

# **Diffraction Shader**

## **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**

# Inhaltsverzeichnis

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Goals . . . . .	1
1.3	Previous work . . . . .	2
1.4	Overview . . . . .	2
<b>2</b>	<b>Theoretical Background</b>	<b>4</b>
2.1	Definitions . . . . .	4
2.1.1	Diffraktion . . . . .	4
2.1.2	Radiometry . . . . .	6
2.1.3	Signal Processing Basics . . . . .	9
2.2	Thesis Basis: J.Stam's Paper about Diffraction Shader . . . . .	12
2.3	Derivations . . . . .	14
2.3.1	BRDF formulation . . . . .	14
2.3.2	Relative BRDF . . . . .	16
2.3.3	Taylor approximation for BRDF . . . . .	18
2.3.4	Sampling: Gaussian Window . . . . .	21
2.3.5	Final Expression . . . . .	22
2.4	Alternative Approach . . . . .	22
2.4.1	Interpolation . . . . .	25
<b>3</b>	<b>Implementation</b>	<b>27</b>
3.1	Precomputations in Matlab . . . . .	28
3.2	Java Renderer . . . . .	30
3.3	GLSL Diffraction Shader . . . . .	32
3.3.1	Vertex Shader . . . . .	32
3.3.2	Fragmant Shader . . . . .	32
3.4	Technical details . . . . .	35
3.4.1	Texture lookup . . . . .	35
3.4.2	Texture Blending . . . . .	37
3.4.3	Color Transformation . . . . .	37
<b>4</b>	<b>Evaluation Data Acquisition</b>	<b>39</b>
4.1	Data Acquisition . . . . .	39
4.2	Diffraction Gratings . . . . .	39
4.3	Evaluation . . . . .	43
4.3.1	Precomputation . . . . .	44
4.3.2	Data evaluation . . . . .	44

<b>5 Results</b>	<b>46</b>
5.1 BRDF maps . . . . .	47
5.2 Snake surface geometries . . . . .	53
5.3 Snake surface geometries . . . . .	59
<b>6 Conclusion</b>	<b>60</b>
6.1 Further Work . . . . .	60
6.1.1 References . . . . .	60
6.2 Acknowledgment . . . . .	60
<b>A Appendix</b>	<b>61</b>
A.1 Schlick's approximation . . . . .	61
A.2 Spherical Coordinates . . . . .	61
<b>List of Tables</b>	<b>62</b>
<b>List of Figures</b>	<b>62</b>
<b>List of Algorithms</b>	<b>64</b>
<b>Bibliography</b>	<b>65</b>

# Kapitel 1

## Introduction

### 1.1 Motivation

In Nature, coloring mostly comes from the inherent colors of materials but sometimes colorization has a pure physical origin such as the effect diffraction or interference of light. Both phenomenon are causing the so called structural coloration, which is the production of color through the interaction of visible light with micriscopically structured surfaces. Color production is due to wave interference with quasiperiodic structures whose periodicity leads to interaction with visible light. Therefore we perceive color when the different wavelengths composing white light are selectively interfered with by matter (absorbed, reflected, refracted, scattered, or diffracted) on their way to our eyes, or when a non-white distribution of light has been emitted. In animals, such as feathers of birds and the scales of butterflies, interference is created by a range of photonic mechanisms, including diffraction grating, selective mirrors, photonic crystals. The connection between microscopic structures and coloration has been observed by Robert Hooke in the early seventeenth century. The discovery of the wave nature of light led to the conclusion that the cause for the coloration lies in wave interference.

In the field of computer graphics, many researchers have been attempted rendering of structural colors by formulating a the bidirectional reflectance distribution function (BRDF) for this purpose. But most of the techniques so far, however, are either too slow for interactive rendering or rely on simplifying assumption, like modeling light as rays, to achieve real-time performance, which are not able capturing the essence of diffraction at all.

### 1.2 Goals

The purpose of this thesis is to simulate realistically by rendering structural colors caused by the effect of diffraction on different biological structures in realtime. We focus on structural colors generated by diffraction gratings, in particular our approach applies to surfaces with quasiperiodic structures at the nanometer scale that can be represented as heightfields. Such structures are found on the skins of snakes, wings of butterflies or the bodies of various insects. we restrict ourselves and focus on different snake skins which are acquired

nanoscaled heightfields using atomic force microscopy.

In order to achieve our rendering purpose we will rely J. Stam's formulation of a BRDF which basically describes the effect of diffraction on a given surface assuming one knows the heightfield of this surface and will further extend this. Apart from Stam's approach, which models the heightfield as a probabilistic superposition of bumps and proceeds to derive an analytical expression for the BRDF, our BRDF representation takes the heightfield from explicit measurement. I.E. in our case, those heightfields are small patches of the microstructured surfaces (in nano-scale) taken by AFM of snake skin patches provided by our collaborators in Geneva.. So this approach is closer to real truth, since we use measured surfaces instead of statistical surface profile.

Therefore, this work can be considered as an extension of J. Stam's derivations for the case one is provided by a explicit height field on a quasiperiodic structure.

Real time performance is achieved with a representation of the formula as a power series over a variable related to the viewing and lighting directions. Values closely related to the coefficients in that power series are precomputed.

The contribution is that this approach is more broadly applicable than the previous work. Although the previously published formula theoretically has this much flexibility already, there is a novel contribution in demonstrating how such generality can be leveraged in practical implementation

### 1.3 Previous work

stam, hooke, see our paper, see stams paper, see own research.

Robert Hooke = observed connection between microscopic structures and colorisation wave nature of light led to conclusion that the cause for the colouration lies in wave interference.

previous

In computer graphics literature, Stam was the first to develop reflection models based on wave optics called diffraction shaders, that can produce colorful diffraction effects. His approach is based on a far field approximation of the Kirchhoff integral. He shows that for surfaces represented as nanoscale heightfields it is possible to derive their BRDF as the Fourier transformation of a function of the heightfield. Nevertheless, this formulation is not immediately useful for efficient rendering of measured complex nanostructures since this would require the on-the-fly evaluation of and integration over Fourier transforms of the heightfield that depend on the light and viewing geometry. In his derivations, Stam models heightfields as probabilistic superpositions of bumps forming periodic like structures. This provides him an analytical identity for this class of heightfields. However, biological nanostructures are way more complex and do not lend themselves to this simplified statistical model.

follow ups

### 1.4 Overview

The remainder of this thesis is organized as follows: due to the fact that this thesis has a rather advanced mathematical complexity the first part of chapter 2

introduces some important definitions which are required in order to be able to follow the derivation in the last third of chapter 2. Before starting the derivations a brief summary of J. Stam's Paper about diffraction shaders is provided since this whole thesis is based on his BRDF representation. Our derivations itself are listed step-wise, whereas there is a final representation provided by the end of chapter 2. Chapter 3 addresses the practical part of this thesis, the implementation of our diffraction model, explaining all precomputation steps and how rendering is performed in our developed framework for this thesis. Chapter 4 evaluates the validity of our brdf model by rendering the effect of diffraction of a blaze grating for a fixed reflection direction for every incident direction. Chapter presents the rendering results of our shader for real world parameters, applying our shader on AFM taken snake patches. The thesis is rounded off by providing a conclusion.

## Kapitel 2

# Theoretical Background

### 2.1 Definitions

#### 2.1.1 Diffraction

ADD IMAGES

A wave is the spatial propagating variation or rather perturbation or also an oscillation of a location-and time-dependent physical quantity. In Mathematics, when talking about a wave, we are referring to the so called wavefunction  $y(x, t) = A\sin(\omega t - kx)$  which is a solution for the so called wave-equation. In general, this function depends on the location  $x$  and in time  $t$ . The maximal amplitude / magnitude of a wave is denoted by  $A$ . The phase of wave indicates in which part within its period with respect to a reference point in time / location the wave is located. Naturally occurring waves are uncommonly purely monochromatic waves, rather an overlapping of many waves with different wavelengths, which is denoted by interference. Interference is a phenomenon in which two waves superimpose to form a resultant wave of greater or lower amplitude. All portions of these corresponding wavelengths are forming the so called spectrum. For example sunlight which is an overlapping of many electromagnetic monochromatic waves of different wavelengths, ranging continuously from infrared across visible light to ultraviolet. Thereby, many effects may occur like interference - by overlapping two waves they either will amplify or cancel out partially or even totally each other in some location. Waves can experience in a different way a change in their form of appearance through reflexion, transmission, refraction or diffraction. In this thesis we are interested in particular in the bending of waves the so called effect of diffraction, which cannot be modeled using the standard ray theory of light.

When a wavefront is occluded partially by an obstacle, the wave is not only moving in the direction provided by the ray geometry, but also occurs complex wave-movements outside of the geometric ray boundaries. This occurrence is called diffraction and occurs on the border of the wave's obstacle.

Generally, the effect of diffraction occurs on the border of obstacle whenever a propagating wave encounters that obstacle. But its effects are generally most pronounced for waves whose wavelength is roughly similar to the dimensions of the diffracting objects. Conversely, if wavelength is hardly similar, then there is almost no diffraction at all. This implies diffraction occurs mostly when the

surface detail is highly anisotropic. If the obstructing object provides multiple, closely spaced openings, a complex pattern of varying intensity can result. This is due to the superposition, or interference, of different parts of a wave that travels to the observer by different paths.

Example: Light rays passing through a small aperture will begin to diverge and interfere with one another. This becomes more significant as the size of the aperture decreases relative to the wavelength of light passing through, but occurs to some extent for any aperture or concentrated light source.

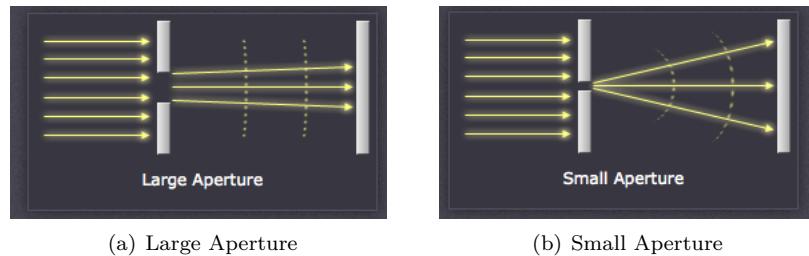


Abbildung 2.1: Diffraction: Single Slit Example

Since the divergent rays now travel different distances, some move out of phase and begin to interfere with each other — adding in some places and partially or completely canceling out in others. This interference produces a diffraction pattern with peak intensities where the amplitude of the light waves add, and less light where they subtract. If one were to measure the intensity of light reaching each position on a line, the measurements would appear as bands similar to those shown below.

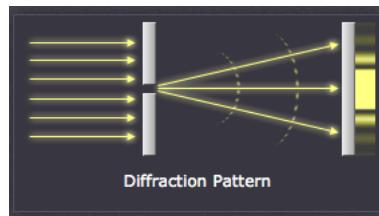


Abbildung 2.2: Diffraction Pattern

In general interference produces colorful effects due to the phase differences caused by a wave traversing thin media of different indices of refraction.

The effects of diffraction are often seen in everyday life. The most striking examples of diffraction are those that involve light; for example, the closely spaced tracks on a CD or DVD act as a diffraction grating to form the familiar rainbow pattern seen when looking at a disk.

In optics, a diffraction grating is an optical component with a periodic structure, which splits and diffracts light into several beams travelling in different directions. The directions of these beams depend on the spacing of the grating and the wavelength of the light so that the grating acts as the dispersive element. The relationship between the grating spacing and the angles of the

incident and diffracted beams of light is known as the grating equation. See chapter Evaluation Data Acquisition for further insight.

### 2.1.2 Radiometry

Light is fundamentally a propagation form of energy, so it is useful to define the SI unit of energy which is joule (J). To aid our intuition, let us describe radiometry in terms of collections of large numbers of photons. A photon can be considered as a quantum of light that has a position, direction of propagation and a wavelength  $\lambda$  measured in nanometers. A photon travels in a certain speed, denoted by  $v = \frac{c}{n}$ , that depends only on the refractive index  $n$  of the medium through which it propagates and the speed of light  $c$ . This allows us to define the frequency  $f = \frac{c}{\lambda}$ . The amount of energy  $q$  carried by a photon is given by the following relationship:  $q = hf = \frac{hc}{\lambda}$  where  $h$  is the Plank's constant.

#### Spectral Energy

If there is a large collection of photons given, their total energy  $Q = \sum_i q_i$  is the sum of energies of each photon  $q_i$  within the collection. But how is the energy distributed across wavelengths? One way in order to determine this distribution is to order all photons by their associated wavelength and then make a histogram from them. This is achieved by a discretization of their spectrum and combine all photons which will fall into the same interval, i.e. compute the sum for each interval from the energy of all their photons. By dividing such an interval by its length, denoted by  $Q_\lambda$ , we get a relatively scaled interval energy, which is called spectral energy and it is an intensive quantity. Intensive quantities can be thought of as density functions that tell the density of an extensive quantity at an infinitesimal point.

#### Power

Power is the estimated rate of energy production for light sources and is measured in the unit Watts, denoted by  $Q$ , which is another name for joules per second. Since power is a density over time, it is well defined even when energy production is varying over time. As with energy, we are really interested in the spectral power, measured in W/nm, denoted by  $\Phi_\lambda$

#### Irradiance

The term irradiance comes into place when we are interested in *how much light hits a given point*. In order to answer this question, we must make use of a density function. Let  $\Delta A$  a finite area sensor that is smaller than the light field being measured. The spectral irradiance  $E$  is just the power per unit area  $\frac{\Delta\Phi}{\Delta A}$  which is

$$E = \frac{\Delta\Phi}{\Delta A \Delta t \Delta \lambda} \quad (2.1)$$

Thus the units of irradiance are  $J m^{-2} s^{-1} (nm)^{-1}$ , the power per unit surface area.

### Radiance

TODO: SHOW IMAGE: SOLID angle WIKI : <http://en.wikipedia.org/wiki/Radiance>  
 CGSlides2012 – 6.shading Book : *Fundamentals of computergraphics*

Although irradiance tells us how much light is arriving at a point, it tells us little about the direction that light comes from. To measure something similar to what we see with our eyes we need to be able to associate the quantity *how much light with a specific direction*.

Radiance is a measure of the quantity of radiation that passes through or is emitted from a surface and falls within a given solid angle in a specified direction. Think of it as the energy carried along a narrow beam of light. This means radiance characterizes the total emission of reflectance. It indicates how much of the power emitted by a reflecting surface will be received by an optical system looking at the surface from some given angle of view. Formally, this leads us to the following definition of radiance:

$$L(\omega) = \frac{d^2\Phi}{dAd\Omega\cos(\theta)} \approx \frac{\Phi}{\Omega A\cos(\theta)} \quad (2.2)$$

where  $L$  is the observed radiance in the unit energy per area per solid angle, which is  $Wm^{-2}sr^{-1}$  in direction  $\omega$  which has an angle  $\theta$  between the surface normal and  $\omega$ ,  $\Phi$  is the total flux or power emitted,  $\theta$  is the angle between the surface normal and the specified direction,  $A$  is the area of the surface and  $\Omega$  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.

### BRDF

The bidirectional reflectance distribution function, in short BRDF, denoted as  $f_r(\omega_i, \omega_r)$  is a four dimensional function that defines *how light is reflected at an opaque surface*. The function takes a negative directed incoming light direction,  $\omega_i$ , and outgoing direction,  $\omega_r$  as input argument. Both are defined with respect to the surface normal  $\mathbf{n}$ . Hence A BRDF returns the ratio of reflected radiance exiting along  $\omega_r$  to the irradiance incident on the surface from direction  $\omega_i$ , which is formally:

$$BRDF(\omega_i, \omega_r) = f_r(\omega_i, \omega_r) \quad (2.3)$$

$$= \frac{dL_r(\omega_r)}{dE_i(\omega_i)} \quad (2.4)$$

$$= \frac{dL_r(\omega_r)}{L_i(\omega_i)\cos(\theta_i)d\omega_i} \quad (2.5)$$

$L$  is the reflected radiance,  $E$  is the irradiance and  $\theta_i$  is the angle between  $\omega_i$  and the surface normal,  $\mathbf{n}$ . The index  $i$  indicates incident light, whereas the index  $r$  indicates reflected light.

