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, animals exhibit structural colors because of the physical interaction of light with the surface nanostructure of their exterior skin. In his pioneering work, J.Stam developed a reflectance model based on wave optics capturing the effect of diffraction from surface nanostructures. His model is limited by an accurate estimate of the correlation function using statistical properties of the surface's height field. We propose an adaption of his BRDF model that can handle complex natural gratings. Furthermore, we describe a method for interactively rendering of diffraction effects due to interaction of light with biological nano-structures such as snake skin. As input data, our method uses discrete height fields of natural gratings acquired by using atomic force microscopy (AFM) and employ Fourier Optics. Based on Taylor Series approximation for the phase shifts at the nanoscale surface, we leverage the precomputation of the discrete Fourier Transformations, involved in our model, to achieve interactive rendering speed (about 5-15 fps). We demonstrate results of our approach using surface nano-structures of two snake species, namely the Elaphe and Xenopeltis species, when applied to a measured snake geometry. Lastly, we evaluate the quality of our method by comparing its (peak) viewing angles with maximum reflectance for a fixed incident beam with those resulting from the grating equation at different wavelengths. We conclude that our method produces accurate results for complex, natural gratings at interactive speed.

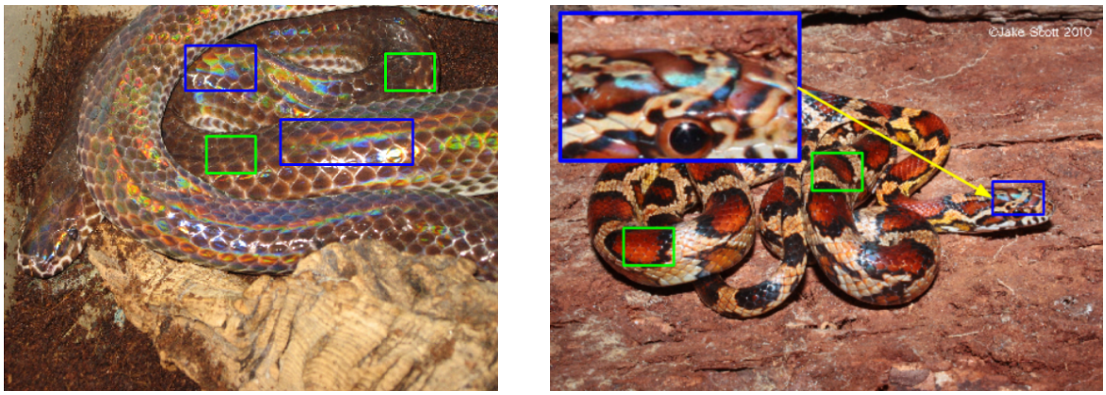
Contents

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 phenomena that involve 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 that exhibits a certain degree of regularity to produce structural colors by diffracting an incident light source. One particular example for such biological color production is the colors we can see when having a closer look at the illuminated skin of snakes, as shown in figure 1.1.



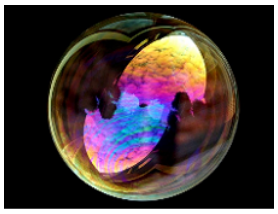
(a) Xenopeltis snake

(b) Elaphe Guttata snake

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

[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`

Some species, like the Xenopeltis, express structural colors in form of iridescent patterns along their scales way stronger than others such as the Elaphe species. The reason lies in the nanostructure of their skins. There are a vast amount of additional reasons for structural color occurrences 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 wings and 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 microstructure of a material. During his examinations of peacock feathers, he found that by wetting the feathers, their colors disappeared. Further, he observed that the feathers have tiny ridges. Building on top of the latest knowledge about interference at that time, 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

[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`

structural colors that are caused by diffraction on natural gratings.

1.2 Goals

The purpose of this thesis is, to simulate physically accurate structural colors caused by the effect of diffraction on various biological structures and then 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 grayscale images.

Natural gratings like this are found on the scale of reptiles, wings of butterflies or the bodies of various insects, but we restrict ourselves 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 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 curved irregularly (indicated by green curves) along the whole surface. This kind of global irregularity is what makes it hard to model the structural complexity of natural gratings[4].

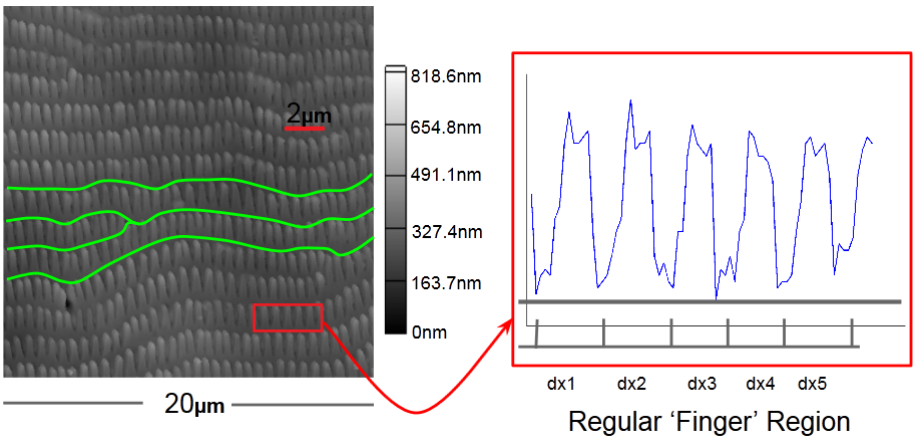


Figure 1.3: Height field of a Xenopeltis snake[5] skin taken using AFM and stored as a grayscale image. Locally, this natural grating consists of regularly aligned (red box) finger-like substructures, but globally we observe a curved alignment of these structures (green curves).

The renderer discussed in this thesis is based on the pioneering work of J. Stam about diffraction shaders [Sta99] where 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

[3]All data is provided by the Laboratory of Artificial and Natural Evolution 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

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 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 brief 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 previously 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 modelling 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 disadvan-

[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

tage 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.

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 Modelling 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 modelled 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 transferred from a source 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 propagates. Its frequency is defined by $f = \frac{v}{\lambda}$ and its carried amount of energy q, measured 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 energy 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 histogram 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 previously seen, each color is associated by a wavelength and is directly related to a certain amount of energy. 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 infinitesimal 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:

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

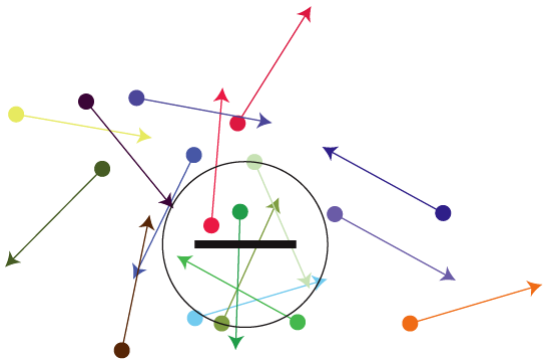
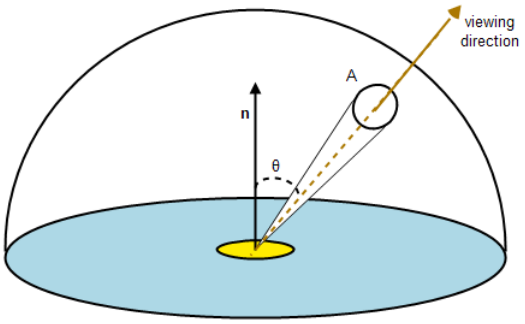


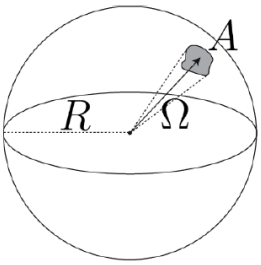
Figure 2.1: Irradiance is the summed up radiance over all directions

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 concept 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 infinitesimal 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 coming 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 figure like used in computer graphics class 2012 in chapter colors

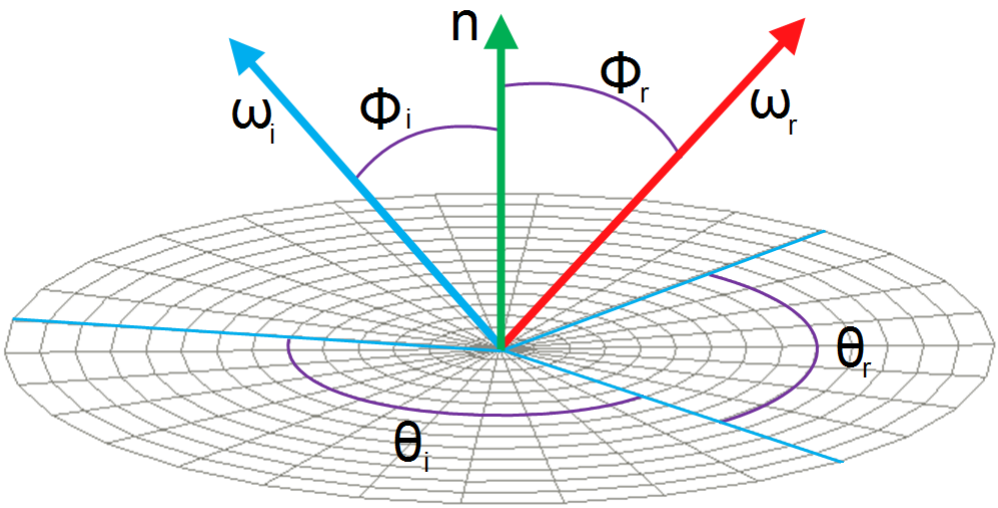


Figure 2.2: Illistration of the BRDF model, where ω_i is pointing to the light source and the existing direction is denoted by ω_r. Both direction unit direction vectors defined w.r.t to a surface normal **n** for every point on the surface.

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

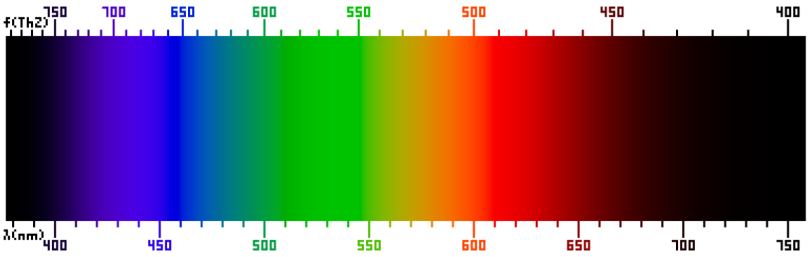


Figure 2.3: Frequency (top) and wavelength (bottom) of colors of the visible light spectrum[4].

In general an eye consists of photoreceptor cells which are responsible for providing ability of color-perception. A schematic of an eye is illustrated 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 brightness levels.

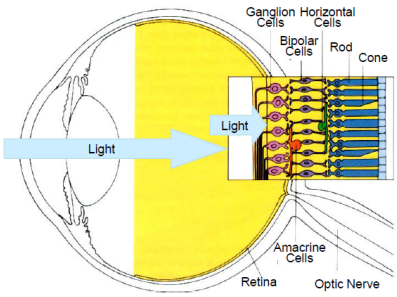


Figure 2.4: Schematic[5] of photoreceptor cells, cones and rods, in human eye

A human eye is made of three different types of cone cells, having their peak sensitivity 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 insensitivity. 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 figure 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 insensitivity. 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.

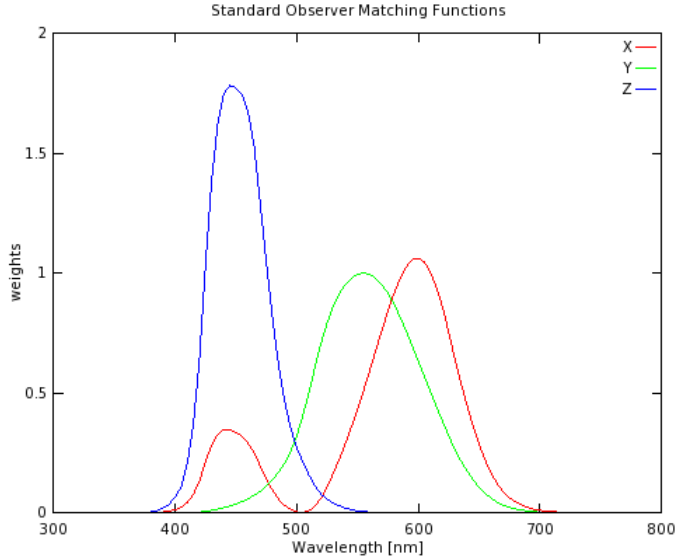


Figure 2.5: Plots of our color matching functions we used for rendering

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 being 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 themselves are inherent to the human visual system. Therefore, many physically light phenomenon are poorly modelled 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 introduction 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 temporarily 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 referring to a complex valued function which is a solution to the so called wave equation which is modelling 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 consisting of periodic oscillations 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:

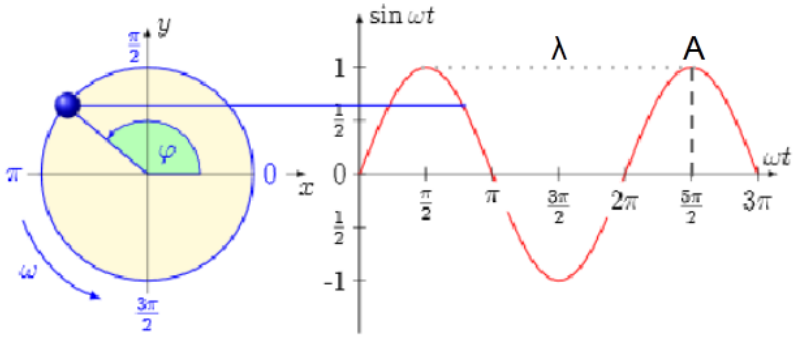


Figure 2.6: Simplified, one dimensionally real valued wave function[7], giving an idea about some important wave properties. We denote the crest of a wave as the highest point relative to the equilibrium line (zero height along time axis) and similarly the trough as the lowest point.

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 interpretations: First, it is a measure of the height from the equilibrium point to the highest 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 brightness to other light waves of the same wavelength. And secondly, it can be interpreted as a measure how much energy a wave carries whereat the greater the absolute amplitude 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 corresponding 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 position 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 traveling

[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 individual wave at those points. Then, the resulting wave is having a greater or lower amplitude than each separate wave and this we can interpret 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:

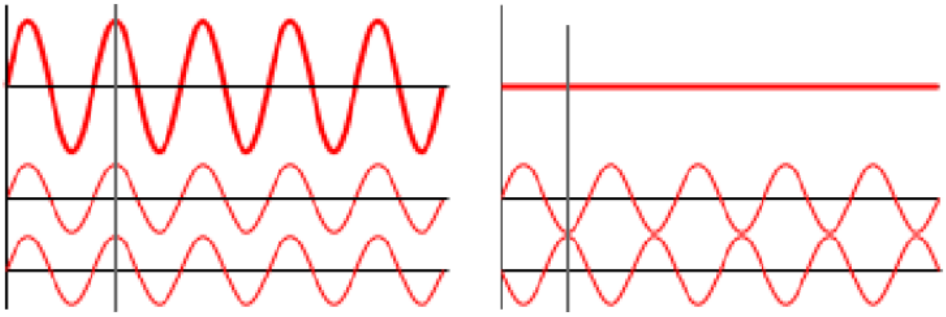


Figure 2.7: Interference scenarios[8] when two waves waves meet: On the left hand-side, there is constructive interference and on the right hand-side there is destructive interference illustrated.

- 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.

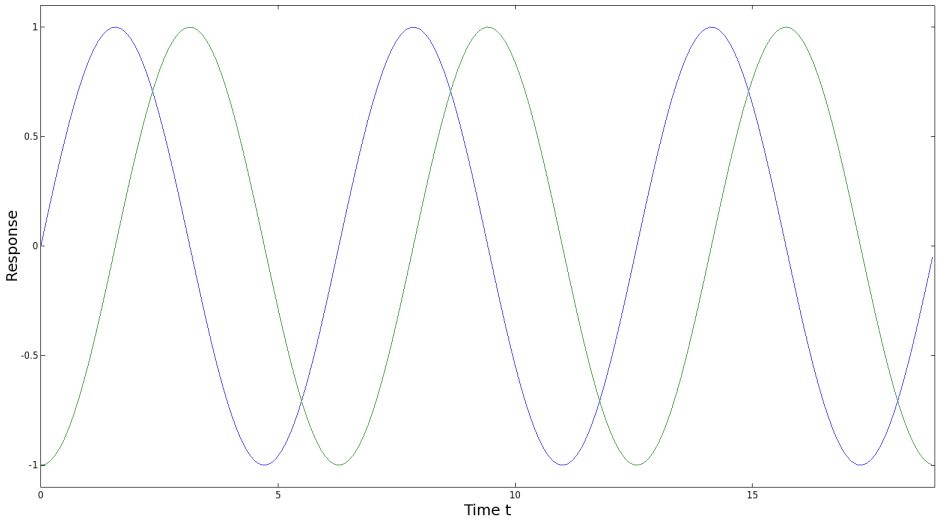


Figure 2.8: Two mathematical sine waves which are perfectly coherent which means that their phase difference is constant for every point in time.

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 completely artificial since in nature there are no mathematical sine waves. Rather, the phase difference is then a function 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 heavily 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 cross-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 occurring 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 interfere 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 the phase and amplitude of a wave, 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 allows 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 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 travel 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 wavefront. Therefore, the new wavefront encloses all emitted wavelets. Figure 2.9 visualizes Huygen's principle for a wavefront reflected off from a plane surface.

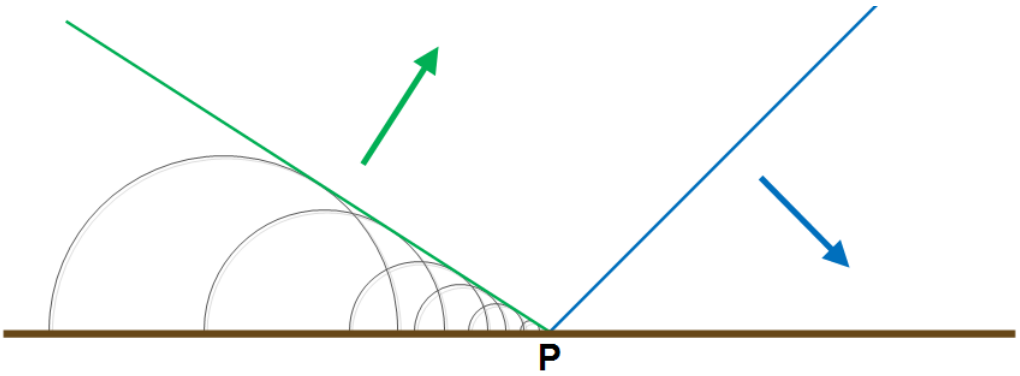


Figure 2.9: A moving wavefront (blue) encounters an obstacle (a surface in brown colors) and produces a new wavefront (green) as a result of superposition among all secondary wavelets.

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.

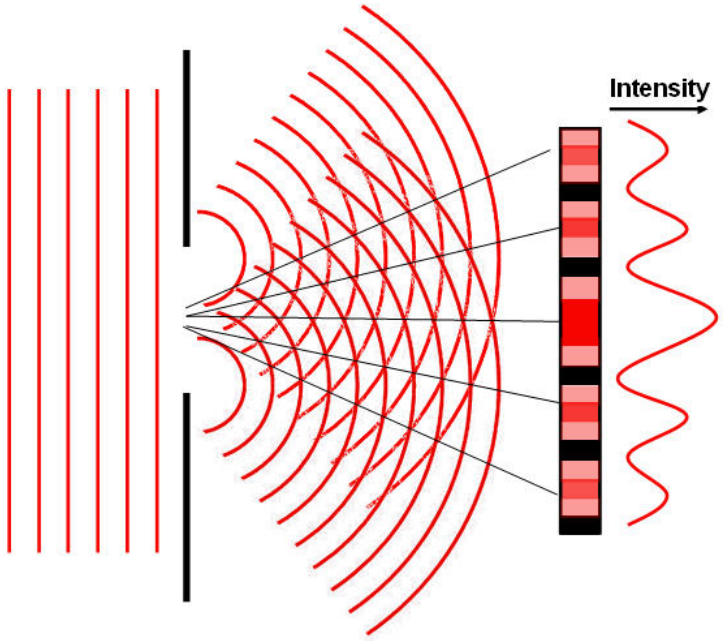
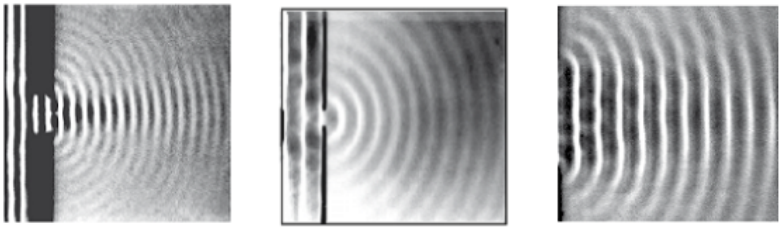


Figure 2.10: Illustration[9] of a diffraction scenario in which a plane wavefront passes through a surface with a certain width and how the wave will be bent, also showing the intensity of the resulting wave.

