Matti Kinnunen

Small-Scale Texture Mapping

Master's Thesis in Information Technology

April 2, 2018

University of Jyväskylä

Department of Mathematical Information Technology

Author: Matti Kinnunen

Contact information: matti.k.kinnunen@jyu.fi

Supervisor: Tuomo Rossi, Jukka Räbinä

Title: Small-Scale Texture Mapping

Työn nimi: Tekstuurien käyttö pienen skaalan pintaominaisuuksien mallintamisessa

Project: Master's Thesis

Study line: Software Engineering

Page count: 63+0

Abstract: This master's thesis examines the applicability of texture mapping in modeling small-scale surface features of materials in computer graphics. A technique is introduced, implemented and examined using a custom software renderer. This technique uses sampled small-scale albedo, normal, and displacement maps to model complex small-scale surface features and their effect on surface reflectance. The implementation of the technique was able to model these features, but further improvements on the technique are needed to address the visual accuracy and computational efficiency.

Keywords: computer graphics, bi-scale, materials

Suomenkielinen tiivistelmä: Tämä pro gradu -tutkielma tarkastelee tekstuurien ja pintakuviointien soveltuvuutta pienen skaalan materiaaliominaisuuksien mallintamiseen tietokonegrafiikassa. Tutkielmassa esitellään tähän liittyvä tekniikka sekä toteutetaan ja tarkastellaan tekniikkaa räätälöidyllä renderöintiohjelmistolla. Tekniikka käyttää albedo-, normaali- ja siirtymäkarttoja mallintamaan kompleksisia pienen skaalan pintaominaisuuksia sekä niiden vaikutusta valon heijastumaan. Ohjelmistolla onnistuttiin mallintamaan näitä ominaisuuksia, mutta tekniikka tarvitsee parannuksia visuaalisen tarkkuuden ja laskennallisen tehokkuuden parantamiseksi.

Avainsanat: tietokonegrafiikka, materiaalit

List of Figures

List of Tables

Contents

1 Introduction

The field of computer graphics is interested in forming synthetic images from 3D environments. In the process of creating these images, material appearance is important to create a visually accurate description (Dorsey, Rushmeier, and Sillion 2008). Material appearance is a term used for the visual impression we get from a material of an object. In computer graphics, it describes the models used to simulate real life materials and render these simulations to form an image. These simulated models build on the knowledge in human perception, physics of light and image formation.

To form an image and a given object's visual appearance in computer graphics, we need to model the object's shape, material, and the incident light of the object (Dorsey, Rushmeier, and Sillion 2008). The object's shape is generally modeled explicitly with polygons or parametric patches estimating the approximate shape of the object (Westin, Arvo, and Torrance 1992). The finer details and the way they interact with the incident light are defined in the material, which can be modeled with different kinds of textures and bidirectional reflectance distribution functions (BRDFs). Usually, the surface features that are larger than one pixel are modeled with textures and the sub-pixel sized features are encoded in the BRDF. Physically-based BRDF models use a theoretical model of the material's microgeometry features to model the light reflectance of a material (Akenine-Moller, Haines, and Hoffman 2008). These BRDF models are able to reproduce many effects of local light scattering quite well, but are limited in their ability to model a wider array of surfaces (Westin, Arvo, and Torrance 1992).

Material appearance can have considerable variation when the same material is viewed at different scales (Wu, Dorsey, and Rushmeier 2011). The small-scale surface features, which are visible at a close range, merge into a single reflectance description as the same material is viewed at a larger distance. Physically this means that averaging the small-scale details determines the large-scale material appearance. These small-scale details of materials can therefore have considerable effects on material appearance, even when the materials are viewed at a distance.

In this thesis, a technique is presented in attempt to model the small-scale surface features of materials explicitly with different texture mapping methods. Concretely, this means that in addition to the scale representations described above (object, texture, BRDF), another representation of the small-scale material features is added. This small-scale texture representation therefore inhabits the space between the larger scale texture representation (meso-scale) and the microscale BRDF.

The previous methods attempting to model these features use explicitly modeled 3D surfaces to model these features and then form a representation of the features that can be computed in an efficient way (Westin, Arvo, and Torrance 1992; Wu, Dorsey, and Rushmeier 2011; Iwasaki, Dobashi, and Nishita 2012). The novelty of the technique presented in this thesis, is the fact that this modeling is done with common texture mapping methods, including normal mapping and different parallax mapping techniques. The parallax mapping techniques in particular are interesting cases for the modeling of small-scale features as they are capable of producing local occlusion and shadowing needed for physically-based lighting.

Chapter 2 of the thesis introduces the previous work on the subject of modeling light and it's interaction with matter, starting with the work done in the field of physics, and moving on to modeling the light-matter interaction in graphics applications. In Chapter 3, bidirectional reflectance distribution functions (BRDFs) are discussed and multiple BRDF models developed in computer graphics research are presented. In Chapter 4, the common approaches to texturing and sampling are introduced. In Chapter 5, the technique presented in this thesis is described in detail. Chapter 6 introduces the results of the implementation and examines the visual results as well as performance figures from the said implementation. In Chapter 7, the results and approach of the introduced method are discussed and future improvements are proposed.

1.1 Terminology

The terminology of scale in this thesis will follow that of Wu, Dorsey, and Rushmeier (2011) and Iwasaki, Dobashi, and Nishita (2012), where the different scales are referred to as small-scale and large-scale. Westin, Arvo, and Torrance (1992) proposed the use of micro/milli-

scale, but recent studies in the area are increasingly using the small/large-scale distinction not to impose absolute sizes for the scales. Small-scale refers to effects that happen on a scale greater than the wave-length of light and large-scale refers to significantly larger scale than the small-scale (Wu, Dorsey, and Rushmeier 2011).

2 Modeling Light

When the topic of rendering photo-realistic materials in digital images is approached, attention should be given to the considerable amount of literature found in the field of physics. In rendering, we often try to simulate the conditions and appearance of our physical world. Our algorithms attempt this by simulating the physical model of light and it's transport. This area has been extensively studied in the field of *radiometry* (Akenine-Moller, Haines, and Hoffman 2008). Radiometry is the core field in the study of physical transmission of light. The understanding of the underlying physical principles is important as we attempt to simulate the appearance of the physical world around us.

In this section, we first turn our attention to light in general and how we will approach it in this study. Next, we turn our attention to radiometry, radiometric units and their relationships to lay the foundation to the following discussion on theory of *bi-directional reflectance distribution functions* (BRDFs).

2.1 Nature of Light

Light is *electromagnetic radiation* which consists of a flow of *photons* in the *visible spectrum* (Akenine-Moller, Haines, and Hoffman 2008). Light has a dual nature as it behaves as either a flow of particles or as a wave depending on the situation. This dual nature is addressed in the field of *quantum optics* (Glassner 1994). Quantum optics give us a basic idea for thinking about photons. It says that a photon can be thought of as a small, localized *wave packet*, that is a wave but doesn't extend infinitely.

In this study, the wave nature of light is mostly ignored. This means that we cannot model certain phenomena, namely *polarization*, *interference*, and *diffraction*, which depend on the wave properties of photons. Although these are important phenomena, they are usually ignored in rendering systems (Akenine-Moller, Haines, and Hoffman 2008). The model of *geometric optics*, where light is regarded as traveling along straight lines between different surfaces, serves to accurately model the light interactions at the scale of human activity (Dorsey, Rushmeier, and Sillion 2008).

The wave-related property of photons that will not be ignored is the photon's frequency or wave-length. Energy of a photon is proportional to it's frequency (Akenine-Moller, Haines, and Hoffman 2008). The energy and frequency of the photon affects it's interaction with sensors, matter, and other photons. In general, humans perceive different photon frequencies between approximately 380 and 780 nanometers as different colors. Electromagnetic frequencies outside of the visible spectrum are not perceptile to the human eye although the frequencies of electromagnetic radiation range from extremely low frequency radio waves to gamma rays.

In geometric optics, light travels along straight ray paths between different surfaces (Dorsey, Rushmeier, and Sillion 2008). This path begins as light is emitted from a light source. Emitted light travels in a straight path until it hits a surface. After the light hits a surface it causes it to change directions either into the encountered matter or away from it.

2.2 Light-Matter Interaction

Light interacts with matter in two ways: it is *scattered* and *absorbed* (Akenine-Moller, Haines, and Hoffman 2008). These two phenomena can describe fundamentally all light-matter interactions. Scattering causes the light to change direction. It does not affect the amount of light, but only it's direction. Scattering is caused by all kinds of optical discontinuities, for example an interface between two different surfaces with different optical properties. In absorbtion, the amount of light is reduced, but the direction is not changed. This happens inside matter. Absorbtion converts light energy to some other form of energy and the light disappears.

Light is scattered in two ways: it is *reflected* out of the material or it is *transmitted* into it (Akenine-Moller, Haines, and Hoffman 2008). In reflection, light is partly or fully propagated outward by the material. Transmission or *refraction* is a similar process in which light passes through the interface of two materials and into the material.

Usually simple models of reflection categorize different kinds of reflection in categories (Glassner 1994). These categories explain how light is propagated by a surface. For the purposes of this thesis, we will introduce the most relevant categories to computer graphics

rendering systems. These categories are *specular* reflection and *diffuse* reflection. Specular reflection changes the direction of the light without actually scattering it, as from a surface of a smooth mirror. Diffuse reflection, on the other hand, scatters light in every direction with equal energy.

There are several categories of transmission of light into the material (Glassner 1994). Most relevant for computer graphics are *specular* transmission and *diffuse* transmission. The categories behave similarly as their respective categories in reflection, with the difference of light being directed into the material, whereas it is directed outward in reflection.

As we consider these different kinds of light-matter interactions, the geometric optics model needs an addition in the form of *scattering distributions* (Dorsey, Rushmeier, and Sillion 2008). These distributions describe how much of the arriving light leaves the surface in each direction. These scattering functions are discussed in detail later in section 2.3. We will lay the mathematical and conceptual foundation for this discussion in the next section.

2.2.1 Radiometry

In the study of light-matter interaction, the concepts and units discussed in radiometry are important to model these interactions mathematically. Radiometry is the core field that studies the physical transmission of light and it deals with the measurement of electromagnetic radiation, that consists of a flow of *photons* (Akenine-Moller, Haines, and Hoffman 2008).

Photons behave as particles or waves depending on the situation. The geometric optics think of photons as particles which works usually quite well in rendering (Akenine-Moller, Haines, and Hoffman 2008). As discussed earlier, some phenomena cannot be modeled without the wave properties of photons, but these phenomena are usually ignored in most rendering systems, and this is the case for this study as well.

Next, the various radiometric units and their relationships are discussed to lay the foundation for the later introduction of the BRDF theory. Rest of this section is mostly based on Dorsey, Rushmeier, and Sillion (2008).

Radiant energy Q is the basic energy unit in radiometry, measured in *joules* (*J*). The *radiant*

power or *radiant flux* is the rate of energy transfer per unit time and it is expressed as *watts* (W). Average flux is denoted as Φ. We get the average flux as the energy ΔQ transferred through some period of time Δt.

$$\Phi = \frac{\Delta Q}{\Delta t} \tag{2.1}$$

As we need to find the $\Phi(t)$ at a particular instant so we use differentials dQ and dt which are the quantities ΔQ and Δt as they approach zero:

$$\Phi(t) = \frac{dQ}{dt}. \tag{2.2}$$

Light travels at the speed of nearly 300 million meters per second so we will not carry the dependence on time in our later definitions because we are not going to be generating images in this time frame.