### Spectral Rendering

In Computer Graphics, spectral rendering is where a scene's light transport is modeled considering the whole span of wavelengths instead of R,G,B values (still relating on geometric optic, which ignore wave phase). The motivation is that real colors of the physical world are spectrum; trichromatic colors are only inherent to Human Visual System.

In Computer Graphics, when talking about Spectral Rendering, we are referring to use the whole span of wavelengths instead just using RGB values in order for rendering a scene's light transport. The motivation for using the whole wavelength spectrum is due to the fact that trichromatic colors are only inherent to human visual system and therefore many phenomenons are poorly represented just using trichromy.

### Colors

In order to see how crucial the role human vision plays, we only have to look at the definition of color: *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.* - Wyszecksi and Siles, 2000 mentioned in Computer Graphics Fundamentals Book.

An eye consists of photoreceptor cells, cones and rods. Cones are responsible for color perception. The human eye has three kinds of cone cells, with spectral sensitivity peaks in short (S, 420–440 nm), middle (M, 530–540 nm), and long (L, 560–580 nm) wavelengths. In principle, any color sensation in human color perception can therefore be described by just three parameters, corresponding to levels of stimulus of the three types of cone cells.

A color space maps a range of physically produced colors from mixed light to an objective description of color sensations registered in the eye in terms of tristimulus values.

Tristimulus values associated with a color space can be conceptualized as amounts of three primary colors in a tri-chromatic additive color model. In some color spaces (including LMS and XYZ spaces), the primary colors used are not real colors, in the sense that they cannot be generated with any light spectrum.

Due to distribution of cones in human eye, the tristimulus values depend on one's field of view. CIE defined a color mapping function called standard observer, to eliminate this variable represent an average human chromatic response.



Abbildung 2.3: Plots of our color matching functions we used for rendering

The CIE's color matching functions  $\bar{x}(\lambda)$ ,  $\bar{y}(\lambda)$ ,  $\bar{z}(\lambda)$  are the numerical description of the chromatic response of the observer (described above). 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:

$$\begin{aligned} X &= \int_{380}^{780} I(\lambda) \bar{x}(\lambda) d\lambda \\ Y &= \int_{380}^{780} I(\lambda) \bar{y}(\lambda) d\lambda \\ Z &= \int_{380}^{780} I(\lambda) \bar{z}(\lambda) d\lambda \end{aligned}$$

where  $\lambda$ , is the wavelength of the equivalent monochromatic light (measured in nanometers).

It is not possible to build a display that corresponds to CIE XYZ, 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 any other color space defined, described as linear transformations.

### 2.1.3 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.

### 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:

$$\mathcal{F}_{FT}\{f\}(w) = \int_{\mathbb{R}^n} f(x)e^{-iwt} dt \quad (2.6)$$

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

$$\mathcal{F}_{FT}^{-1}\{f\}(w) = \int_{\mathbb{R}} \mathcal{F}\{f\}(w)e^{iwt} dt \quad (2.7)$$

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

$$\mathcal{F}_{DTFT}\{f\}(w) = \sum_{-\infty}^{\infty} f(x)e^{-iwxk} \quad (2.8)$$

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:

$$\mathcal{F}_{DFT}\{f\}(w) = \sum_{n=0}^{N-1} f(x)e^{-iwx_n k} \quad (2.9)$$

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  $f(t)$ .

$$f(t) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty} a_n \cos(nt) + \sum_{n=1}^{\infty} b_n \sin(nt) \quad (2.10)$$

where

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

### Convolution

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

$$(f * g)(t) = \int_{\mathbb{R}^n} f(t)g(t-x) dx \quad (2.11)$$

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 2.4. Table 2.1 tells what Fourier transformation operator has to be applied to which kind of input signal and what properties its resulting Fourier transform will have.

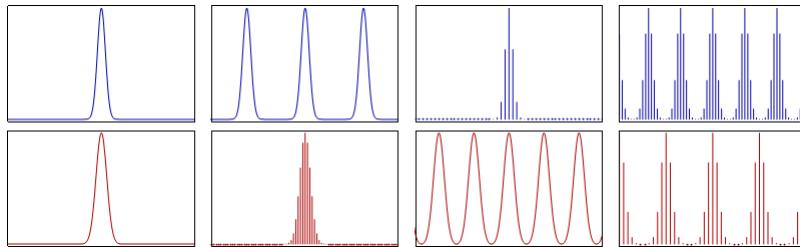


Abbildung 2.4: Relationship between the continuous Fourier transform and the discrete Fourier transform: Left column: A continuous function (top) and its Fourier transform (bottom). Center-left column: Periodic summation of the original function (top). Fourier transform (bottom) is zero except at discrete points. The inverse transform is a sum of sinusoids called Fourier series. 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 (bottom) computes discrete samples of the continuous DTFT. The inverse DFT (top) is a periodic summation of the orginal samples.

## 2.2. THESIS BASIS: J.STAM'S PAPER ABOUT DIFFRACTION SHADER<sup>2</sup>

Spacial signal $f(t)$ is	Operator	Transformed frequency signal $\hat{f}(\omega)$ is
continuous and periodic in $t$	FS	only discrete in $\omega$
only continuous in $t$	FT	only continuous in $\omega$
only discrete in $t$	DTFT	continuous and periodic in $\omega$
discrete and periodic in $t$	DFT	discrete and periodic in $\omega$

Tabelle 2.1: Fourier operator to apply for a given spatial input signal and the properties of its resulting output signal in frequency space

### 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:

$$\mathcal{T}(f; a)(x) = \sum_{n=0}^{\infty} \frac{f^n(a)}{n!} (x - a)^n \quad (2.12)$$

## 2.2 Thesis Basis: J.Stam's Paper about Diffraction Shader

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 (ADD REF TO WIKI), 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  $\psi_1 = e^{ik\mathbf{x} \cdot \mathbf{s}}$  on the surface is known, the field  $\psi_2$  everywhere else away from the surface can be computed. More precisely, we want to compute the wave  $\psi_2$  equal to the reflection of an incoming planar monochromatic wave  $\psi_1 = e^{ik\omega_i * x}$  traveling in the direction  $\omega_i$  from a surface  $S$  to the light source. Formally, this can be written as:

$$\psi_2(\omega_i, \omega_r) = \frac{ik e^{iKR}}{4\pi R} (F(-\omega_i - \omega_r) - (-\omega_i + \omega_r)) \cdot I_1(\omega_i, \omega_r) \quad (2.13)$$

with

## 2.2. THESIS BASIS: J.STAM'S PAPER ABOUT DIFFRACTION SHADE

$$I_1(\omega_i, \omega_r) = \int_S \hat{\mathbf{n}} e^{ik(-\omega_i - \omega_r) \cdot \mathbf{s} d\mathbf{s}} \quad (2.14)$$

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:

$$\sigma^0 = 4\pi \lim_{R \rightarrow \infty} R^2 \frac{\langle |\psi_2|^2 \rangle}{\langle |\psi_1|^2 \rangle} \quad (2.15)$$

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:

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

$$BRDF = \frac{1}{4\pi} \frac{1}{A} \frac{\sigma^0}{\cos(\theta_i) \cos(\theta_r)} \quad (2.16)$$

where  $\theta_i$  and  $\theta_r$  are the angles of incident and reflected directions on the surface with the surface normal  $n$ . See 2.5.

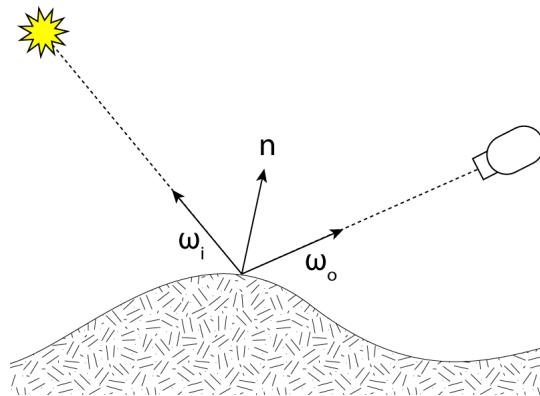


Abbildung 2.5:  $\omega_i$  points toward the light source,  $\omega_r$  points toward the camera,  $n$  is the surface normal

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

$$(u, v, w) = -\omega_i - \omega_r \quad (2.17)$$

explicitly and introduces the equation:

$$I(ku, kv) = \int_S \hat{\mathbf{n}} e^{ik(u, v, w) \cdot \mathbf{s} d\mathbf{s}} \quad (2.18)$$

which is a first simplification of 2.14. Note that the scalar  $w$  is the third component of 2.17 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 2.18 he will end up with:

$$\mathbf{I}(ku, kv) = \int \int (-h_x(x, y), -h_y(x, y), 1) e^{ikwh(x, y)} e^{ik(ux+vy)} dx dy \quad (2.19)$$

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

$$p(x, y) = e^{iwh(x, y)} \quad (2.20)$$

which will allow him to further simplify his equation 2.19 to:

$$\mathbf{I}(ku, kv) = \int \int \frac{1}{ikw} (-p_x, -p_y, ikwp) dx dy \quad (2.21)$$

where he used that  $(-h_x(x, y), -h_y(x, y), 1) e^{ikwh(x, y)}$  is equal to  $\frac{(-p_x, -p_y, ikwp)}{ikw}$  using the definition of the partial derivatives applied to the function 2.20.

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 2.19:

$$\mathbf{I}(ku, kv) = \frac{1}{w} P(ku, kv) \cdot (u, v, w) \quad (2.22)$$

Let us consider the term  $g = (F(-\omega_i - \omega_r) - (-\omega_i + \omega_r))$ , which is a scalar factor of 2.13. 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 2.22 into 2.13 he will end up with:

$$\psi_2(\omega_i, \omega_r) = \frac{ike^{iKR}}{4\pi R} \frac{2F(1 + \omega_i \cdot \omega_r)}{w} P(ku, kv) \quad (2.23)$$

By using the identity 2.16, this will lead us to his main finding:

$$BRDF_\lambda(\omega_i, \omega_r) = \frac{k^2 F^2 G}{4\pi^2 Aw^2} \langle |P(ku, kv)|^2 \rangle \quad (2.24)$$

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

$$G = \frac{(1 + \omega_i \cdot \omega_r)^2}{\cos(\theta_i) \cos(\theta_r)} \quad (2.25)$$

## 2.3 Derivations

### 2.3.1 BRDF formulation

Lets assume we have given an incoming light source with solid angle  $\omega_i$ ,  $\theta_i$  is its angle of incidence,  $\omega_r$  is the solid angle for the reflected light,  $\lambda$  wavelength,  $\Omega$  is the hemisphere we of integration for the incoming light. Then, we are able to formulate a BRDF by using its definition:

$$\begin{aligned}
f_r(\omega_i, \omega_r) &= \frac{dL_r(\omega_r)}{L_i(\omega_i)\cos(\theta_i)d\omega_i} \\
\Rightarrow f_r(\omega_i, \omega_r)L_i(\omega_i)\cos(\theta_i)d\omega_i &= dL_r(\omega_r) \\
\Rightarrow \int_{\Omega} f_r(\omega_i, \omega_r)L_i(\omega_i)\cos(\theta_i)d\omega_i &= \int_{\Omega} dL_r(\omega_r) \\
\Rightarrow L_r(\omega_r) &= \int_{\Omega} f_r(\omega_i, \omega_r)L_i(\omega_i)\cos(\theta_i)d\omega_i \quad (2.26)
\end{aligned}$$

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

$$L_{\lambda}(\omega) = I(\lambda)\delta(\omega - \omega_i) \quad (2.27)$$

where  $I(\lambda)$  is the intensity of the relative spectral power for the wavelength  $\lambda$ . Since all light rays are parallel whenever we are provided by a directional light source and we can think of radiance as a measure of the light emitted from a particular surface location into a particular direction, above's radiance identity will follow immediately. By plugging this identity into our current rendering equation, we will get:

$$\begin{aligned}
L_{\lambda}(\omega_r) &= \int_{\Omega} BRDF_{\lambda}(\omega_i, \omega_r)L_{\lambda}(\omega_i)\cos(\theta_i)d\omega_i \\
&= BRDF_{\lambda}(\omega_i, \omega_r)I(\lambda)\cos(\theta_i) \quad (2.28)
\end{aligned}$$

where  $L_{\lambda}(\omega_i)$  is the incoming radiance and  $L_{\lambda}(\omega_r)$  is the radiance reflected by given surface Note that above's integral vanishes since  $\delta(\omega - \omega_i)$  is only equal one if and only if  $\omega = \omega_i$ .

For the  $BRDF(\omega_i, \omega_r)$  we are going to use Stam's main derivation (2.24) applying the fact that the wavenumber is equal  $k = \frac{2\pi}{\lambda}$ :

$$\begin{aligned}
BRDF(\omega_i, \omega_r) &= \frac{k^2 F^2 G}{4\pi^2 A w^2} \langle |P(ku, kv)|^2 \rangle \\
&= \frac{k^2 F^2 (1 + \omega_i \cdot \omega_r)^2}{\cos(\theta_i)\cos(\theta_r)4\pi^2 A w^2} \langle |P(ku, kv)|^2 \rangle \\
&= \frac{4\pi^2 F^2 (1 + \omega_i \cdot \omega_r)^2}{\cos(\theta_i)\cos(\theta_r)4\pi^2 A \lambda^2 w^2} \langle |P(ku, kv)|^2 \rangle \\
&= \frac{F(w_i, w_r)^2 (1 + \omega_i \cdot \omega_r)^2}{\cos(\theta_i)\cos(\theta_r)A \lambda^2 w^2} \langle |P(ku, kv)|^2 \rangle \quad (2.29)
\end{aligned}$$

going back to the definition of  $(u, v, w) = -\omega_i - \omega_r$  and using spherical coordinates, we get for  $w$  the following identity  $w = -\omega_i - \omega_r = -(\omega_i + \omega_r) = -(\cos(\theta_i) + \cos(\theta_r))$  and therefore  $w^2$  is equal  $(\cos(\theta_i) + \cos(\theta_r))^2$  This new fact will allow us to get even further:

$$\begin{aligned}
L_\lambda(\omega_r) &= \frac{F(\omega_i, \omega_r)^2 (1 + \omega_i \cdot \omega_r)^2}{A \lambda^2 \cos(\theta_i) \cos(\theta_r) (\cos(\theta_i) + \cos(\theta_r))^2} \left\langle \left| P_{cont}\left(\frac{2\pi u}{\lambda}, \frac{2\pi v}{\lambda}\right) \right|^2 \right\rangle \cos(\theta_i) I(\lambda) \\
&= I(\lambda) \frac{F(\omega_i, \omega_r)^2 (1 + \omega_i \cdot \omega_r)^2}{\lambda^2 A (\cos(\theta_i) + \cos(\theta_r))^2 \cos(\theta_r)} \left\langle \left| P_{cont}\left(\frac{2\pi u}{\lambda}, \frac{2\pi v}{\lambda}\right) \right|^2 \right\rangle \\
&= I(\lambda) \frac{F(\omega_i, \omega_r)^2 (1 + \omega_i \cdot \omega_r)^2}{\lambda^2 A (\cos(\theta_i) + \cos(\theta_r))^2 \cos(\theta_r)} \left\langle \left| T_0^2 P_{dtft}\left(\frac{2\pi u}{\lambda}, \frac{2\pi v}{\lambda}\right) \right|^2 \right\rangle \quad (2.30)
\end{aligned}$$

$P_{cont}$  denotes the continuous inverse Fourier-Transform for the Taylor-Series of our height field representing the nano-scaled surface structure, i.e.  $P(k, l) = \mathcal{F}^{-1}\{p\}(k, l)$  and  $P_{dtft}$  is the inverse Discrete Time Fourier Transform of  $p(x, y) = e^{ikwh(x, y)}$ . Furthermore  $T_0$  the sampling distance for the discretization of  $p(x, y)$  assuming equal and uniform sampling in both dimensions  $x, y$ .