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 occurs among all wavelets when applying Huygen's Principle for the case when a wavefront 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 everyday'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 iridescent 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. Further information about diffraction gratings can be found 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 modelling 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

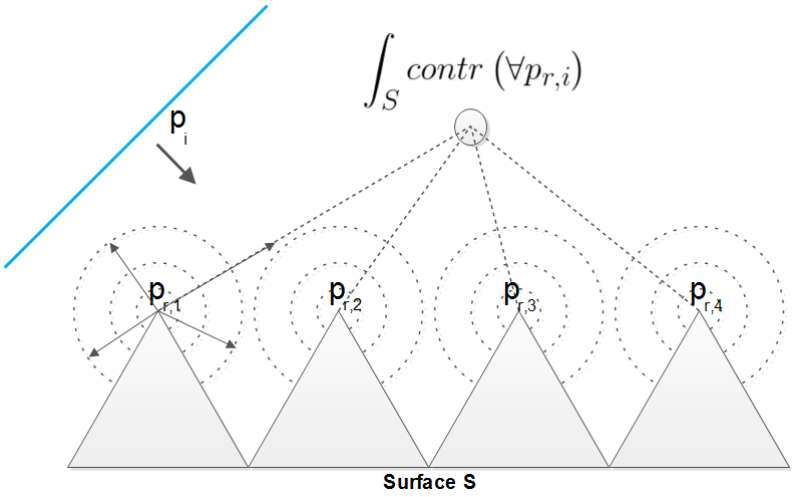


Figure 2.12: Illustration of idea behind Stam's approach: Integration over all secondary sources according to Huygen's principle resulting from an incident wave will give us an identity for the total contribution at a certain point in space.

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 contribution of all wavelets along the surface S (in the figure indicated by a integration symbol), which will 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 modelling 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 modelled by probabilistic distributions or have a direct analytical representation. Both cases allow him to derive an analytical solution for his BRDF model.

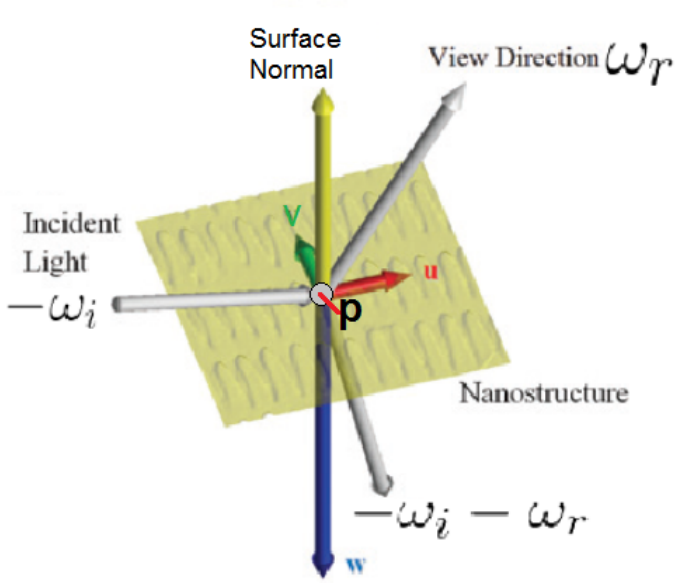


Figure 2.13: Illustration[11] of geometrical setup of Stam's approach where ω_i is a direction, pointing towards the light source, ω_r points towards the camera, n is the surface normal, (u, v, w) are the components of the vector $-\omega_i - \omega_r$.

The direction vector of the secondary wavelet can be computed 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 contribution of all secondary sources used in Stam's BRDF in equation 2.8. For simplification, let us introduce an auxiliary 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, modelling 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.S. Dhillon's poster[DD14].

[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 Transformation, we are dealing with the inverse Fourier Transformation. However, especially in electrical engineering, 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 substitute 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 enough 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 following a brief comparison 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 $fig : stameggratings$.

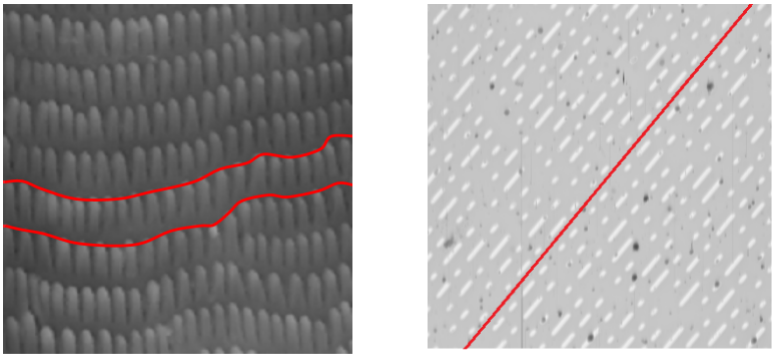


Figure 2.14: Alignment of nano-structures in diffraction gratings. On the left a complex, natural grating of the Elaphe snake species and on the right a synthetic, very regularly aligned grating of a CD.

[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.

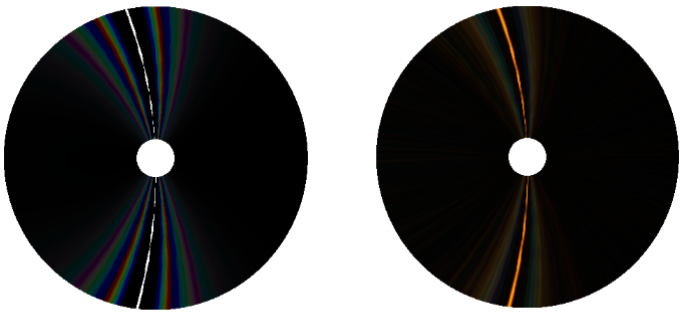


Figure 2.15: Comparison of our approach against a reference implementation of Stam's provided by Nvida Gem. For synthetic diffraction gratings, which have a very regular structure, Stam's approach is doing well. All angles of the diffraction pattern look the same as in the images produced by our approach.

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 specular regions, or such as the color distribution which is rather discrete in Stam's approach. Furthermore, Stam's approach has at some places color contribution, where it should not have.

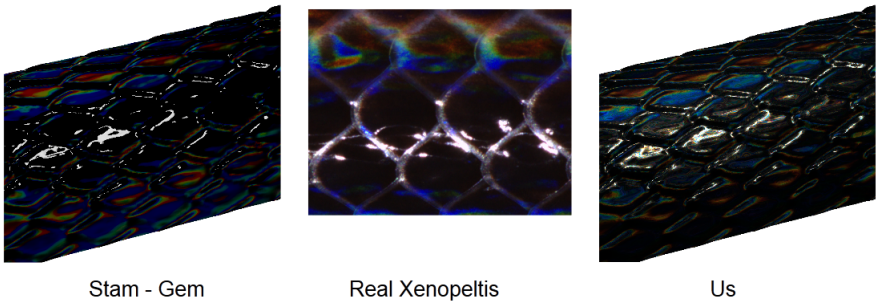


Figure 2.16: Comparison of our approach(on the right) against a reference implementation of Stam's(on the left) provided by Nvida Gem by trying to render a reproduction of a real Elaphe skin (center) under similar lightning and viewing conditions. For natural diffraction gratings, which have a rather complex structure, Stam's approach 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 dealing with and even will have a runtime complexity which allows to perform interactive rendering.

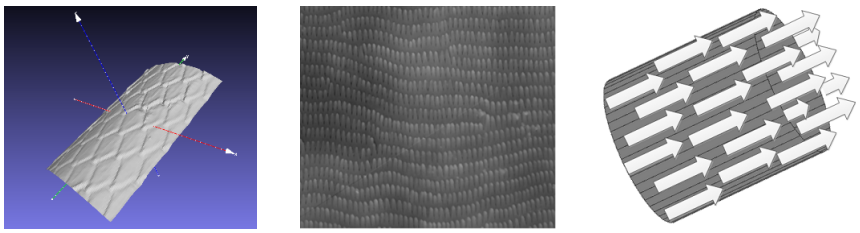
Chapter 3

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 vector field which describes how fingers on a provided surface of the nano-structure 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 formulates a BRDF modelling 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 Evolution 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 corresponds to in this provided image.

is simple enough being modelled relying on statistical methods. These assumptions guarantee that 2.8 has an explicit solution. However, the complexity of a biological nanostructure cannot sufficiently and accurately modelled 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.

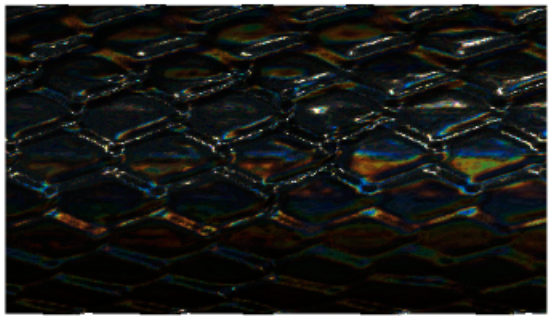


Figure 3.2: Output: Rendered Structural Colors

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 original signal. 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 Transformation operator on the discretized signal $\hat{s}$ which gives us the following expression:

[3] actually it requires the computation of the inverse Fourier Transform of a transformed version of the given height field, 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.

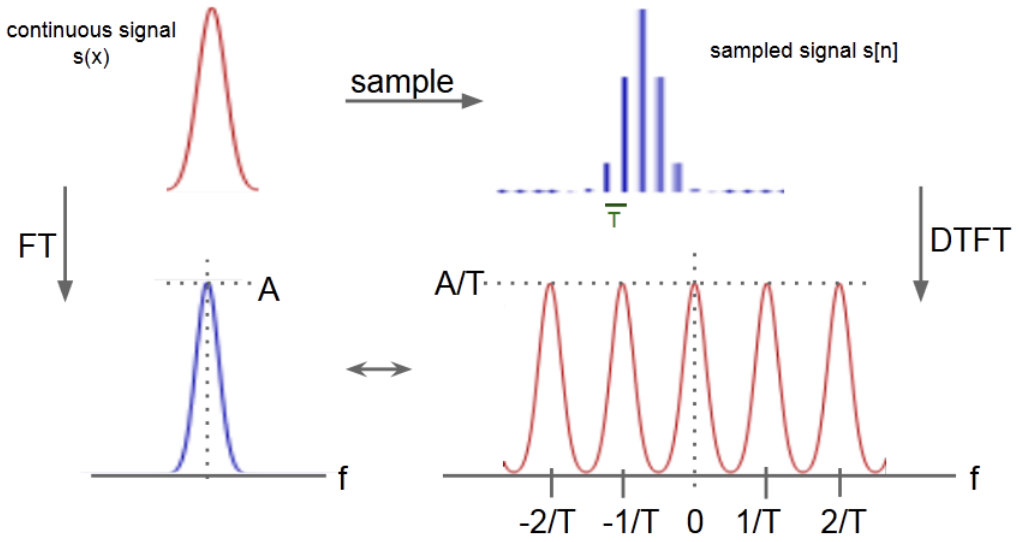


Figure 3.3: Illustration of how to approximate the analytical Fourier Transform (FT) [5] of a given continuous signal by a Discrete Time Fourier Transform (DTFT). The DTFT applied on a band-limited, discretized signal yields a continuous, periodic response in frequency space.

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

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.Previously 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 we plug h into equation 2.6 and this expression again plug into equation 2.7.

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 interference is limited by the quantity coherence area. For filtered sunlight on earth this is equal to $65\mu m$ [8].

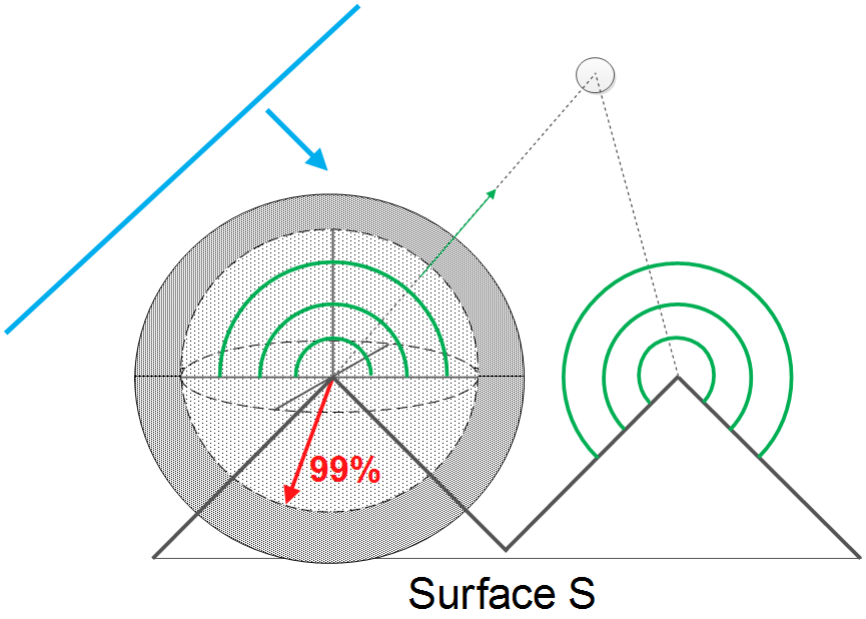


Figure 3.4: A plane wave encounters a surface. According to Huygens principle, secondary wavelets are emitted of from this surface. The resulting wave at a certain point in space (here indicated by a gray circle) depends on the interference among all waves encountering at this position. The amount of 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 Huygen'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 modelled 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 approach.

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.

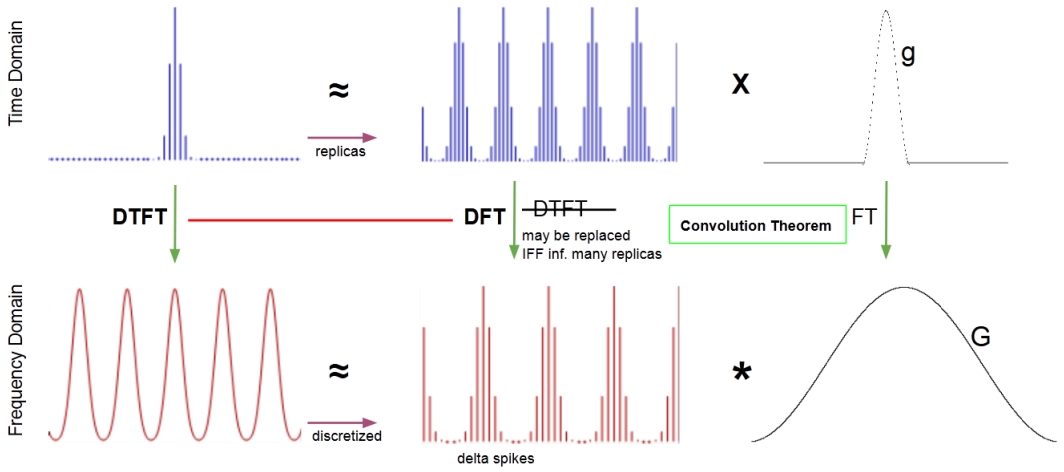


Figure 3.5: Illustration of how to approximate the DTFT [11] by the DFT relying on the Convolution Theorem, using a gaussian window function.

Given a spatial, band-limited 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 previous 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).

Remember that in general, the signal response 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 band-limited. 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 approximately 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 yields 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 algorithms 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 follows:

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

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 dropped all λ subscripts for spectral radiance quantities.

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 arguments.

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 denotes 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 separate 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

[14] Even a fast variant of computation the Fourier Transform has a runtime complexity 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.

of signal processing and Fourier Transformations please consult the appendix A.

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 substitute 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 corresponding 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 equal to $(cos(\theta_i)+cos(\theta_r))^2$[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. Nevertheless, 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 separate P_{dtft} in a certain way such that some computional expensive 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 wavelength 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 determine 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 valid and proven (see appendix C.1) implies that there actually exists such an N. Thus, 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 numerically 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 constrained 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

In this section we describe how our final model for rendering structural colors due to diffraction will look like. For this purpose we use all our previous findings and plug them together to one big equation. For a given height field h representing the surface of a grating, we want to compute the resulting color due to light diffracted on that grating. For rendering we rely on the CIE_{XYZ} colorspace. For given direction vectors w_i and w_r as shown in figure 2.13 the DFT of the height field is equal to:

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

From section 3.2.3 we know that we can reproduce a FT by applying a Gaussin window on the DFT from equation 3.31. This Windowing approach will then look like:

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

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).

In section 3.4 we derived equation 3.30 which tells us how to compute the CIE_{XYZ} color value of a particular color channel using our relative reflectance brdf model from section 3.3.3. Plugging all these findings together and using our windowing approach, listed in equation 3.32, 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:

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

Where $C(\omega_i, \omega_r)$ is the defined in equation 3.18. Note that equation 3.33 integrates over a given wavelength spectrum, denoted by Λ. Usually, this Λ is equal to $[\lambda_{min}, \lambda_{max}]$ where $\lambda_{min} = 380nm$ and $\lambda_{max} = 780nm$.

3.6 Alternative Approach

3.6.1 PQ factors

In this section we are presenting an alternative approach to the previous Gaussian window approach described in section 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 repetitive 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:

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

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

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:

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

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 nano-scaled surface snake shed represented as a two dimensional height field $h(x,y)$. 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.

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

Note that a detailed derivation of equation 3.41 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 section 3.6.1 we derived an alternative approach to the gaussian window approach described in section 3.2.3 in order to approximate our height field. We assume that our height field is a superposition of periodically aligned substructured (i.e. finger structures). This so called PQ approach allows us to integrate over one period of a substructure in our height field, instead iterating over the whole domain. Nevertheless, this main finding, described in equation 3.41, is using the DTFT. Thus, since our original height field is supposed to be a continuous-time band-limited function we can reconstruct it by applying a sinc-interpolation.

In general, for a sinc-interpolation, we are interested in recovering an original analog signal $x(t)$ from its samples. Therefore, 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 illustrates a reconstruction of a 1d signal relying on a sinc-interpolation.

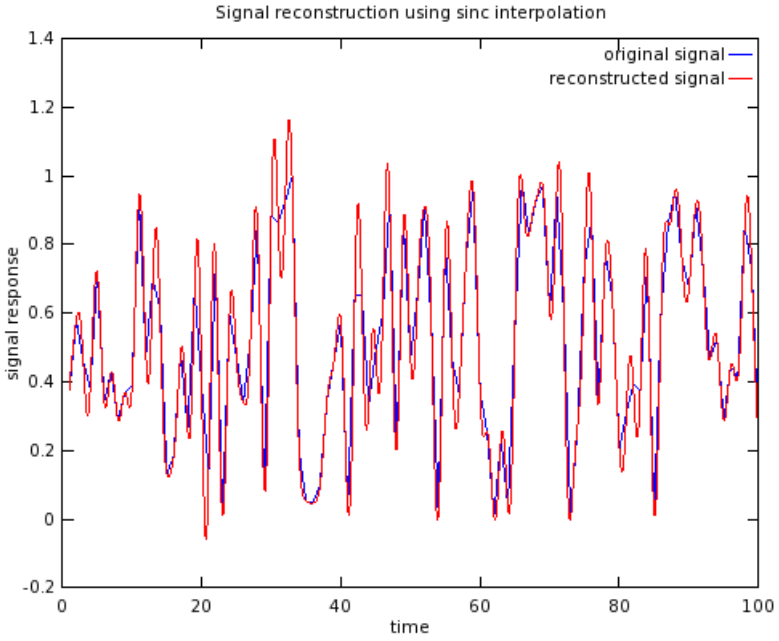


Figure 3.6: Comparison between a given random one dimensional input signal $s(t)$ and its sinc interpolation $\hat{s}(t)$. Notice that for the interpolation there were $N = 100$ samples from the original signal provided.

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, modelled by a virtual camera, light sources and geometries placed in the world[2], having some material properties[3] assigned to. In our implementation, scene geometries are modelled 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 scene 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 stages 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 programming interface of OpenGL[4], a cross-language, multi-platform 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 gratings, based on the model we have derived in the previous 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 autumn 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, modelled by a virtual camera, light sources and geometries placed in the world, having some material properties assigned to.