The *average flux per unit area* is referred to as the *radiant exitance* M. It is the total flux Φ divided by surface area A:

$$M = \frac{\Phi}{A} \tag{2.3}$$

As we did with time, we divide the surface area A into infinitesimally small areas dA and consider the flux $d\Phi$ from these areas individually. These infinitesimally small areas are located at a single point (x,y). We define the *radiant exitance* $M(x,y)$ at a specific position as

$$M(x,y) = \frac{d\Phi(x,y)}{dA}. \tag{2.4}$$

Irradiance E refers to radiant energy arriving at the surface. It is defined in a similar manner as radiant exitance M, with the only difference being that the radiant energy is arriving in the surface.

Next, we consider the radiant flux in a particular direction. As before, we start with considering the set of all directions. We can think of the set of all directions as an unit sphere. Any single direction can be specified as a point on the unit sphere and any set of directions as a surface on the unit sphere. A set of directions is called a solid angle, which is measured in steradians. Steradian is the three-dimensional analog of radian. An unit sphere's area is 4π and the solid angle of a sphere is 4π steradians.

Radiant intensity is the radiant flux per unit solid angle. The average radiant intensity I over the set of all directions is defined as

$$I = \frac{\Phi}{4\pi}. \tag{2.5}$$

Again, we can divide this to smaller sets of directions and express a particular point as an infinitesimally small set of directions $d\omega$. Additionally, we need coordinates to specify a particular direction.

To specify coordinates, we set up a spherical coordinate system for the directions. We define the specific angle a direction makes with a fixed zenith ("up") as the *polar angle* θ. As a second reference, we use an arbitrary direction that is perpendicular to the zenith and define a plane from the zenith and perpendicular direction. The projection of a direction to the plane makes an angle with the reference direction and this angle ϕ is called the *azimuthal angle*.

These two angles θ and ϕ form a direction and from now on Θ will be used to specify a single direction (θ, ϕ). The radiant intensity I in a particular direction is defined as

$$I(\Theta) = \frac{d\Phi(\Theta)}{d\omega}. \tag{2.6}$$

Radiance is the amount of radiant intensity per projected unit area and unit solid angle. Let dA be an infinitesimal area on a surface perpendicular to the direction where the polar angle is zero. Then, the projection of the area in direction Θ is $dA\cos\theta$. Let $d\omega$ be an infinitesimal solid angle around direction Θ.

Thus, we can define the surface radiance L in direction Θ as

$$L(\Theta) = \frac{d^2\Phi(\Theta)}{\cos\theta dA d\omega}. \tag{2.7}$$

In computer graphics we are interested in the amount and color of light that arrives to the pixel being computer in the shader. This means that we are interested in the radiance that would arrive from a visible object in a specific view direction. Radiance is the key quantity in defining light transfer and important when computing the lighting in computer graphics.

2.2.2 Color

Light consists of photons in a distribution of *wavelengths* and this distribution is called the light's *spectrum* (Akenine-Moller, Haines, and Hoffman 2008). The wavelenghts of perceivable light (or the visible band of light) are between 380 and 780 nanometers. Human color perception in the eye utilises three different types of cone receptors located in the retina and these receptors respond differently to various wavelength's of light. This means that given a spectrum, the brain only receives three signals. For this reason, only three numbers can be used to represent any spectrum that humans can perceive.

Many different kinds of *color systems* have been proposed (Akenine-Moller, Haines, and Hoffman 2008). The most commonly used system in computer graphics is the RGB color model. This divides the spectrum to 3 signals r, g and b. The RGB model cannot directly represent all visible colors. Other color representations include the XYZ color space, HSB (hue, saturation, brightness) and HLS (hue, lightness, saturation). In this study, the RGB model is used because of it's practicality and common use in computer graphics.

2.3 Bidirectional Reflectance Distribution Function

Next, we will turn our attention to the *bidirectional reflectance distribution function* (BRDF) before continuing to a detailed description of *reflectance*. These descriptions in sections 2.3 and 2.4 are based on Akenine-Moller, Haines, and Hoffman (2008).

In order to shade a surface in computer graphics, we need to compute the outgoing radiance using the quantities and directions of incoming light. In radiometry, a bidirectional

reflectance distribution function (BRDF) is used for this purpose. Given the light direction $\mathbf{l}$ and view direction $\mathbf{v}$, the BRDF describes how the incoming light is reflected from a surface. The BRDF is defined as the ratio between differential radiance (outgoing radiant flux) and differential irradiance (incoming radiant flux):

$$f(\mathbf{l},\mathbf{v}) = \frac{dL_o(\mathbf{v})}{dE(\mathbf{l})}. \tag{2.8}$$

Here light is coming from a very small solid angle around $\mathbf{l}$ is measured at the surface area as irradiance dE. Radiance dL_o is proportional to the irradiance dE in any view direction $\mathbf{v}$. This ratio is the BRDF and it depends on the two vectors $\mathbf{l}$ and $\mathbf{v}$. As this value of the BRDF depends on wavelength, it is represented as an RGB vector in rendering.

The earlier section on radiometry used an angular parametrization of directionality. The BRDF can also be presented as a function of four scalar variables. This parametrization uses two angles where θ is the elevation relative to surface normal and ϕ is the rotation about the normal vector. In rendering literature, it is common to represent directionality with vectors and as we continue from physics more towards rendering, vectors will be used to represent the directionality.

In this study we will limit ourselves to non-area light sources, and as such the definition of the BRDF can be given in a non-differential form:

$$f(\mathbf{l},\mathbf{v}) = \frac{L_o(\mathbf{v})}{E_L(\mathbf{l})\overline{\cos}\theta_i}. \tag{2.9}$$

Here E_L is the irradiance of a light source on a perpendicular plane to $\mathbf{l}$. The overlined cosine represents a clamped cosine, which gives a value of 0 if the result of the cosine function is less than zero. When E_L is multiplied with the clamped cosine of the angle θ_i between the surface normal $\mathbf{n}$ and light direction vector $\mathbf{l}$, we get the irradiance on the surface.

The BRDF is an abstraction of the light-matter interaction and there are various phenomena that take place in this interaction. Some of the incoming light is transmitted through the surface or reflected away from it. As some light is transmitted through the surface it can

partially absorbed and scattered before eventually exiting the surface. This phenomena is called *subsurface scattering*.

For BRDFs used in rendering, the large-scale subsurface scattering is usually approximated as happening at a single point even though in reality the entry and exit points are not the same. In this study, we will use the BRDF with the single point subsurface scattering approximation which allows us to model all light interaction with a BRDF or a *spatially varying BRDF* (SVBRDF). The SVBRDF also captures the spatial variation of the BRDF that changes from one surface point to the next.

2.3.1 Physical constraints of BRDFs

There are two physical constraints on BRDFs that are to be considered as we attempt to model physically accurate BRDFs. These constraints are the *Helmholtz reciprocity* and *energy conservation*. Helmholtz reciprocity states that when the input and output angles are switched the function value stays the same:

$$f(\mathbf{l},\mathbf{v}) = f(\mathbf{v},\mathbf{l}). \tag{2.10}$$

Often the BRDFs that are used in computer graphics violate this principle without introducing noticeable artifacts. Energy conservation means that the total outgoing energy cannot be greater than the incoming energy (excluding light emitting surfaces). Energy conservation is not always required by rendering systems and approximate energy conservation is sufficient.

2.4 Reflectance

Light-matter interaction happens on the object's surface and in it's interior. It is important to differentiate between the two to achieve understanding of light's behaviour with real-world objects. Using BRDFs, the surface phenomena are modeled as *surface reflectance* and interior phenomena as *body reflectance*. The surface acts as an optical discontinuity which scatters light. The matter in the object's interior can absorb some of the transmitted light. The interior might also have optical discontinuities which scatter the light. In computer graphics,

the surface reflectance is usually modeled with specular terms and body reflectance using diffuse terms.

2.4.1 Surface Reflectance and Fresnel Equations

Fresnel equations describe the interaction of light with a perfectly planar interface between two substances. The requirement of a perfect plane is not plausible in real-world objects, but any surface irregularities that are considerably smaller than the shortest wavelength of light will have no effect. For this reason, we can consider such plane as an optically perfect plane.

As *Fresnel reflectance* is only a surface reflectance phenomenon, we can use it to describe the surface reflectance independent of body reflectance phenomena. Optical discontinuities cause light to change direction (scattering) as discussed earlier. An optically perfect planar interface between two substances scatters incoming light in only two directions. These directions are the ideal reflection direction and the ideal refraction direction.

The ideal reflection direction $\mathbf{r}_i$ has the same angle θ_i with the surface normal $\mathbf{n}$ as the incoming light direction $\mathbf{l}$. Fresnel reflectance R_F describes the reflected light and depends on the incoming light angle θ_i.

The reflection vector $\mathbf{r}_i$ for the Fresnel equations is defined as

$$\mathbf{r}_i = 2(\mathbf{n} \cdot \mathbf{l})\mathbf{n} - \mathbf{l}. \tag{2.11}$$

Light that is not reflected outward is transmitted, so the proportion of transmitted flux is $1 - R_F$. The proportion for radiance is different because of the differences in projected area and solid angle between the incident and transmitted rays. This relationship between incident and transmitted radiance is

$$L_t = (1 - R_F(\theta_i))\frac{\sin^2\theta_i}{\sin^2\theta_t}L_i. \tag{2.12}$$

In addition to the incident angle θ_i, the Fresnel reflectance R_F and the transmission angle θ_t depend on the *refractive index n* of the substances. The dependence of θ_t on θ_i and the

refractive indices of the two substances is described by Snell's Law which states that

$$n_1 \sin(\theta_i) = n_2 \sin(\theta_t), \tag{2.13}$$

where n_1 is the refractive index of the substance that the light propagates from and n_2 is the refractive index of the substance the light propagates to. Snell's Law gives a different form for the transmitted radiance:

$$L_t = (1 - R_F(\theta_i))\frac{n_2^2}{n_1^2} L_i. \tag{2.14}$$

The direct use of Fresnel equations in rendering is very challenging because of their complexity and other properties. Schlick presented an approximation of Fresnel reflectance, which is based on the charasteristic specular color $R_F(0^\circ)$ of the material. The Schlick approximation is defined as

$$R_F(\theta_i) \approx R_F(0^\circ) + (1 - R_F(0^\circ))(1 - \overline{\cos}\theta_i)^5. \tag{2.15}$$

Substances with different ranges of values for $R_F(0^\circ)$ can be divided into three groups. This division is between insulators (dielectrics), metals (conductors), and semiconductors. Semiconductors are rarely found in rendered scenes so our focus will be on the first two.

Insulators have low values of $R_F(0^\circ)$, typically 0.05 or lower. The *Fresnel effect* refers to the increased reflectance at glancing angles. This effect is clearly visible on insulators because of the low value of reflectance at normal incidence. Insulators usually have colorless reflectance values due to their low variation of optical properties in the visible spectrum.