### 2.3.2 Relative BRDF

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 BRDF 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  $\omega_r = \omega_0 := (0, 0, *)$  i.e. specular reflection case, 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 by simply dividing  $L_\lambda(\omega_r)$  by  $L_\lambda^{spec}(\omega_0)$ , denoted as

$$\rho_\lambda(\omega_i, \omega_r) = \frac{L_\lambda(\omega_r)}{L_\lambda^{spec}(\omega_0)} \quad (2.31)$$

But first, let us derive the following expression:

$$\begin{aligned}
L_\lambda^{spec}(\omega_0) &= I(\lambda) \frac{F(\omega_0, \omega_0)^2 (1 + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix})^2}{\lambda^2 A (\cos(0) + \cos(0))^2 \cos(0)} \left\langle \left| T_0^2 P_{dtft}(0, 0) \right|^2 \right\rangle \\
&= I(\lambda) \frac{F(\omega_0, \omega_0)^2 (1 + 1)^2}{\lambda^2 A (1 + 1)^2 1} \left| T_0^2 N_{sample} \right|^2 \\
&= I(\lambda) \frac{F(\omega_0, \omega_0)^2}{\lambda^2 A} \left| T_0^2 N_{sample} \right|^2 \quad (2.32)
\end{aligned}$$

Where  $N_{samples}$  is the number of samples of the DTFT.

Thus, we can plug our last derived expression into the definition for the relative reflectance radiance in the direction  $w_r$  and will get:

$$\begin{aligned}
\rho_\lambda(\omega_i, \omega_r) &= \frac{L_\lambda(\omega_r)}{L_\lambda^{spec}(\omega_0)} \\
&= \frac{I(\lambda) \frac{F(\omega_i, \omega_r)^2 (1 + \omega_i \cdot \omega_r)^2}{\lambda^2 A (\cos(\theta_i) + \cos(\theta_r))^2 \cos(\theta_r)} \langle \left| T_0^2 P_{dtft}(\frac{2\pi u}{\lambda}, \frac{2\pi v}{\lambda}) \right|^2 \rangle}{I(\lambda) \frac{F(\omega_0, \omega_0)^2}{\lambda^2 A} \left| T_0^2 N_{sample} \right|^2} \\
&= \frac{F^2(\omega_i, \omega_r) (1 + \omega_i \cdot \omega_r)^2}{F^2(\omega_0, \omega_0) (\cos(\theta_i) + \cos(\theta_r))^2 \cos(\theta_r)} \langle \left| \frac{P_{dtft}(\frac{2\pi u}{\lambda}, \frac{2\pi v}{\lambda})}{N_{samples}} \right|^2 \rangle
\end{aligned} \tag{2.33}$$

for simplification and a better overview, let us introduce the following expression, the so called gain-factor:

$$C(\omega_i, \omega_r) = \frac{F^2(\omega_i, \omega_r) (1 + \omega_i \cdot \omega_r)^2}{F^2(\omega_0, \omega_0) (\cos(\theta_i) + \cos(\theta_r))^2 \cos(\theta_r) N_{samples}^2} \tag{2.34}$$

Using this substitute, we will end up with the following expression for the relative reflectance radiance:

$$\rho_\lambda(\omega_i, \omega_r) = C(\omega_i, \omega_r) \langle \left| P_{dtft}(\frac{2\pi u}{\lambda}, \frac{2\pi v}{\lambda}) \right|^2 \rangle \tag{2.35}$$

Using the previous definition for the relative reflectance radiance

$$\rho_\lambda(\omega_i, \omega_r) = \frac{L_\lambda(\omega_r)}{L_\lambda^{spec}(\omega_0)} \tag{2.36}$$

which we can rearrange to the expression

$$L_\lambda(\omega_r) = \rho_\lambda(\omega_i, \omega_r) L_\lambda^{spec}(\omega_0) \tag{2.37}$$

Let us choose  $L_\lambda^{spec}(\omega_0) = S(\lambda)$  such that it has the same profile as the relative spectral power distribution of CIE Standard Illuminant *D65*. Further, when integration over  $\lambda$  for a specular surface we should get *CIEXYZ* values corresponding to the white point for *D65*. The corresponding tristimulus values using CIE colormatching functions for the *CIEXYZ* values look like:

$$X = \int_\lambda L_\lambda(\omega_r) \bar{x}(\lambda) d\lambda \tag{2.38}$$

$$Y = \int_\lambda L_\lambda(\omega_r) \bar{y}(\lambda) d\lambda \tag{2.39}$$

$$Z = \int_\lambda L_\lambda(\omega_r) \bar{z}(\lambda) d\lambda \tag{2.40}$$

where  $\bar{x}$ ,  $\bar{y}$ ,  $\bar{z}$  are the color matching functions

Using our last finding for  $L_\lambda(\omega_r)$  with the definition for the tristimulus values we can actually derive an expression for computing the colors for our BRDF formula. Since X, Y, Z are defined similarly, it satisfies to derive an

explicit expression for just one tristimulus term, for example X. The other two will look the same, except the we have to replace all X with Y or Z respectively. Therefore, we get:

$$\begin{aligned}
X &= \int_{\lambda} L_{\lambda}(\omega_r) \bar{x}(\lambda) d\lambda \\
&= \int_{\lambda} \rho_{\lambda}(\omega_i, \omega_r) L_{\lambda}^{spec}(\omega_0) \bar{x}(\lambda) d\lambda \\
&= \int_{\lambda} \rho_{\lambda}(\omega_i, \omega_r) S(\lambda) \bar{x}(\lambda) d\lambda \\
&= \int_{\lambda} C(\omega_i, \omega_r) \left\langle \left| P_{dft} \left( \frac{2\pi u}{\lambda}, \frac{2\pi v}{\lambda} \right) \right|^2 \right\rangle S(\lambda) \bar{x}(\lambda) d\lambda \\
&= C(\omega_i, \omega_r) \int_{\lambda} \left\langle \left| P_{dft} \left( \frac{2\pi u}{\lambda}, \frac{2\pi v}{\lambda} \right) \right|^2 \right\rangle S(\lambda) \bar{x}(\lambda) d\lambda \\
&= C(\omega_i, \omega_r) \int_{\lambda} \left\langle \left| P_{dft} \left( \frac{2\pi u}{\lambda}, \frac{2\pi v}{\lambda} \right) \right|^2 \right\rangle S_x(\lambda) d\lambda
\end{aligned} \tag{2.41}$$

Where we used the definition  $S_x(\lambda) \bar{x}(\lambda)$  in the last step.

### 2.3.3 Taylor approximation for BRDF

In this section, we will deliver an approximation for the inverse Fourier Transformation of Stam's auxiliary function  $p(x,y)$ s. This derivation will rely on the definition of Taylor Series expansion. Further, we will provide an error bound for our approximation approach for a given number of iterations. Last, we will extend our current BRDF formula by the findings derived within this section.

Given  $p(x, y) = e^{ikwh(x, y)}$  form Stam's Paper where  $h(x, y)$  is a given height field. Let be  $y$  real or even complex value, and lets consider the power series for the the exponential function

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

Let us define  $t := t(x, y) = ikwh(x, y)$  where  $i$  is the imaginary number. For simplification, let us denote  $h(x, y)$  as  $h$ . Then it follows by our previous stated identities:

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

Hence it holds  $p(x, y) = \sum_{n=0}^{\infty} \frac{(ikwh(x, y))^n}{n!}$ .

Let us now compute the Fourier Transformation of  $p(x,y)$  form above:

$$\begin{aligned}\mathcal{F}\{p\}(u,v) &= \mathcal{F}\left\{\sum_{n=0}^{\infty} \frac{(ikwh)^n}{n!} \cdot \right\}(u,v) \\ &= \text{Fourier Lin Operator} \sum_{n=0}^{\infty} \mathcal{F}\left\{\frac{(ikwh)^n}{n!}\right\}(u,v) \\ &= \sum_{n=0}^{\infty} \frac{(ikw)^n}{n!} \mathcal{F}\{h^n\}(u,v)\end{aligned}$$

Hence it follows:  $P(\alpha, \beta) = \sum_{n=0}^{\infty} \frac{(ikw)^n}{n!} \mathcal{F}\{h^n\}(\alpha, \beta)$  for which  $\mathcal{F}_{FT}\{h^n\}(u, v)$ . Next we are going to look for an  $N \in \mathbb{N}$  such that

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

is a good approximation. But first the following two facts have to be proven:

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  $\epsilon$ .

### Proof Sketch of 1.

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

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

$$\forall y : \limsup_{k \rightarrow \infty} \left| \frac{a_{k+1}}{a_k} \right| = \limsup_{k \rightarrow \infty} \frac{y}{k+1} = 0$$

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

### 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,

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

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:

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

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

$$x_{max} = \|i\| k_{max} w_{max} h_{max}$$

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:

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

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

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

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  $\epsilon \approx 8.8150 * 10^{-8}$ .

Using now our approximation for  $P_{dtft} = \mathcal{F}^{-1}\{p\}(u, v)$  for the tristimulus value  $X$ , we will get:

$$\begin{aligned} X &= C(w_i, w_r) \int_{\lambda} \left| P_{dtft}\left(\frac{2\pi u}{\lambda}, \frac{2\pi v}{\lambda}\right) \right|^2 S_x(\lambda) d\lambda \\ &= C(w_i, w_r) \int_{\lambda} \left| \sum_{n=0}^N \frac{(wk)^n}{n!} \mathcal{F}^{-1}\{i^n h^n\}\left(\frac{2\pi u}{\lambda}, \frac{2\pi v}{\lambda}\right) \right|^2 S_x(\lambda) d\lambda \end{aligned}$$

### 2.3.4 Sampling: Gaussian Window

Practically, we cannot compute the DTFT numerically due to finite computer arithmetic, since  $w$  is a continuous function for the DTFT. The DFT of a discrete height field patch is equivalent to the DTFT of an infinitely periodic function consisting of replicas of the same discrete patch. By windowing with a window function that is zero outside the central replica, the convolution of either the DFT or the DTFT of height field with the Fourier Transform of the window becomes equivalent.

Let  $window_g$  denote the gaussian window with  $4\sigma_s \mu m$  where  $\sigma_f = \frac{1}{2\pi\sigma_s}$  let us further substitute  $\mathbf{t}(\mathbf{x}, \mathbf{y}) = i^n h(x, y)^n$

$$\mathcal{F}_{dtft}^{-1}\{\mathbf{t}\}(u, v) = \mathcal{F}_{fft}^{-1}\{\mathbf{t}\}(u, v) window_g(\sigma_f) \quad (2.42)$$

Therefore we can deduce the following expression from this:

$$\begin{aligned} \mathcal{F}_{dtft}^{-1}\{\mathbf{t}\}(u, v) &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F_{fft}^{-1}\{\mathbf{t}\}(w_u, w_v) \phi(u - w_u, v - w_v) dw_u dw_v \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \sum_i \sum_j F_{fft}^{-1}\{\mathbf{t}\}(w_u, w_v) \\ &\quad \delta(w_u - w_i, w_v - w_j) \phi(u - w_u, v - w_v) dw_u dw_v \\ &= \sum_i \sum_j \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F_{fft}^{-1}\{\mathbf{t}\}(w_u, w_v) \\ &\quad \delta(w_u - w_i, w_v - w_j) \phi(u - w_u, v - w_v) dw_u dw_v \\ &= \sum_i \sum_j F_{fft}^{-1}\{\mathbf{t}\}(w_u, w_v) \phi(u - w_u, v - w_v) \end{aligned}$$

where

$$\phi(x, y) = \pi e^{-\frac{x^2+y^2}{2\sigma_f^2}} \quad (2.43)$$

### 2.3.5 Final Expression

As the last step of our series of derivations, we plug all our findings together to one big equation in order to compute the color for each pixel on our mesh in the  $CIE_XYZ$  colorspace:

For a given height-field  $h(x, y)$ , representing a small patch of the nano-structure of our surface, the resulting  $CIE_{XYZ}$  caused by the effect of diffraction can be computed like the following:

$$\text{Let } P(u, v, \lambda) = F_{fft}^{-1}\{i^n h^n\}(\frac{2\pi u}{\lambda}, \frac{2\pi v}{\lambda})$$

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = C(\omega_i, \omega_r) \int_{\lambda} \sum_{n=0}^N \frac{(wk)^n}{n!} \sum_{(r,s) \in \mathcal{N}_1(u,v)} |P_{\lambda}(u - w_r, v - w_s)|^2 \phi(u - w_r, v - w_s) \begin{pmatrix} S_x(\lambda) \\ S_y(\lambda) \\ S_z(\lambda) \end{pmatrix} d\lambda \quad (2.44)$$

where  $\phi(x, y) = \pi e^{-\frac{x^2+y^2}{2\sigma_f^2}}$  is the gaussian window. where  $w_s$  and  $w_r$  are ... explain them

## 2.4 Alternative Approach

REASON FOR WHICH CASE THIS CAN BE USED, ADD AN IMAGE OF BOXFUNCTION The approach introduced within this section is an alternative to the gaussian window approach. Let us consider the so called 1-dimensional Box-function with length  $T$  which is defined as the following:

$$Box(x) = \begin{cases} 1 & \text{if } x \leq T \\ 0 & \text{if else} \end{cases}$$

We assume, that our given height field can be represented as a 2-dimensional box-function. Note that we can use any explicit given constrained 2-dimensional function and will get some identities like we get from the box-function.

Further we are assuming that we can model the overall surface by assuming this height field being distributed in a periodic manner. Therefore, the whole surface can be represented like this  $f(x) = \sum_{n=0}^N Box(x + nT_1, y + mT_2)$  assuming the given height field has the dimensions  $T_1$  by  $T_2$ . But let us first consider the 1-dimensional Box-function case before deriving an identity for the Fourier transform of our 2-dimensional Box-function, i.e. the Fourier Transform of our height field.

Note: A function  $f$  periodic with period  $T$  means:  $\forall x \in \mathcal{R} : Box(x) = Box(x + T)$

A so called bump can be represented by our 1-dimensional Box-function. We assume periodicity which is equivalent to:  $f(x) = \sum_{n=0}^N Box(x + nT)$

We are interested in the 1-dimensional inverse Fourier transform of the 1-dimensional Box-function:

$$\begin{aligned}
\mathcal{F}^{-1}\{f\}(w) &= \int f(x)e^{iwx}dx \\
&= \int_{-\infty}^{\infty} \sum_{n=0}^N Box(x+nT)e^{iwx}dx \\
&= \sum_{n=0}^N \int_{-\infty}^{\infty} Box(x+nT)e^{iwx}dx
\end{aligned}$$

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

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

Plugging this substitution back to the equation from above we will get

$$\begin{aligned}
\mathcal{F}^{-1}\{f\}(w) &= \int f(x)e^{iwx}dx \\
&= \sum_{n=0}^N \int_{-\infty}^{\infty} Box(y)e^{iw(y-nT)}dy \\
&= \sum_{n=0}^N e^{-iwnT} \int_{-\infty}^{\infty} Box(y)e^{iwy}dy \\
&= \sum_{n=0}^N e^{-iwnT} \mathcal{F}\{f\}(w) \\
&= \mathcal{F}^{-1}\{f\}(w) \sum_{n=0}^N e^{-iwnT}
\end{aligned}$$

We used the fact that the term  $e^{-iwnT}$  is a constant when integrating along  $dy$  and the identity for the inverse Fourier transform of the Box function. Next, let us consider  $\sum_{n=0}^N e^{-uwnT}$  further:

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

We recognize the geometric series identity for the left-hand-side of this equation. Since our series is bounded we can derive our right-hand-side.