[2] With the term world we are referring 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 communicate 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 color patterns 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 modelling pixel-wise 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 Transformations for a given input height field, representing a natural grating. For that purpose we have written a small Matlab [10] script conceptualized 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 Transformation 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 Evolution 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 equation 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 separately for each DFT term. The normalization is computed the following way:

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

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

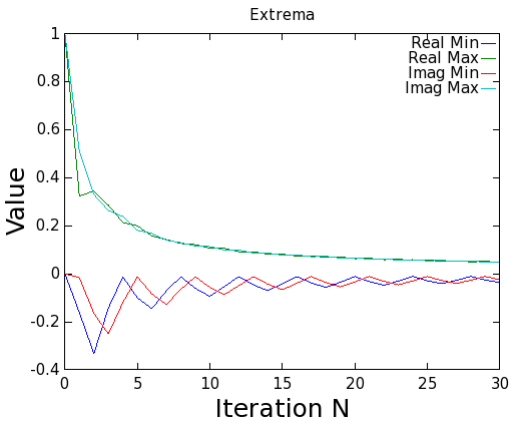
INPUT *heightfieldImg, maxH, dH, termCnt*
OUTPUT *DFT terms stored in Files*

```
% 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.

Input: Precomputed DFT Terms

(a) Heightfield of a Blazed Grating



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

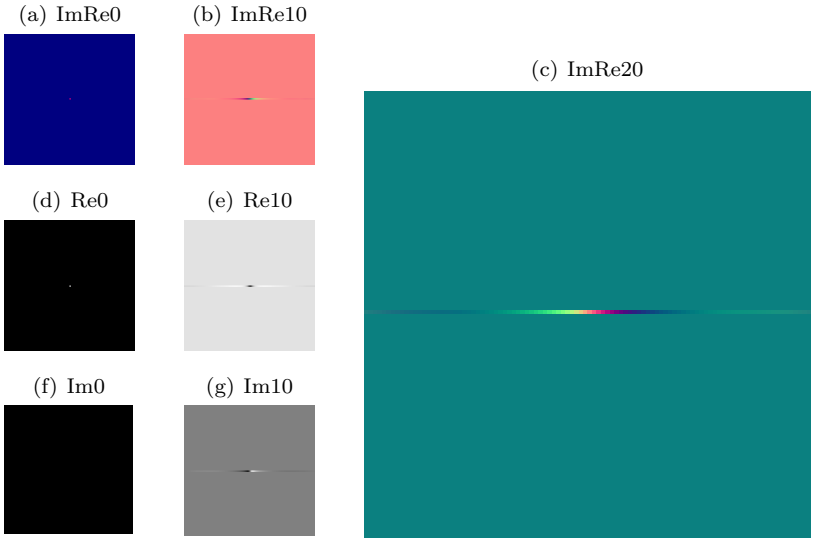


Figure 4.1: A visualization of DFT terms for a height field of a Blazed Grating.

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 4.0(a) until figure 4.0(b) show us exemplarily, 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 4.0(d) till figure 4.0(e) show us the real

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

part images from above's line corresponding figures. Similarly for the figure line from figure 4.0(f) until figure 4.0(g) showing the corresponding imaginary part DFT term images. Figure 4.0(c) is a visualization of a scaled DFT term of a Blazed Grating. In this visualization of a blazed grating, we only see color contribution along the x-axis. Reason for this is, that, since a blazed grating like shown in figure 4.1(a) only varies along the x-axis, also its corresponding 2d DFT image will vary only along the x-axis.

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.

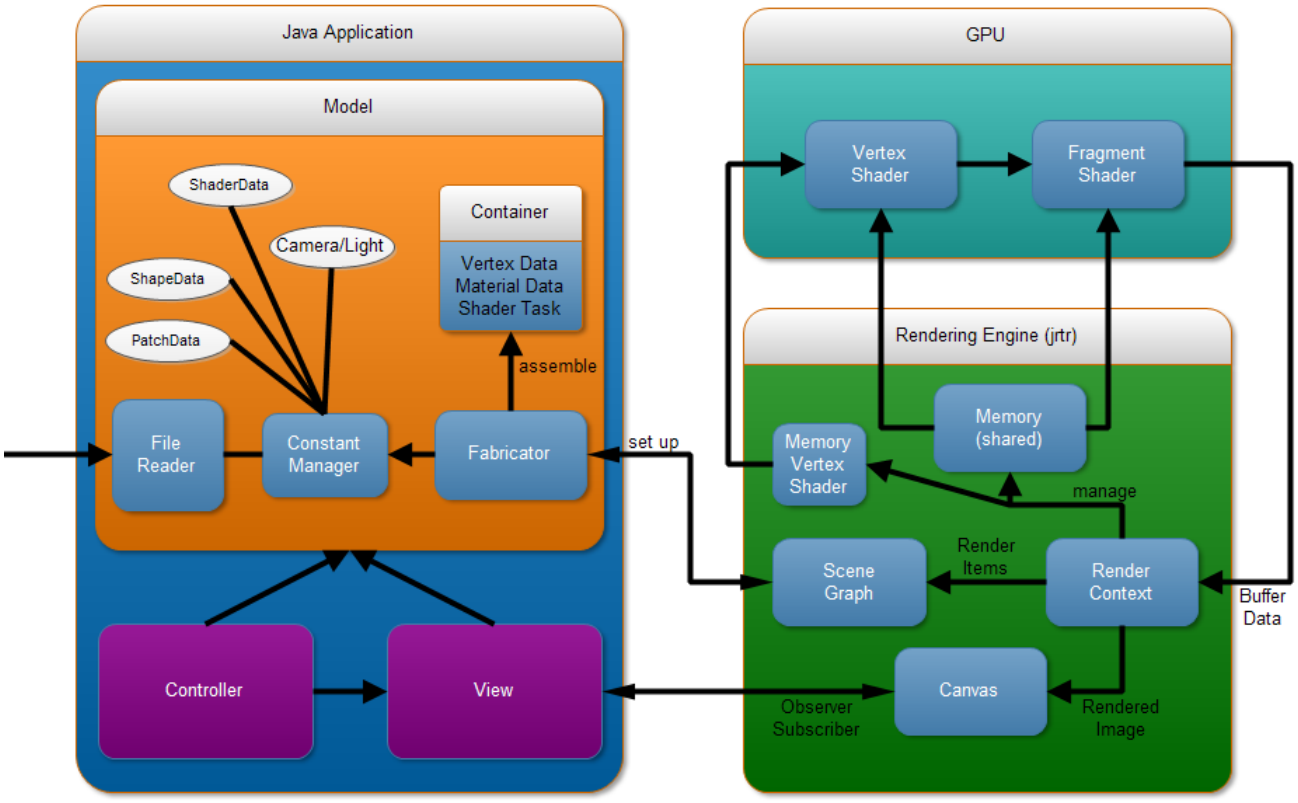


Figure 4.2: Schematical architecture of our Java renderer.

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

[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.

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.

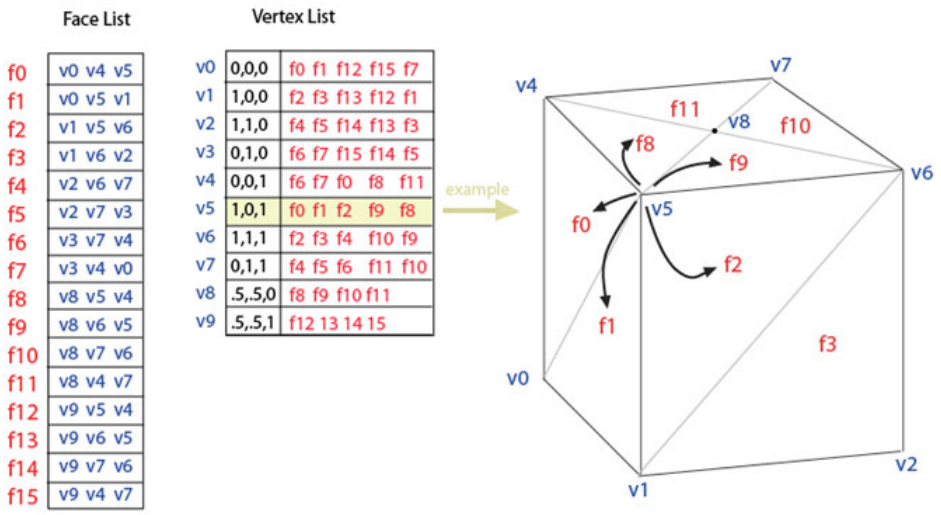


Figure 4.3: Representation[17] of a triangular mesh represents an object as a set of triangles and a set of vertices.

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 whole resource-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.

[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.

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

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 render an image whereas the vertex shader is the first shading stage in the rendering pipeline.

As an input, a vertex shader, like illustrated in figure 4.4, 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 transform the position of vertices. Notice that a vertex shader can manipulate the position of a vertices, but cannot generate additional mesh vertices. Therefore, 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.

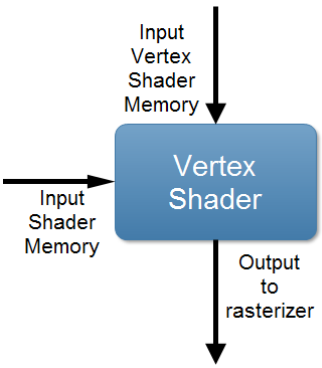


Figure 4.4: Illustration of vertex shader in OpenGL's rendering pipeline.

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:

[18] Defining the positions of a shape in a local coordinate system simplifies its modelling 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.

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 4.5. 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 the image plane, i.e. for any given position in camera space, P determines the corresponding 2d image coordinate. Perspective projections project along rays that converge in center of projection.

Since we are interested in modelling how a viewer sees structural colors on a given scene shape as shown in figure 4.5, modelling 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.

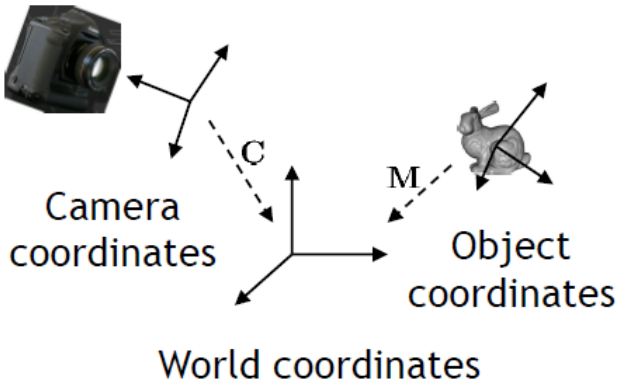


Figure 4.5: Illustration[21] of the Camera coordinate system where its origin defines the center of projection of camera.

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 4.6 illustrates geometrical setup required in order to construct C.

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

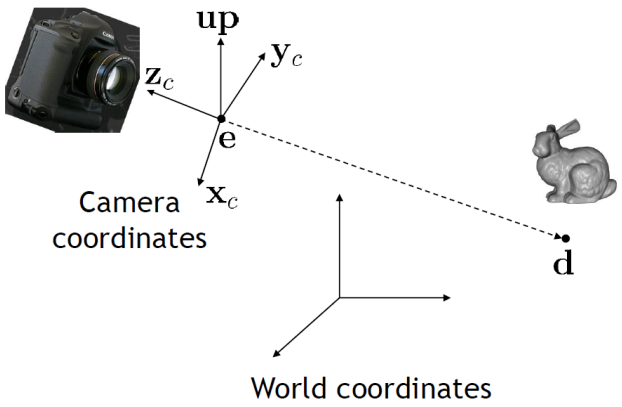


Figure 4.6: Illustration[22] of involved components in order to construct the camera matrix C. The eye-vector e denotes the position of the camera in space, d is the position the camera looks at, and up denotes the cameras height. The camera space is spanned by the vectors helper vectors x_c, y_c and 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 4.6, looks like the the following:

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

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 vectors. 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 have 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.

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

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* ∈ *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 algorithm 2 takes a mesh with of a given scene shape. Each of this vertex 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 4.7.

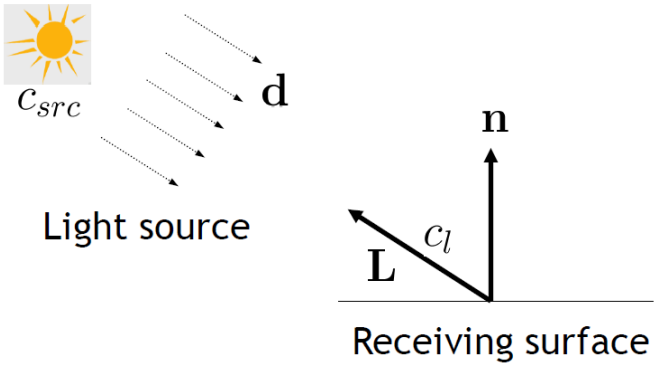


Figure 4.7: Illustration[23] of our light source setup. For a directional light source, all light rays are in parallel.

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:

[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.

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

span(): Assembles a matrix from a given set of vectors. This matrix spans 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 spanned by the given argument matrix.

The output of our vertex shader is on the one side the transformed vertex position 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

In the previous section we gave an introduction to the first shading stage of the OpenGL rendering pipeline by explaining the basics of a vertex shaders. Furthermore, we conceptually discussed the idea behind our vertex shading algorithm formulated in algorithm 2. Summarized, the main purpose of our vertex shader is to compute the light- and viewing-direction vectors ω_i and ω_r defined like in figure 2.13.

After the vertex-shading stage, the next stage in the OpenGL rendering pipeline is the *rasterization* of mesh triangles. As an input, a rasterizer takes a triple of mesh-triangle spanning vertices, each previously processed by a vertex shader. For each pixel lying inside the current mesh triangle, a rasterizer computes its corresponding position in the triangle. According to its computed position, a pixel also gets interpolated values of the vertices attributes of its mesh triangle assigned. The set of interpolated vertex attributes together with the computes position of a pixel is denoted as a fragment. Figure 4.8 conceptualizes the idea of processed set of fragment computed by a rasterizer.

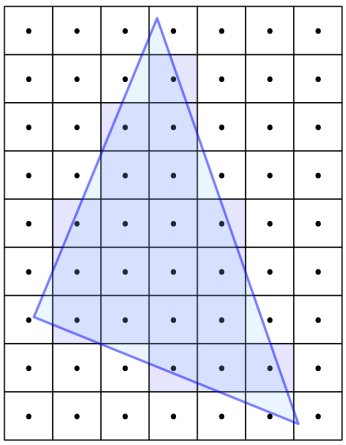


Figure 4.8: Illustration[25] of fragments covered by mesh triangle computed by the OpengGL rasterizer.

A fragment shader as shown in figure 4.9, is the OpenGL pipeline stage subsequent after a mesh triangle is rasterized in the rasterization stage. As input value, a fragment shader takes at least a fragment computed by the rasterizer. It is also possible to assign custom, non-interpolated values to a fragment shader. For each fragment in the fragment shading stage, a color value is computed. Furthermore, a fragment shader computes a depth value for each of its fragments, determining their visibility. Since all existing scene vertices were perspectively projected onto the 2d image plane, the rasterizer could have produced several fragments having assigned the same pixel position. Therefore, among all fragments with the same pixel position, the output pixel color is equal to the color of that fragment, which depth is closest to the viewer.

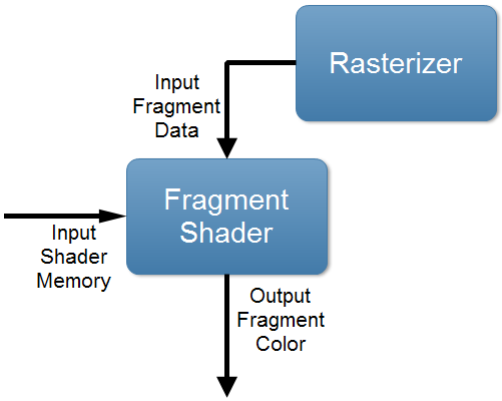


Figure 4.9: Illustration of fragment shader in OpenGL's rendering pipeline.

In this section we explain how to render structural colors resulting due to light diffracted on a natural grating, based on the model described in section 3.5 in equation 3.33. The color values of the produced structural colors, resulting from our model, are computed by our fragment shader (later denoted as **FLSS**) which expects being provided by the following input:

- Precompute DFT terms of the provided height field as explain in section 4.1.
- The processed output, produced during the vertex shading stage (see section 4.3.1), which is, the light- and viewing-direction vectors ω_i and ω_r.
- Fragments produced during the rasterization stage using the output of our vertex shader.

Apart from these basic, varying input values, the following set of shading constants are initialized:

- The number of iterations used for the taylor series approximation, determining the the approximation accuracy.
- The wavelength spectrum $\Lambda = [\lambda_{min}, \lambda_{max}]$ with a certain discretizing level λ_{step}.
- The color weights of the CIE_{XYZ} color matching functions.
- The height field image resolution and its pixel to width correspondence.

[25]This image was taken from `http://en.wikibooks.org/wiki/GLSL_Programming/Rasterization`

By Using all these inputs, our fragment shader performs a numerical integration over the given wavelength spectrum Λ for our final derived expression, stated in equation 3.33. For this integration we use the trapezoidal-rule with uniform discretization of the wavelength spectrum at $\lambda_{step} = 5nm$ step sizes. This implies we are compressing sampled frequencies to the region near the origin due to the fact we are dividing the (u, v) by the wavelength λ and this implies that the (u, v) space is sampled non-linearly.

In section 3.2.3 we have seen that we have to multiply our DFT terms by a Gaussian Window in order to approximate the DTFT which our model is based on. This windowing approach is performed for each discrete λ value using a window large enough to span $4\sigma_f$ in both dimensions. Our DFT terms are computed from height fields that span at least $65\mu m^2$ and are sampled at a resolution of at least $100\mu m$. This ensures that the spectral response encompasses all the wavelengths in the visible spectrum.

Next, we will discuss the actual fragment shading algorithm, listed in algorithm 3. Note, that our fragment shading algorithm uses some helper procedures. The two most fundamental one is the **getlookupCoords** which computes the lookup coordinates in the DFT terms for a given (u,v) defined like in equation 2.5 and a wavelength λ. The actual computation these coordinates is described in section 4.4.1. Notice that the routine **getLocalLookUp** computes a local lookup coordinates used during the gaussian window approximation explained in section 3.2.3. The routine **distVecFromOriginTo** is used to compute the direction vector from the current position of a fragment in texture space, to one of its texture space living one-neighborhood neighbors.

Algorithm 3 Fragment diffraction shader pseudo code

Input: Precomputed DFT Terms
Mesh Triangles
ω_i and ω_r
Output: Structural Color of a pixel