Metallic substances have high values of $R_F(0^\circ)$ (0.5 and above). Metallic substances can also have optical properties with variation over the visible spectrum. This can be observed in their colored reflectance. Subsurface scattering or transparency is not present with metals as they absorb all transmitted light.

In *external reflection*, incoming light is reflected from an object's external surface. Re-

flectance function $R_F(\theta_i)$ is defined for a given substance and this function depends only on the incoming light direction. The value of $R_F(\theta_i)$ varies over the visible spectrum and for rendering it is treated as an RGB vector.

The function $R_F(\theta_i)$ has two characteristics. First, the $R_F(0^\circ)$ of a given substance is the property of the substance that defines it's specular color. Second, as θ_i increases, the value of $R_F(\theta_i)$ increases until $\theta_i = 90^\circ$ where $R_F(90^\circ)$ is 1 for all frequencies.

Internal reflection occurs as light is traveling inside a transparent object and encounters the surface from inside the object. In this case the refractive indexes are reversed as the light is propagating from the substance with higher refractive index. The Schlick approximation can be used for internal reflection with substituting the transmission angle θ_t for θ_i.

External reflection is the most commonly encountered phenomenon in rendering, but internal reflection can be important to note.

2.4.2 Body Reflectance

Surface reflectance is enough to model light-matter interaction with metals as they absorb all light that is not reflected. For insulators however, body reflectance has to be considered.

Insulators can be *homogenous* or *heterogenous*, depending on the amount of internal optical discontinuities in the substance. Homogenous substances have a low amount of internal discontinuities that scatter light. This means that these substances are transparent. Hmogenous substances, such as clear liquids and glass, can partially absorb light but do not scatter it. Light continues unscattered after it's initial transmission until it hits the surface.

Most insulators contain various discontinuities and this means they are heterogenous. These discontinuities include density variations, structural changes, foreign particles, and other discontinuities which scatter the light inside the substance. In addition, light may be partially or completely absorbed. The light that is not absorbed will be re-emitted through the surface. As previously stated, when we model reflectance with a BRDF, we assume that light will be emitted from the point of entry. This is called *local subsurface scattering*.

The ratio between the light that leaves the surface from the interior and the light transmitted

into the substance is called *scattering albedo*. For insulators, the scattering albedo ρ is visually more prominent than the Fresnel reflectance $R_F(\theta_i)$ as insulators transmit most of the incoming light. As they are results of different physical processes, ρ and $R_F(\theta_i)$ can have different spectral distributions (represented as an RGB vector in rendering).

Lhe local subsurface scattering is often modeled as a Lambertian diffuse term in BRDFs. As such, the directional-hemispherical reflectance of R_{diff} (diffuse term) is set to a constant diffuse color value $\mathbf{c}_{\text{diff}}$. This yields the following diffuse BRDF term:

$$f_{\text{diff}}(\mathbf{l},\mathbf{v}) = \frac{\mathbf{c}_{\text{diff}}}{\pi}. \tag{2.16}$$

There is a tradeoff between surface and body reflectance that depends on the incoming light angle θ_i. Body reflectance will decrease as θ_i increases towards a glancing angle. This can be accounted for with a modification to the diffuse term as

$$f_{\text{diff}}(\mathbf{l},\mathbf{v}) = (1 - R_F(\theta_i))\frac{\rho}{\pi}, \tag{2.17}$$

where R_F is the Fresnel reflectance and ρ is the scattering albedo.

This equation results in an uniform distribution of light and as such does not depend on the view direction $\mathbf{v}$. If we consider the numerous scattering effects happening inside the material, it might seem plausible that the outgoing direction is randomized and has an uniform distribution. However, the discussions on the Helmholtz reciprocity and internal reflection imply that the outgoing distribution is not uniform.

Helmholtz reciprocity implies that as the term varies by incoming direction, it should vary in the outgoing direction also. Additionally, the light goes through Fresnel reflectance as it is leaving the substance. This imposes a directional preference on the outgoing light. There are diffuse terms in literature which address these issues. Nevertheless, the simple Lambertian diffuse term with an uniform distribution remains the most commonly used diffuse term in practice.

3 Microstructure and BRDF models in Rendering

Until now, we have discussed the light-matter interaction with the assumption that the surface is optically flat. Most surfaces have surface structures which affect the way light reflects from them. The surface structure which is modeled in the BRDF is *microscale*. This means that it is smaller than a single pixel but considerably larger than the wavelengths of visible light. In this chapter, we examine how this microstructure is approached in rendering and review different BRDF models that are used in rendering systems. This review is mostly based on Akenine-Moller, Haines, and Hoffman (2008).

3.1 Surface Microgeometry

The *microgeometry* of an object surface is too small to be seen directly. It's effect is expressed as a statistical approximation of light scattering at the object surface. The most important visual effect of the microscale structure in rendering is that there are multiple surface normals in the area covered by a pixel instead of a single one. This distribution is called the *microscale normal distribution*. This distribution of normals causes the light to be reflected in multiple directions.

Increased microscale roughness of the surface results in blurring of reflected environmental detail. Also, the appearance of specular highlights are broader and dimmer because of the spreading of light energy into a larger set of directions.

The microscale normal distribution can be *isotropic* or *anisotropic*. Most commonly, the distribution of microsurface normals is rotationally symmetrical and as such the distribution is isotropic. Anisotropic surface normal distributions result in directional reflections and highlights.

Shadowing is another important effect of microgeometry. This refers to the occlusion of light by microscale surface details. *Masking* is a similar effect which refers to the occlusion of visibility by the microsurface structure.

The light that is occluded by shadowing does not disappear. It is reflected away from the

object or to other microfacets. There might be several microsurface interreflections before light reaches the viewer. In insulators, the interreflections are often not visible due to the attenuation of Fresnel reflectance at each bounce. However in metals, this interreflection is the only source of visible diffuse reflection as they don't exhibit any subsurface scattering.

Microfacet theory is a mathematical analysis of the effects of microscale surface structure on reflectance and various BRDF models used in rendering are based on this theory. In this theory, the microgeometry is modeled as a collection of *microfacets* that are flat Fresnel mirrors on the surface. The distribution of these microfacets is defined by the *normal distribution function* (NDF) of the surface. This function is a probability distribution of the microfacet normals. The NDF is a normalized function that integrates to 1 over the sphere, because the probability of a microfacet pointing somewhere is always 1.

The NDF captures the most important visual effect of spreading the reflected light. Microfacet theory does not model multiple bounces of light or body reflectance so BRDFs that are based on microfacet theory include a diffuse term. Masking and shadowing are modeled, but without a full surface representation, this is done in an ad hoc manner.

As a flat Fresnel mirror, each microfacet reflects light in a single reflected direction. When a ray of light coming from light direction $\mathbf{l}$ meets the surface and is reflected to the direction of the view vector $\mathbf{v}$, only the microfacets that are aligned correctly will participate in the reflection. The participating microfacets are the ones that have their surface normal aligned between $\mathbf{l}$ and $\mathbf{v}$. This vector that is aligned between $\mathbf{l}$ and $\mathbf{v}$ is called the *half vector* $\mathbf{h}$.

The proportion of microfacets aligned with the half vector $\mathbf{h}$ will participate in the reflectance and it is given by the NDF evaluated at the half vector. Reflectance will also depend on the Fresnel reflectance of the microfacets equal to $R_F(\alpha_h)$. Value of α_h is the angle between $\mathbf{l}$ and $\mathbf{h}$ and it is the incoming light angle for participating microfacets.

In BRDFs based on the microfacet theory, masking and shadowing are accounted for by introducing a *geometric attenuation factor* $G(\mathbf{l},\mathbf{v})$. This is a function that represents how much of the incoming light remains after masking and shadowing, and gives a value between 0 and 1.

3.1.1 BRDF Model

Different types of BRDF models usually fall within one of two groups: empirical models or analytical models. Empirical models are designed for specific class of surface types. Analytical models are based on physical theory. In this section, some classic empirical models are introduced first. Next, some approaches for physically-based analytical models are presented.

First empirical model used in rendering was the *Lambertian BRDF* (Dorsey, Rushmeier, and Sillion 2008). This model is still in wide use because of it's computational simplicity. Real-world materials often differ from the model when the view or incident angle approaches a glancing angle.

Lambertian shading equation is named after Lambert's law which defines the outgoing radiance for ideally diffuse surfaces as proportional to $\overline{\cos}\theta_i$. As we saw before, this holds not only to Lambertian surfaces but for irradiance in general.

The Lambertian BRDF gives a constant value across all view directions. This constant value is usually referred to as the *diffuse color* $\mathbf{c}_{\text{diff}}$. The BRDF is defined as

$$f(\mathbf{l}, \mathbf{v}) = \frac{\mathbf{c}_{\text{diff}}}{\pi}. \tag{3.1}$$

The division by π results from the integration of a cosine factor over the hemisphere yielding π. This division is often moved to the irradiance in shader applications, but it is present in the definitions of BRDFs in academic literature.

The first specular model in computer graphics was presented by Phong (1975). Despite it's age, this model still remains in common use. Many newer models can be viewed as variations or improvements on this classic model (Dorsey, Rushmeier, and Sillion 2008).

Model presented by Phong (1975) is given in a slightly different form than the other BRDFs presented here. For the purposes of consistency in presentation, the Phong model presented here is in a more physically plausible and simpler form. This version of the Phong model is defined as

$$f(\mathbf{l},\mathbf{v}) = \frac{\mathbf{c}_{\text{diff}}}{\pi} + \frac{\mathbf{c}_{\text{spec}}\overline{\cos}^{m}\alpha_r}{\pi} \tag{3.2}$$

where the α_r is the angle between the *reflection vector* $\mathbf{r}_i$ and the view vector $\mathbf{v}$. The reflection vector $\mathbf{r}_i$ is the ideal reflection direction presented earlier in Section 2.4.1.

The reflectance in the model is divided to the diffuse and specular elements. Key aspect of the model is the term that raises the cosine of the angle α_r to the power of m. This is commonly referred to as the *Phong reflectance* (Dorsey, Rushmeier, and Sillion 2008) and it defines the specular reflectance spreading in a set of directions around the mirror direction. This can be interpreted as an effect of a slightly rough surface.

Blinn (1977) introduced a half vector based variant of the model, which is often referred to as the *Blinn-Phong* BRDF. The half vector $\mathbf{h}$ represents the halfway direction between view vector $\mathbf{v}$ and light vector $\mathbf{l}$ and it is computed as

$$\mathbf{h} = \frac{\mathbf{l}+\mathbf{v}}{\|\mathbf{l}+\mathbf{v}\|}, \tag{3.3}$$

where the halway vector is divided by it's length to get an unit-length result.

Computation of the half vector and it's dot product with the normal is more convenient for lighting computations (Dorsey, Rushmeier, and Sillion 2008). Blinn also noted that ideal mirror reflections are observed only with the alignment of the normal and half vector so the specular lobe can then be thought of as the result of the probability of microfacet normals oriented with the half vector.

The Blinn-Phong BRDF was originally given in non-normalized form, but in it's normalized energy-conserving form it is defined as

$$f(\mathbf{l},\mathbf{v}) = \frac{\mathbf{c}_{\text{diff}}}{\pi} + \frac{m+8}{8\pi}\mathbf{c}_{\text{spec}}\overline{\cos}^{m}\theta_h, \tag{3.4}$$

where θ_h is the angle between the half vector $\mathbf{h}$ and the surface normal $\mathbf{n}$. The multiplication of the specular term with a value is done to ensure that parameters such as $\mathbf{c}_{\text{spec}}$ are equivalent

to reflectance values. This process is called *BRDF normalization*.