Since  $e^{-ix}$  is a complex number and every complex number can be written in its polar form, i.e.  $e^{-ix} = \cos(x) + i\sin(x)$  we can go even further, using the trigonometric identities that  $\cos(-x) = \cos(x)$  and  $\sin(-x) = -\sin(x)$ :

$$\frac{1 - e^{iwT(N+1)}}{1 - e^{-uwT}} = \frac{1 - \cos(wT(N+1)) + i\sin(wT(N+1))}{1 - \cos(wT) + i\sin(wT)}$$

Which is still a complex number  $(p + iq)$ . Every complex number can be written as a fraction of two complex numbers. This means that the complex number  $(p + iq)$  can be written as  $(p + iq) = \frac{(a+ib)}{(c+id)}$  for any  $(a+ib), (c+id) \neq 0$ . For our case, let us use the following substitutions:

$$a := 1 - \cos(wT(N+1)) \quad b = \sin(wT(N+1)) \quad (2.45)$$

$$c = 1 - \cos(wT) \quad d = \sin(wT) \quad (2.46)$$

hence it follows  $\frac{1-e^{iwT(N+1)}}{1-e^{-iwT}} = \frac{(a+ib)}{(c+id)}$ . By rearranging the terms it follows  $(a+ib) = (c+id)(p+iq)$  and multiplying the right hand-side out we get the following system of equations:

$$(cp - dq) = a \quad (2.47)$$

$$(dp + cq) = b \quad (2.48)$$

Which gives lead us we some further math (trick: mult first eq. by  $c$  and 2nd by  $d$ , then adding them together. using distributivity and we have the identity for  $p$  for example, similar for  $q$ ) to

$$p = \frac{(ac + bd)}{c^2 + d^2} \quad (2.49)$$

$$q = \frac{(bc + ad)}{c^2 + d^2} \quad (2.50)$$

Putting our substitution for  $a, b, c, d$  back into the current representation for  $p$  and  $q$  and using some trigonometric identities, this we then get:

$$p = \frac{1}{2} + \frac{1}{2} \left( \frac{\cos(wTN) - \cos(wT(N+1))}{1 - \cos(wT)} \right) \quad (2.51)$$

$$q = \frac{\sin(wT(N+1)) - \sin(wTN) - \sin(wT)}{2(1 - \cos(wT))} \quad (2.52)$$

Since we have seen, that  $\sum_{n=0}^N e^{-uwnT}$  is a complex number and can be written as  $(p + iq)$  and we know now the explicit identity for those  $p$  and  $q$  we get for the 1-dimensional Fourier transform of the 1-dimensional Box-function the following final identity:

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

In oder to derive next a identity for the Fourier transform for our 2-dim height field, we can proceed similarly, the only fact which changes is, that we are now in a 2-dimensional domain, i.e. we are about to compute a two-dimensional Fourier transform: Let us again us again a Box-function, this time a 2-dimensional Box-function  $Box(x, y)$  just for the sake of convenience.

$$\begin{aligned}
\mathcal{F}^{-1}\{f\}(w_1, w_2) &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \sum_{n_2=0}^{N_1} \sum_{n_2=0}^{N_2} \text{Box}(x_1 + n_1 T_1, x_2 + n_2 T_2) e^{iw(x_1+x_2)} dx_1 dx_2 \\
&= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \sum_{n_2=0}^{N_1} \sum_{n_2=0}^{N_2} \text{Box}(y_1, y_2) e^{iw((y_1-n_1 T_1)+(y_2+n_2 T_2))} dy_1 dy_2 \\
&= \sum_{n_2=0}^{N_1} \sum_{n_2=0}^{N_2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \text{Box}(y_1, y_2) e^{iw(y_1+y_2)} e^{-iw(n_1 T_1+n_2 T_2)} dy_1 dy_2 \\
&= \sum_{n_2=0}^{N_1} \sum_{n_2=0}^{N_2} e^{-iw(n_1 T_1+n_2 T_2)} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \text{Box}(y_1, y_2) e^{iw(y_1+y_2)} dy_1 dy_2 \\
&= \left( \sum_{n_2=0}^{N_1} \sum_{n_2=0}^{N_2} e^{-iw(n_1 T_1+n_2 T_2)} \right) \mathcal{F}^{-1}\{\text{Box}\}(w_1, w_2) \\
&= \left( \sum_{n_2=0}^{N_1} e^{-iwn_1 T_1} \right) \left( \sum_{n_2=0}^{N_2} e^{-iwn_2 T_2} \right) \mathcal{F}^{-1}\{\text{Box}\}(w_1, w_2) \\
&= (p_1 + iq_1)(p_2 + iq_2) \mathcal{F}^{-1}\{\text{Box}\}(w_1, w_2) \\
&= ((p_1 p_2 - q_1 q_2) + i(p_1 q_2 + q_1 p_2)) \mathcal{F}^{-1}\{\text{Box}\}(w_1, w_2) \\
&= (p + iq) \mathcal{F}^{-1}\{\text{Box}\}(w_1, w_2)
\end{aligned}$$

Where we define  $p := (p_1 p_2 - q_1 q_2)$  and  $q := (p_1 p_2 + q_1 q_2)$ . For this identity we used 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 Box-function. We applied the same substitution like we did in for the 1 dimensional case, but this time twice, once for each variable separately. The last step, substituting with  $p$  and  $q$  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  $\mathcal{F}^{-1}\{f\}(w_1, w_2)$  which will then be equal  $(p^2 + q^2)^{\frac{1}{2}} |\mathcal{F}^{-1}\{\text{Box}\}(w_1, w_2)|$

#### 2.4.1 Interpolation

$$f(t) = \sum_{n=-\infty}^{\infty} x[n] (2F_0 T_s) \text{sinc}(F_0(t - nT_s)) \quad (2.53)$$

$$= \sum_{n=-\infty}^{\infty} x[n] \text{sinc}\left(\frac{t - nT_s}{T_s}\right) \quad (2.54)$$

method to construct a continuous-time bandlimited function from a sequence of real numbers. sampling theorem: a bandlimited function can be perfectly reconstructed from a countable sequence of samples if the bandlimit,  $B$ , is no greater than half the sampling rate. In practice, infinite sequences, perfect sampling, and perfect interpolation are all replaced by approximations, deviating from the ideal mathematical reconstruction.

Issue: given discrete signal, want to perform computation using its DFT values, but given DTFT values. solution: convolve by a sinc filter. show plots sinc- vs linear- signal interpolation

Given a sequence of real numbers  $x[n]$  then the continuous function

$$x(t) = \sum_{n=-\infty}^{\infty} x[n] \text{sinc}\left(\frac{t - nT}{T}\right) \quad (2.55)$$

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.

## Kapitel 3

# Implementation

In computergraphics, we are interested in synthesizing 2d images from a given scene containing our 3d geometries by using so called shader programs. This process is denoted as rendering. The purpose of shader programs, which are executed directly on the GPU hardware device, is to compute the colorization and illumination of the objects living in our scene. All these computations happen in several stages and depend on the provided scene-input parameters like the camera, light sources, objects material constants and the desired rendering effect one is interested in to model. The shader stages are implemented sequencially as small little programs, the so called vertex-, geometry- and fragment-shaders. Those stages are applied within the rendering pipeline sequencially.

Our shaders which we use are written in a high-level language called GLSL, the OpenGL Shading Language. The decission for using OpenGL has been made since my underlying framework, which is responsible for the precomputation of all scene date, is based on another framework, written in Java using JOGL in oder to communicate with the GPU and is also responsible to precompute all the relevant scene data. This framework, the so called jrtr framework, has been developed as an exercise during the class computer graphics held by M. Zwicker which I attended in autumn 2012. The framework itself has been used and further extended during this thesis quite a lot. All necessary input data required for our java framework in order to perform the shading is precomputed by using Matlab. This is basically addressing all the required precomputations for the provided heigh-fields, refering to computation of the inverse two dimensional Fourier transformations which are further explained within this chapter. The Matlab scripts themself rely on the provided snake nano-scaled sheds images, taken by AFM.

It's noteworthy that all the vertices are processed within the vertex-shader, whereas the fragement shader's responsibility is to perfrom pixelwise rendering, using the input from the vertex shader. Just remember, fragments are determined by a triple of vertices. hence each pixel has assigned a trilinear interpolated value of all input parameters of its spanning vertices. Usually, all necessary transformations are applied vertex-wise, considering the vertex-shader as the precomputation stage for the later rendering within the rendering pipeline, in the fragment-shader. In the geometry shader, new vertices around a considered vertex can be created. this is useful for debugging - displaying normals graphically for example.

In this part of thesis we are going to explain how we render our BRDF formulation derived in the last section in practice. all the necessary computations in order to simulate the effect of diffraction are performed within a fragment shader. This implies that we are modeling the effect of diffraction pixelwise and hence the overall modeling quality and computational pace depends on rendering window's resolution.

By the end of this chapter we will have seen how our render works, what we have to precompute and how our shaders work.

### 3.1 Precomputations in Matlab

Our first task is to precompute the inverse two dimensional discrete Fourier Transformations of a given snake shed patch of interest taken by AFM. For that purpose we have written a small Matlab script which offers a huge collection of mathematically, numerically fast and stable algorithms. Our Matlab script reads a given image, which is representing a nano-scaled heightfield, and computes its inverse two dimensional DFT by using Matlab's internal inverse fast Fourier Transformation function, denoted by *ifft2*. Note that we only require once color channel of the input image since the input image is representing an heightfield, encoded by just one color. Basically, we are interested in computing the *ifft2* for different powers of the input image since our taylor series approximation for the overall computation relies on this. 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 each pixel of our input image. Therefore, for each input image we get as many output images, representing the two dimensional inverse Fourier Transformation, as the minimal amount of taylor terms required for a well-enough approximation. In order to store our output images, we have to use 2 color channels instead just one like it was for the given input image. As an optimization step, we do not directly store images, rather we output binary files which contain all RGB values for each pixel in a row first, column last format. This allows us to have much higher precision for the output values and also it does not waste any color channels. Note that we have scaled each pixel value in a range between 0 and 1. Therefore, we have to remember four scaling factors for each output image as well, which are the real and imaginary minimum and maximum values. Later, using linear interpolation within the shader, we will get back the image's original values.

---

**Algorithm 3.1** Precomputation: Fourier images

---

```
% 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 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(maxH, dh, termCnt)
dH = dh*1E-6;
% load patch into patchImg
patchImg = patchImg.*maxH;
% perform imrotate(patchImg, angle)
for t = 0 : termCnt
    patchFFT = power(1j*patchImg, t);
    fftTerm{t+1} = fftshift(ifft2(patchFFT));

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

    % perform imrotate(imOut, -angle)
    % find real and imaginary extrema of
    % write imOut, extrema, dH, into files .
end
```

---

The command `fftshift` rearranges the output of the `ifft2` by moving the zero frequency component to the centre of the image. This is useful for visualizing a Fourier Transform with zero frequency components in the middle of the spectrum.

mention rescaling

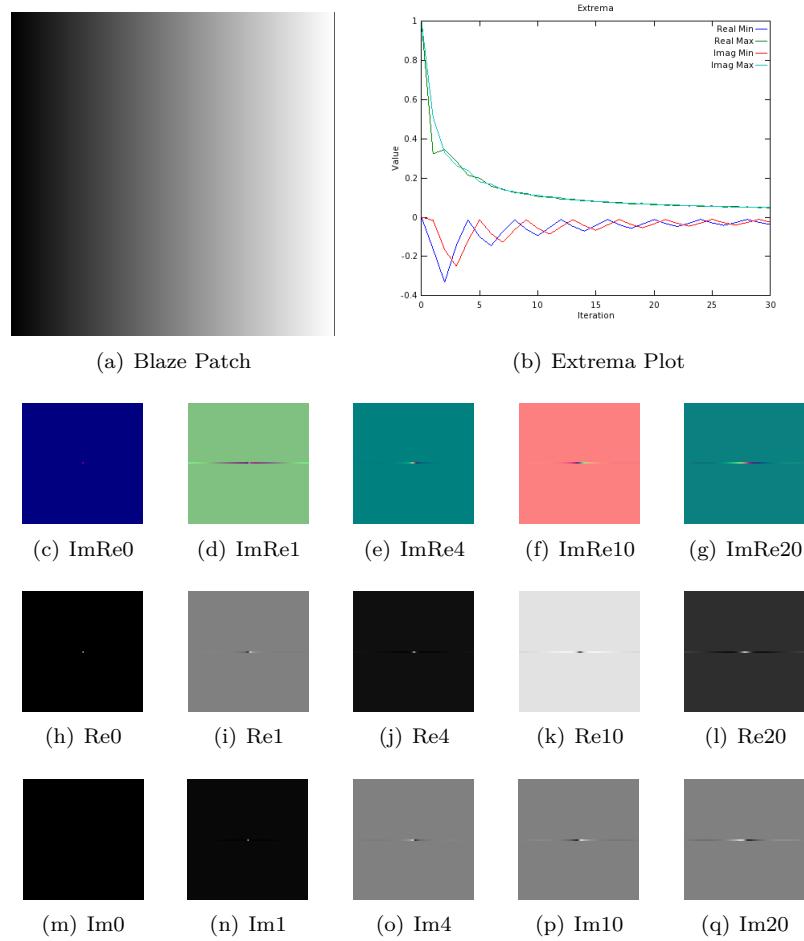


Abbildung 3.1: Blaze

TODO: say something about output?

## 3.2 Java Renderer

In autumn 2012, during the semester I have attended the class computer graphics held by M. Zwicker. During the whole class we have developed a so called real time renderer program written in java as a series of homework assignments in order to be admitted to the final exam. 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, denoted as scene.