Procedures[26]**:** *getColorWeights*: get colormatching value for wavelength λ(see section 2.1.8)
getlookupCoords: get lookup coordinate(Eq.4.8) by viewing-& light direction
distVecFromOriginTo: get direction vector from origin to a given position
getLocalLookUp: get lookup coordinate(Eq.4.9) of DFT windowing values
rescaledFourierValueAt: rescales DFT terms according to equation 4.1
gaussWeightOf: computes gaussian window according to equation 3.8
dot: computes the dot-product of two given vectors
gammaCorrect: apply gamma correction on RGB vector(see section 4.4.3)

Foreach *Pixel* $p \in Fragment$ **do**
  INIT $BRDF_{XYZ}, BRDF_{RGB}$ **TO** $vec4(0.0)$
  $(u, v, w) = -\omega_i - \omega_r$
  for $(\lambda = \lambda_{min}; \lambda \leq \lambda_{max}; \lambda = \lambda + \lambda_{step})$ **do**
    $xyzWeights = getColorWeights(\lambda)$
    $lookupCoord = getlookupCoords(u, v, \lambda)$
    INIT P **TO** $vec2(0.0)$
    $k = \frac{2\pi}{\lambda}$
    for $(n = 0$ **TO** $T)$ **do**
      $taylorScaleF = \frac{(kw)^n}{n!}$
      INIT F_{fft} **TO** $vec2(0.0)$
      $anchorX = int(floor(center.x + lookupCoord.x * fftImWidth)$
      $anchorY = int(floor(center.y + lookupCoord.y * fftImHeight)$
      for $(i = (anchorX - winW)$ **TO** $(anchorX + winW))$ **do**
        for $(j = (anchorY - winW)$ **TO** $(anchorY + winW))$ **do**
          $dist = distVecFromOriginTo(i, j)$
          $pos = getLocalLookUp(i, j, n)$
          $fftVal = rescaledFourierValueAt(pos)$
          $fftVal *= gaussWeightOf(dist)$
          $F_{fft} += fftVal$
        end for
      end for
      $P += taylorScaleF * F_{fft}$
    end for
    $xyzPixelColor += dot(vec3(|P|^2), xyzWeights)$
  end for
  $BRDF_{XYZ} = xyzPixelColor * C(\omega_i, \omega_r) * shadowF$
  $BRDF_{RGB}.xyz = D_{65} * M_{XYZ-RGB} * BRDF_{XYZ}.xyz$
  $BRDF_{RGB} = gammaCorrect(BRDF_{RGB})$
end for

[26]Please have a look at section 4.3.2 to see further descriptions of these procedures.

Please note that for simplification purposes we omitted some input values in algorithm 3. A complete input value list can be found in section 4.3.2. Last, a a brief description of some code sections of our fragment shading algorithm:

From line 4 to 26:
This loop performs uniform sampling along wavelength spectrum Λ for the spectral integration seen in section 3.5. The procedure $ColorWeights(\lambda)$ computes the color weight for the current wavelength λ by a linear interpolation between the color weight and the values $\lceil\lambda\rceil$ and $\lfloor\lambda\rfloor$. The color weights are stored in a external table accessed by our fragment shader[27]. At line 6 the procedure call $lookupCoord(u, v, \lambda)$ returns the coordinates for the texture lookup which are computed like described in equation 4.7. At Line 25 the diffraction color contribution, computed during the integration over the wavelength spectrum for each wavelength λ, is accumulated.

From line 9 to 24:
This loop performs the Taylor series approximation using a predefined number of iterations. Basically, the spectral response is approximated for our current value for (u, v, λ). According to section 3.2.3, we can approximate a DTFT by multiplying a gaussian window by DFT terms. When interpreting our DFT terms by a set of matrices, a particular DFT term value, corresponding to the position of a given fragment, can be looked up at the index ($anchorX$, $anchorY$). For our the Gaussian-Windowing approach we use a one-neighborhood around ($anchorX$, $anchorY$) in order to approximate the DFT value for a fragment.

From line 14 to 22:
In this inner most loop, the convolution of the gaussian window with the DFT terms of the given height field is performed. The routine $gaussWeightOf(dist)$ computes the weights in equation (3.8) from the distance between the current fragment's position and the current neighbor's position in texture space. Local lookup coordinates for the current Fourier coefficient $fftVal$ value, stored in the DFT terms, are computed at line 17 and computed like described in equation 4.9. The actual texture lookup is performed at line 18 using those local coordinates. Inside the procedure $rescaledFourierValueAt$ the values of $fftVal$ are rescaled by its extrema values[28], since $fftVal$ is normalized according to the description from section 4.1. The current $fftVal$ value in iteration is scaled by the current Gaussian Window weight and then summed to the final neighborhood FFT contribution at line 20.

After line 26:
At line 27 the gain factor $C(\omega_i, \omega_r)$ from equation 3.18 is multiplied by the computed pixel color like formulated in equation 3.19. The gain factor contains the geometric term of equation 2.9 and the Fresnel term F. For computing the Fresnel term we use the Schlick approximation defined in equation D.1, using an refractive index of 1.5 since this is close to the measured value from snake sheds. Our BRDF values are scaled by a shadowing function as described in the appendix of the paper[Sta99]. The reason or this is that the nanostructure of a snake skins is grooved forming V-cavities. therefore some regions on the snake surface is shadowed. Last, we transform our colors from the CIE_XYZ colorspace to the CIE_RGB space using the CIE Standard Illuminant $D65$. Last we apply a gamma correction on our computed RGB color values. Consult the section 4.4.3 for further insight.

[27] This color weight table contains data for lambda steps in size of 1nm

[28] This extrema values were precomputed in Matlab during the precomputation stage and are passed as an input to our fragment shader.

4.4 Technical Details

4.4.1 Texture Lookup

In our fragment shader we want to access DFT coefficients of our height field at a given location (u, v) (defined like in equation 2.5) to compute structural colors according to equation 3.33.

For any given nano-scaled surface patch P with a resolution of A by A μm, stored as a N by N pixel image I, one pixel in any direction corresponds to $dH = \frac{A}{N}\mu m$. In Matlab we computed a series of n output DFT terms $\{I_{out_1}, ..., I_{out_n}\}$ from this image I, where the n-th image represents the DFT coefficients for the n-th DFT term in our Taylor Series approximation. Notice, that every DFT image value (u, v) is in the range $[-\frac{f_s}{2}, \frac{f_s}{2}]^2$ where f_s is the sampling frequency equal to $\frac{1}{dH}$.

In order to get access to these image within our shader, we have to pass them as GLSL textures. 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.

However, u, v coordinates are assumed to be in range $[-\frac{f_s}{2}, \frac{f_s}{2}]$, but GLSL texture coordinates are in range $[0, 1]$. Therefore, for every computed lookup coordinate pair (u, v), we have to apply a affine transformation to be in the correct range. Figure 4.10. illustrates this issue of different lookup ranges. In general, an affine transformation is a mapping of the form $f(x) = sx + b$ where s is a scaling and b is a translation.

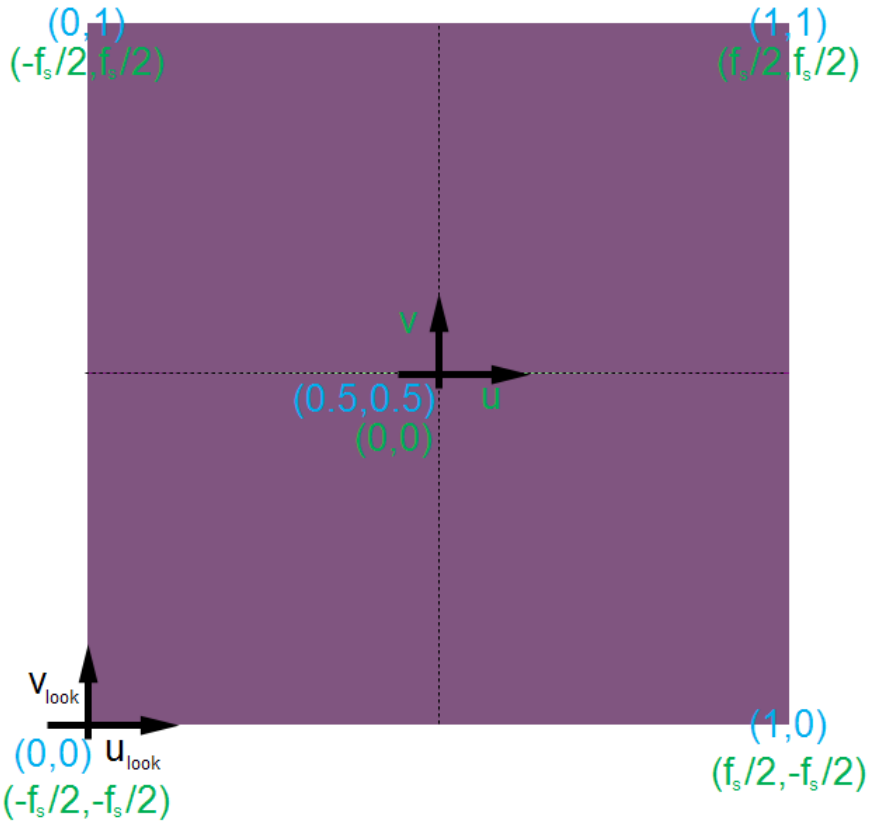


Figure 4.10: Illustration of (u, v) lookup using GLSL textures: texture coordinates are shown in blue color, u-v coordinates are shown in green color.

Figure 4.10 visualized a particular DFT coefficients image for a blazed grating. In GLSL texture coordinates the range of such an image is $[0, 1]^2$ whereat in a (u,v) coordinates system this would correspond to the range $[-\frac{f_s}{2}, \frac{f_s}{2}]^2$. Regarding the way the DFT terms were computed in Matlab, their zero frequency, i.e. $(u, v) = (0, 0)$, is located at the center of the given DFT image. Since by convention the bottom left corner of a GLSL texture corresponds to the value $(0, 0)$ but we want to work with (u, v) coordinates, we have to introduce an helper coordinate system (u_{lookup}, v_{lookup}). This helper coordinates can be computed by a affine transformation applied on (u, v) coordinates. Hence, we will explain how to compute the scaling s and translation b components of our affine transformation:

Translation component b of affine transformation:
Since the zero frequency component of DFT images is shifted towards its centre position, we have to shift the coordinates u and v to the center of the current N by N pixel image by a bias b. The translation b is a constant value and is computed like the following:

$$b = \begin{cases} \frac{N}{2} & \text{if } N \equiv_2 0 \\ \frac{N-1}{2} & \text{otherwise} \end{cases} \tag{4.5}$$

Scaling component s of affine transformation:
For the scaling part of our affine transformation, we have to think a little further: let us 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 temporal interval dH.

Let k denote the wavenumber which is equal to $\frac{2\pi}{\lambda}$ for a given wavelength λ. Then the signal is both, periodic with time period T and discrete with temporal interval dh. This implies that its spectrum should be discrete with frequency interval w_0 and periodic with frequency period $\Omega = \frac{2\pi}{dH}$. This gives us an idea how to discretize the spectrum. For amy surface patch P which is periodically distributed on its surface, its frequency interval along the x-axis is equal to:

$$w_0 = \frac{2\pi}{T} = \frac{2\pi}{N \cdot dH} \tag{4.6}$$

Thus, only wavenumbers 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:

$$\begin{aligned} s(u) &= \frac{ku}{w_0} \\ &= \frac{kuNdH}{2\pi} \\ &= \frac{uNdH}{\lambda} \end{aligned} \tag{4.7}$$

The lookup for the v-direction will be equal $s(v)$ defined like in equation 4.7.

Final affine transformation:
Using the translation from equation 4.5 and the scaling from equation 4.7, the transformed texture lookup-coordinates (u_{look}, v_{look}) for a given wavelength λ is equal to:

$$\begin{aligned} (u_{look}, v_{look}) &= (s(u) + b, s(v) + b) \\ &= \left(\frac{uNdH}{\lambda} + b, \frac{vNdH}{\lambda} + b \right) \end{aligned} \tag{4.8}$$

Note that for the Windowing Approach, used in algorithm 3, we are visiting a one-pixel-neighborhood around each (u, v) coordinate pair. For any position (i, j) of its neighbor-pixels, these local coordinates $(u_{look}^{local_i}, v_{look}^{local_j})$ around the origin (u_{look}, v_{look}) from equation 4.8 are equal to:

$$(u_{look}^{local_i}, v_{look}^{local_j}) = (i, j) - (u_{look}, v_{look}) \tag{4.9}$$

4.4.2 Texture Blending

So far, we have seen how to render structural colors caused when light is diffracted on a grating. But usually, many objects, such as a snake mesh, also have a texture associated with. Therefore, will have a closer look how to combine colors of a given texture with computed structural colors. For this purpose we will use a so called texture blending approach. This means that, for each pixel,

its final rendered color 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 $w_{diffuse}$, denoting the weight of the diffuse color part. The texture color is once scales by the absolute value of the Fresnel Term F and once by $1 - w_{diffuse}$. Algorithm 4 shows in depth how our texture blending is implemented:

Algorithm 4 Texture Blending

Input: $c_{texture}$: Texture color at given fragment position
$c_{diffraction}$: Structural color at given fragment position
Output: c_{out} : Mixed fragment texture- and structural-color

1: $\alpha = abs(F)$
2: **if** $(\alpha > 1)$ **then**
3: $\quad \alpha = 1$
4: **end if**
5: $diffraction_{diff} = (1 - w_{diffuse}) \cdot c_{diffraction}$
6: $text_{spec} = (1 - \alpha) \cdot c_{texture}$
7: $text_{diff} = w_{diffuse} \cdot c_{texture}$