This version of the Blinn-Phong has commonalities to microfacet based analytical BRDFs and we can start interpreting the model using the microfacet theory. The cosine power term can be seen as a normal distribution function with m as the parameter of the NDF. We can also replace the $\mathbf{c}_{\text{spec}}$ with the Fresnel term and thus adding the Fresnel effect. The Blinn-Phong presented here is missing the geometric attenuation factor, which models shadowing and masking in the microstructure. The model above can be interpreted as having the simplest possible visibility factor which always has the value of 1.

Torrance and Sparrow (1967) introduced a reflectance model based on measured data and their approach was later introduced to graphics by Blinn (1977). Revising the Blinn model, Cook and Torrance (1982) later introduced their own model. The Cook-Torrance reflectance model is defined as

$$f(\mathbf{l},\mathbf{v}) = \frac{F(\mathbf{v},\mathbf{h})D(\mathbf{h})G(\mathbf{v},\mathbf{l},\mathbf{h})}{\pi(\mathbf{n}\cdot\mathbf{l})(\mathbf{n}\cdot\mathbf{v})} \tag{3.5}$$

where D is the normal distribution function, F is the Fresnel reflectance, and G is the geometric attenuation factor. Here, the different individual pieces of the model can be mixed to create the desired type of analytical BRDF model (Akenine-Moller, Haines, and Hoffman 2008). Cook and Torrance introduced their model with the π term in the denominator, but this π multiplier is sometimes replaced with 4 in newer models. In this study, a modification of a model proposed by Walter et al. (2007) is used and this model has the 4 term in the denominator.

3.1.2 Normal Distribution Functions

Previously, we introduced Fresnel equations and we will be using the Schlick approximation for the Fresnel reflectance portion of the BRDF model used in this study. Next, we will introduce some common normal distribution functions and geometric attenuation factors including the ones used in this study. Sections 3.1.2 and 3.1.3 are mostly based on Hoffman (2013).

In BRDFs, the statistical distribution of microgeometry surface orientations is defined by the *normal distribution function* $D(\mathbf{h})$. The normal distribution function (NDF) is evaluated at the half vector $\mathbf{h}$ and it describes the concentration of surface points which are oriented in a way that they could reflect light from $\mathbf{l}$ to $\mathbf{v}$. Many different NDFs have been proposed in literature and in this section we review some of the most commonly used NDFs, mainly the *Blinn-Phong NDF*, the *Beckmann distribution*, and the *Trowbridge-Reitz NDF*.

As discussed earlier, the classic Phong shading equation was later modified to better fit the structure of the microfacet BRDF by Blinn (1977). The Blinn-Phong BRDF introduced a normal distribution function that can be defined in it's normalized form as

$$D_p(\mathbf{h}) = \frac{\alpha_p + 2}{2\pi}(\mathbf{n} \cdot \mathbf{h})^{\alpha_p} \tag{3.6}$$

where $\mathbf{n}$ is the surface normal and α_p is the roughness parameter of the NDF. High values of α_p represent smooth surfaces and low values represent rough surfaces.

Later, Cook-Torrance proposed replacing the NDF with a Beckmann distribution. In it's normalized form the Beckmann distribution is defined as

$$D_b(\mathbf{h}) = \frac{1}{\pi\alpha_b^2(\mathbf{n} \cdot \mathbf{h})^4} e^{-\left(\frac{1-(\mathbf{n}\cdot\mathbf{h})^2}{\alpha_b^2(\mathbf{n}\cdot\mathbf{h})^2}\right)}. \tag{3.7}$$

For smooth surfaces, the Beckmann distribution and Blinn-Phong give similar results. The difference between the NDFs becomes more apparent in rough surfaces. The roughness parameter α_b in the Beckmann distribution is equal to the root mean square slope of the microscale surface. This gives a different meaning of microgeometry roughness compared to Blinn-Phong's increased randomness of the microgeometry. This means that the Beckmann distribution is able to represent very rough surfaces, which are less random than the uniform distribution, but more rough as the microgeometry is not as flat.

Blinn (1977) recommended the use of the Trowbridge-Reitz NDF. The NDF was not normalized by Blinn but was later normalized and this form is often referred as the GGX distribution (Walter et al. 2007). The Trowbridge-Reitz NDF is defined as

$$D_{\text{tr}}(\mathbf{h}) = \frac{\alpha_{tr}^2}{\pi\big((\mathbf{n}\cdot\mathbf{h})^2(\alpha_{tr}^2-1)+1\big)^2} \tag{3.8}$$

,

where α_{tr} is the roughness parameter of the Trowbridge-Reitz NDF. 3.8 represents the Trowbridge-Reitz NDF in it's original form as presented by Hoffman (2013).

The Trowbridge-Reitz NDF is capable of producing both the uniform distribution of the Blinn-Phong NDF and the very rough surfaces of the Beckmann distribution.

In the implementation of the proposed technique of this thesis, the Trowbridge-Reitz NDF will be used. This NDF was chosen due to it's more realistic highlight shapes, recommendation in literature, and it's common use in film and game production.

3.1.3 Geometric Attenuation Factors

The *geometric attenuation factor* $G(\mathbf{l},\mathbf{v},\mathbf{n})$ represents the probability of the microsurface facets that are visible from both $\mathbf{l}$ and $\mathbf{v}$. As with the NDF, the geometry function's value is a scalar value between 0 and 1. There are many geometric attenuation factors proposed in graphics literature and in this section we will introduce few common ones for the purposes of this thesis.

Some BRDFs such as the Blinn-Phong model do not define a geometric attenuation factor, which is equivalent to setting the visibility term to 1. This kind of approach implicitly defines the geometric attenuation factor as

$$G_{\text{Implicit}}(\mathbf{l},\mathbf{v},\mathbf{n}) = (\mathbf{n}\cdot\mathbf{l})(\mathbf{n}\cdot\mathbf{v}). \tag{3.9}$$

The implicit geometry function is a plausible approach for heightfield microsurfaces such as the Blinn-Phong NDF. It approaches 0 as $\mathbf{v}$ or $\mathbf{l}$ approaches a glancing angle. The computational efficiency of this geometry function is very good. However, the implicit geometry function goes to 0 too fast for moderate glancing angles, is too dark compared to other geometric attenuation function, and is not affected by surface roughness.

The geometric attenuation function used in the original Cook-Torrance BRDF is based on the geometry function introduced by Torrance and Sparrow. It is defined as

$$G_{\mathrm{CT}}(\mathbf{l},\mathbf{v},\mathbf{h}) = \min\left(1, \frac{2(\mathbf{n}\cdot\mathbf{h})(\mathbf{n}\cdot\mathbf{v})}{(\mathbf{v}\cdot\mathbf{h})}, \frac{2(\mathbf{n}\cdot\mathbf{h})(\mathbf{n}\cdot\mathbf{l})}{(\mathbf{v}\cdot\mathbf{h})}\right). \tag{3.10}$$

This geometric attenuation factor has seen considerable use, but it does have some problems. It is based on an unrealistic microgeometry model and it is not affected by surface roughness.

For increased accuracy, the Smith family of geometric attenuation factors provide a good solution. These functions take into account the surface roughness and are considered to be more accurate than the model presented above. The Smith function was originally designed for the Beckmann distribution, but has been later generalized into a form that can be used with any NDF.

For the purposes of this thesis, a modified Schlick (1994) model was used. This geometric attenuation factor belongs in the Smith family of functions and in it's original form was approximated for a Beckmann distribution. This modified model was developed to better fit the Smith model of Trowbridge-Reitz (Karis 2013). It is defined as

$$G(\mathbf{l},\mathbf{v},\mathbf{h}) = G_1(\mathbf{l})G_1(\mathbf{v}) \tag{3.11}$$

$$G_1(\mathbf{v}) = \frac{\mathbf{n}\cdot\mathbf{v}}{(\mathbf{n}\cdot\mathbf{v})(1-k)+k} \tag{3.12}$$

$$k = \frac{(\alpha+1)^2}{8}. \tag{3.13}$$

As the Trowbridge-Reitz NDF was chosen for this study, this reformulation of the Schlick model was chosen to better fit the Smith model of Trowbridge-Reitz (Karis 2013).

3.2 Representing Materials on Multiple Scales

While there has been considerable research effort to model materials on a single scale, the modeling of multi-scale materials is not as well studied. There has been some development

in the area in recent years. In this section, the most relevant literature on multi-scale material modeling is presented.

An early method for computing large-scale appearance from small-scale geometric structures was introduced by Westin, Arvo, and Torrance (1992). In their method, they provide a representation of the BRDF as a matrix of spherical harmonic coefficients, use a Monte Carlo technique to estimate the coefficients from a geometric optics simulation, and create a small-scale BRDF from microscale scattering events. Their method works for arbitrarily rough geometries.

This method requires the storage of the BRDF as a large matrix of spherical harmonic coefficients that is produced through precomputation. A Monte Carlo estimation of this matrix is then computed and this matrix is used in rendering. Westin, Arvo, and Torrance (1992) display examples of materials such as simple isotropic and anisotropic surfaces, velvet, and woven cloth using Monte Carlo ray tracing.

In their work on accurate and efficient normal map filtering in the frequency domain, Han et al. (2007) show that normal map filtering can be formalized as a spherical convolution of the NDF and the BRDF. They use spherical harmonics for low-frequency specular BRDFs and spherical von Mises-Fisher distributions for high-frequency materials. This is because they acknowledge the impracticality of using spherical harmonics with high-frequency materials due to the large amount of coefficients required. Han et al. (2007) introduce a function called the *effective BRDF*, which is a convolution of the NDF and the BRDF.

Han et al. (2007) also note that an extension of their technique to associate additional spatially variable properties to the normal map. This extensions enables the correct filtering of different materials across wide variety of scales. They display the use of this extension on a complex surface on many different scales and it compares quite favorably with the ground truth approach of supersampling pixels with hundreds of samples per pixel.

Wu, Dorsey, and Rushmeier (2009) introduce the *charasteristic point map* (CPM) as method to represent a hierarchy of view-independent points that preserve the appearance of a detailed model across different scales. They introduce this method to represent objects as simplified mesh hierarchies that are combined with their CPM hierarchies. In rendering the appear-

ance across scales, they use a weighted average of reflectances from charasteristic points. This representation enabled them to compute the macro-scale reflectance from micro-scale geometries with shadowing-and-masking effects.

Wu, Dorsey, and Rushmeier (2011) later extended their method and introduced the first interactive physically-based bi-scale material editing system, which manipulates small-scale geometry and BRDFs to facilitate appearance design at two scales consistently. They introduce a rendering pipeline that converts the edited small-scale details into effective large scale appearance. After taking the small-scale details as an input, the pipeline discretises the geometry into facet samples, computes their directional visibility information, and converts this into a *bidirectional visible normal distribution function* (BVNDF). Combining the BVNDF with the rotated small-scale BRDFs, the pipeline produces the effective BRDF and performs the final large-scale appearance rendering.