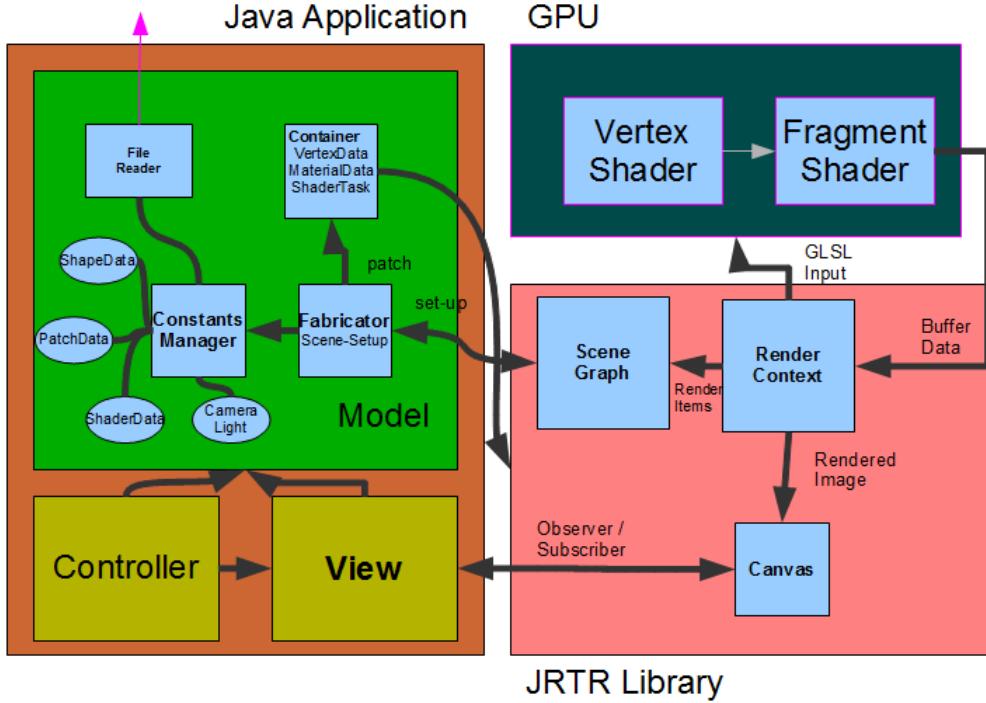


Abbildung 3.2: Renderer Architecture

the scene application program task was basically to define the whole scene we are going to be rendered within jrtr later. A scene consists of the setup of the world camera, definitions of light source, frustum, geometries which live in our scene and their material constants. Such materials are textures for example. All those scene attributes are managed within jrtr. In the application program, there only happens their definition. The minimal definition of a geometry is given by its wireframe mesh. This is a datastructures consisting of vertices each stored as a triple of xyz positions in an float array and triangles each defined by a triple of vertex-indices which form a fragment each stored in an integer array. It is possible to assing additional geoemtry data like a color for each vertex, normals and texture coordinates. All whole scene is stored within container data-structures, defined and managed within jrtr, like a scene graph, which contains all geometries and their transformations in a tree like structured hierarchy. The geometries themself are stored within an container, containing all vertex attributes and the material constants. The jrtr rendering engine uses a low-level API, called OpenGL in order to communicate with the grpahcis processor unit (GPU). Within jrtr, the whole resource-management for the rendering pipeline place, i.e. all required low-level buffers are allocated, flushed and assigned by the scene data attributes. jrtr also offers also the possibility to assign arbitrary shaders.

### 3.3 GLSL Diffraction Shader

#### 3.3.1 Vertex Shader

The first computational stage within our rendering pipeline is computing all necessary per vertex data. Those computations are performed in the vertex shader. In our case, we compute for any vertex of our current geometry the direction vectors  $k_1$  and  $k_2$  described like previously in the tangent space. Initially all input data lives in its own space. Hence, we first have to transform all input data into the same space in order to use it for later computations within the fragment shader. We are going to transform  $k_1$  and  $k_2$  into the so called tangent space which. Furthermore, we have also to realign our local coordinate system. This is why there is an Rodrigues rotation also involved. In order to avoid scaling issues and since we are only interested in the direction of the vectors  $k_1$  and  $k_2$ , we have to normalize them, too. Last, we also output the position of the current vertex transformed into the projective camera space.

show CG graphic unify  $k_1, k_2$  to  $w_i, wr$

---

#### Algorithm 3.2 Vertex diffraction shader

---

```
foreach Vertex  $v \in Shape$  do
     $vec3N = normalize(modelM * vec4(normal, 0.0)).xyz$ 
     $vec3T = normalize(modelM * vec4(tangent, 0.0)).xyz$ 
     $T = rotateRodrigues(T, N, phi)$ 
     $vec3B = normalize(cross(N, T))$ 
     $vec3Pos = ((cop_w - position)).xyz$ 
     $vec4lightDir = (directionArray[0])$ 
     $lightDir = normalize(lightDir)$ 
     $l = projectVectorOnTo(lightDir, TangentSpace)$ 
     $p = projectVectorOnTo(Pos, TangentSpace)$ 
     $normalize(l); normalize(p)$ 
     $glposition = projection * modelview * position$ 
end for
```

---

explain  $cop_w$  modelM other shader assigned inputs

#### 3.3.2 Fragment Shader

The purpose of a fragment shader is to render per fragment. A fragment is spanned by three vertices of a given mesh. For each pixel within all the output from the vertex shaders of its corresponding vertices is then trilinearly interpolated, depending on the pixel's position within the fragment, and passed into its fragment shader program. Furthermore, there can be additional input be assigned which is not directly interpolated from the output of vertex shader programs. Our fragment shader just relies on  $k_1$  and  $k_2$  from its vertex shaders for the computation of the effect of diffraction. There are some values preliminarily assigned to our fragment shader during the OpenGL setup within our java program, like all references to the image buffers, containing the Fourier Transformations, the number of Taylor step approximations, the minimal and maximal wavelength, other lookup values like the scaling factors, a reference to

a lookup table containing the  $CIE_{XYZ}$  color weights for our wavelength domain and other scaling constants.

Our shader performs an on-the-fly numerical integration for the integral in the derivation using trapezoidal rule with uniform discretization of the  $\lambda$  dimension at a resolution of 5nm. To compute  $F_{dtft}\{p\}$  terms the shader uses the precomputed DFTs for the Taylor series terms given in the derivation. The Gaussian window approach is performed for each discrete  $\lambda$  value using a window large enough to span  $4\sigma_f$  in both dimensions. For computing DFT tables we generally use nanostructure heightfields that span at least  $65\mu m^2$  and are sampled with resolution of at least 100nm. This ensures that the spectral response encompasses all the wavelengths in the visible spectrum, i.e. from 380nm to 780nm. Note that this shader is not very fast in hardly can be denoted being interactive.

mention we uniform discretize  $\lambda$  for a given  $(u,v)$  which implies compressing sampled frequencies to the region near to the origin (of their frequency domain). For natural structures in nano-scale, most of their spectral energy lies at lower spatial frequencies which maps closer to region  $(u,v) = (0,0)$  than higher frequencies. This is why We have chosen to sample  $(u,v)$  space non-linearly. We use 30 Taylor-terms for our approximation approach which has an error below Y, proven in the previous derivation chapter.

---

**Algorithm 3.3** Fragment diffraction shader

---

```

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

```

---

**From line 4 to 26:**

Within this loop happens the uniform sampling along wavelength-space.  $ColorWeights(\lambda)$  computes the color weight for the current wavelength  $\lambda$  by bilinear interpolation between the color weight for  $\lceil \lambda \rceil$  and  $\lfloor \lambda \rfloor$  which are stored in a external weights-table (assuming this table contains wavelengths in 1nm steps). At line 6:  $lookupCoord(u, v, \lambda)$  the coordinates for the texture lookup are computed - See 3.3. Line 25 sums up the diffraction color contribution for the current wavelength in iteration  $\lambda$ .

**From line 9 to 24:**

Within this loop happens the Taylor series approximation until a predefined upper bound, denoted by  $T$ , has been reached. Basically, the spectral response is approximated for our current  $(u, v, \lambda)$ . Furthermore, neighborhood boundaries for the gaussian-window sampling are computed, denoted as  $anchorX$  and  $anchorY$ .

**From line 14 to 22:**

In this most inner loop, the convolution of the gaussian window with the inverse FFT of the patch is pixel-wise performed.  $gaussWeightOf(dist)$  computes the weights (2.43) from the distance between the current pixel's coordinates and the current neighbor's position in texture space. Local lookup coordinates for the current fourier coefficient  $fftVal$  value are computed at line 17 and computed like described in 3.5. The actual texture lookup is performed at line 18 using those local coordinates. Inside  $rescaledFourierValueAt$  the values  $fftVal$  is rescaled by its extrema, i.e.  $(fftVal * Max + Min)$  is computed, since  $fftVal$  is normalized 3.1. The current  $fftVal$  values in iteration is scaled by the current gaussian weight and then summed to the final neighborhood FFT contribution at line 20.

**After line 26:**

At line 27 the gain factor  $C(\omega_i, \omega_r)$  2.34 is multiplied by the current computed pixel color like formulated in 2.35. The gain factor contains the geometric term  $refeq : geometricterm$  and the Fresnel term  $F$ . We approximate  $F$  by the Schlick approximation A.1, using an reactive index at 1.5 since this is close to the measured value from snake sheds. Our BRDF values are scaled by s shadowing function as described in (SEE REFERENCES - PAPER), since most of the grooves in the snake skin nano-structures would form a V-cavity along the plane for a wave front with their top-edges at almost the same height.

Last, we transform our colors from the  $CIE_XYZ$  colorspace to the  $CIE_RGB$  space using the CIE Standard Illuminant D65, followed by a gamma correction. See 3.4.3 for further insight.

## 3.4 Technical details

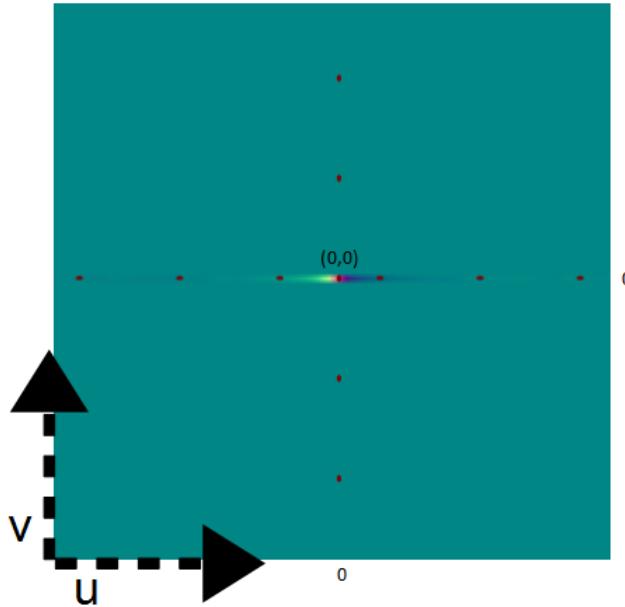
### 3.4.1 Texture lookup

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

Given a nano-scaled surface patch  $P$  with a resolution  $A$  by  $A$  microns stored as an  $N$  by  $N$  pixel image  $I$ . Then one pixel in any direction corresponds to  $dH = \frac{A}{N} \mu m$ . In Matlab we compute a series of  $n$  output images  $\{I_{out_1}, \dots, I_{out_n}\}$  from  $I$ , which we will use for the lookup in our shader - See figure 3.3. For the lookup we use scaled and shifted  $(u, v)$  coordinates from 2.17.

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

$$b = (N \% 2 == 0) \quad ? \quad \frac{N}{2} : \frac{N - 1}{2} \quad (3.1)$$

Abbildung 3.3:  $(u, v)$  lookup image

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

$$\frac{ku}{w_0} = \frac{kuNdH}{2\pi} \quad (3.2)$$

$$= \frac{uNdH}{\lambda} \quad (3.3)$$

Using those findings 3.1, 3.3, the final  $(u, v)$  texture lookup-coordinates for the current wavelength  $\lambda$  in iteration, will then look like:

$$(u_{lookup}, v_{lookup}) = \left( \frac{uNdH}{\lambda} + b, \frac{vNdH}{\lambda} + b \right) \quad (3.4)$$

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

$$(u_{lookup}, v_{lookup}) = (i, j) - (u_{lookup}, v_{lookup}) \quad (3.5)$$

### 3.4.2 Texture Blending

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

---

#### Algorithm 3.4 Texture Blending

---

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

---

### 3.4.3 Color Transformation

In our shader we access a table which contains precomputed CIE's color matching functions values from  $\lambda_{min} = 380nm$  to  $\lambda_{max} = 780nm$  in  $5nm$  steps. Such a function value table can be found at downloaded at [cvrl.ioo.ucl.ac.uk](http://cvrl.ioo.ucl.ac.uk) for example. We compute the  $(X, Y, Z)$   $CIE_{XYZ}$  color values as described in 2.1.2.

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

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = M \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (3.6)$$

where one possible transformation is:

$$M = \begin{bmatrix} 0.41847 & -0.15866 & -0.082835 \\ -0.091169 & 0.25243 & 0.015708 \\ 0.00092090 & -0.0025498 & 0.17860 \end{bmatrix} \quad (3.7)$$

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

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = M \cdot \begin{bmatrix} X \cdot D65.x \\ Y \cdot D65.y \\ Z \cdot D65.z \end{bmatrix} \quad (3.8)$$

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

## Discussion

The fragment shader algorithm described in 3.3 performs the gaussian window approach by sampling over the whole wavelength spectrum in uniform step sizes. This algorithm is valid but also slow since we iterate for each pixel over the

whole lambda spectrum. Furthermore, for any pixel, we iterate over its 1 neighborhood. Considering the loop for the taylor approximation as well, we will have a run-time complexity of  $O(\#spectrtumIter \cdot \#taylorIter \cdot neighborhoodRadius^2)$ . Hence, Instead sampling over the whole wavelength spectrum, we could instead integrate over just a few required lambdas which are elicited like the following: Lets consider  $(u, v, w)$  defined as 2.17. 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  $\lambda_n^v = \frac{dv}{n}$ . For  $n = 0$  there it follows  $(u, v) = (0, 0)$ . If  $u, v > 0$

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

If  $u, v < 0$

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

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

Another variant is the  $PQ$  approach described in chapter 2 2.4. Since this approach does not require to iterate over a pixel's neighborhood, it is also faster than the gaussian window approach. One could think of a combination of those two optimization approaches. Keep in mind, both of these approaches are further approximation. The quality of the rendered images will suffer using those two approaches.

---

**Algorithm 3.5** Sinc interpolation for pq approach

---

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

---

sdfgdf

## Kapitel 4

# Evaluation Data Acquisition

### 4.1 Data Acquisition

For measurement on the true surface topography of snake sheds, samples are stuck on glass plates using double face tape, the animal was pushed up below a hollowed plate letting the skin emerging from the top of the plate. the surface of the scale under measurement should be large compared to the size of the drop to avoid wetting the plasmic membrane that would corrupt the reasing of the contact angle.

Measurements were carried out using intermittent contact mode in a Burker Dimension 3100 atomic force microscope (AFM) under ambient conditions using a Nanoscope V controller.

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 but does not touch the surface. As the probe traverses the surface, attractive and repulsive forces arising between it and the atoms on the surface induce forces on the probe that bend the cantilever. The amount of bending is measured and recorded, providing a map of the atoms on the surface. Atomic force microscopes can achieve magnification of a factor of  $5 \times 10^6$ , with a resolution of 2 angstroms, sufficient to resolve individual carbon atoms. Also called scanning force microscope, is a very high-resolution type of scanning probe microscopy, with demonstrated resolution on the order of fractions of a nanometer, more than 1000 times better than the optical diffraction limit.

The tips used were etched silicon TESP tips with a nominal frequency and force constant of 320 kHz and 42 N/m respectively.

SHOW AFM IMAGE

### 4.2 Diffraction Gratings

An idealised grating is made of a very large number of parallel, evenly spaced slits in an opaque sheet. Typically, it would have about 10,000 slits. In order to cause diffraction, the spacing between slits must be wider than the wavelength of interest. Each slit in the grating acts as a quasi point source from which light propagates in all directions. The diffracted light is composed of the sum of interfering wave components emanating from each slit in the grating.

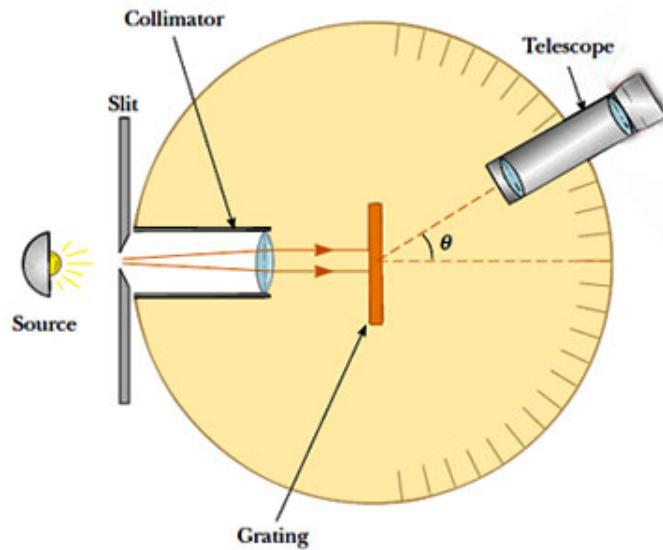


Abbildung 4.1: Spectrometer: When a beam of monochromatic light passes through a grating placed in a spectrometer, images of the sources can be seen through the telescope at different angles.