8: $c_{out} = diffraction_{diff} + text_{spec} + text_{diff}$

4.4.3 Color Transformation

In our fragment shader, we access a table which contains precomputed CIE's color matching functions values[29] from $\lambda_{min} = 380nm$ to $\lambda_{max} = 780nm$ in $5nm$ steps. In algorithm 3 we describe how to compute the CIE_{XYZ} color values as described in section 2.1.8. We can transform the color values into CIE_{sRGB} gamut 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:

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

One aspect we have to keep in mind is what is the tristimulus value of the color defining our white point in our images. Defining what tristimulus value whit corresponds to, usually depends on the application. For our shaders we use use the CIE Standard Illuminant $D65$. $D65$ is intended to represent an average midday sun daylight. Applying the D65 illuminant, the whole colorspace transformation will look like:

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

[29] Such a function value table can be found at `cvrl.ioo.ucl.ac.uk` for example

Each component of the Standard Illuminant acts as rescaling factor according to the white-point definition of $D65$ for our computed color values. Last, we perform a gamma correction on each pixel's (R, G, B) value. A gamma correction is a non linear transformation which controls the overall brightness of an image[30].

4.5 Discussion

4.5.1 Comparison: per Fragment-vs. per Vertex-Shading

In this chapter we have seen how our render is implemented. We discussed our fragment shader which is responsible for computing the structural color values resulting by diffracted light. Our fragment shader's runtime performance depends on the resolution of the final rendered image, since there is a correspondence between pixels and fragments. Notice that we also could have computed the structural colors during the vertex shading process. Then, the whole rendering performance of our shading approach would only depend on the vertex count of our meshes. Shading on a per vertex basis is usually bad since in order to get a nice and accurate rendering, the vertex count of a mesh has to be big. Furthermore, a vertex shader produces poor results for shapes like a cube for example, according to our structural color model. Therefore, shading on a per fragment shader scales depending on the rendered image resolution and is independent of the mesh vertices (their distribution and count).

4.5.2 Optimization of Fragment Shading: NMM Approach

The fragment shader algorithm described in algorithm 3 performs a gaussian window approach by sampling over the whole wavelength spectrum in uniform step sizes. This algorithm slow, since we for each pixel we iterate over the whole lambda spectrum. Furthermore, for any pixel, we iterate over its one-neighborhood. When also considering the number of taylor series approximation steps, we will have a run-time complexity of

$$O(\#spectrtumIter \cdot \#taylorIter \cdot neighborhoodRadius^2) \tag{4.13}$$

Our goal is to optimize this runtime behaviour. Instead sampling over the whole wavelength spectrum, we instead could integrate over a minimal number of wavelengths contributing the most to our shading result like shown in figure 4.11. These values are elicited like the following: Lets consider (u, v, w) defined as in equation 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 $\lambda_n^v = \frac{du}{n}$. Therefore we can derive the following boundaries of n:

$$\begin{aligned}
&\text{If } u, v > 0 && N^u_{min} = \tfrac{du}{\lambda_{max}} \leq n_u \leq \tfrac{du}{\lambda_{min}} = N^u_{min} \\
& && N^v_{min} = \tfrac{dv}{\lambda_{max}} \leq n_v \leq \tfrac{dv}{\lambda_{min}} = N^v_{min} \\
&\text{If } u, v < 0 && N^u_{min} = \tfrac{du}{\lambda_{min}} \leq n_u \leq \tfrac{du}{\lambda_{min}} = N^u_{max} \\
& && N^v_{min} = \tfrac{dv}{\lambda_{min}} \leq n_v \leq \tfrac{dv}{\lambda_{min}} = N^v_{max} \\
&\text{If } (u, v) = (0, 0) && n_u = 0 \\
& && n_v = 0
\end{aligned}$$

[30] For further information about gamma correction, please have a look in the book *Fundamentals of Computer Graphics*[PS09].

By transforming those equations (equations 4.5.2) to $(\lambda^u_{min}, \lambda^u_{min})$ and $(\lambda^v_{min}, \lambda^v_{min})$ respectively, we can reduce the total number of required iterations in our fragment shader. We denote this optimization by the n_{min}, n_{max} (NMM) shading approach.

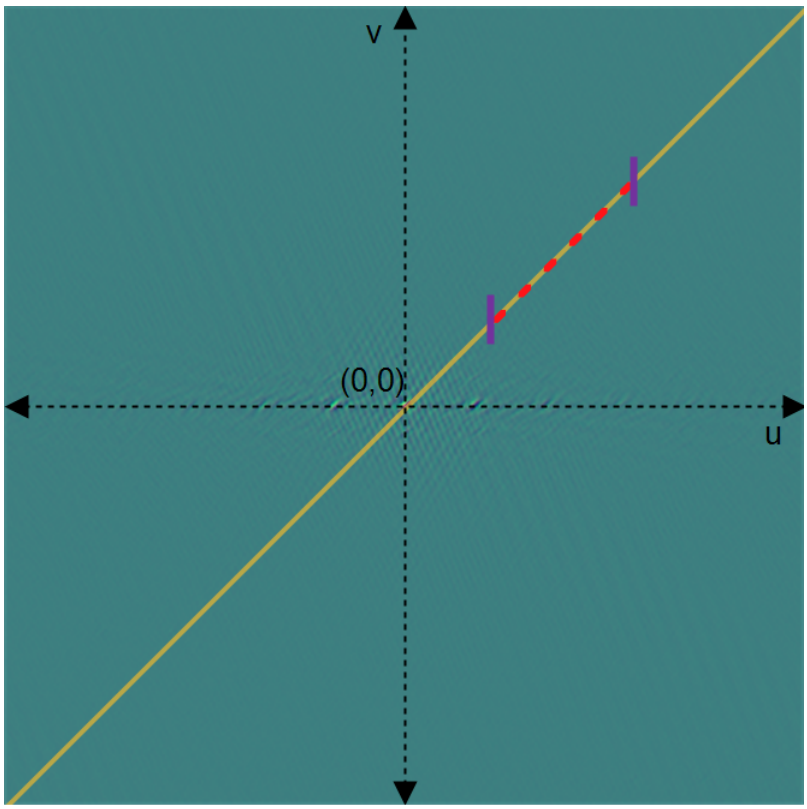


Figure 4.11: Illustration of NMM shading approach's idea.

Figure 4.11 illustrated the idea of the NMM shading approach by showing a visualization of a DFT term. Instead integrating over the whole wavelength spectrum (yellow line), we can find a constrained wavelength range by using equation 4.5.2, and then only integrate over a reduced, discrete set of λ values (indicated by the dotted red line). Since we are uniformly integrating over the wavelength spectrum but this approach selects non-uniformly a reduced wavelength spectrum, this can lead to issues. Close to zero frequency, i.e. (u, v) equal to $(0, 0)$, we could have too few or even no samples, using the NMM approach. Thus, our workaround according to this issue is to use a default color close to the zero frequency. Technically, this means that return white color around a small ϵ circumstance.

4.5.3 The PQ Shading Approach

Another variant is the *PQ* approach described in section 3.6.1. In section 3.6.2 we described how to interpolate the values relying on sinc functions. Another approach would be simple to perform a linear interpolation. This last variant does not require to iterate over a pixel's neighborhood. This it has a faster runtime complexity faster than any windowing approach. 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 pixel's neighborhood within the fragment shader. This once again will slow down the whole shading. Algorithm 5 describes the modification of the fragment shader in algorithm 3 in oder to use sinc interpolation for the PQ approach instead using a gaussian window:

Algorithm 5 Sinc interpolation for PQ approach

Input: same input as in algorithm 3
Output: c_p : Structural Color based in the PQ approach (See section 3.6.1)

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

In a fragment shader, for every fragment p we compute its reconstructed function value $f(p)$ stored in w_p. The value w_p is the reconstructed signal value at $f(p)$ by the sinc function as described in section 3.6.2. Similar like for the gaussian windowing approach, we calculate the distance $\Delta_{p,(i,j)}$ between the current fragment position p and each of its neighbors $(i,j) \in \mathcal{N}_1(p)$ in its one-neighborhood. Multiplying this distance by π gives us the an angle used for the sinc function interpolation. Notice that, in order to avoid division by zeros side-effects, we add a small integer ϵ to our angle value.

Chapter 5

Evaluation and Data Acquisition

5.1 Data Acquisition

Our goal is to perform physically accurate simulations of diffraction effects due to natural gratings. As for every simulation, its outcome highly depends on the input data and thus we also require measurements[1] of real natural gratings. For that purpose, samples of skin sheds 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 AFM 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 depth-map of the atoms on the surface. An Atomic force microscope is a very high-resolution probe scannings, with demonstrated resolution on the order of a fraction of a nanometer, which is 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 all underlying components involved in the rendering process. One particular component, which we have not investigated any further, is the diffraction gratings that represents our height fields. Thus, in this section we will examine in detail, what such a diffraction grating actually is and how it works.

By the term *diffraction grating* we are referring to the surface of a flat piece of an opaque material that contains a large number of parallel, closely and evenly spaced slits[2]. Therefore, these slits are forming a periodically packed, groove-like structure along the surface of the material.

A diffraction grating alters the state of an incident light beam by diffracting its component waves. According to the definition of Huygen's principle (see section 2.2.4), when an incident light beam hits the grating, the slits of the grating will act as a point light source that emits spherical wavelets in every direction. Basically, a diffraction grating can be either transmissive (see figure

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

[2] Usually, these slits are either engraved or etched into the surface of the material.

5.2(a)) or reflective (see figure 5.7). As light transmits through or reflects off a grating, the grooves on the grating cause the light to diffract and divide the light into its component wavelengths. This also implied that the emitted wave will have a different outgoing angle with peak intensity than what it had initially. In general, the closer the spacing of the slits is to the wavelength of the incident wave, the more the emitted wave will be diffracted.

Figure 5.1 illustrates what happens when monochromatic light source passing through a transmissive grating. Using a spectometer, we see that the outgoing angle of the emitted wave 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.

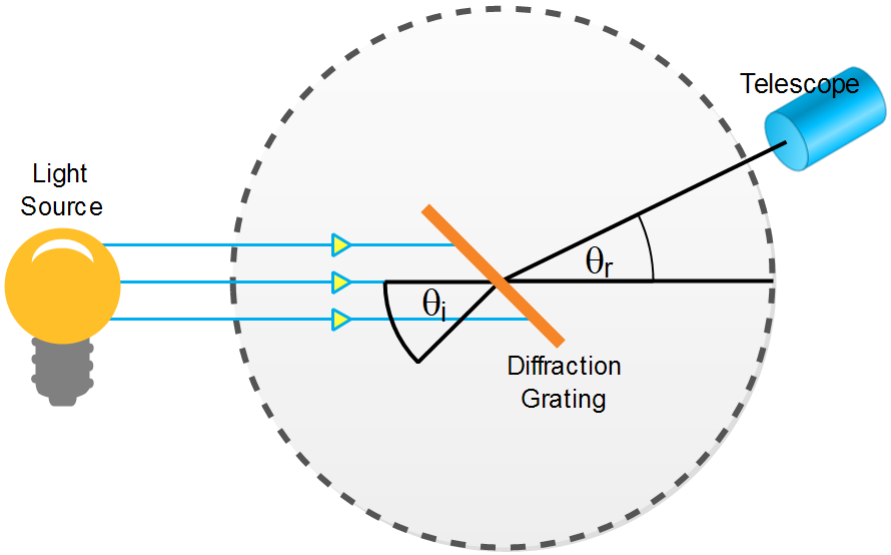
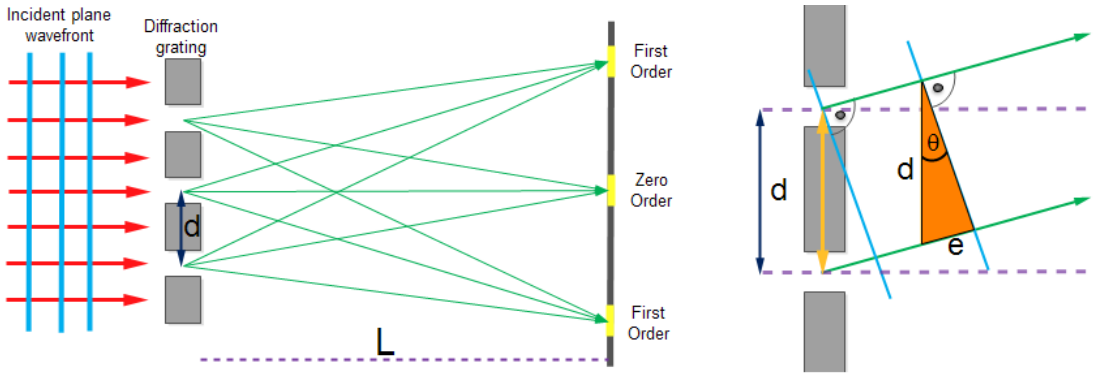


Figure 5.1: Spectometer: When a beam of monochromatic light passes through a grating placed in a spectrometer, images of the sources can be seen through the telescope at different angles.

Suppose an incident plane wavefront, composed of a monochromatic light source, is directed at a transmissive diffraction grating, parallel to its axis as shown in figure 5.1. Let the distance between successive slits on the grating be equal the value d. Furthermore, at a distance L, there is a screen parallel to the grating. Then the emitted waves will from a diffraction pattern on the screen which is the result of interference effects (constructively or destructively interference) among outgoing wavelets like shown in figure 5.2(a).



(a) Idealized Transmissive Grating (b) Parallel reflected Rays

Figure 5.2: Light directed to parallel to grating

If the distance to the screen is much larger than the slits width, i.e. $L >> d$, then all the waves emanating from the surface and ending up at the receiver are parallel. Thus, the path difference between waves from any two adjacent slits can be derived by drawing a perpendicular line between the parallel waves. Applying simple trigonometry gives us this path difference as $e = dsin(\theta)$ as shown in figure 5.2(b). 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. Without constraints of generality, either for a transmissive or a reflective diffraction grating type, an analogous derivation for equation 5.1 can be derived.

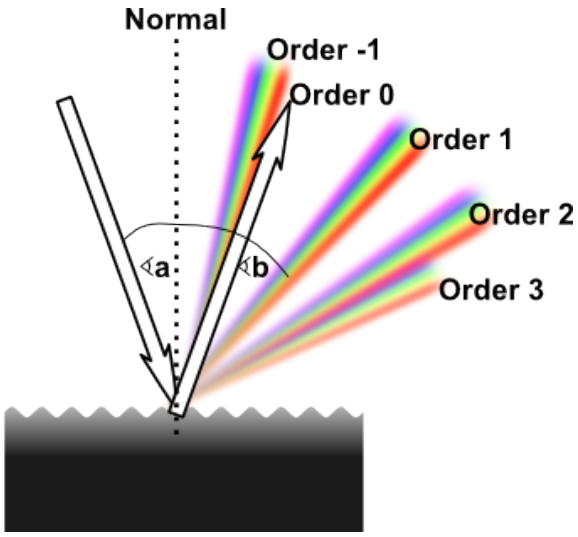
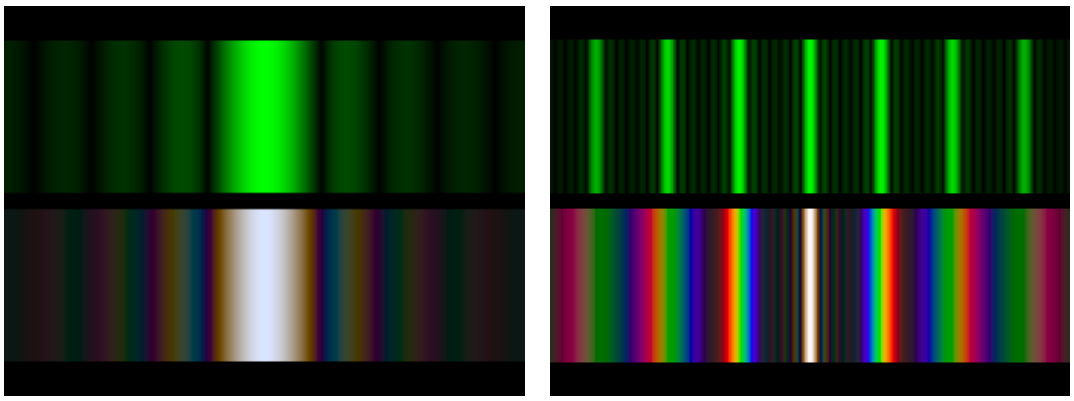


Figure 5.3: Different[3] Orders of diffraction when light is diffracted on a reflective diffraction grating.

When a 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. Figure 5.4 illustrates this for different number of slits on a diffraction grating.



(a) one slit (b) seven slits

Figure 5.4: Difference of diffraction pattern[4] between a monochromatic (top) and a white (bottom) light spectra for different number of slits.