This pipeline was implemented with a slider-based interface that allowed manipulation of small-scale geometry and materials. The small-scale geometry could be created either by setting parameters for a procedural model, or by specifying a height-field. The procedural models in their paper include pyramids, grooves, rods and woven threads. It is also possible to vary other properties of the different parts of the small-scale geometry.

Various kinds of materials were introduced in the paper from the procedural models. These include anisotropic metallic appearance using pyramids as the small-scale geometry, unusual 5-way anisotropic reflection using pentagonal pyramids (no existing BRDF model could represent this with physical plausibility), the appearance of velvet using small-scale rods, and a two-hue silk appearance using a woven structure of 2 colors (cannot be achieved using standard BRDF models).

Most recently, Iwasaki, Dobashi, and Nishita (2012) presented a technique for rendering and editing bi-scale materials under all frequency lighting in real time. They were able to achieve real-time performance on the bi-scale materials by representing the BVNDF introduced by Wu, Dorsey, and Rushmeier (2011) with *spherical gaussians* (SG). This enabled them to represent the small-scale reflectance without dense sampling of different directions and the large matrices used for storing these precomputed BVNDF and BRDF values needed by the

previous methods.

The technique introduced by Iwasaki, Dobashi, and Nishita (2012) has similarities to the method proposed by Han et al. (2007), but they extend it in their method to account for shadowing and masking effect by changing the SG lobe amplitudes according to visibility. This results in an accurate representation of the small-scale materials that is able to fit comfortably in GPU memory. They calculate the BVNDF from small-scale geometries similarly as Wu, Dorsey, and Rushmeier (2011), but use the GPU to do this which results in higher performance.

4 Other relevant techniques

To describe the implementation of the technique used in this study, it is necessary to describe some additional concepts and methods. The technique presented in this thesis uses *texture maps* to represent the small-scale structures of the materials. These texture maps are then *sampled* accordingly to calculate final reflection values. In this chapter, the relevant methods and concepts on *texturing* and *sampling* are introduced. Section 4.1 on texturing is based on Akenine-Moller, Haines, and Hoffman (2008). Sections 4.2 and 4.2.1 on sampling are based on Glassner (1994).

4.1 Texturing

Texturing is a process in which a given surface appearance is modified at each location using images, functions, or other data sources. It is a technique for modeling various surface properties and for modifying the parameter values in shading.

In the texturing process, a location in object space is given to a *projector* function to obtain a set of numbers called the *parameter-space values*. This process of combining a location in space to accessing the texture called *texture mapping*. The parameter-space values are then used in *corresponder* functions to transform them into the texture space and retrieve the correct value from the texture, which can be used to modify surface properties.

The most common property to be modified by an image texture is the diffuse color $\mathbf{c}_{\text{diff}}$ and the texture used for this purpose is called *diffuse color map*. Specular color $\mathbf{c}_{\text{spec}}$ is another commonly textured attribute and this texture is called the *specular color map*. *Gloss maps* refer to the textures that modify the surface roughness parameter α.

The surface detail can be traditionally classified into three scales. *Macrogeometry* describe detail that cover multiple pixels and is often represented by geometric primitives such as triangles. *Microgeometry* are features that are substantially smaller than a pixel and are encapsulated in the shading model. *Mesogeometry* describe the detail between these two scales. These details are too expensive to render using polygons and too large to implement

in the shading. This is a scale in which a family of methods called *bump mapping techniques* are used.

Bump mapping adjusts the shading parameters in a way that small perturbations from the base geometry appear while the actual base geometry stays flat. The bump mapping techniques differ mainly in the way they represent these details. In modern hardware, the preferred method of bump mapping is to store a *normal map*. A normal map contains the perturbed normals directly encoded as (x, y, z) with individual values mapped to $[-1, 1]$. Usually, this perturbed normal is retrived in *tangent space* (relative to the surface itself) and then used in shading the surface.

Normal mapping will only alter the normal of the surface and will not show occlusion of other details. A method called *parallax mapping* was introduced to add approximated occlusion of surface details on the meso-scale. In parallax mapping, the bumps are stored in a displacement texture. As the surface point is viewed, the value of the displacement is retrieved at that point. This displacement value is then used to alter the texture coordinates and retrieve values from different location of the texture. The amount of coordinate change is done according to the retrieved displacement and the viewing angle. The displacement can be stored in a separate texture or a single channel in other textures. These values are scaled and biased before being used. Scale is used to determine how much the value is to extend the above or below the surface. Bias tells the height at which no change takes place.

The new parallax-adjusted texture position $\mathbf{p}_{\text{adj}}$ can be computed as

$$\mathbf{p}_{\text{adj}} = \mathbf{p} + \frac{h \cdot \mathbf{v}_{xy}}{v_z} \tag{4.1}$$

where $\mathbf{p}$ is the location of the original texture coordinates, h is the adjusted displacement value and v_z and $\mathbf{v}_{\text{xy}}$ are components of a normalized view vector $\mathbf{v}$.

This method was further developed by Welsh (2004) with the addition of offset limiting. Offset limiting is used to limit the amount of change in texture coordinates to the displacement value that was originally retrieved. Parallax mapping with offset limiting is in wide use in games and it is now considered the practical standard for bump mapping.

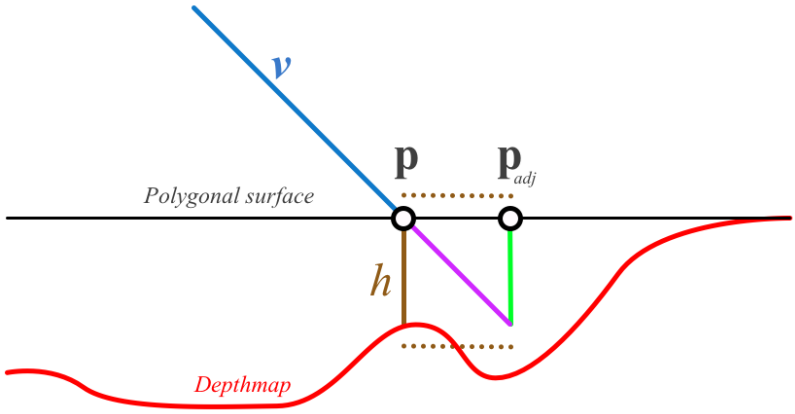

Figure 1. Representation of parallax mapping with a depth map, where **v** is the view vector, **p** is location of the original texture coordinate, h is the height at the original texture coordinates, and $\mathbf{p}_{\text{adj}}$ is parallax-adjusted texture position.

Parallax mapping provides an approximation of the displacement, but assumes that the displacement between pixels are somewhat similar. Furthermore, the bias setting affects how the heightfield is displayed.

Another group of techniques emerged to accurately find where the view vector first intersects the displacement map. These techniques have the same approach of ray tracing the path of the view vector. *Parallax occlusion mapping* (POM), *relief mapping*, and *steep parallax mapping* are the names given by different researchers to the variations of this technique.

The idea of these algorithms is to test a number of texture samples along the view vector. The amount of tested texture samples increases as the view angle increases in order to not miss the closest intersection point. These tested texture samples are processed to determine if the view vector is above or below the displacement. At the first occurence of the sample found under the displacement, that sample and the previous sample (above the displacement) are used to find the intersection. This texture location is then used to shade the surface. The technique can also be used to have the surface cast shadows onto itself by applying the same process to determining light visibility.

The displacement is considered as a depthfield in all of the approaches with the plane of

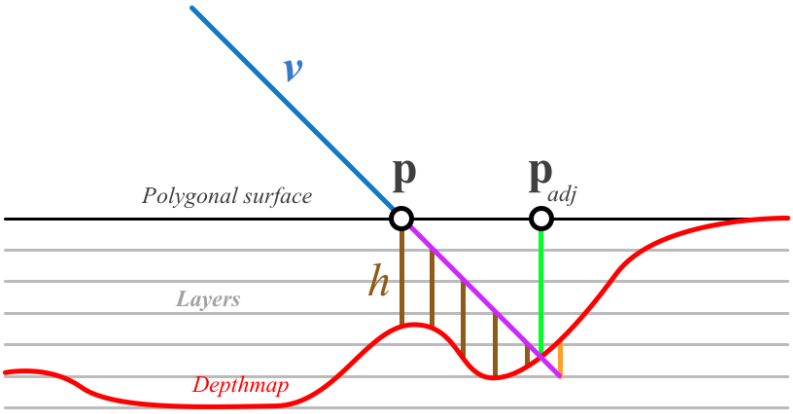

Figure 2. Representation of the basic approach of parallax occlusion mapping, relief mapping, and steep parallax mapping. The path of the view vector is sampled at the layer intersections and after the first occurence of a sample below the depthfield, the intersection point is calculated and adjusted texture coordinates are returned.

the polygon representing the upper limit. However, the root-finding problem of finding the actual intersection point between the two texture samples is approached differently with these algorithms. This refers to the method used to find the intersection between the last sample above, and first sample below the depthfield. The methods used are a binary search, single step of the secant method, and iteration using the secant method. There are also various additional algorithms attempting to improve on the efficiency of these methods that mainly focus on skipping intervening empty spaces instead of sampling the depthfield at a regular interval. Examples of these methods include *cone-stepping* and *quad-tree relief mapping*.

4.2 Sampling

As our approach in finding the average reflectance over a microstructure texture is based on sampling the texture multiple times, we should look closer at *sampling* and *reconstruction*. The general approach of *uniform* and *non-uniform* sampling will be covered as well as their common implementation techniques and problems.

Sampling is a process where a continuous signal is turned into a discrete signal by taking

sample values. Reconstruction is the process of turning a discrete sampled signal back to a continuous signal. Computer graphics is usually dealing with the discrete versions of aperiodic, continuous signals as most of the performed operations are too complex for computer graphics when they are attempted analytically.

If the signal is sampled at equal intervals, the sampling is uniform. *Uniform sampling* is an approach where sampling patterns can be represented as a regular *lattice*. These lattices can be in different forms and in our 2D setting, but they are usually grids of some regular form. Uniform sampling is an attractive solution if we can assume that the signal is bandlimited.

Uniform Sampling Theorem states that a bandlimited signal $f(t)$ that has a cutoff frequency ω_F can be reconstructed from its samples $f(nT_0)$ if $2\pi/T_0 \geq 2\omega_F$. The sampling rate T_0 is called the *Nyquist rate* and the ω_F is called the *Nyquist frequency* of the signal. This effectively means that in order to perfectly reconstruct a signal, we must sample $f(t)$ at least twice as often as the highest frequency in $f(t)$.

However, the assumption of a bandlimited signal is unlikely to be true in a computer graphics setting. Also, a limited amount of samples is prone to errors in uniform sampling. These errors are known as *aliasing*. For example, when an image is sampled with a regular pattern of a too low sampling rate, *aliasing structures* can appear that are often visible patterns.

In nonuniform sampling however, we can use a variable sampling density and trade structured aliasing for noise. Nonuniform sampling refers to all techniques that produce a sampling pattern that is not periodic. Two types of nonuniform sampling are *patterned* and *random* sampling. Patterned nonuniform sampling is often used when the sampling pattern is known but nonuniform in its nature.

A common approach to nonuniform sampling are the various *stochastic* techniques. These are techniques that are based on random, or *quasi-random*, sampling. Stochastic techniques often turn structured aliasing for high-frequency noise. This means that the reconstructed signal is still "wrong", but noise is generally preferred over structured aliasing in images. The human visual system is surprisingly adept in ignoring noise while good at spotting structured aliasing artefacts.