Suppose monochromatic light is directed at the grating parallel to its axis as shown in figure. Let the distance between successive slits be equal  $d$ .

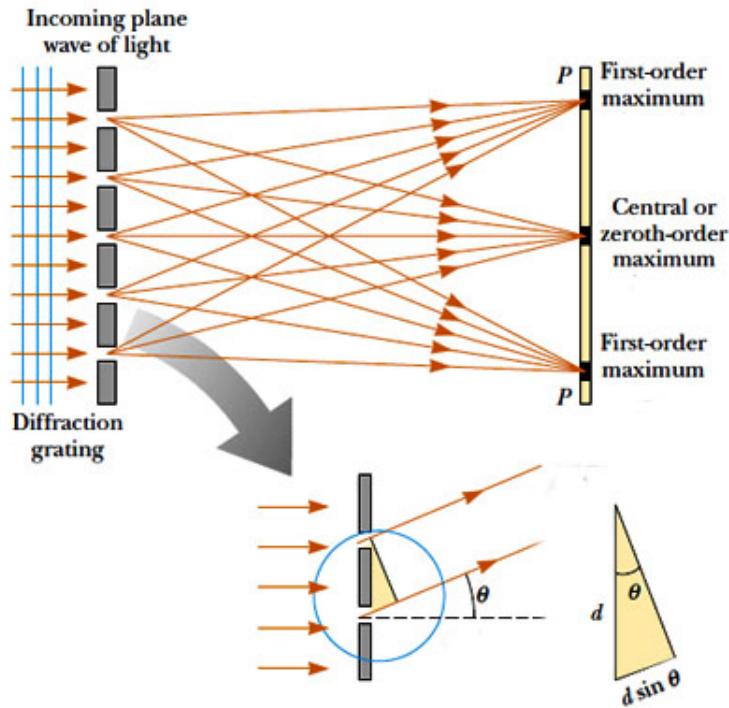


Abbildung 4.2: Light directed to parallel to grating:

The diffraction pattern on the screen is the result of the combined effects of diffraction and interference. Each slit causes diffraction, and the diffracted beams in turn interfere with one another to produce the pattern. The path difference between waves from any two adjacent slits can be found by dropping a perpendicular line between the parallel waves. By geometry, this path difference is  $d \sin(\theta)$ . If the path difference equals one wavelength or some integral multiple of a wavelength, 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  $\theta$  is:

$$d \sin(\theta) = m\lambda \quad (4.1)$$

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 splitted into very narrow bright fringes at large angles  $\theta$ .

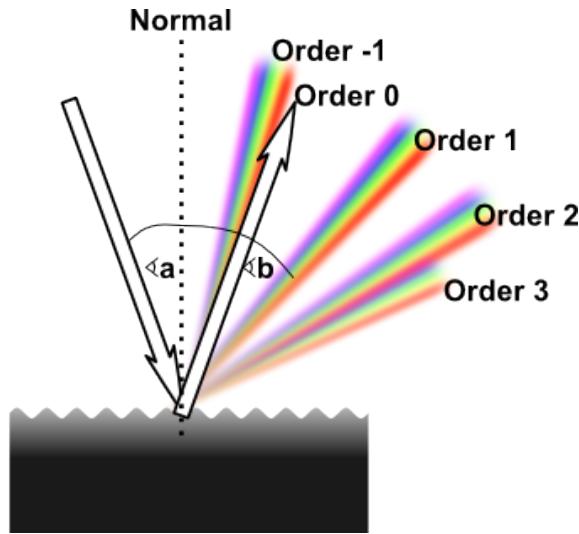


Abbildung 4.3: Different Orders of diffraction: When light is directed at the grating not parallel to its axis, there is another  $\sin(a)$  involved. See grating equation 4.2.

When a narrow beam of white light is directed at a diffraction grating along its axis, instead of a monochromatic bright fringe, a set of coloured spectra are observed on both sides of the central white band as shown in figure 4.3.

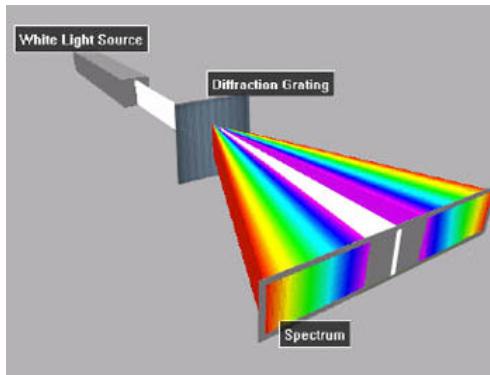


Abbildung 4.4: White Light beam causes coloured diffraction spectra

Since  $\theta$  increases with wavelength  $\lambda$ , red light which has the longest wavelength is diffracted through the largest angle. Violet light has the shortest wavelength and is diffracted the least. Thus, white light is split into its component colours from violet to red light. The spectrum is repeated in the different orders of diffraction. Only the zeroth order spectrum is pure white.

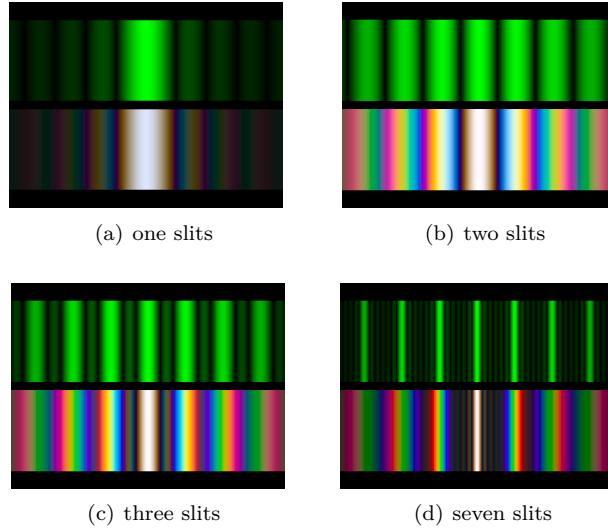


Abbildung 4.5: Difference of diffraction pattern between a monochromatic (top) and a white (bottom) light spectra for different number of slits.

### 4.3 Evaluation

#### UNIFY ANGLE NAMES:

In order to check the physical reliability of our method we applied it on a synthetic blazed grating, Elaphe and Xenopeltis snake sheds and evaluated its response using the grating equation. This equation models the relationship between the grating spacing and the angles of the incident and diffracted beams of light.

When light at a wavelength  $\lambda$  falls on a sample presenting a periodicity  $d$  along the incident plane under an incident angle  $\theta$  compared to the normal of the surface the angle  $\phi$  corresponding to the direction of the emerging beam showing constructive interferences (maximum in intensity) is given by the grating equation:

$$\sin(\theta) = \sin(\phi) + \frac{m\lambda}{d} \quad (4.2)$$

In our evaluation we are interested in the zero order diffraction, i.e.  $m$  equals zero which corresponds to direct transmission or specular reflection in the case of a reflection grating. Within our evaluation we further assume that the incident light direction  $w_i$  is given. In contrast the direction of the reflected wave  $w_r$  is not given. In Mathematics, a three dimensional direction vector is fully defined by two two angles, i.e. it can be represented by spherical coordinates with radius  $r = 1$ . By convention, we denote those two vectors by  $\theta$  and  $\phi$ . Hence,  $\theta_i$ ,  $\phi_i$  and  $\phi_r$  are given constants whereas  $\theta_i$  is a free parameter for our evaluation simulation. Therefore, we are going to compare the maxima for peak viewing angle corresponding to each wavelength using data produced by our method against the maxima resulting by the grating equation.

### 4.3.1 Precomputation

#### SHOW IMAGE OF GRID

Before being able to compare the output produced by our method by the grating equation, we have to discretise the wavelength space  $\Lambda$  and the range  $\Theta$  of our free parameter  $\theta_i$ . We also have to initially assign  $\theta_i$ ,  $\phi_i$  and  $\phi_r$ . For our experiments we choose the following initial setup:  $\theta_i = 75$   $\phi_i = 0$   $\phi_r = 180$  degree. Further we discretise the lambda space  $\Lambda = \{\lambda | \lambda = \lambda_{min} + k\lambda_{step}, k \in \{0, \dots, C - 1\}\}$  where  $\lambda_{step} = \frac{\lambda_{max} - \lambda_{min}}{C - 1}$  and  $C$  is the discretisation level of the lambda space, in our scenario  $C = 42$ . We similarly discretise the angle space by predefining a minimal and maximal angle boundary and  $ceil(\text{angMax} - \text{angMin})/\text{angInc}$  is the number of angles. We are going implement a similar algorithm as the diffraction fragment shader algorithme on the grid  $[\Lambda, \Theta]$  and will store its spectral response in a matrix  $R = \{\text{response}(\lambda_i, \theta_r^j) | i\text{index}(\Lambda), j\text{index}(\Theta)\}$ . We also have evaluated our other shaders, mentioned within the discussion of the derivation and implementation chapters.

### 4.3.2 Data evaluation

---

#### Algorithm 4.1 Evaluation: lambda theta graph

---

```
% load all variables computed in java
lInc = (lMax - lMin)./(lambdaCnt - 1);
lambda = lMin + lInc*(-1+[1:lambdaCnt]);
[maxC maxI] = max(response.');
viewAngForMax = angMin + angInc * (maxI-1);
plot(lambda, viewAngForMax,'-r');

for thetaI=baseAngle-eps:0.5:baseAngle+eps,
    % grating equation
    thetaV = asin(lambda./dPeriod - sin(thetaI*pi()/180))*180/pi();
    if(thetaI==75)
        plot(lambda, thetaV,'+ b');
    else
        plot(lambda, thetaV, '. g');
    end
end
```

---

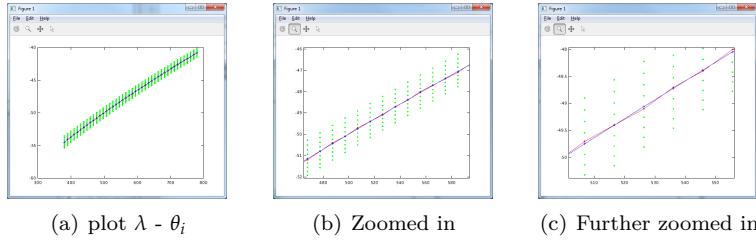


Abbildung 4.6: Evaluation: blaze grating

red graph is based on data produced by our method, the blue and green graphs are plots from the grating equation for different angles. If the blue graph is close to the red one, then our method performs well.

SHOW PLOTS AND TALK ABOUT THEM

subvariants: sample whole lambda space, just a few lambdas, pq approach.  
discussion

# Kapitel 5

## Results

show comparision between all shader variants: pq, adaptive, all, nvidiaGem. mention system specs and how long it took in order to precompute mention GEM results. show nminmax renderings



(a) A



(b) B

Abbildung 5.1: Species Elaphe Guttata



(a) A



(b) B

Abbildung 5.2: Species Xenopeltis Unicolor

Mention basic set up

brdf map for blazed grating shows typical high relative brightness for the first order of diffraction.

elaphe skin, finger-like nanostructure quite regularly placed, thus exhibits diffraction across these periodic arrangements, i.e. there is a long horizontal axis for brdf map. furthermore fingers overlap across layers thus do not exhibit any definite periodicity along finger directions. hence do not see strong diffraction color along other directions in brdf map.

for xeno, layers of nano-fingers do not overlap and they manoeuvre significantly along their length but with some local consistency. thus see diffraction along many different directions in brdf map. similar argument for diffraction across locally periodic finger patches with slightly different orientations.

the elapse does not look that regular under the electron scanning microscope. This is why it is much less iridescent than the other species.

In addition, the micro-geometry is highly similar among snake species, it is the geometry of the nanostructures that are highly different among species and that cause the snake to be or not be iridescent. So, even Xenopeltis would not give you very different geometry than Elaphe.

Xeno has a brownish body with no pattern that makes the iridescence more spectacular than on Ellaphe

## 5.1 BRDF maps

Following passage explains the structure of the partial BRD maps that are presented in this report. Consider a sample the partial BRD maps shown below :

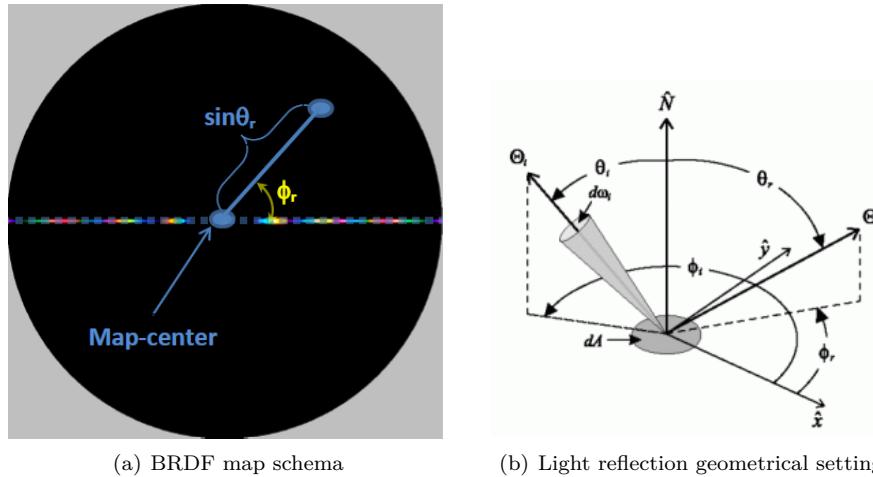


Abbildung 5.3: BRDF maps for different patches:  $\Theta = (\theta, \phi)$  is the direction of light propagation

Distance from the center of the (circular) map =  $\sin(\theta_r)$  Angle with the positive X-Axis in the map =  $\phi_r$   $\theta_i$  and  $\phi_i$  are constant for a given map, plot. For all the results presented in this report  $\phi_i = \pi$

BRDF map shows shader output for all viewing directions and a fixed incident light direction. viewing direction represented in spherical coordinates  $(\theta_v, \phi_v)$  is represented in map at point  $(x, y) = (\sin(\theta_v)\cos(\phi_v), \sin(\theta_v)\sin(\phi_v))$  with origin at map center. fix light direction for normal incidence, i.e.  $(\theta_i, \phi_i) = (0, 0)$  unless specified otherwise. See figure 5.3.