Since the reflected angle θ_r 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. 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

[3]This image has been taken from
http://www.tau.ac.il/~phchlab/experiments_new/SemB01_Hydrogen/02TheoreticalBackground.html

[4]These images have been taken from http://www.itp.uni-hannover.de/~zawischa/ITP/multibeam.html

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 5.4.

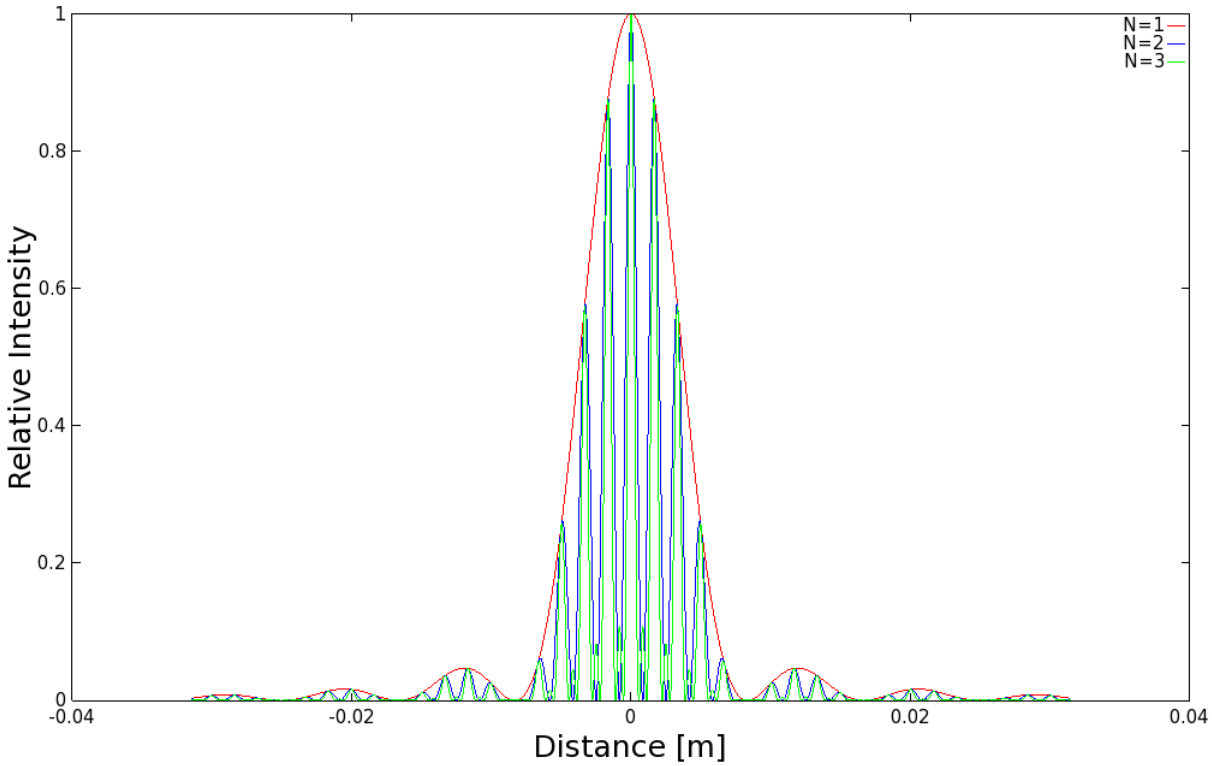


Figure 5.5: Relative intensities of a diffracted beam of light at wavelength $\lambda = 500nm$ on a grating for different number of periods N width slit width of 30 microns and slit separation of 0.15 mm each. The viewer is 0.5m apart from the grating.

5.3 Verifications

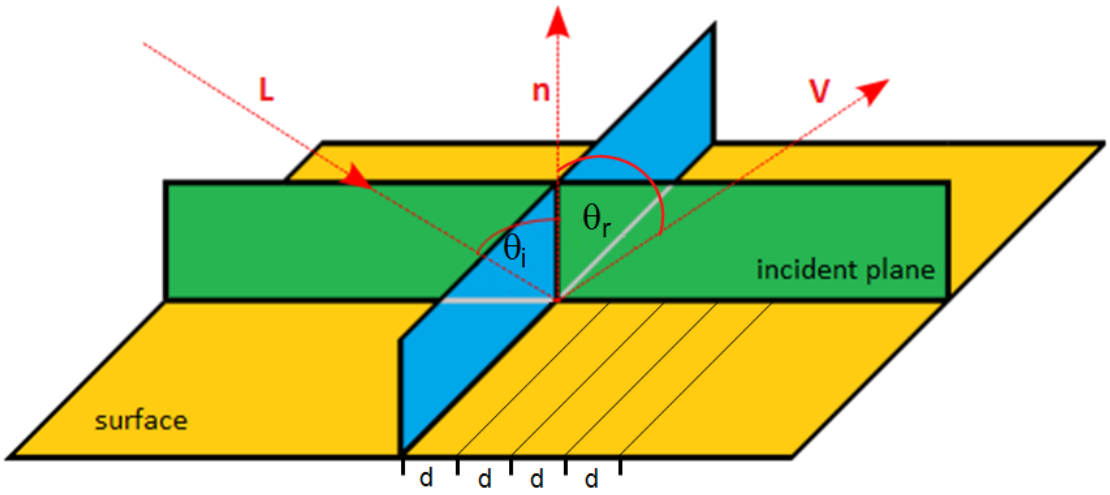


Figure 5.6: Experimental setup for evaluation: A light beam with direction L hits the surface, representing a grating pattern with periodicity d, at the incident plane relative to the surface normal n at angle θ_i and emerges at angle θ_r with viewing direction V.

The physical reliability of our BRDF models has been verified by applying it to a height field for a synthetic blazed grating. Figure 5.6 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_i relative to the normal n along its incident plane. The beam emerges from the surface at the angle θ_r with certain intensity as predicted by our model. Note that actually two angles are necessary in order to define a direction vector, using spherical coordinates. However, in our evaluation we fixed the azimuthal angle of these directional vectors (Further information can be found in section 5.3.1).

In our evaluation we compare the local peak angles predicted by our model from those resulting by the grating equation. The grating equation models the relationship between the grating spacing, the incident light angle and the maximum angle for the diffracted light beams.

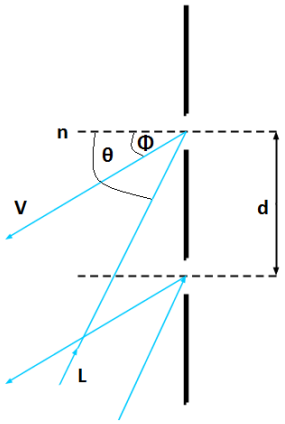


Figure 5.7: Reflecting grating: When the incident light direction is not parallel to its axis at the grating, there is another $sin(\phi)$ involved. See also the grating equation 5.2.

Figure 5.7 shows that if the incident light is not along the axis of the diffraction gratings then it effects the optical path differences. The maximum in intensity is given by the grating equation derived from the equation 5.1 following figure 5.7:

$$sin(\theta_i) = sin(\theta_r) + \frac{m\lambda}{d} \tag{5.2}$$

In our evaluation we are interested in the first order diffraction, i.e. $m = 1$. We further assume that the incident light direction L is given. In contrast the direction of the reflected wave V is a free parameter.

In Mathematics, a three dimensional direction vector is fully defined by two two angles, i.e. it can be represented by spherical coordinates with. Hence, θ_i, is a given constant whereas θ_r is a free parameter for our evaluation simulation. Therefore, we are going to compare the maxima of the 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 Numerical Comparisons

In this section we explain how we evaluated the quality of our BRDF models. For a fixed incident light direction we want to compare the peak viewing angles with maximum reflectance of our methods with those resulting from the grating equation at different wavelengths. For this purpose we implemented the corresponding BRDF model for each of our shading approaches, FLSS, NMM and PQ, in Java. By fixing the azimuth angle[5] of the incident light L- and viewing direction V, we reduced the degrees of freedom these directions, during our evaluation. Thus, any BRDF is then defined by function that expects as input arguments a wavelength λ and the inclination angles of the incident light θ_i - and viewing direction θ_r. The return value of such a function, denoted by $BRDF(\lambda, \theta_i, \theta_r)$, is the intensity value of the corresponding BRDF at these given input arguments.

[5]Each direction vector in space can be expressed by spherical coordinates. Then, such a unit vector is defined by a pair of two angles, the inclination and the azimuth angle. For further information, please have a look at appendix D.2

In our evaluation program we set the incident light angle θ_i equal to $75°$. This allows us to remove the argument θ_i from our BRDF function. Our java program computes each $BRDF$ function at a given discrete wavelength-viewing-angle grid, denoted by $[\Lambda, \Theta]$. The wavelength space $\Lambda = [\lambda_{min}, \lambda_{max}]$ and the viewing angle range $\Theta = [\alpha_{min}, \alpha_{max}]$ of our free parameter θ_r are discretized in equidistant steps whereas their step sizes. These step-sized, denoted by $(\lambda_{step}, \alpha_{step})$, are provided as input arguments for our Java evaluation program.

Next, let us have a closer look how our discrete $[\Lambda, \Theta]$ grid is constructed. The wavelength space Λ, which is ranging from λ_{min} to λ_{max}, is discretized like the following:

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

where $steps_\lambda = \frac{\lambda_{max} - \lambda_{min}}{\lambda_{step}}$. We similarly discretise the viewing angle space Θ by setting an minimal and maximal viewing-angle boundary α_{min} and α_{max}. Then $\lceil \frac{\alpha_{max} - \alpha_{min}}{\alpha_{step}} \rceil$ is the number of angles$steps_\alpha$. And thus, our Θ space it defined like the following:

$$\Theta = \{\alpha = \alpha_{min} + k \cdot \alpha_{step} | k \in \{0, .., steps_\alpha - 1\}\} \tag{5.4}$$

Then, every BRDF java function is applied to the grid $[\Lambda, \Theta]$ and the resulting spectral response is stored in a matrix

$$R = \{BRDF(\lambda_i, \theta_r^j) | i \in Index(\Lambda), \quad j \in Index(\Theta)\} \tag{5.5}$$

The generation process of the evaluation plots, which we discuss in section 5.3.2, is described in algorithm 6. This algorithm takes the matrix R from equation 5.5 as input argument. For the maximal reflectance of our methods at any wavelength, it computes the corresponding peak viewing angles and compares it to the angle resulting from the grating equation.

Algorithm 6 BRDF Evaluation Graph Plotter

Input: R Matrix with $BRDF$ intensity values of (Λ, Θ) grid
Λ discretized wavelength space used to compute R
(α_{min}) minimum value of viewing angle space Θ
(α_{step}) discretization level of viewing angle space Θ
d estimated periodicity of height field
θ_i fixed incident angle

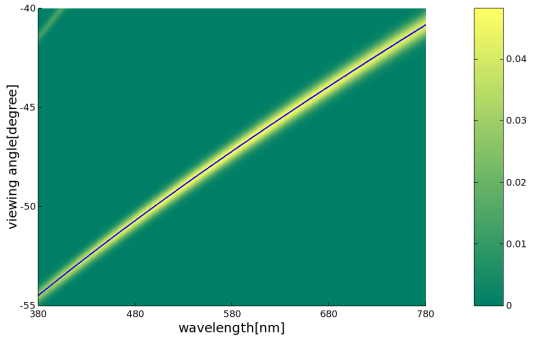
Procedure: $getMaxIntensGridPointsOf(matrix, r)$: get the column-index of the largest intensity value in the row $matrix(r, *)$
$plotPoint(x, y)$: draw a point at (x, y)

Output: Evaluation Plot of given BRDF model applied on given height field

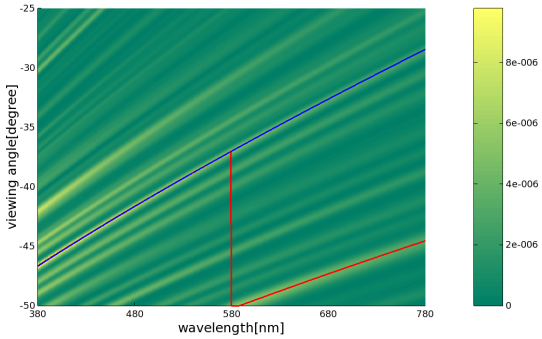
1: **Foreach** $\lambda_k \in \Lambda$ **do**
2: $\quad \widetilde{\alpha} = getMaxIntensGridPointsOf(R, \lambda_k)$ ▷ $\widetilde{\alpha} \equiv$ index viewing angle of max. R
3: $\quad \widetilde{\alpha}_{r_k} = \alpha_{min} + \alpha_{step} \cdot \widetilde{\alpha}$
4: $\quad \theta_{r_k} = asin\left(\frac{\lambda}{d} - sin(\theta_i)\right)$
5: $\quad plotPoint(\lambda_k, \widetilde{\alpha}_{r_k})$ ▷ graph resulting by our BRDF model
6: $\quad plotPoint(\lambda_k, \theta_{r_k})$ ▷ graph resulting by grating equation
7: **end for**

Algorithm 6 iterates over the wavelength space Λ and generates our evaluation plots. For any wavelength it computes the viewing angle of the the maximal reflectance and the angle resulting from the grating equation as defined in equation 5.2. Both angles are then plotted for the current wavelength in the current iteration. In the next section we will discuss the generated evaluation plots.

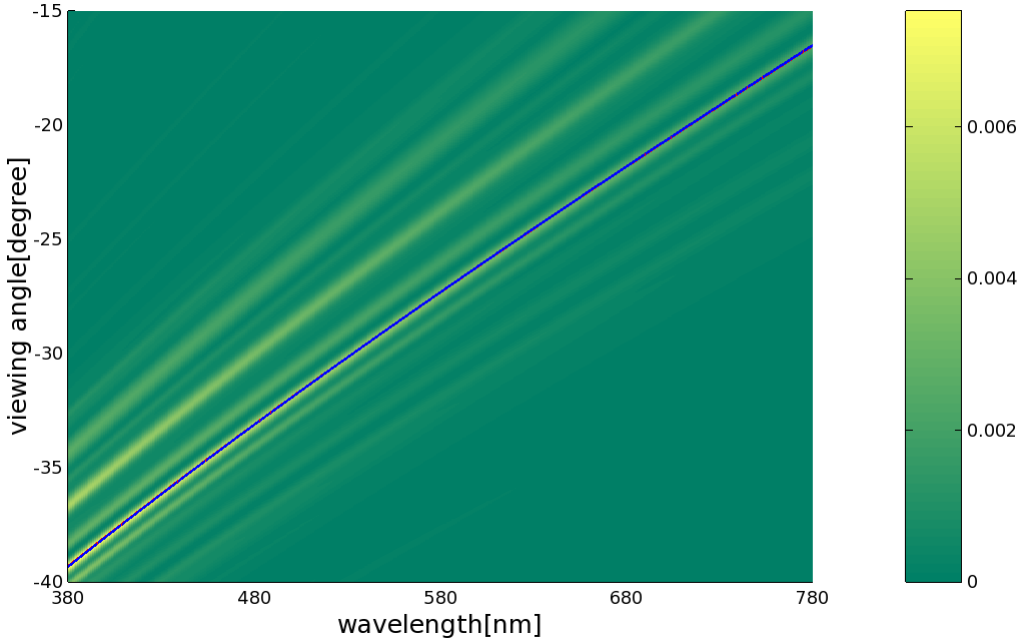
5.3.2 Virtual Testbench



(a) Blazed Grating



(b) Xenopeltis Grating



(c) Elaphe Grating

Figure 5.8: Reflectance obtained by using the FLSS approach described in algorithm 3.

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.

	FLSS		NMM		PQ	
	mean	variance	mean	variance	mean	variance
Blazed Grating	2499.997	0.377	2499.997	0.377	2491.861	248.044
Elpahe Grating	1144.262	0.401	1144.179	0.677	1052.308	49.678
Xenopeltis Grating	1552.27	0.45	-	-	-	-

Table 5.1: Statistics of periodicity d of our used gratings 6.2 estimated[6] by using the grating equation 5.2.

Figure 5.8 shows the reflectance graphs resulting by the FLSS approach described in algorithm 3. This evaluation was applied to a idealized periodic structures, namely to the Blaze- 5.8(a) and to two natural gratings, the Elaphe- 5.8(c) and Xenopeltis grating 5.8(b). For all our evaluation plots, we used an illumination angle of θ_i equal to 75°.

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 each wavelength and then plot these peak viewing angles versus corresponding wavelengths 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.

By rearranging the terms of the grating equation defined in equation 5.2 and using the peek reflectance angle θ_{r_k} derived like in algorithm 6 using the matrix R of equation 5.5, we can compute a periodicity vale d_k for any wavelength λ_k.

$$d_k = \frac{\lambda}{sin(\theta_{r_k}) + sin(\theta_i)} \tag{5.6}$$

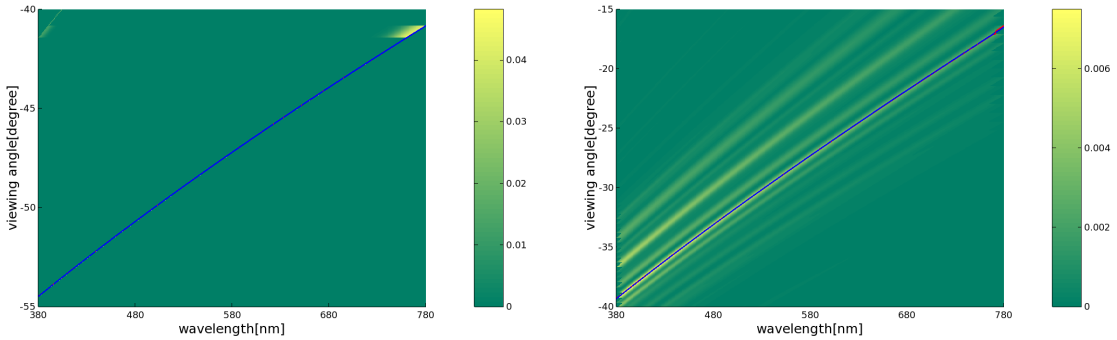
By computing the mean value of these d_k values for all λ in Λ we can compute an estimated periodicity value d. We estimated these periodicity values for every grating structure and every method we are using. Thes corresponding periodicity values tabulated in table 5.1.

The red and blue curve are closely overlapping in our figures 5.8(a) and 5.8(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 5.9 shows the evaluation plots for the NMM approach applied to the Blazed- (see figure 5.9(b)) and the Elaphe-grating (see figure 5.9(b)). The NMM approach is an optimization of the FLSS approach and is discussion in section 4.5.2. The response curve of each plotted NMM

[6]This tabulated periodicity values are estimates similarly like described in the evaluation section of D.S.Dhillon's paper [D.S14].

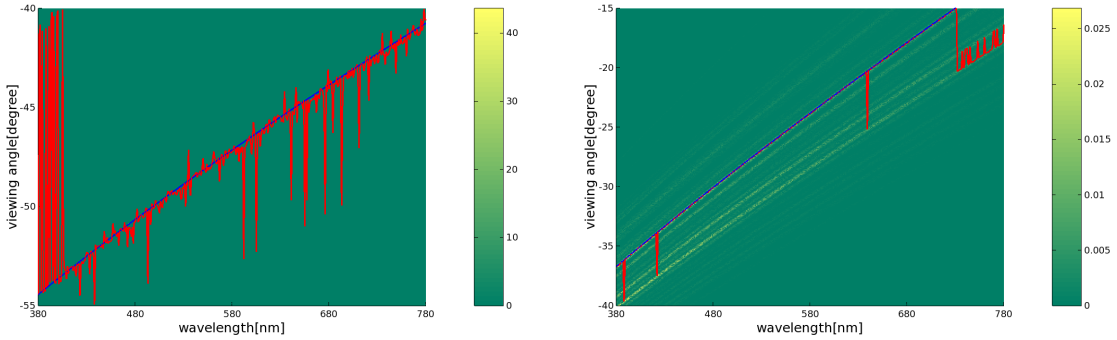
approach graph closely matches the corresponding grating equation curve. Furthermore, the evaluation graphs of the NMM approach look similar to the corresponding evaluation plots of the FLSS approach shown in figure . Thus, this confirms that the NMM optimization works well.



(a) Blaze grating (b) Elaphe grating

Figure 5.9: Reflectance obtained using NMM optimization apporach.

Last, let us consider the evaluation graphs in figure 5.10 for the PQ approach described in algorithm 5. The PQ approach assumes the given grating being periodically distributed on the surface of a shape. For this approach we have plotted evaluation graphs of the Blaze- (See figure 5.10(a)) and Elaphe grating (See figure 5.10(b)). The response curves of both PQ evaluation graphs exhibit 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 well. This could be due to the PQ's assumption that a given height field has to be periodically distributed along the surface. But in general, for natural gratings, this is assumption does usually not hold true. Nevertheless, the red curve fits one of the response curves. We conclude that the PQ approach will produce not accurate results compared to the FLSS approach. Thus, the PQ approach is sub-optimal.



(a) Blaze grating (b) Elaphe grating

Figure 5.10: Reflectance obtained using PQ optimization approach.

Chapter 6

Results

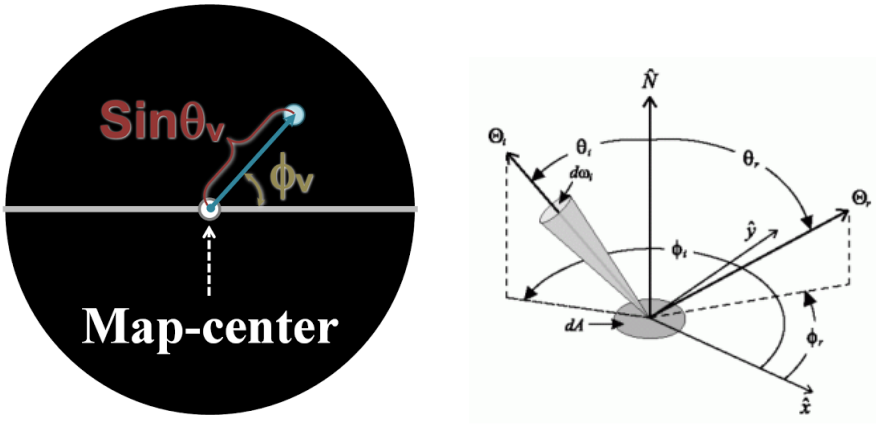