4.2.1 Sampling techniques

Uniform sampling is based on different patterns that can be defined in respect to *lattices*. The most commonly used lattice is the *rectangular lattice*, that is formed by two perpendicular vectors. It is used frequently in image sampling by thinking of the frame buffer as a group of *cells* where each holds $m \times n$ pixels. Other lattices more rarely used in computer graphics include the *hexagonal*, *triangular*, and *diamond* lattices.

Poisson sampling is the simplest random sampling technique. It is effectively a group of random samples that have no relationship to each other. They are generated in R^n by picking n uniformly distributed random numbers and using them as the location of the sample.

N-rooks sampling is a technique that can be used for sampling a signal on a $N \times N$ grid. In this technique, a sample is taken in each row and column of that grid. N-rooks pattern is constructed on the grid by initially placing the samples along the main diagonal and then randomly shuffling the columns.

Jittered sampling is based on randomly perturbing samples on a sampling pattern. This means that a random displacement is added on all samples in their respective domains. These sample domains are defined by the sampling pattern, for example as an uniform rectangular grid, or an N-rooks pattern.

Poisson-disk distribution is a random pattern with the requirement that no two samples are closer than distance r_p. This requirement is called the *Poisson-disk criterion* and it uses the idea of surrounding the random samples with disks that do not overlap but are as close to each other as possible. As generating this pattern is inefficient, the precomputation of the pattern is common.

5 Technique

The technique presented and examined in this thesis applies the different techniques of *texture mapping* to modeling of complex small-scale structures. This *texture mapping of small-scale surface features* is an interesting approach for several reasons. As stated earlier, the various texture mapping techniques are able to model of complex structures on surfaces. Also, parallax mapping techniques enable us to model the masking and shadowing effect needed in physically plausible materials.

Shadowing and masking effect is included in physically based analytical BRDF models by using the normal distribution function and the geometric attenuation factor. These models work well for a wide range of common materials. However, they lack the possibility of customizing small-scale structure (beyond the scalar roughness value) as they are statistical approximations of the surface's structure. Meso-scale texture mapping, that is commonly used in rendering, is applied on a larger scale and as such cannot be used to model the small-scale features of materials.

Using small-scale texture maps in addition to the meso-scale texture maps and BRDFs, we gain a new degree of freedom to model our materials. The idea is to add another set of texture maps between the meso-scale texture maps and BRDFs. These texture maps are used to define the small-scale features of materials. For example, the small-scale features can include the modeling of weaves in different fabrics or the anisotropic grooves in brushed metals.

Small-scale texture mapping enables us to add considerable complexity to the small-scale geometry and material features. The small-scale geometry is modeled with normal maps and displacement maps, but other surface features present in meso-scale texture mapping can be added. These possible additions include albedo, metalness, roughness, and ambient occlusion maps.

In this study, albedo, normal, and displacement maps were used in modeling the small-scale features of the material. The displacement map stores the depth values of the surface and was used as an individual texture. This texture can also be used to calculate surface normals.

For this study, a separate normal map texture was used in addition to a displacement map. An albedo colour map was used to distinguish the correct response of the calculations.

These maps represent the small-scale features of the material and are sampled multiple times to get the average amount of light reflected. In a single sample, the displacement map is used to determine the correct texture coordinates for the sample. At this point, the masking effect has been achieved as the parallax map calculation has determined the actual visible texture coordinates. After obtaining the coordinates, the amount of light reflected is determined using the other texture maps with the chosen BRDF for that particular sample. Here, the possible shadowing effect is also calculated for the sample by determining the light visibility. When all the samples are calculated, the sample reflection values are then averaged to get the final output.

This process is done in the fragment shader individually for each pixel. The small-scale texture maps are essentially treated as a repeating texture that always covers the area of the pixel. This represents a slight problem as the small-scale features can then be considered as varying in scale in regards to the pixel coverage. However, as we constrain the small-scale features to repeating patterns that are always displayed as the surface reflection features, this approach gives an adequate approximation for the use of this master's thesis.

5.1 Implementation

A detailed description of the implemented *pixel shader* (also known as the *fragment shader*) is given in this section. This pixel shader is evaluated for every pixel that the material covers. The implementation is given in GLSL code format. There are some simplifications in the code presented here to improve the readability and presentation. For the full working code, please refer to the shader implementation provided in the code repository (Kinnunen 2018).

5.1.1 Sampling

The entry point of the pixel shader is the main function. We start by presenting the simplest case of normal mapping with a single point light and describe the modifications to implement parallax mapping later in the section.

```
void main()
{
  vec4 result;
  for(int sx = 0; sx < SAMPLES_X; sx++)
  {
    for (int sy = 0; sy < SAMPLES_Y; sy++)
    {
      float x = (sx + 0.5) / SAMPLES_X;
      float y = (sy + 0.5) / SAMPLES_Y;
      vec2 textureCoordinates = vec2(x, y);
      // Normal mapping. Parallax Mapping described later.
      vec2 finalTexCoord = textureCoordinates;
      result += CookTorrance(viewDir, lightDir);
    }
  }
  out = result / (SAMPLES_X * SAMPLES_Y);
}
```

Here we can observe the sampling procedure described earlier with full sampling of the texture map. Samples are taken for every pixel of the texture map and for every sample we do the calculation of the Cook-Torrance BRDF. These lighting calculations are summed and and averaged, which results in the outputted color value for the pixel. When we consider the addition of parallax mapping later, we store the texture coordinates from those functions in the `finalTexCoord` variable on the corresponding row of the main function.

5.1.2 Shading

```
vec4 CookTorrance(vec3 viewDir, vec3 lightDir) {
  // Specular.
  float NDF = TrowbridgeReitzNDF(normal, halfway, roughness);
  float G = GeometrySmith(normal, viewDir, lightDir, roughness);
  vec3 F = FresnelSchlick(max(dot(halfway, viewDir), 0.0), F0);
  vec3 nom = NDF * G * F;
  float denom = 4 * max(dot(normal, viewDir), 0.0) * max(dot(normal, ↩
     lightDir), 0.0);
  vec3 spec = nom / denom;
```

```
  //Diffuse.
  float NdotL = max(dot(normal, lightDir), 0.0);
  vec3 refracted = (vec3(1.0) - F) * (1 - metalness);
  vec3 Lo = ((refraction * albedo / PI) + spec) * radiance * NdotL;
  return vec4(Lo, 1.0);
}
```

Lighting calculation for individual samples is presented in the function above. First, we obtain the proportion of reflecting microstructure facets using the normal vector, halfway vector between view and light direction, and the roughness parameter given by the material properties. This value is stored in the `NDF` variable. Next, the geometric attenuation factor is calculated. We will need both the view direction and light direction explicitly to calculate this function. Lastly, the Fresnel reflectance is calculated using the Schlick approximation. For the first parameter of the Fresnel function, we use the clamped dot product of the half vector and the view vector. The second parameter `F0` is the refractive index of the given material, which is needed by the Schlick approximation.

After obtaining the NDF value, geometric attenuation factor and, Fresnel reflectance, we are able to input these values in the Cook-Torrance BRDF nominator and calculate the denominator from the clamped dot products. Finally, this gives us the specular reflectance that will be used later.

Next, we calculate the diffuse term of the reflectance. For this, we first approximate the proportion of light refracted into the material. This is done by subtracting the proportion of the light that is directly reflected off the surface. Then, we use the material's metalness parameter to take into account the proportion that is absorbed by the material. Combining these calculations, we use the Lambertian reflectance with the material's albedo property (the "color" of the material), that gives us the amount of diffusely reflected light.

The specular and diffuse reflection are then multiplied by the amount of light arriving at the surface. This gives us the final amount and color of the light that is reflected from this sample.

```
vec3 FresnelSchlick(float cosTheta, vec3 F0)
```

```
{
  return F0 + (1.0 - F0) * pow(1.0 - cosTheta, 5.0);
}
float TrowbridgeReitzNDF(vec3 N, vec3 H, float roughness)
{
  float a = roughness*roughness;
  float a2 = a*a;
  float NdotH = max(dot(N, H), 0.0);
  float NdotH2 = NdotH*NdotH;
  float nom = a2;
  float denom = (NdotH2 * (a2 - 1.0) + 1.0);
  denom = PI * denom * denom;
  return nom / denom;
}
```

Above are code examples of the functions that define the Schlick approximation and Trowbrigde-Reitz NDF. They are straight-forward implementations of Equation 2.15 and Equation 3.8. In the Schlick approximation, we use the material's refractive index `F0` to calculate the approximation of the Fresnel reflection and `cosTheta` which is equal to the clamped dot product of the normalized view vector and the halfway vector. In the NDF implementation, we use the normal vector, halfway vector, and the material's microstructure roughness value to approximate the proportion of the contributing microstructure facets.

```
float GeometrySchlickTR(float NdotV, float roughness)
{
  float r = (roughness + 1.0);
  float k = (r*r) / 8.0;
  float nom = NdotV;
  float denom = NdotV * (1.0 - k) + k;
  return nom / denom;
}
float GeometrySmith(vec3 N, vec3 V, vec3 L, float roughness)
{
  float NdotV = max(dot(N, V), 0.0);
  float NdotL = max(dot(N, L), 0.0);
  float visibilityV = GeometrySchlickTR(NdotV, roughness);
```

```
  float visibilityL  = GeometrySchlickTR(NdotL, roughness);
  return visibilityV * visibilityL;
}
```

The geometric attenuation factor is calculated by the functions above as defined in Equation 3.11. The first point of entry is the `GeometrySmith` function that takes as it's parameters the tangent space normalized view, light and normal vectors in addition to the material's roughness value. The dot product of the normal is then calculated for both the view vector and the light vector. These dot products are used to calculate the visiblity with both the light and view directions.

The visibility of both directions is returned by the `GeometrySchlickTR` function. After attaining the visibility of both the view direction and the light direction, multiplying them gives the final proportion of microfacets that are not occluded or shadowed.

5.1.3 Parallax Mapping

Parallax mapping techniques aim to shift the texture coordinates based on the displacement map and the view direction. Applying these methods in the technique relies on replacing the `finalTexCoords` assignment in the code of the main function with one of the function calls that are presented next.

```
vec2 ParallaxMapping(vec2 texCoords, vec3 viewDir)
{
  float height =  texture(texture_displacement1, texCoords).r;
  vec2 p = viewDir.xy / viewDir.z * (height * height_scale);
  vec2 final = texCoords - p;
  return final;
}
```

The original parallax mapping approach performs a single step approximation of the correct texture coordinates using the original texture coordinates, displacement map value, and normalized tangent space view vector. The change in texture coordinates is calculated as defined in Equation 4.1. This value is stored in the variable `p`. The change in texture coordinates is then subtracted from the original texture coordinates and the final texture coordinates are

then returned by the function.