Mention brdf map basic idea figure + text

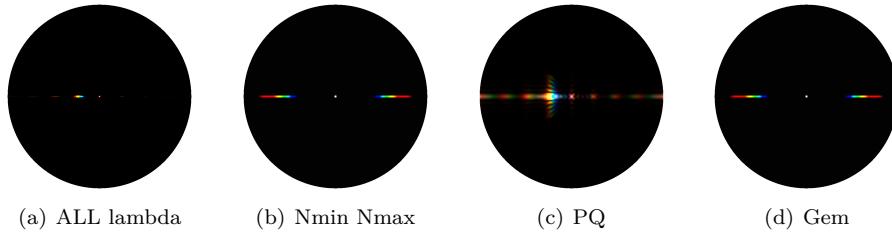


Abbildung 5.4: BRDF maps for blaze grating comparing our different rendering approaches

brdf maps for different nanoscale surface gratings

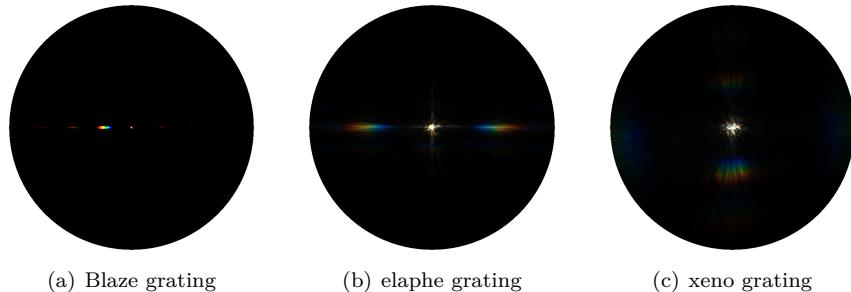
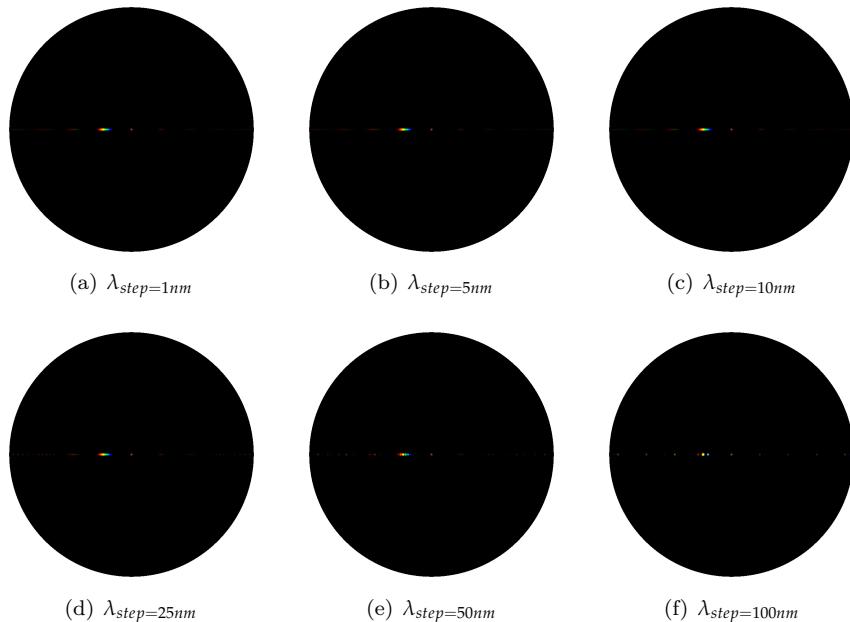
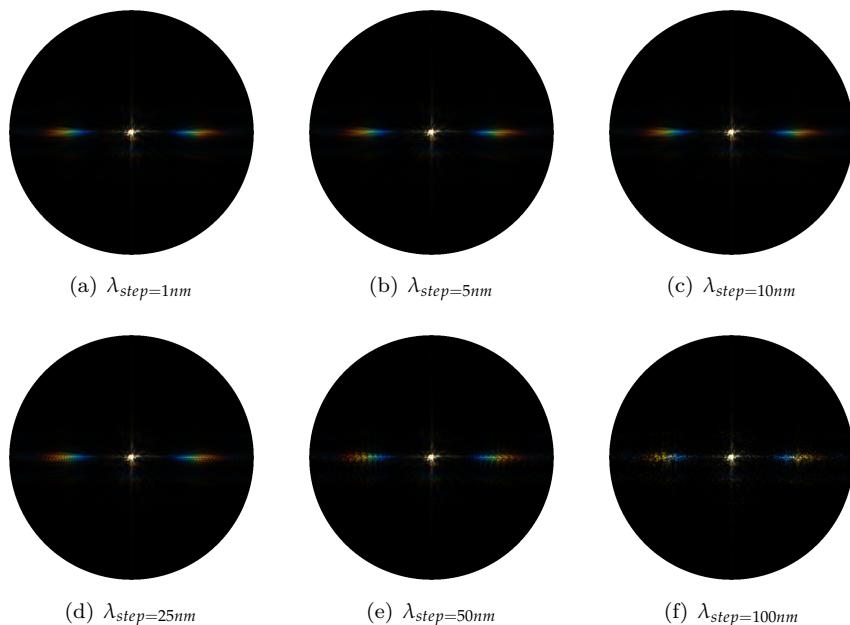


Abbildung 5.5: BRDF maps for different patches

for blaze grating different wavelength step sizes in fragment shader wider steps fewer iterations, loss of quality. for elaphe step bigger than 10nm there are visible artifacts. for blaze grating bigger step sizes possible.

Abbildung 5.6: Blaze grating at  $2.5\mu m$ : Different  $\lambda$  step sizesAbbildung 5.7: Elaphe grating at  $65\mu m$ : Different  $\lambda$  step sizes

compare pq qith full lambda sampling. see similarities but also differences

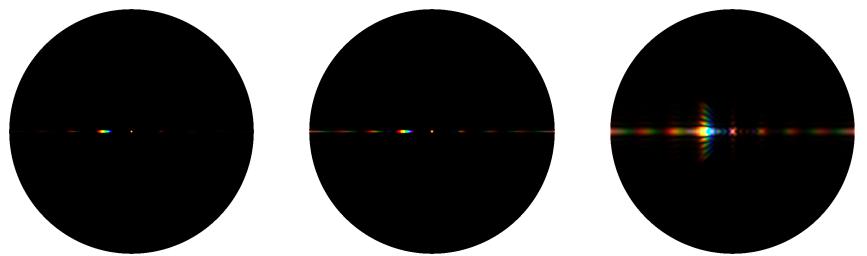
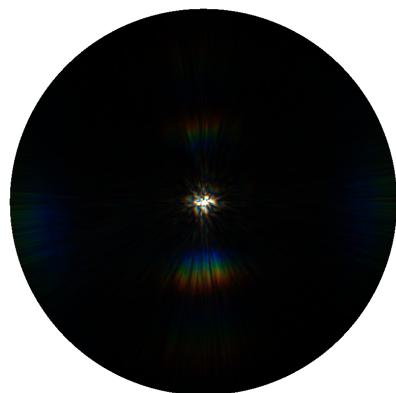
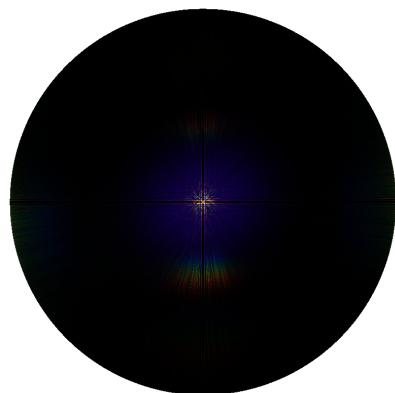


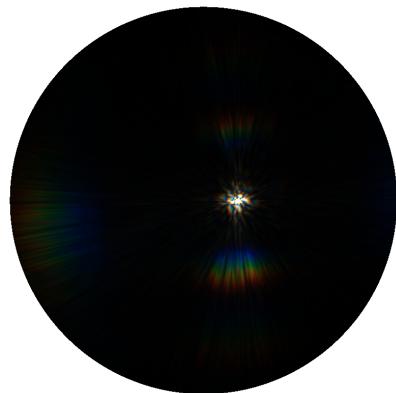
Abbildung 5.8: Blaze grating: PQ approach vs full lambda space sampling



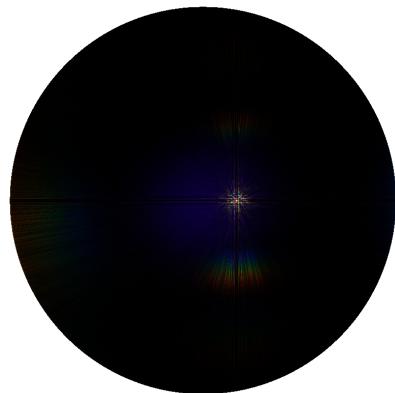
(a) Full Lambda Sampling: Xeno grating  $\theta_i = 0\text{degree}$



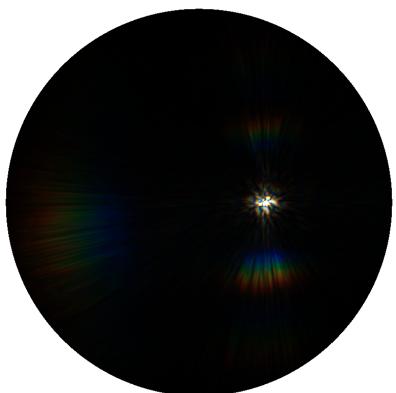
(b) PQ Approach: Xeno grating  $\theta_i = 0\text{degree}$



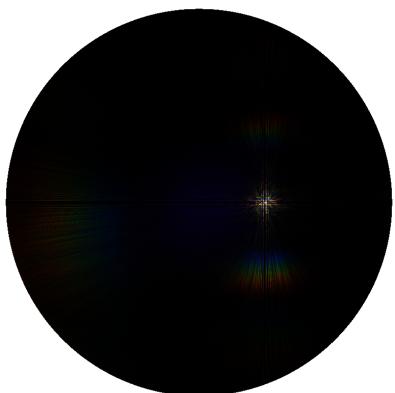
(c) Full Lambda Sampling: Xeno grating  $\theta_i = 10\text{degree}$



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



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



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

Abbildung 5.9: Xeno grating: PQ approach vs full lambda space sampling

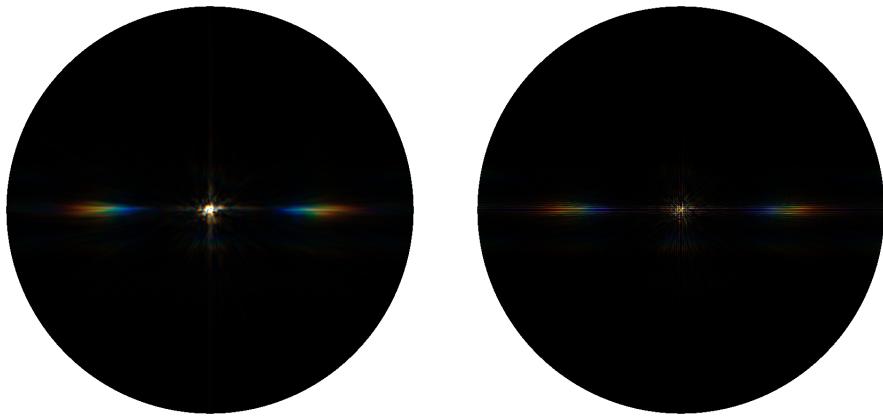


Abbildung 5.10: Elaphe grating: PQ approach vs full lambda space sampling

blaze grating lower coherence length, fewer interacting grating periods, produce blurred diffraction bands for different lambda which overlap to produce poorly resolved colors.

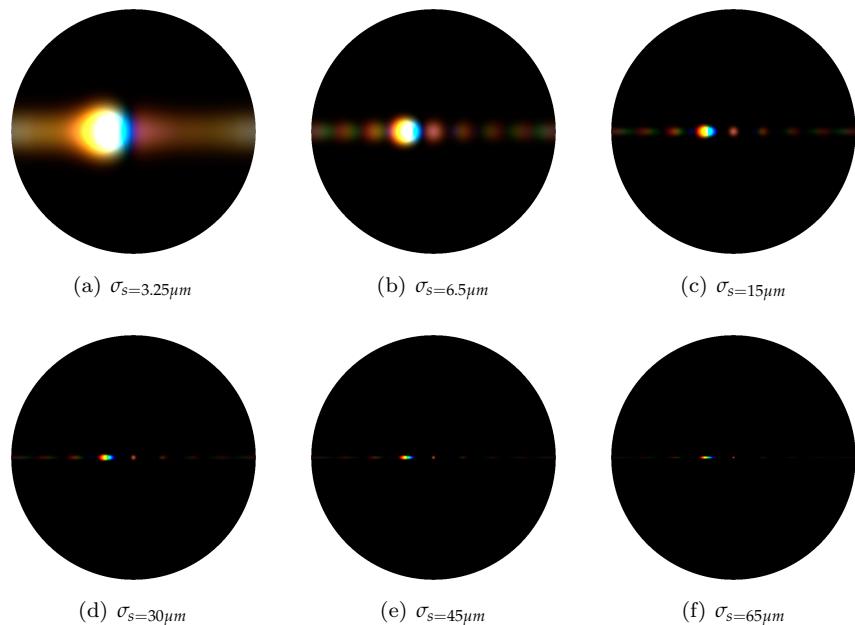
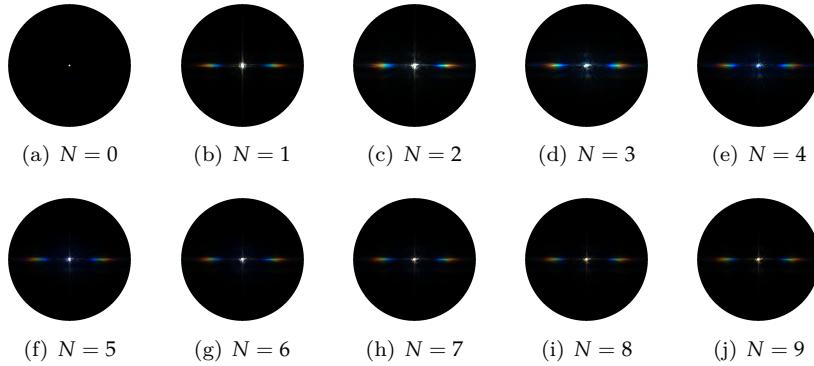
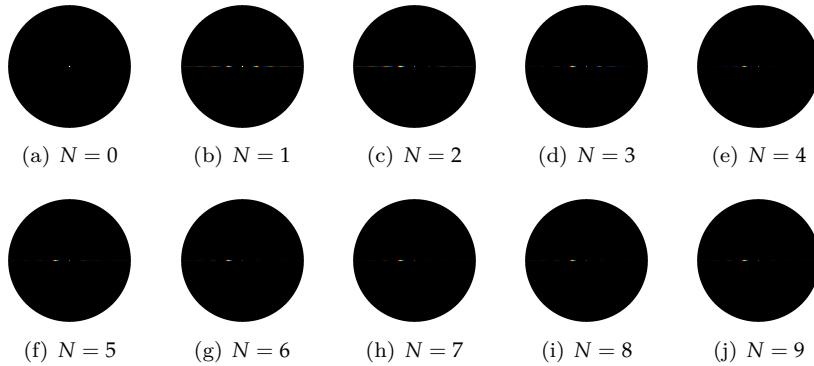
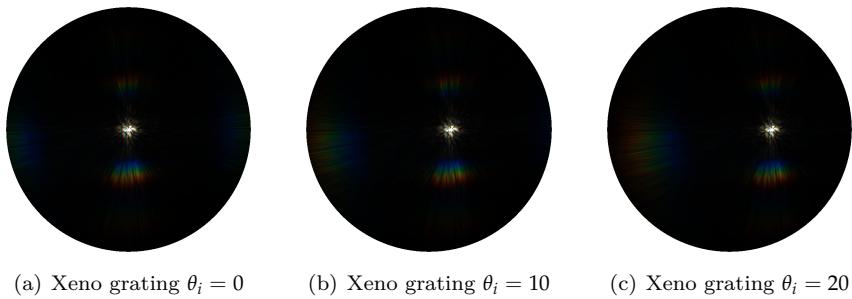


Abbildung 5.11: Blaze grating at  $2.5\mu m$ : Different  $\sigma_s$  sizes

visually convergence taylor series for higher values for N.