In this chapter we examine the rendered output results of our implementation of our BRDF models applied to different input patches such as Blaze grating or Elaphe 1.1(b) and Xenopeltis 1.1(a) snake nano-scaled surface sheds. We are discussing and comparing both, their BRDF maps 6.1 and the corresponding renderings on a snake geometry like shown in section 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.

6.1 BRDF maps

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 expressed in spherical coordinates (See appendix $D.2$) (θ_v, ϕ_v) and is represented in the map at point

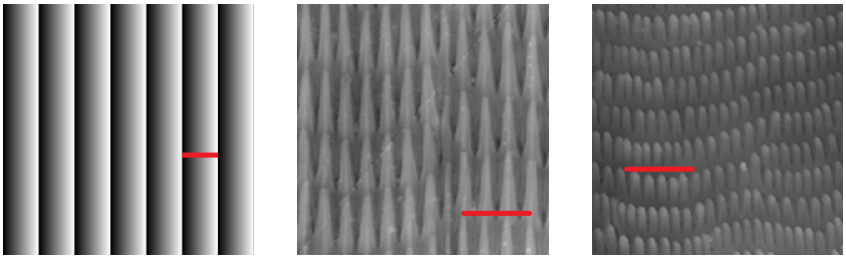
$$(x, y) = (sin(\theta_v)cos(\phi_v), sin(\theta_v)sin(\phi_v)) \tag{6.1}$$

with its origin at the map center. The light direction for normal incidence (θ_i, ϕ_i) has been fixed to $(0, 0)$ for our rendered results.



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

Figure 6.1: BRDF maps[1] for different patches: $\Theta = (\theta_i, \phi_i)$ is the direction of light propagation



(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 (**FLSS** like introduced in section 4.3.2) as described in section 4.3.2 applied on different nanoscale surface gratings as shown in figure 6.2. In Subfigure 6.3(a) we see the BRDF map for the Blazed grating, showing high relative brightness for its first order diffraction, i.e. for the Blazed gratings most of the diffracted spectral energy lies in its first order. Notice that the surface of blazed grating is forming a step structure for which the angle between the step normal and the grating normal is denoted by *blaze angle*. Every blazed grating is manufactred in the Littrow[2] configuration. This means that the blaze angle is chosen such that the diffraction angle and incidence angle are identical. Thus it a blazed grating it has a maximal efficiency for the wavelength of the used light. Higher diffraction modes are still perceivable (second and higher diffraction orders) but with a much lower relative

[1] image source of figure:

- 6.1(a): Taken from D.S.Dhillon's Paper [D.S14]
- 6.1(b): Taken from `http://math.nist.gov/~FHunt/appearance/brdf.html`

[2] For further information please see `http://en.wikipedia.org/wiki/Blazed_grating`.

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 well-formed periodicity along finger direction.

For Xenopeltis surface grating 6.2(c), we observe diffraction along many different, almost 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

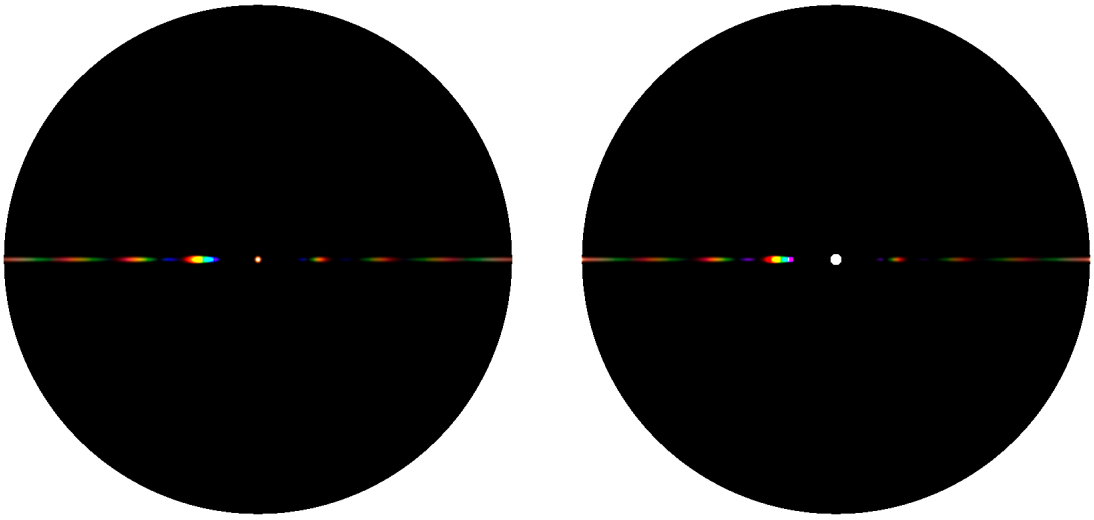
Figure 6.4 shows BRDF maps of all our BRDF models applied on the Blaze grating. Figure 6.4(a) shows the FLSS shading approach result for our blazed grating and it is used in order to compare with our other rendering approaches.

Figure 6.4(b) shows the BRDF map for the NMM approach, introduces in section 4.5.2, which is close to the FLSS approach, verified in section 5.3, just like in the case of corresponding evaluation in figure 5.9(a). Nevertheless there is a small, noticeable difference: For the NMM approach we see a white, circular spot around the map center. Nevertheless, apart from this white spot, the NMM approach resembles the FLSS approach. The reason for this differences is due to the fact that the NMM approach threats the center of a BRDF map as a special case, like described in section 4.5.2. Technically, every location around a small ϵ-circumference from the map center gets white color assigned.

Figure 6.4(c) shows the BRDF map for the PQ approach which relies on sinc-interpolation. The PQ BRDF map and the FLSS results are visual alike. Compared to the evaluation plots in figure 5.10, the BRDF maps even persuade more. Compared to FLSS, one difference we notice is that the first order of diffraction is a little spread for the PQ approach. Without[3] applying a sinc-interpolation, this spreading effect would be even strengthened.

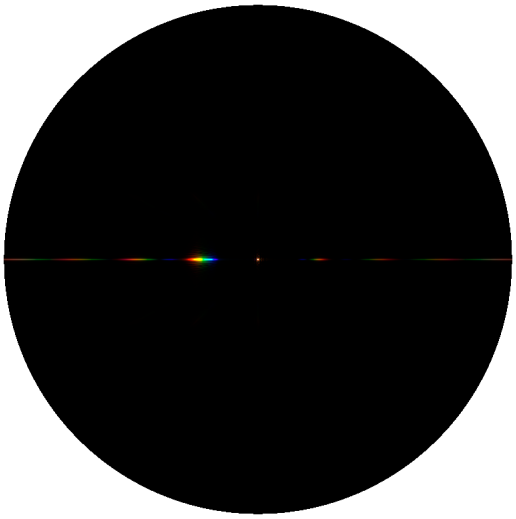
[3]Note that if we do not perform a sinc-interpolation this would correspond to apply a linear interpolation instead.

Last, let us consider figure 6.4(d) which shows the BRDF map produced by using Nvidia Gem's implementation [JG04] of Stam's BRDF model when constraining the y-axis of the BRDF map. This corresponds to a 1d diffraction grating, along the x-axis. This model only uses the spacing d of a given grating. It also always produces highly symmetric results.

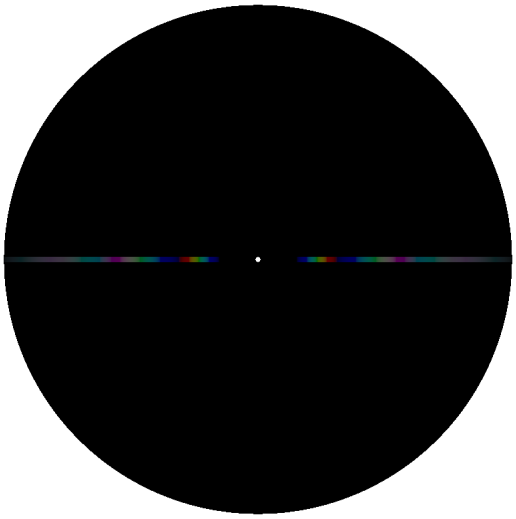


(a) FLSS

(b) NMMS



(c) PQ

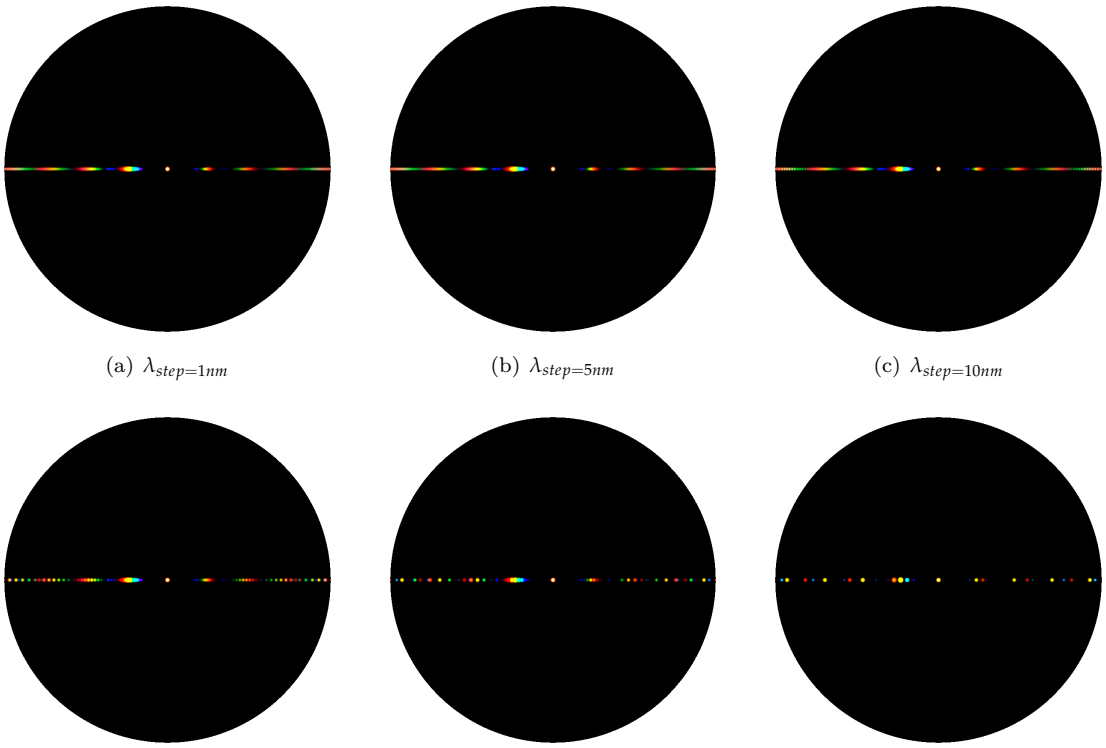


(d) Nvidia Gem

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 FLSS approach applied on the blazed grating and the Elpahe snake shed, respectively. Within our fragment shaders the outermost loop iterates over the range $[380nm, 780nm]$

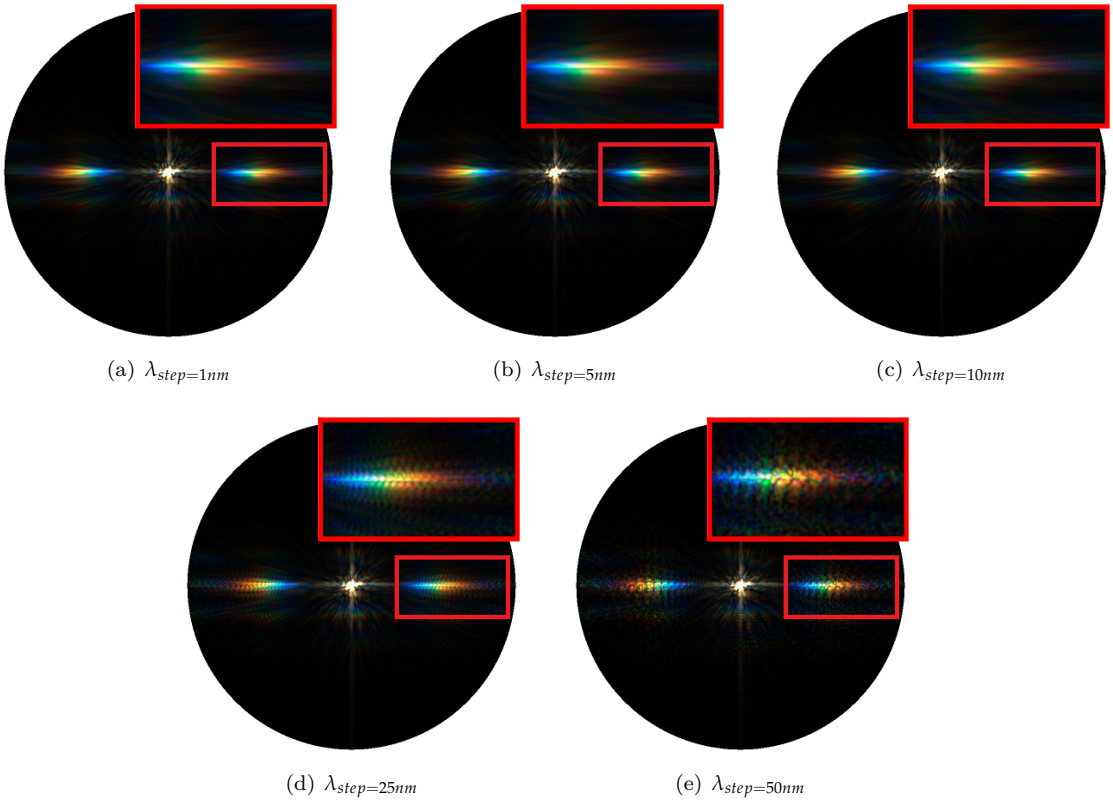
for a given step size λ_{step} to integrate over the wavelength spectrum. Having bigger step sizes implies having fewer λ-samples which will reduce the overall runtime of a shader but, it will also introduce artifacts and therefore lower the overall shading quality. For Elaphe surface grating, artifacts are perceivable when $\lambda_{step} \leq 10nm$. Results produced by using $5nm$ step sizes do not differ from those produced by using $\lambda_{step} = 1nm$. This allows us to set λ_{step} at 5nm. For a Blazed grating we may chose even bigger step sized without losing any rendering quality(see figure 6.5).



(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



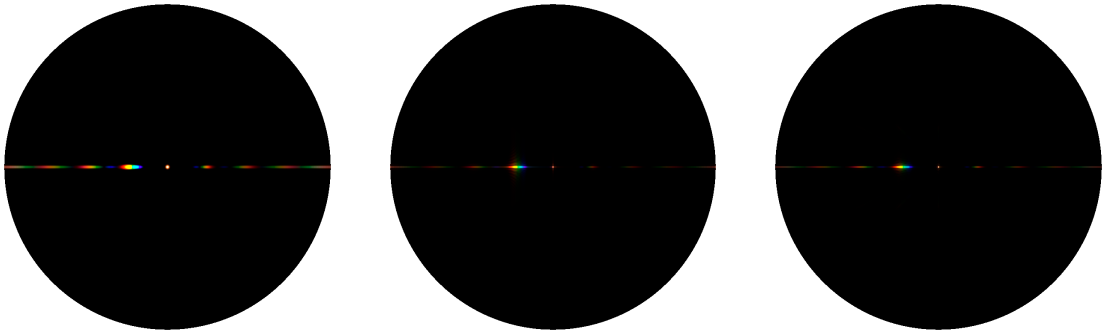
(a) $\lambda_{step=1nm}$ (b) $\lambda_{step=5nm}$ (c) $\lambda_{step=10nm}$

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

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

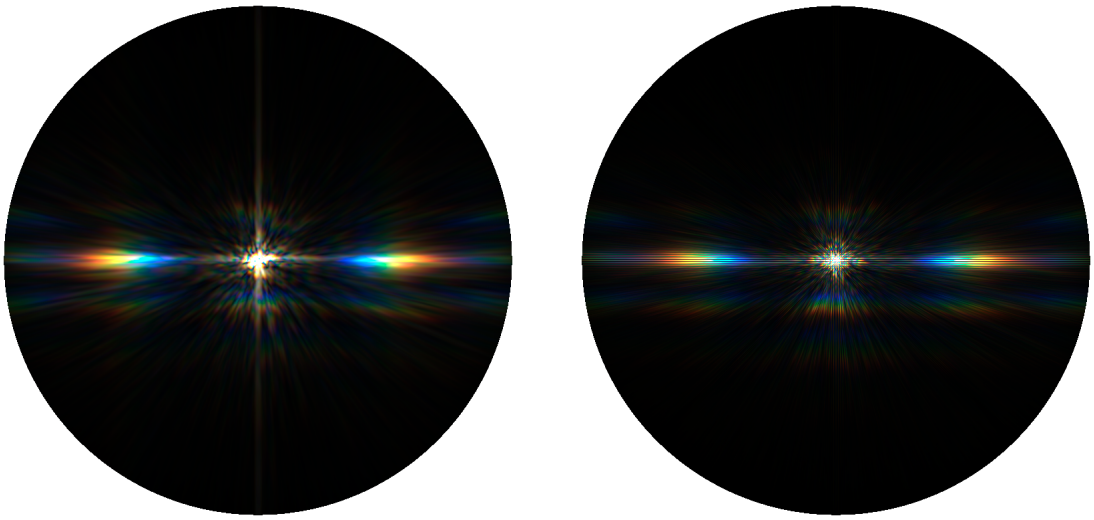
The figures 6.7, 6.8, 6.9 show a comparison of the BRDF maps produced by the FLSS approach (on the left) and the PQ shading approach (on the right) applied on all our patches. For Blazed grating, as already mentioned, we notice that both approaches, FLSS and PQ, resemble each other. We also notice that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

In genral, a Blazed Grating is manufactured in a way that a large fraction of the incident light is diffracted preferentially into the first order. Therefore, most of the energy in its BRDF map lies in the first order of diffraction at its blaze angle. This implies that largest portion of the color contribution, visible on the corresponding BRDF map, lies at that angle. In figure 6.7, in contrast to the results produced by the FLSS approach, we see color fringes at the first order modes in the BRDF map produced by our PQ approach. This implies that the PQ approach does not produce reliable results which also affirms our evaluation plots shown in figure 5.10. A similar argumentation hold true for the PQ approach, when we do not apply we a sinc-interpolation like shown in figure 6.7(c).



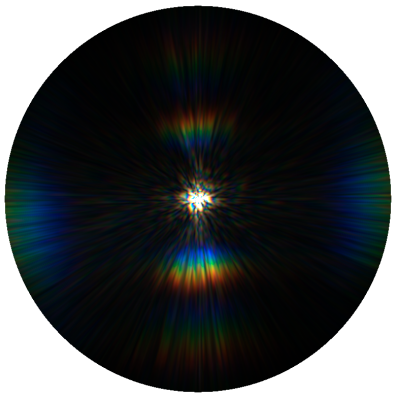
(a) FLSS Approach: Blazed grating (b) PQ Approach: Blazed grating (c) PQ Sinc Interpolation Approach: Blazed grating

Figure 6.7: A comparison between the PQ- and the FLSS approach applied on an Blazed grating.

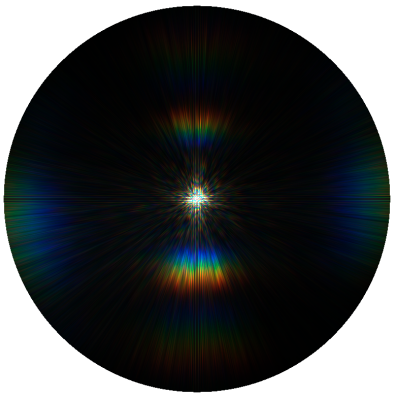


(a) FLSS Approach: Elaphe grating (b) PQ Approach: Elaphe grating

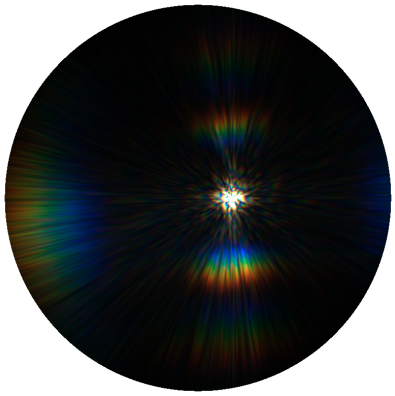
Figure 6.8: A comparison between the PQ- and the FLSS approach applied on an Elaphe grating.



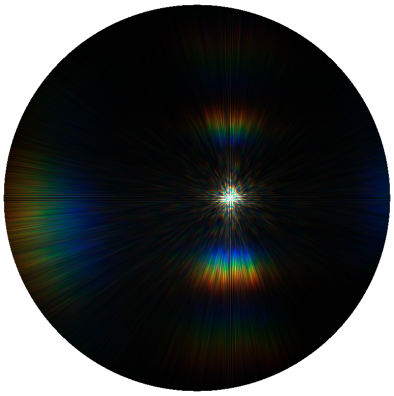
(a) FLSS Approach: Xenopeltis grating $\theta_i = 0°$



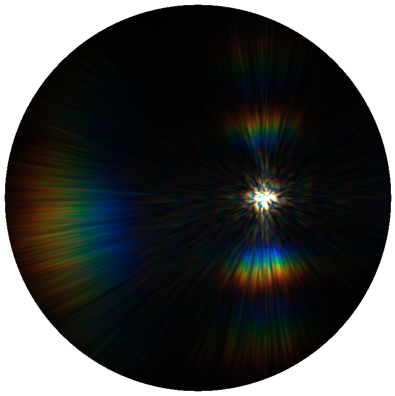
(b) PQ Approach: Xenopeltis grating $\theta_i = 0°$



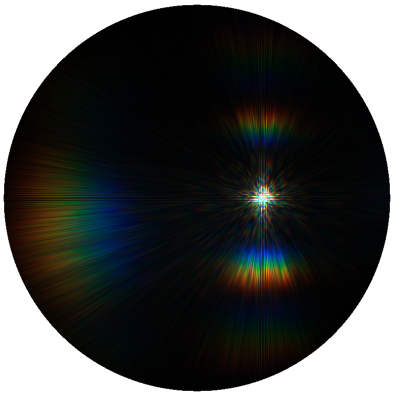
(c) FLSS Approach: Xenopeltis grating $\theta_i = 10°$



(d) PQ Approach: Xenopeltis grating $\theta_i = 10°$



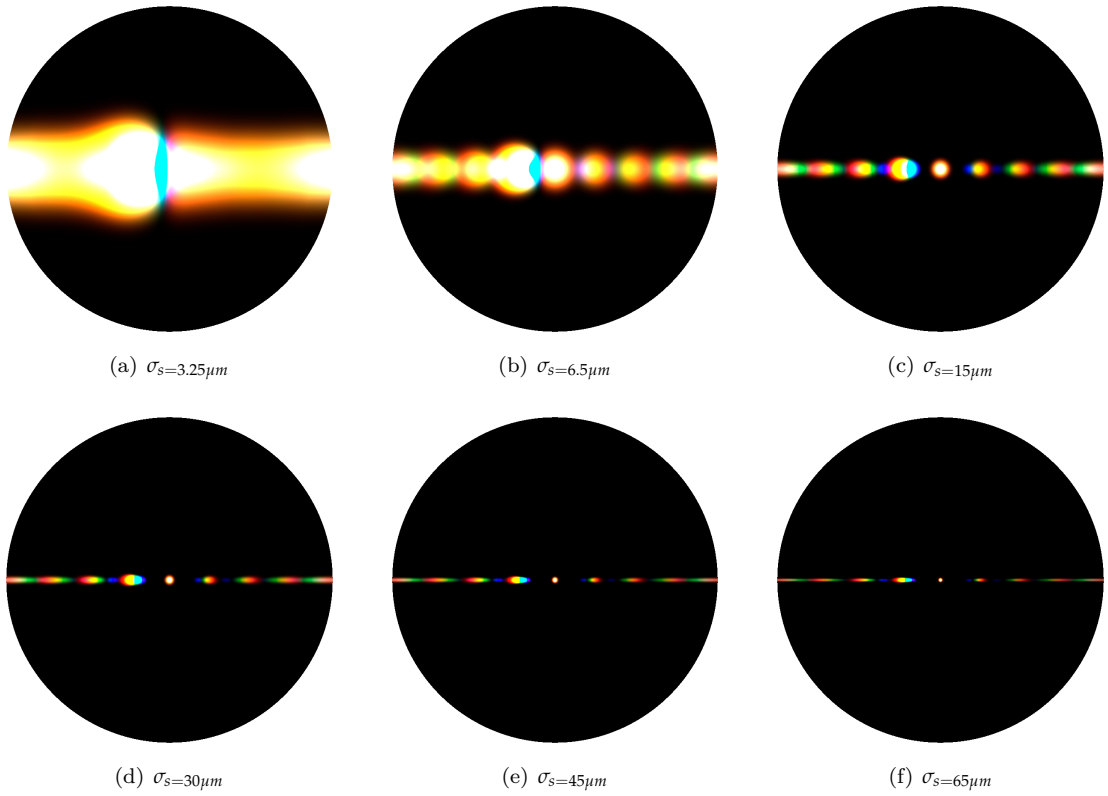
(e) FLSS Approach: Xenopeltis grating $\theta_i = 20°$



(f) PQ Approach: Xenopeltis grating $\theta_i = 20°$

Figure 6.9: A comparison between the PQ- and the FLSS approach applied on an Xenopeltis grating.

Figure 6.10 shows BRDF maps for the full lambda sampling approach applied to the Blazed grating, while varying the value for the spatial variance σ_s. This akin to changing the coherence length for the incident light. The lower the coherence length, the fewer interacting grating periods 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 with periodicity of $2.5\mu m$: Different σ_s

Figures 6.11 and 6.12 show the BRDF maps the reference-, FLSS approach using different values for N in the taylor series approximation. For both input patches we clearly visually observe the convergence of the taylor series for higher values of N. We visually observe convergence of the Taylor series for all our patches a very large value of N[4].

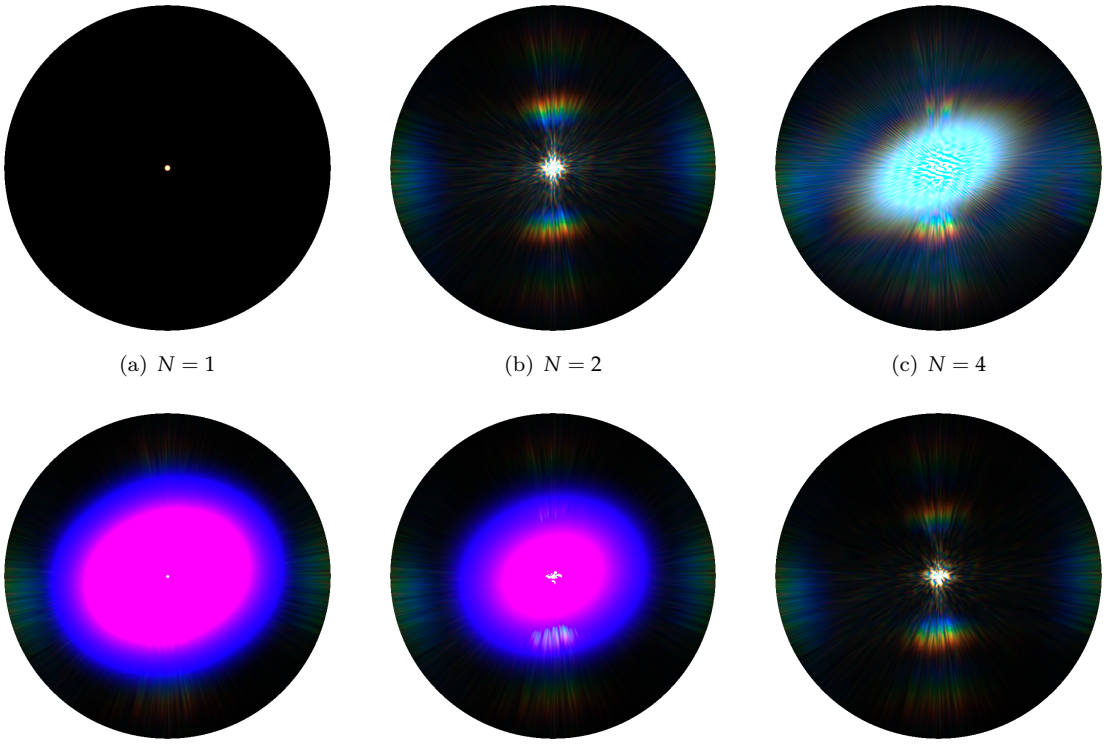
Like discussed in section 3.4 there exists a certain value of N for which our approach converges. For all our shading approaches, applied on our gratings, we visually observed a convergence of their BRDF maps when using $N \geq 39$ DFT terms. Furthermore, for a Blazed grating it satisfies to use only $N \geq 7$ - and for an Elaphe grating only $N \geq 9$ DFT terms. Notice, that these numbers of required DFT terms were empirically determined by trial and error strategy.

However, by making use of taylor error term estimates, like introduced in the appendix section C.1, we can derive an upper bound for N. Since this computation is dependent on many aspects, such as on the grating spacing, the pixel-width correspondence, the used lambda space for sam-

[4]Using N equal to 39 lead to visual convergence for all our used gratings.

pling, it is usually simpler to determine empirically actual values for N.

In algorithm 1 we compute the DFT terms of a provided height field h raised to the power of the imaginary number i times an integer, i.e. we evaluate the expression $DFT(h)^n \cdot i^n$. Since we multiply our height field by i^n and then apply the DFT operator, basically, there exist four possible convergence images, each having its own convergence radius.



(a) $N = 1$ (b) $N = 2$ (c) $N = 4$

(d) $N = 24$ (e) $N = 26$ (f) $N = 40$

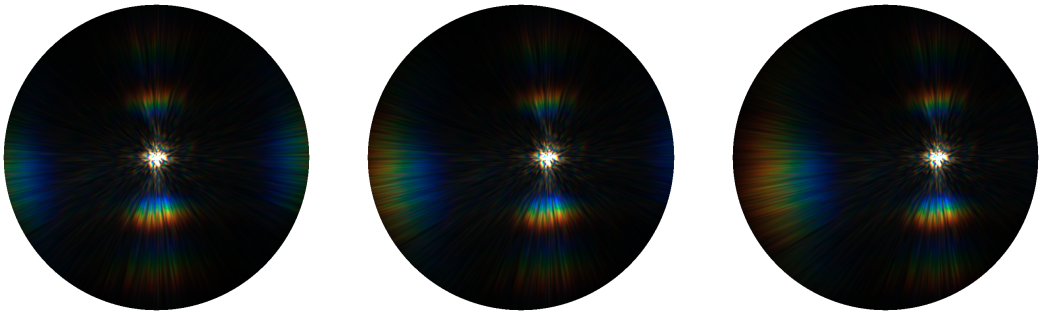
Figure 6.11: Blazed grating at $65\mu m$: N Taylor Iterations

(a) $N = 1$ (b) $N = 2$ (c) $N = 5$

(d) $N = 10$ (e) $N = 11$ (f) $N = 40$

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

Figure 6.13 shows the BRDF maps of the FLSS 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) Xenopeltis grating $\theta_i = 0°$ (b) Xenopeltis grating $\theta_i = 10°$ (c) Xenopeltis grating $\theta_i = 20°$

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

6.2 Rendering 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 is typical for diffraction colors observed in nature. For rendering we are going to rely on our FLSS approach. Unfortunately, this approach is rather slow and can barely be considered as being interactively performing. Nevertheless, we have introduced some optimizations in order to make it interactive.

All rendered results shown in this section are produced by the FLSS approach since we have proven its validity in figure 5.8. Therefore, we can trust its renderings and may consider them as being accurate. Furthermore, as support of our evaluation plots regarding the NMM approach, that are shown in figure 5.9, we also show results produced by the NMM approach.

The Laboratory of Artificial and Natural Evolution in Geneva provided us by a triangular mesh of a snake. This mesh was produced by a 3d scan of a Elaphe snake species and consists of 11696 vertices and 22950 faces. Note that, for all our renderings, we used this snake mesh.