Parallax occlusion mapping was the second approach that was used to calculate the view dependent change in texture coordinates. This method calculates the new texture coordinates by dividing the depth of the displacement range into layers and finding the first layer below the displacement map along the view vector. After this, we linearly interpolate between the depth values of the last layer above and first layer below the displacement.

```
vec2 ParallaxOcclusionMapping(vec2 texCoords, vec3 viewDir, float ↩
    numLayers)
{
  float layerDepth = 1.0 / numLayers;
  float currentLayerDepth = 0.0;
  vec2 P = viewDir.xy / viewDir.z * height_scale;
  vec2 deltaTexCoords = P / numLayers;
  vec2  currentTexCoords     = texCoords;
  float currentDepthMapValue = texture(texture_displacement1, ↩
      currentTexCoords).r;
  while(currentLayerDepth < currentDepthMapValue)
  {
    currentTexCoords -= deltaTexCoords;
    currentDepthMapValue = texture(texture_displacement1, ↩
        currentTexCoords).r;
    currentLayerDepth += layerDepth;
  }
  vec2 prevTexCoords = currentTexCoords + deltaTexCoords;
  float afterDepth  = currentDepthMapValue - currentLayerDepth;
  float beforeDepth = texture(texture_displacement1, prevTexCoords).r -↩
      currentLayerDepth + layerDepth;
  float weight = afterDepth / (afterDepth - beforeDepth);
  vec2 finalTexCoords = prevTexCoords * weight + currentTexCoords * ↩
      (1.0 - weight);
  return finalTexCoords;
}
```

The first step is to calculate the change in texture coordinates per layer. This is stored in the

`deltaTexCoords` variable. After that, we iterate through the layers along the view vector until the first occurence where the layer's depth value is below the displacement map. Next, the depth values of the layers before and after the intersection are linearly interpolated and stored in the `weight` variable. Finally, this weight is used to calculate the approximation of the new texture coordinates and returned by the function.

The number of layers can vary according to the view direction so that a more accurate calculation can be achieved at grazing angles. The addition of shadows for parallax mapping and parallax occlusion mapping is done by applying the same respective calculations with light vectors instead of the view vectors to determine the visibility of the light. For a complete representation of the implemented code please refer to the provided code repository (Kinnunen 2018). In this code the shadowing for both techniques is implemented and used for the small-scale shadowing. In the actual implementation, the number of layers are also calculated based on the view and light vectors.

6 Results

In this section, the results of the software implementation is discussed. This includes the evaluation of the visual outputs and the performance of the proposed technique. The main focus of the thesis was to determine the applicability the proposed idea. In this section, the findings from implementing these ideas are presented.

In order to test the texture mapping of small-scale surface features, a real-time rendering system was implemented using C++ and OpenGL. The custom renderer application uses common open-source libraries (GLFW, GLEW, and Assimp) to do tasks such as windowing, OpenGL extension handling and model loading. The source code for the renderer and shaders are provided in the code repository (Kinnunen 2018).

The main software contribution of this thesis is the shader code that implements different small-scale texture mapping techniques. This GLSL shader calculates per-pixel shading of modeled small-scale structures as described in the previous chapter. The shader supports the use of normal mapping, two parallax mapping techniques, two sampling schemes, and two BRDFs.

Parallax mapping techniques supported by the shader are the original parallax mapping and parallax occlusion mapping. Shadowing effect is available for both setups. These techniques are used to determine the masking and shadowing effects of the small-scale texture samples in the pixel shader. Sampling schemes that are supported include full sampling of the small-scale texture maps and jittered sampling. The BRDFs supported by the shader are Blinn-Phong and Cook-Torrance.

In the performance evaluation, different test setups are examined with varying models, texture mapping techniques, textures, displacement depths and other shader parameter setups. As a performance baseline, performance measurements of normal mapped, parallax mapped and a parallax occlusion mapped meso-scale materials are presented. These baseline setups all have the same textures and same shader parameters for each measurement.

6.1 Visual results

A number of different rendered results are presented in this section. The technique proposed in this thesis is capable of producing large-scale appearance from small-scale details for both isotropic and anisotropic cases. The effect of the small-scale structure is presented on a single 3D object such as the Stanford bunny, a cylinder, or a disk. A rendering of the small-scale surface structure is displayed with the presented model that shows the repeating surface structure on a rectangular plane. The materials imitated in this section are silk, brushed metal, and a colored grooved surface with a flat bottom section. Finally, a visual comparison of different techniques is displayed with a multicolored material consisting of small-scale pyramids.

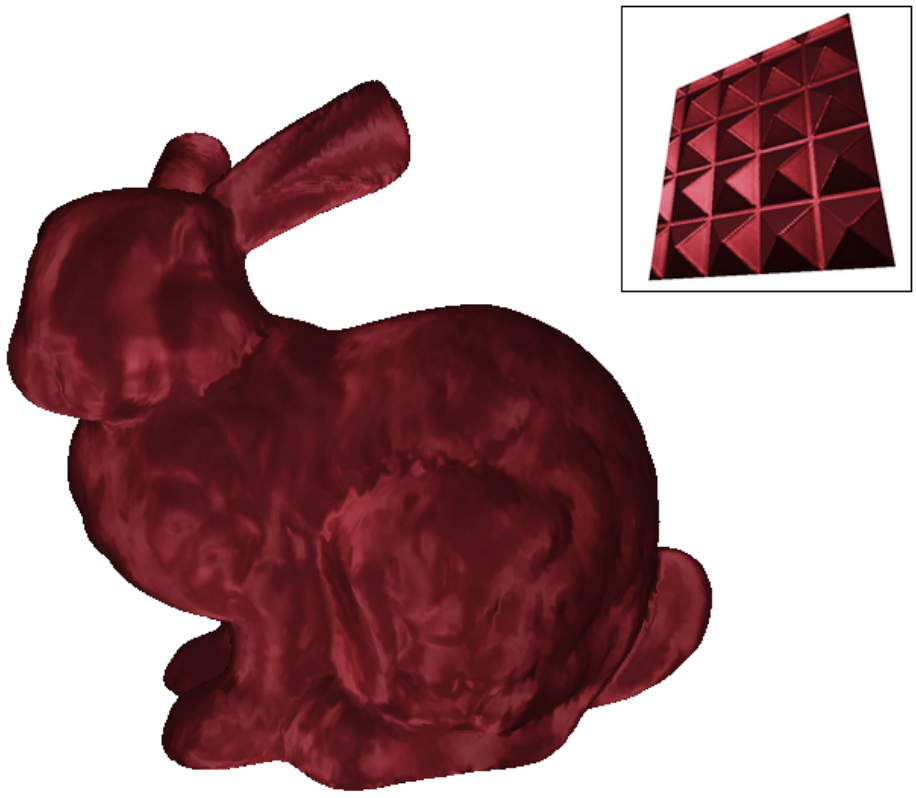

Figure 3. Stanford bunny with a small-scale material imitating silk.

First, Figure 3 shows a Stanford bunny with a silk-like material. Silk is a woven fabric with a high reflectance metallic appearance. The highly reflective woven structure creates an

anisotropic reflection as the individual fibers are arranged in a perpendicular fashion. This woven effect was imitated with a pyramid microstructure, small-scale parallax occlusion mapping, and metallic settings for a Cook-Torrance shader.

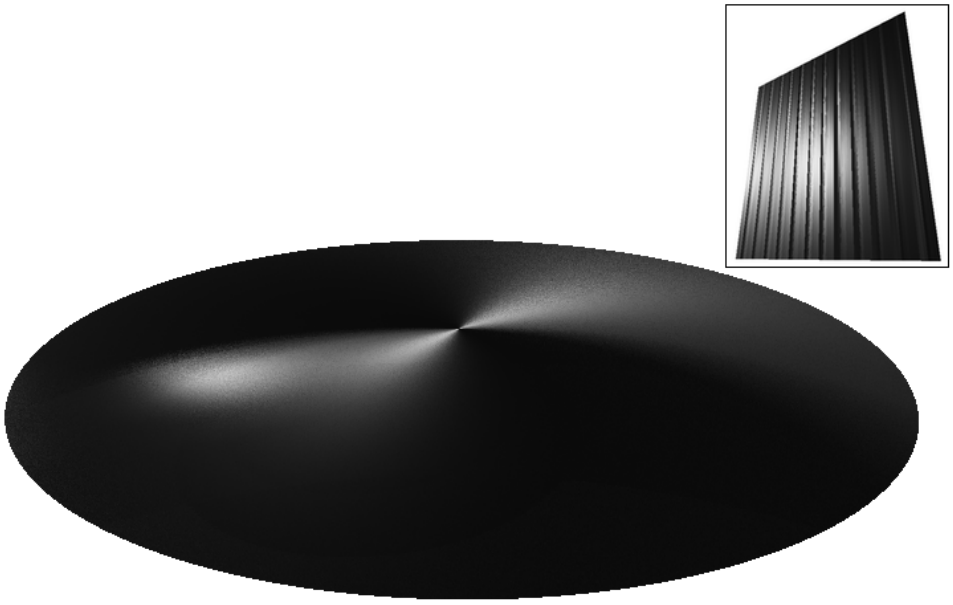

Figure 4. A disk with a small-scale material of a grooved surface imitating a brushed metal appearance.

Next, a highly anisotropic surface reflection is presented which is common for brushed metal. The circular anisotropic reflection shown in Figure 4 is often found on metallic disks or vinyl records. The effect was achieved with a grooved small-scale surface where the grooves run parallel to each other. These parallel grooves, which are shown in the upper right, run on the disk as a circular path around the center point. The settings used for the Cook-Torrance shading are set as highly metallic and the small-scale surface structure technique used was parallax occlusion mapping with no shadows enabled.

For rough surfaces, the technique offers interesting effects. Figure 5 shows a rough flat-bottomed inverted pyramid small-scale geometry with multicolored surfaces. This can be used for creating a matted fur-like effect where the reflectance changes as the viewing angle approaches a grazing angle. This gives an appearance of a light coat vertical green and

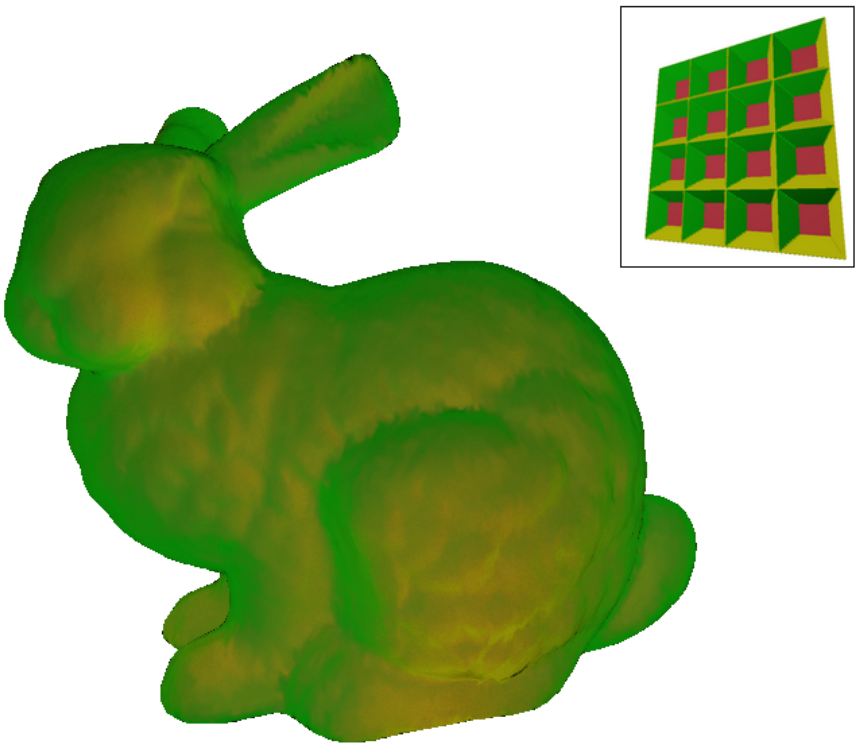

Figure 5. Rough surface with a small-scale structure of inverted pyramids with a flat colored bottom.

yellow strands on top of a red surface.

