-cos(wT))} \end{aligned} \tag{C.15}$$

Since we have seen, that $\sum_{n=0}^{N} e^{-uwnT}$ is a complex number and can be written as $(p+iq)$, we now know an explicit expression for p and q. Therefore, the one dimensional inverse Fourier transform of S is equal:

$$\begin{aligned} \mathcal{F}^{-1}\{S\}(w) &= \mathcal{F}^{-1}\{f\}(w) \sum_{n=0}^{N} e^{-iwnT} \\ &= (p+iq)\mathcal{F}^{-1}\{f\}(w) \end{aligned} \tag{C.16}$$

C.2.2 Two dimensional case

$$\begin{aligned} \mathcal{F}^{-1}\{S\}(w_1,w_2) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \sum_{n_2=0}^{N_1}\sum_{n_2=0}^{N_2} h(x_1+n_1T_1, x_2+n_2T_2) e^{iw(x_1+x_2)} dx_1 dx_2 \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \sum_{n_2=0}^{N_1}\sum_{n_2=0}^{N_2} h(y_1,y_2) e^{iw((y_1-n_1T_1)+(y_2+n_2T_2))} dx_1 dx_2 \\ &= \sum_{n_2=0}^{N_1}\sum_{n_2=0}^{N_2} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(y_1,y_2) e^{iw(y_1+y_2)} e^{-iw(n_1T_1+n_2T_2)} dy_1 dy_2 \\ &= \sum_{n_2=0}^{N_1}\sum_{n_2=0}^{N_2} e^{-iw(n_1T_1+n_2T_2)} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} Box(y_1,y_2) e^{iw(y_1+y_2)} dy_1 dy_2 \\ &= \left(\sum_{n_2=0}^{N_1}\sum_{n_2=0}^{N_2} e^{-iw(n_1T_1+n_2T_2)}\right) \mathcal{F}^{-1}\{h\}(w_1,w_2) \\ &= \left(\sum_{n_2=0}^{N_1} e^{-iwn_1T_1}\right)\left(\sum_{n_2=0}^{N_2} e^{-iwn_2T_2}\right) \mathcal{F}^{-1}\{h\}(w_1,w_2) \\ &= (p_1+iq_1)(p_2+iq_2)\mathcal{F}^{-1}\{h\}(w_1,w_2) \\ &= ((p_1p_2-q_1q_2)+i(p_1p_2+q_1q_2))\mathcal{F}^{-1}\{h\}(w_1,w_2) \\ &= (p+iq)\mathcal{F}_{DTFT}\{h\}(w_1,w_2) \end{aligned} \tag{C.17}$$

Where we have defined

$$\begin{aligned} p &:= (p_1 p_2 - q_1 q_2) \\ q &:= (p_1 p_2 + q_1 q_2) \end{aligned} \tag{C.18}$$

Appendix D

Miscellaneous Transformations

D.1 Fresnel Term - Schlick's approximation

The Fresnel's equations describe the reflection and transmission of electromagnetic waves at an interface. That is, they give the reflection and transmission coefficients for waves parallel and perpendicular to the plane of incidence. Schlick's approximation is a formula for approximating the contribution of the Fresnel term where the specular reflection coefficient R can be approximated by:

$$R(\theta) = R_0 + (1 - R_0)(1 - \cos\theta)^5 \tag{D.1}$$

and

$$R_0 = \left(\frac{n_1 - n_2}{n_1 + n_2}\right)^2$$

where θ is the angle between the viewing direction and the half-angle direction, which is halfway between the incident light direction and the viewing direction, hence $\cos\theta = (H \cdot V)$. And n_1, n_2 are the indices of refraction of the two medias at the interface and R_0 is the reflection coefficient for light incoming parallel to the normal (i.e., the value of the Fresnel term when $\theta = 0$ or minimal reflection). In computer graphics, one of the interfaces is usually air, meaning that n_1 very well can be approximated as 1.

D.2 Spherical Coordinates and Space Transformation

$\forall \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{R}^3 : \exists r \in [0, \infty) \exists \phi \in [0, 2\pi] \exists \theta \in [0, \pi]$ s.t.

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} rsin(\theta)cos(\phi) \\ rsin(\theta)sin(\phi) \\ rcos(\theta) \end{pmatrix}$$

From the definition 2.5 of $(u, v, w) = -\omega_i - \omega_r$ and using spherical coordinates $D.2$, we get for w the following identity:

$$\begin{aligned} w &= -\omega_i - \omega_r \\ &= -(\omega_i + \omega_r) \\ &= -\left(cos(\theta_i) + cos(\theta_r)\right) \end{aligned} \tag{D.2}$$

and therefore w^2 is equal $(cos(\theta_i) + cos(\theta_r))^2$.

D.3 Tangent Space

The concept of tangentspace-transformation of tangent space is used in order to convert a point between world and tangent space. GLSL fragment shaders require normals and other vertex primitives declared at each pixel point, which mean that we have one normal vector at each texel and the normal vector axis will vary for every texel.