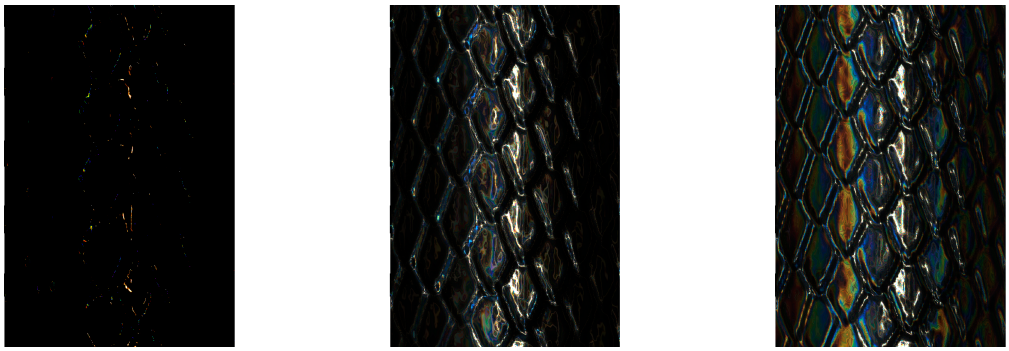
Among all the snake species under consideration, their macroscopic geometry is highly similar. Only the geometry of their nano-structures varies and is responsible for a snake's iridescence. Thus, we can use the same snake surface model to render diffraction for different species. Table 6.1 lists the 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
Graphics Clock	1150 MHz
Graphics Memory	4096 MB
Graphics Memory Bandwidth	230.4 GB/s MHz

Table 6.1: Hardware specifications of the machine used to render snake surface. Statistics are provided using the tool *NVIDIA Geforce Experience*.

Figure 6.14 shows renderings produced by the FLSS approach applied on our snake 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 i.e. is non-flat, we can expect rather less diffraction color contribution like shown in figure 6.14(b).

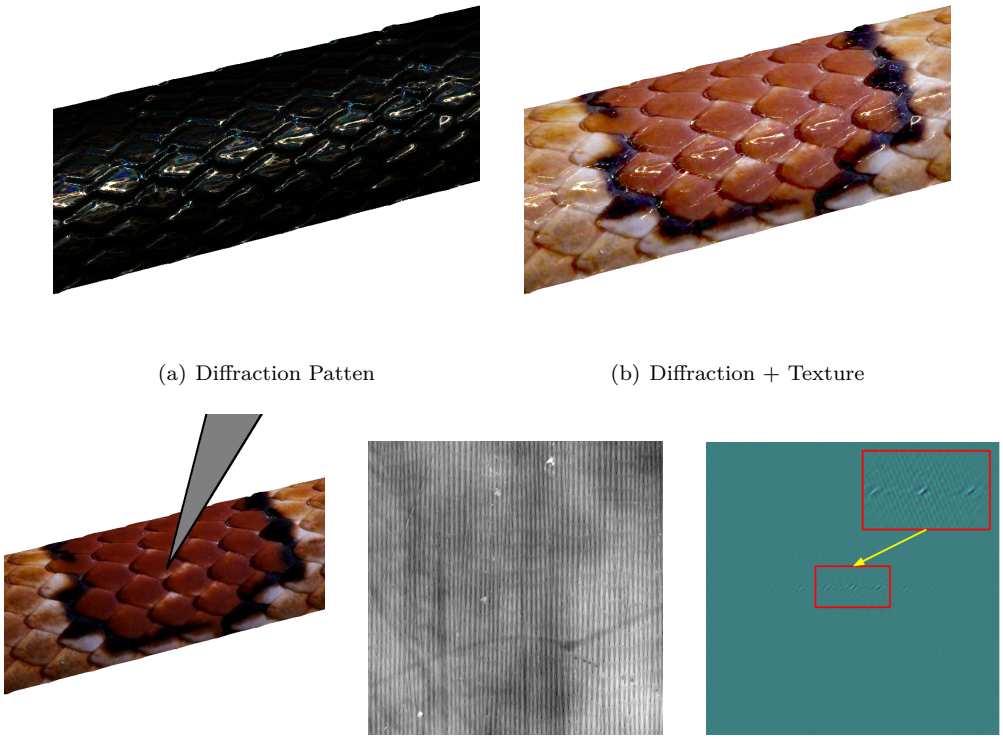
In contrast, For both the renderings, we see colorful patterns on the skin of our snake species, Elaphe and Xenopeltis, due to the effect of diffraction. We see much less colorful patterns for Elaphe like shown in figure 6.14(b) than for Xenopeltis like shown in figure 6.14(c). This is consistent with the observations in the real world as shown in figure 1.1. As observable figure 6.2(b), the substructures (the finger like structures) in the height field of a Elaphe snake skin are not very regularly aligned along the y-axis. This is why the Elaphe species is less iridescent than the other specie. The Xenopeltis snake has a brownish body with no pigmentation, which makes the iridescence more spectacular than on Elaphe.



(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 shows a set of subfigures for rendering the effect of diffraction produced by the FLSS approach (used as our reference approach), applied on our snake mesh using the Elaphe nano structure. Figure 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 nano-scale surface structure 6.15(d).

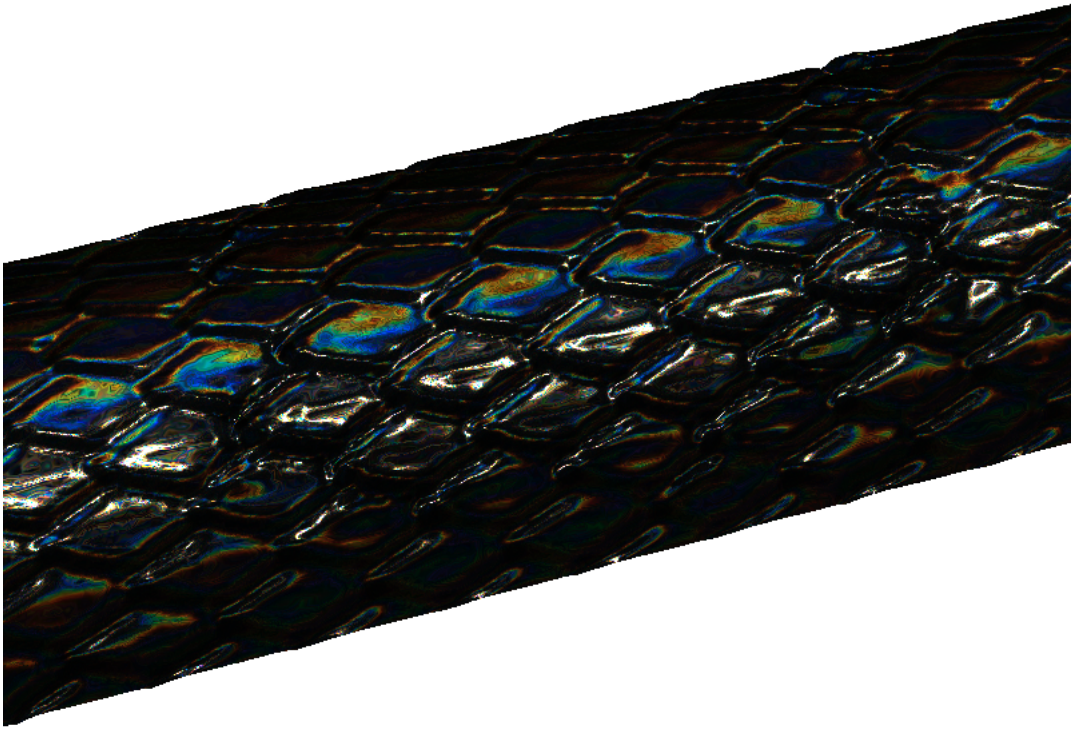


(a) Diffraction Patten (b) Diffraction + Texture

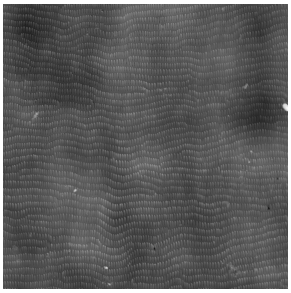
(c) Texture with Light Direction (d) Nanostructure (e) 8th DTF Term

Figure 6.15: Diffraction for Elaphe snake skin produced by our reference apporach.

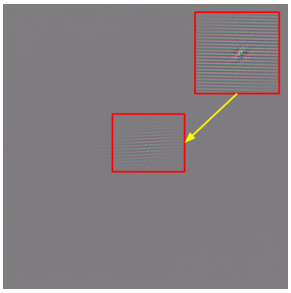
Figure 6.16 shows a set of subfigures for the effect of diffraction for the Xenopeltis snake surface. For Xenopeltis we see quite a lot color contribution due the phenomenon of diffraction like shown in figure 6.16(c). Comparing this to a real image 1.1(a) we notice much resemblance regarding the reflectance strength and colorful pattern.



(a) Diffraction Patten



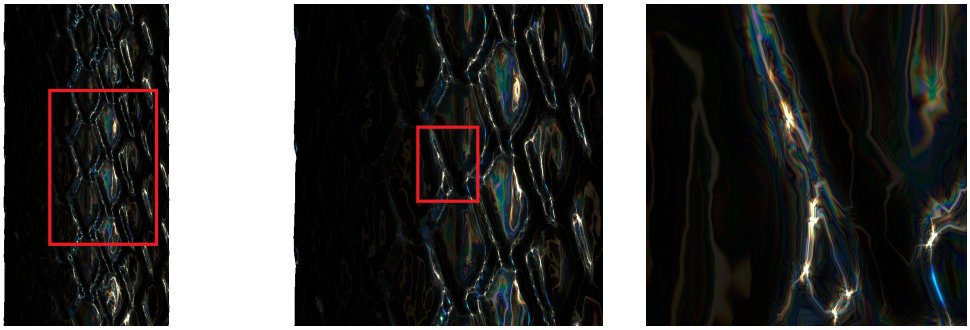
(b) Nanostructure



(c) 3th DTF Term

Figure 6.16: Diffraction for Xenopeltis snake skin produced by our reference approach.

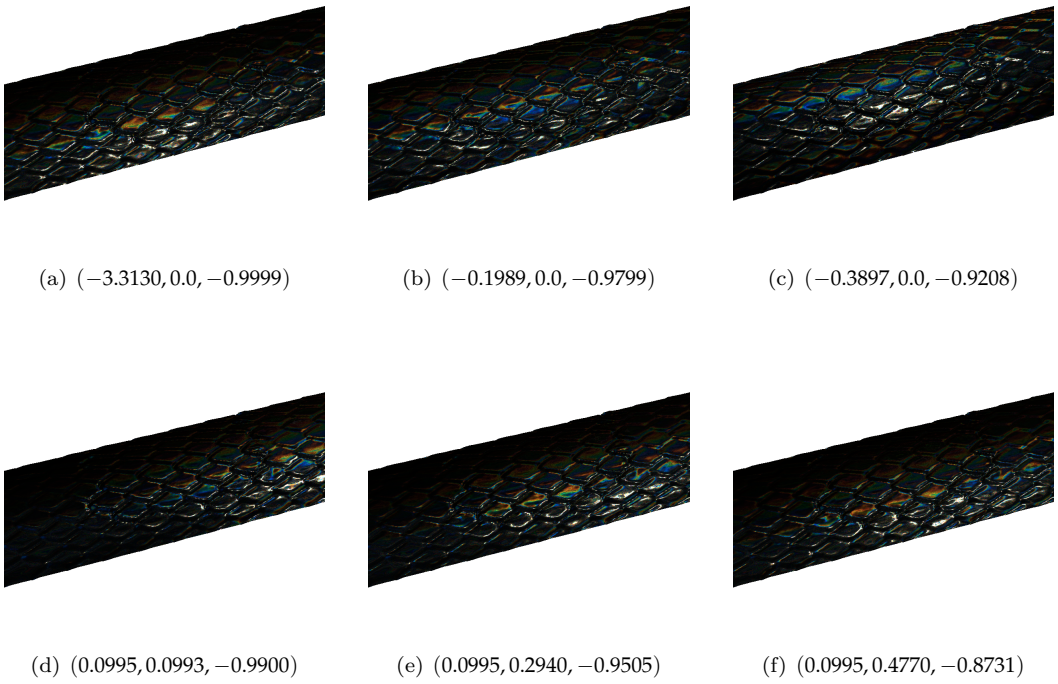
Figure 6.17 shows the diffraction pattern for Elaphe snake shed at different zoom levels for fixed incident light and viewing direction. We changed the zoom-levels by adjusting the field of view angle of our camera. For each image in this figure, the one to its right side is a five times zoomed-in version of the region within its red box. The close up perspectives exhibit complex and colorful diffraction patterns.



(a) $zoom = 1$ (b) $zoom = 5x$ (c) $zoom = 25x$

Figure 6.17: Diffraction on Elaphe snake skin grating: Different camera zoom levels by varying the field of view.

Figure 6.18 shows how the diffraction pattern changes when the incident light direction is moved slightly. This Figure gives us an impression what kind of complex, perspective-dependent pattern the diffraction phenomenon produces.



(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 Elaphe 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.

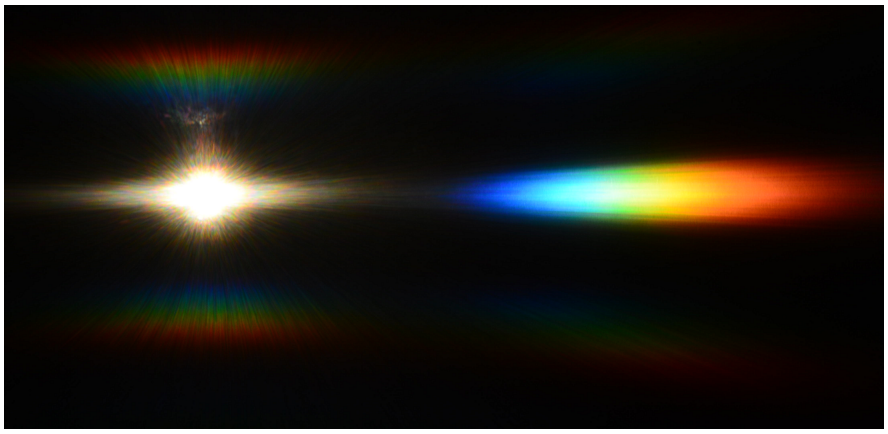
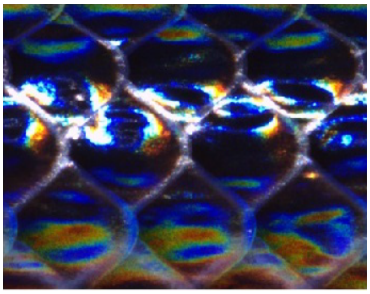
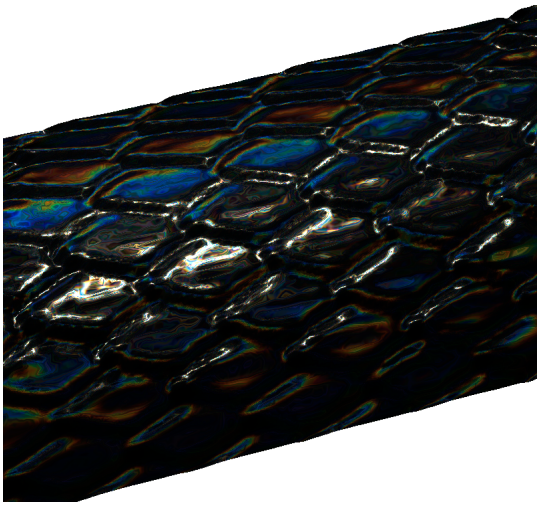


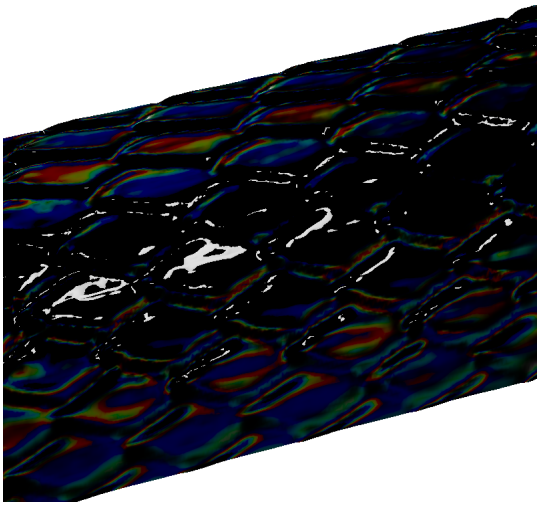
Figure 6.19: Diffraction Elaphe: experimental setup



(a) Xenopeltis Image



(b) FLSS



(c) Stam

Figure 6.20: Attempt to reproduce a real world image: FLSS- vs Stam's Approach

Chapter 7

Conclusion

7.1 Summary

The aim of this thesis was to simulate structural colors produced by a directional light source upon diffraction by quasyperiodic nanostructures in nature. Our goal also included to use explicitly measured nano-scaled patches to simulate these structural colors at interactive rates. For this purpose we developed a BRDF model based on J. Stam's work described in his paper about diffraction shaders. Furthermore, we implemented our model as a rendering program.

Rendering the effect of diffraction for a measured nanostructure, described as a discrete height field was previously not done and may therefore be considered as a novelty.

Our BRDF model is based on a Taylor series approximation and a Gaussian windowing approach, formulated in a way such that many computationally expensive terms[1] in our model can be precomputed. These precomputed terms can be accessed by performing a lookup during the shading stages, which reduce the overall runtime complexity of our shader. Nevertheless, our basic implementation (FLSS) is not interactive for most of the present GPU platforms. Therefore, we also devised some optimization into my NMM shading approach, to render scenes interactively. We also formulated a completely different approach, the PQ shading approach, which performs sinc-interpolation, assuming the given height field patch is periodically distributed on a surface. Since the height field of natural diffraction gratings is only a quasiperiodic function, the PQ approach does not appropriately model the reality and thus does not produce reliable results. In addition, we evaluated the quality of all our shading approaches applied on different surface gratings. Finally, we produced some renderings BRDF maps and renderings on a snake mesh using all our shading approaches applied using different surface gratings.

For future work we could think of extending our model in a way such that it can handle multilayer diffraction grating. This would allow us to simulate structural colors due to multilayer interferrence wave-effects, which can be observed on wings of certain butterfly species.

This thesis contributed also to a follow up paper which discusses a fast implementation of the FLSS shading approach. This paper also contains a complete comparison with Stam's Shading approach. For further information, please have a look at the paper [D.S14].

[1] Such as computing different powers of the two dimensional Fourier transformation of the given height field.

7.2 Personal Experiences

I always knew that eventually I will write a thesis in the field of computer graphics. After having attended the *Computer Graphics* class held by Mr. Zwicker, I was certain that I will definitely write my thesis at the Computer Graphics Group at the University of Bern. Since I already acquired a Minor degree in Mathematics I and am very interested in topics like numerics, differential equations and differential geometry, I asked for a thesis subject involving some Mathematics. I have to admit that, after having read J. Stam's paper about diffraction shaders at the very beginning of my thesis, I thought this topic would exceed my knowledge in Mathematics and Physics. And it actually did, but I decided not to give up.

During working on my thesis I learnt new concepts in Physics relating to wave-interference, diffraction of waves, diffraction gratings and deepened my knowledge of wave theory for light. Regarding Mathematics I learned quite a lot about the different kind of Fourier transforms, about Windowing techniques for signal processing and about BRDF models. I felt it as a satisfactory experience to use and apply all the knowledge which I have acquired during the time as a bachelor student. Furthermore, this thesis gave me the opportunity to program quite a lot. Hence I could 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 Dhillon for his continuous support of my study, his patience, motivation, enthusiasm, and knowledge. His guidance and active support helped me quite a lot while deriving our BRDF model, developing it and evaluating the shaders and while writing this thesis.

Last but not least, I would like to thank my mother, Manuela Single and my brother Patrik Single and also to my close friend, Radischa Iyadurai for supporting me morally 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:

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

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

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

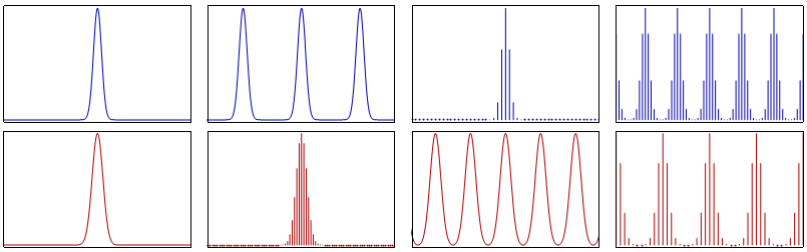


Figure A.1: Relationship[1] between the continuous Fourier transform and the discrete Fourier transform: Left column: A continuous function (top) and its Fourier transform $A.1$ (bottom). Center-left column: Periodic summation of the original function (top). Fourier transform (bottom) is zero except 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: $\mathbb{R}^n \to \mathbb{C}$ 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

In order to prove equation 3.29 from section 3.4 we have to show the following: For any $N \in \mathbb{N}$ and

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

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** relying on Taylor's Theorem[1] It is possible to show that the series $\sum_{n=0}^{N} \frac{(ikwh)^n}{n!} \mathcal{F}\left\{h^n\right\}(\alpha, \beta)$ 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,

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

[1]Please have a look at `http://en.wikipedia.org/wiki/Taylors_theorem` in order to see a proper definiton of the ratio test.

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 $\mathbf{M}$ 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 $\mathbf{a}$ 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

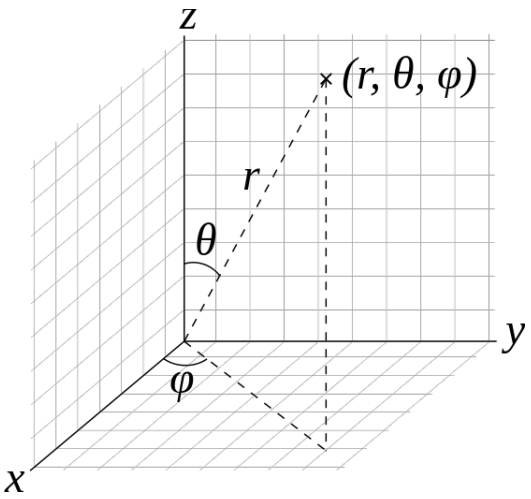


Figure D.1: Illustration[1] of Spherical coordinates $(r,,)$ radius r, polar (inclination) angle θ, and azimuthal angle ϕ.

To define a spherical coordinate system like shown in figure $D.1$, two angles, tho polar angle θ and the azimuthal ϕ plus a radius r are required. Then the Cartesian coordinates (x,y,z) may be retrieved from the spherical coordinates like the following: $\forall \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{R}^3 : \exists r \in [0,\infty) \exists \phi \in [0,2\pi] \exists \theta \in [0,\pi]$ 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

[1] image source of figure $D.1$ `http://en.wikipedia.org/wiki/Spherical_coordinate_system`

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 $D.2$, 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.

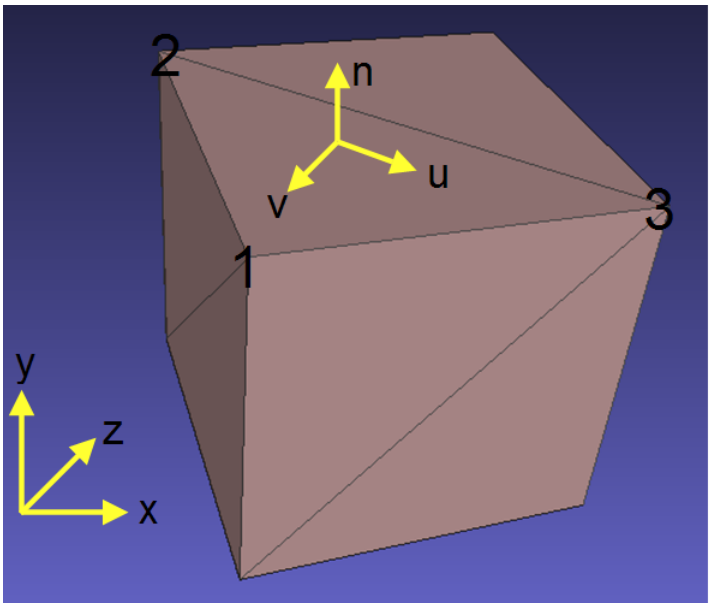


Figure D.2: Cube in world space (x, y, z) showing the tangen space (u, v, n) of its face $(2, 1, 3)$

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

[DD14] D.S. Dhillon, et a.: Interactive Diffraction from Biological Nanostructures. In: *EUROGRAPHICS 2014/ M. Paulin and C. Dachsbacher* (2014), January

[D.S14] D.S.Dhillon, M.Single I.Gaponenko M.C. Milinkovitch M. J.Teyssier: Interactive Diffraction from Biological Nanostructures. In: *Submitted at Computer Graphics Forum* (2014)

[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