Abbildung 5.12: Elaphe grating at  $65\mu\text{m}$ :  $N$  Taylor IterationsAbbildung 5.13: Blaze grating at  $2.5\mu\text{m}$ :  $N$  Taylor IterationsAbbildung 5.14: BRDF maps for Xeno grating: different  $\theta_i$  angles

## 5.2 Snake surface geometries

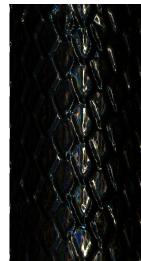
initially using fully lambda shader for those renderings, slow but accurate.

diffraction colors change dramatically with changes in light direction, surface normals and viewing direction, which is typical for diffraction colors observed in nature.

Mention resolution of patches and their runtimes with used hardware - specify this hardware too.



(a) Blaze grating



(b) Elaphe grating



(c) Xeno grating

Abbildung 5.15: Diffraction of different snake skin gratings rendered on a snake geometry

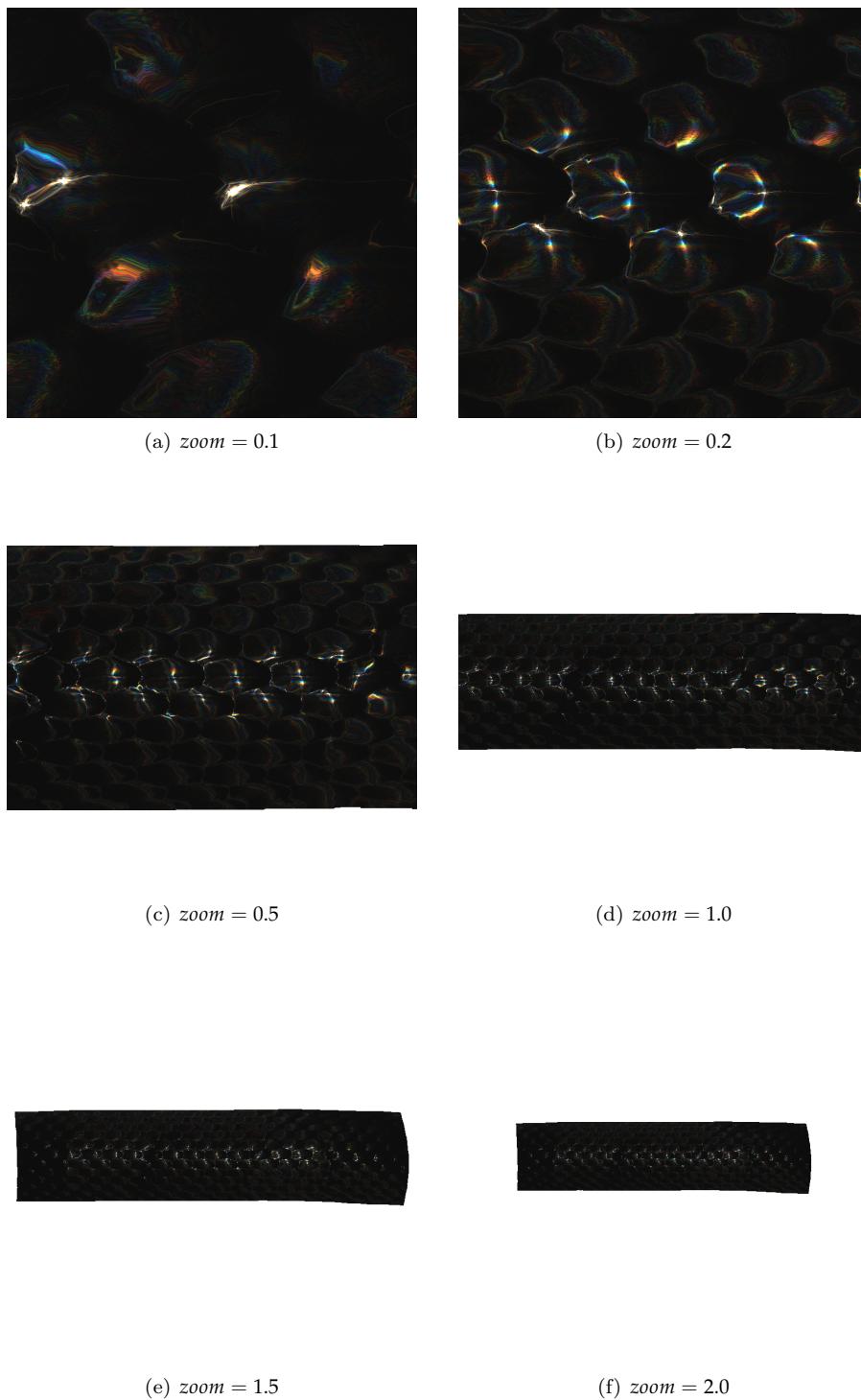


Abbildung 5.16: Diffraction on Elaphe snake skin grating: Different camera zoom levels

(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)  $(0.0995, 0.4770, -0.8731)$ 

Abbildung 5.17: Diffraction on Elaphe snake skin grating: Different light directions

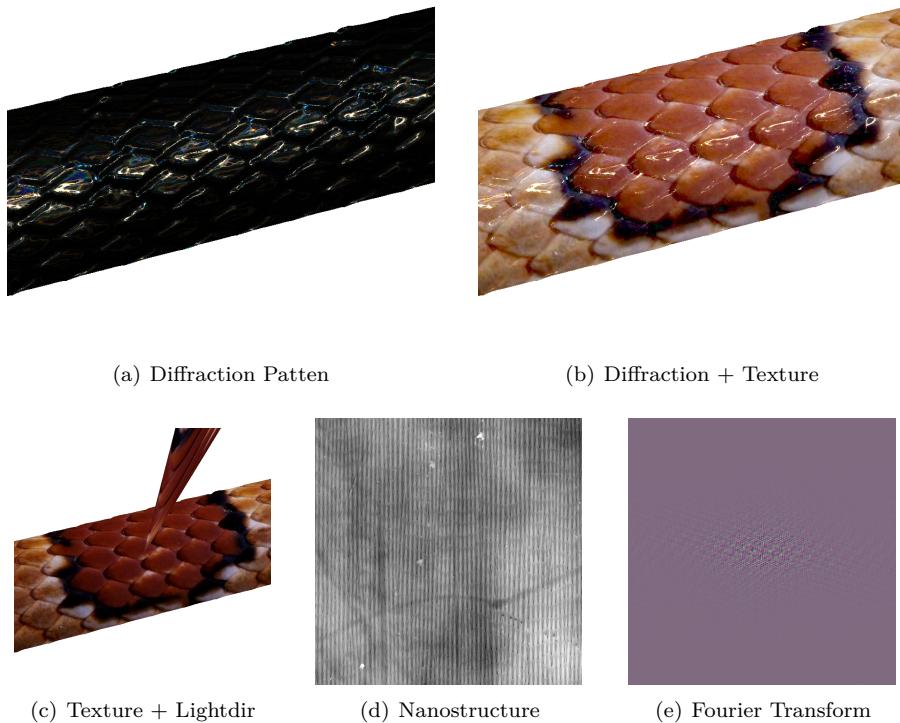


Abbildung 5.18: Diffraction for Elaphe snake skin

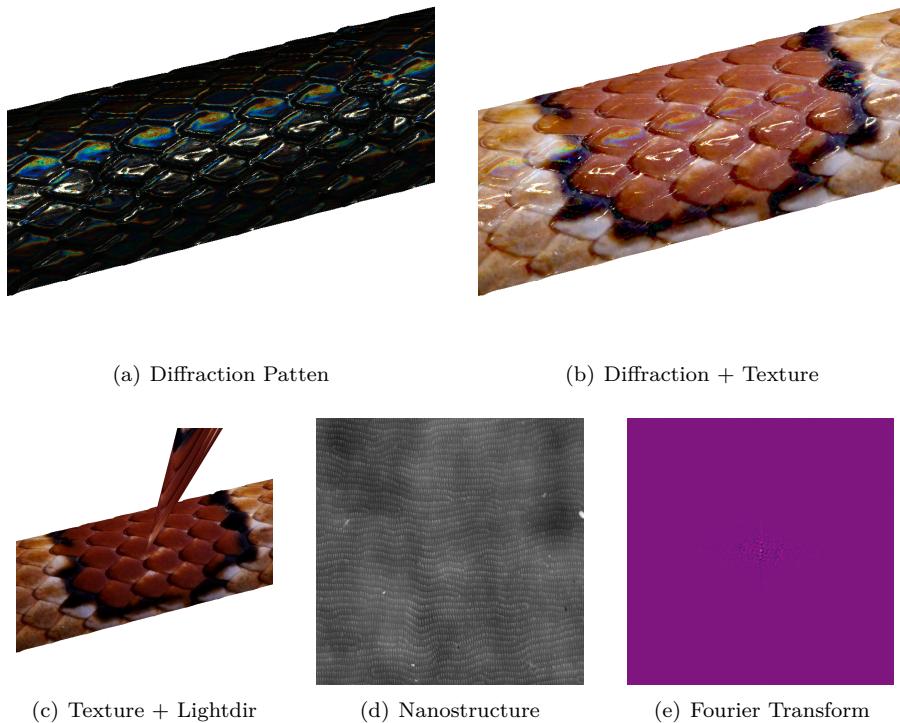
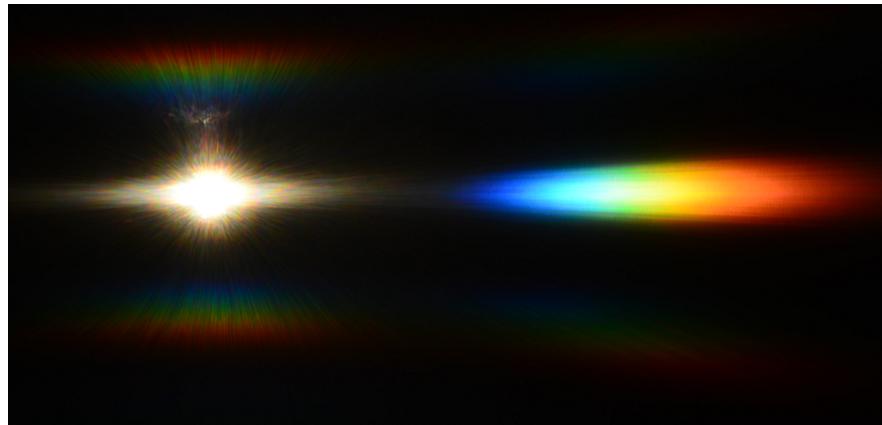


Abbildung 5.19: Diffraction for Xeno snake skin

### 5.3 Snake surface geometries



(a) Simulation



(b) Experiment

Abbildung 5.20: Diffraction Elpahe: simulation vs. experimental setup

# Kapitel 6

# Conclusion

can we do better?

brief overview pf results achieved, what was the most important in the work, appropriate to provide an introduction to possible future work in this field. reflect the emotions associated with the work, what was especially difficult or particularly interesting, one may elaborate on open questions within subjects related to the thesis without giving any answer. discuss follow-ups.

statement what you've researched and what your original contribution of the fild is explain why our approach is a good idea explain how the straight forward approach would behave compared to our approach, computing the fourier transformations straight away. explain what we achieved, summary say something about draw-backs and about limitations of current apporach say something about the ongoing paper future work maybe say something about runtime complexity

## 6.1 Further Work

### 6.1.1 References

refs

## 6.2 Acknowledgment

# Anhang A

# Appendix

## A.1 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:

$$R(\theta) = R_0 + (1 - R_0)(1 - \cos \theta)^5 \quad (\text{A.1})$$

and

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

where  $\theta$  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.

## A.2 Spherical Coordinates

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

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

# Tabellenverzeichnis

- |   |    |
|---|----|
| 2.1 Fourier operator to apply for a given spatial input signal and the properties of its resulting output signal in frequency space . . . . | 12 |
|---|----|

# Abbildungsverzeichnis

2.1	Diffraction: Single Slit Example . . . . .	5
2.2	Diffraction Pattern . . . . .	5
2.3	Plots of our color matching functions we used for rendering . . . . .	9
2.4	Relationship between the continuous Fourier transform and the discrete Fourier transform: Left column: A continuous function (top) and its Fourier transform (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. 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 (bottom) computes discrete samples of the continuous DTFT. The inverse DFT (top) is a periodic summation of the orginal samples.	11
2.5	$\omega_i$ points toward the light source, $\omega_r$ points toward the camera, $n$ is the surface normal . . . . .	13
3.1	Blaze . . . . .	30
3.2	Renderer Architecture . . . . .	31
3.3	( $u, v$ ) lookup image . . . . .	36
4.1	Spectrometer: 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. . . . .	40
4.2	Light directed to parallel to grating: . . . . .	41
4.3	Different Orders of diffraction: When light is directed at the grating not parallel to its axis, there is another $\sin(a)$ involved. See grating equation 4.2. . . . .	42
4.4	White Light beam causes coloured diffraction spectra . . . . .	42
4.5	Difference of diffraction pattern between a monochromatic (top) and a white (bottom) light spectra for different number of slits. .	43
4.6	Evaluation: blaze grating . . . . .	45
5.1	Species Elaphe Guttata . . . . .	46
5.2	Species Xenopeltis Unicolor . . . . .	46
5.3	BRDF maps for different patches: $\Theta = (\theta, \phi)$ is the direction of light propagation . . . . .	47

5.4	BRDF maps for blaze grating comparing our different rendering approaches . . . . .	48
5.5	BRDF maps for different patches . . . . .	48
5.6	Blaze grating at $2.5\mu m$ : Different $\lambda$ step sizes . . . . .	49
5.7	Elaphe grating at $65\mu m$ : Different $\lambda$ step sizes . . . . .	49
5.8	Blaze grating: PQ approach vs full lambda space sampling . . . . .	50
5.9	Xeno grating: PQ approach vs full lambda space sampling . . . . .	51
5.10	Elaphe grating: PQ approach vs full lambda space sampling . . . . .	52
5.11	Blaze grating at $2.5\mu m$ : Different $\sigma_s$ sizes . . . . .	52
5.12	Elaphe grating at $65\mu m$ : $N$ Taylor Iterations . . . . .	53
5.13	Blaze grating at $2.5\mu m$ : $N$ Taylor Iterations . . . . .	53
5.14	BRDF maps for Xeno grating: different $\theta_i$ angles . . . . .	53
5.15	Diffraction of different snake skin gratings rendered on a snake geometry . . . . .	54
5.16	Diffraction on Elaphe snake skin grating: Different camera zoom levels . . . . .	55
5.17	Diffraction on Elaphe snake skin grating: Different light directions . . . . .	56
5.18	Diffraction for Elaphe snake skin . . . . .	57
5.19	Diffraction for Xeno snake skin . . . . .	58
5.20	Diffraction Elpahe: simulation vs. experimental setup . . . . .	59

# List of Algorithms

3.1	Precomputation: Fourier images . . . . .	29
3.2	Vertex diffraction shader . . . . .	32
3.3	Fragment diffraction shader . . . . .	34
3.4	Texture Blending . . . . .	37
3.5	Sinc interpolation for pq approach . . . . .	38
4.1	Evaluation: lambda thetar graph . . . . .	44

# Literaturverzeichnis

[Doe00] DOE, John: *Title*. Publisher, 0000. – ISBN 0000000000

## Erklärung

gemäss Art. 28 Abs. 2 RSL 05

Name/Vorname: .....

Matrikelnummer: .....

Studiengang: .....

Bachelor

Master

Dissertation

Titel der Arbeit: .....

.....

.....

LeiterIn der Arbeit: .....

.....

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

.....  
Ort/Datum

.....  
Unterschrift