Think of it as a bumpy suface defined on a flat plane. If those normals were declared in the world space coordinate system, we would have to rotate these normals every time the model is rotated, even when just for a small amount. Since the lights, cameras and other objects are usually defined in world space coordinate system, and therefore, when they are involved in an calculation within the fragment shader, we would to have to rotate them as well for every pixel. This would involve almost countless many object to world matrix transformations meed to take place at the pixel level. Therefore, instead doing so, we transform all vertex primitives into tangent space within the vertex shader.

To make this point clear an example: Even we would rotate the cube in figure $D.1$, the tangent space axis will remain aligned with respect to the face. Which practically speaking, will save us from performing many space transformations applied pixel-wise within the fragment shader and instead allows us to perform us the tangenspace transformation of every involved vertex primitive in the vertex-shader.

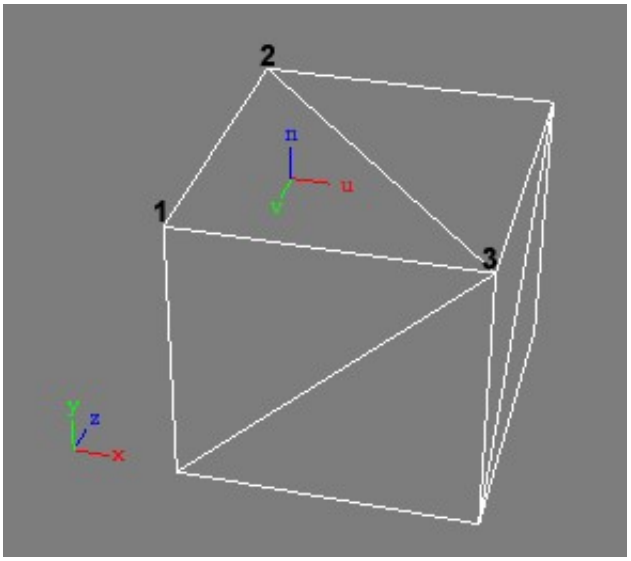


Figure D.1: Cube in world space (x, y, z) showing the tangen space (u, v, n) of its face $(2, 1, 3)$

List of Tables

List of Figures

List of Algorithms

Bibliography

[Bar07] Bartsch, Hans-Jochen: *Taschenbuch Mathematischer Formeln*. 21th edition. HASNER, 2007. – ISBN 978–3–8348–1232–2

[CT12] Cuypers T., et a.: Reflectance Model for Diffraction. In: *ACM Trans. Graph. 31, 5* (2012), September

[DSD14] D. S. Dhillon, et a.: Interactive Diffraction from Biological Nanostructures. In: *EUROGRAPHICS 2014/ M. Paulin and C. Dachsbacher* (2014), January

[For11] Forster, Otto: *Analysis 3*. 6th edition. VIEWEG+TEUBNER, 2011. – ISBN 978–3–8348–1232–2

[I.N14] I.Newton: *Opticks, reprinted*. CreateSpace Independent Publishing Platform, 2014. – ISBN 978–1499151312

[JG04] Juan Guardado, NVIDIA: Simulating Diffraction. In: *GPU Gems* (2004). `https://developer.nvidia.com/content/gpu-gems-chapter-8-simulating-diffraction`

[LM95] Leonard Mandel, Emil W.: *Optical Coherence and Quantum Optics*. Cambridge University Press, 1995. – ISBN 978–0521417112

[MT10] Matin T.R., et a.: Correlating Nanostructures with Function: Structurral Colors on the Wings of a Malaysian Bee. (2010), August

[PAT09] Paul A. Tipler, Gene M.: *Physik für Wissenschaftler und Ingenieure*. 6th edition. Spektrum Verlag, 2009. – ISBN 978–3–8274–1945–3

[PS09] P. Shirley, S. M.: *Fundamentals of Computer Graphics*. 3rd edition. A K Peters, Ltd, 2009. – ISBN 978–1–56881–469–8

[R.H12] R.Hooke: *Micrographia, reprinted*. CreateSpace Independent Publishing Platform, 2012. – ISBN 978–1470079031

[RW11] R. Wright, et a.: *OpenGL SuperBible*. 5th edition. Addison-Wesley, 2011. – ISBN 978–0–32–171261–5

[Sta99] Stam, J.: Diffraction Shaders. In: *SIGGRPAH 99 Conference Proceedings* (1999), August

[T.Y07] T.Young: *A course of lectures on natural philosophy and the mechanical arts Volume 1 and 2*. Johnson, 1807, 1807

Erklärung

gemäss Art. 28 Abs. 2 RSL 05

Name/Vorname: .....

Matrikelnummer: .....

Studiengang: ……

Bachelor ☐ Master ☐ Dissertation ☐

Titel der Arbeit: .....

.....

.....

LeiterIn der Arbeit: .....

.....

Ich erkläre hiermit, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäss aus Quellen entnommen wurden, habe ich als solche gekennzeichnet. Mir ist bekannt, dass andernfalls der Senat gemäss Artikel 36 Absatz 1 Buchstabe o des Gesetztes vom 5. September 1996 über die Universität zum Entzug des auf Grund dieser Arbeit verliehenen Titels berechtigt ist.

.....
Ort/Datum

.....
Unterschrift