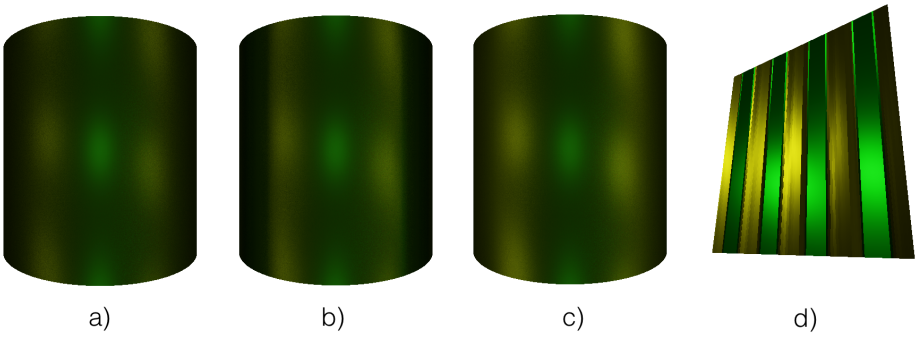

Figure 6. Comparison of different small-scale texture mapping techniques. Small-scale normal mapping (a), parallax mapping (b), and parallax occlusion mapping (c) are displayed on a cylinder. Last image (d) shows a rendering of the small-scale structure.

Figure 6 shows a comparison between different techniques of small-scale normal mapping, parallax mapping, and parallax occlusion mapping. The small-scale structure used in this comparison is a grooved surface with a flat surface between the grooves. The flat surface is colored green and the parallel grooves are colored yellow to display which parts of the small-scale surface are contributing in the reflection. As seen in the different rendered cylinder models, the yellow reflection of the grooves is more prominent on the sides of the cylinder, whereas the flat portion of the small-scale surface shows more prominently in the center.

This comparison shows the visual difference between the techniques. The reflection in the normal mapped version of the cylinder (a) shows a dimmer reflection as there is no masking effect in place. The technique calculates all of the texture map samples with similar weighting because of this. Parallax mapping (b) shows an improvement in this regard. As an approximated masking effect is present, the yellow reflection of the grooves is more pronounced. The comparison also shows the problem parallax mapping has when approaching grazing angles. The one pass approximation in the technique has a distorting effect at grazing angles and this is shown with the darkened edges of the cylinder. Parallax occlusion mapping (c) is the most accurate approximation of these techniques and does not show this warping at grazing angles. The technique gives a smooth falloff when approaching the grazing angles with prominent reflection from the grooves due to the more accurate masking effect.

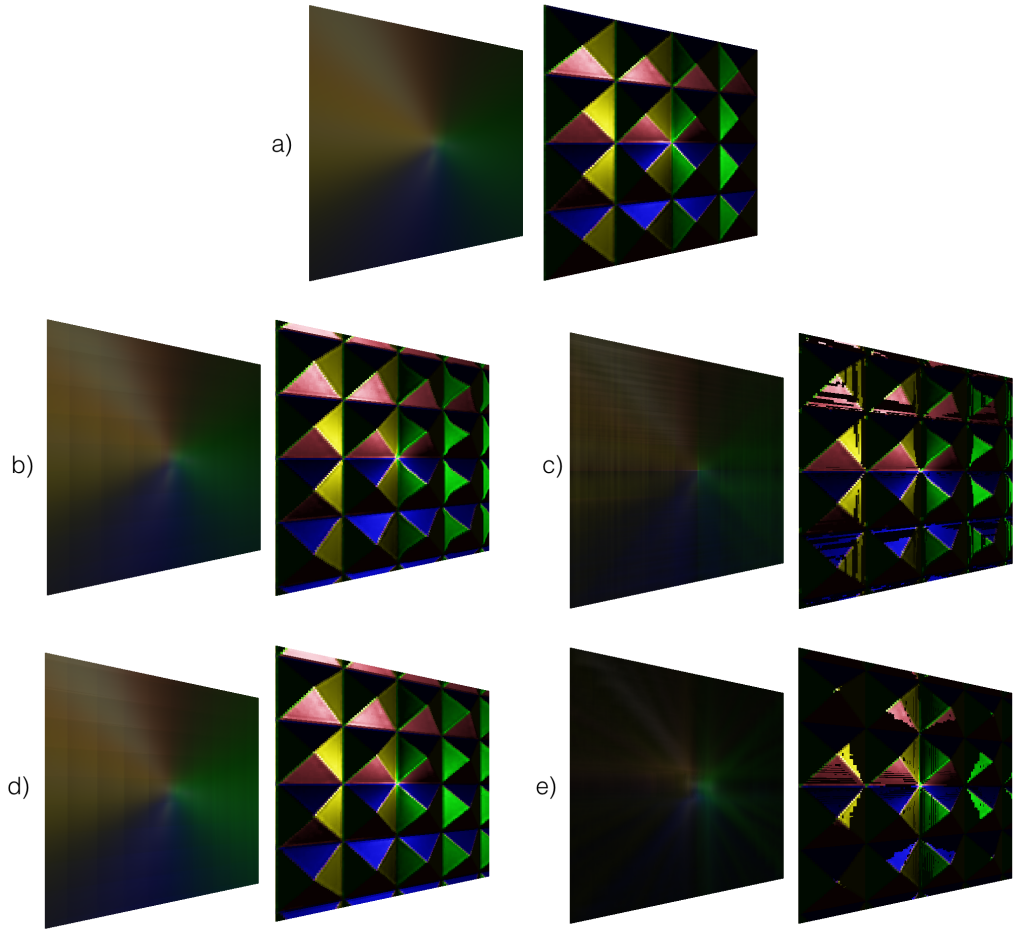

Figure 7. Small-scale texturing techniques with and without shadows. The planes are grouped by technique with the first plane applying the small-scale technique to calculate large-scale appearance and the second showing the surface structure. Techniques presented are normal mapping (a), parallax mapping (b), parallax mapping with shadows (c), parallax occlusion mapping (d), and parallax occlusion mapping with shadows (e).

A more comprehensive comparison of the techniques with and without shadowing is presented in Figure 7 with the corresponding small-scale structure renderings. In this comparison, we can also see the problematic artefacts visible in some of the techniques. The techniques are rendered using full sampling of the 32 by 32 pixel textures. The comparison shows normal mapping (a), parallax mapping (b), parallax mapping with shadows (c), parallax occlusion mapping (d), and parallax occlusion mapping with shadows (e).

Normal mapping approach does not show any particular artefacts and the microstructural effect is visible in the image even if dimmed compared to more pronounced effects in parallax mapping techniques. In the parallax mapping technique, the previously discussed warping effect is slightly visible on the left hand side green triangles. The parallax mapping effect also already shows the slight rectangular artefacts in rendering of the small-scale texture technique. These artefacts become considerably more visible in the shadowed parallax mapping approach. Here, the approximation of the shadows are very inaccurate. This can be seen most visibly in the top left pyramid of the small-scale structure visualization. The technique approximates a ringing shadow artefacts that result from incorrect approximation.

In the parallax occlusion case, the slight rectangular artefacts are also visible. The reflections are again more prominent than in the normal mapping and smoother than in the parallax mapping case. With the addition of shadows the artefacts become more prominent but the shadowing is considerably more accurate than with parallax mapping. As before, the parallax occlusion mapping gives the most accurate results of these techniques and provides both masking and shadowing effects.

6.2 Performace

The runtime performance of the implementation is examined in this section. For this purpose, frame times of different configurations of the implementation are documented and analyzed with data gathered by running the software. These configurations introduce the major factors that contribute to the frame times by varying different test parameters. Different 3D objects, mapping techniques, sample rates, and textures are implemented in a controlled environment as to provide an overview of their effect on performance.

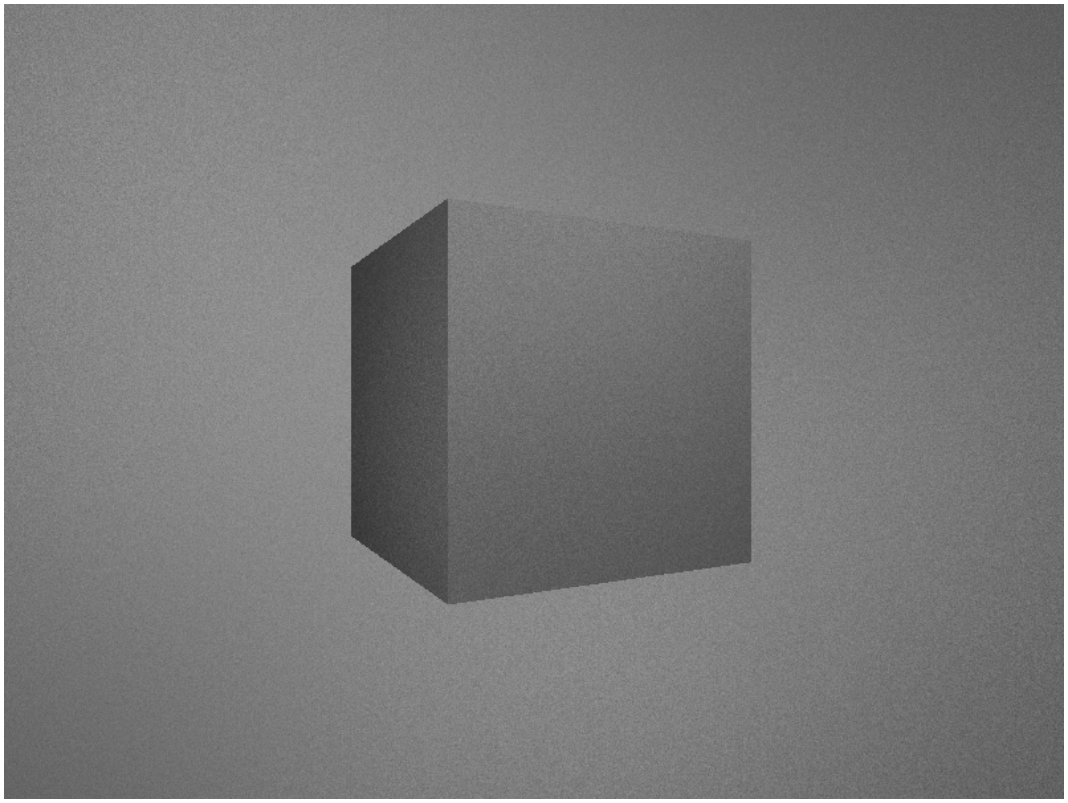

Figure 8. Representation of the controlled test environment, which consists of a central rotating 3D object (the cube) and a stationary background. Both use the same shader confiquration ensuring full pixel coverage for the shader in question.

The controlled environment used in this study is a scene that runs in the custom software renderer developed by the author. The scene consists of a central rotating 3D object, two point lights, and a stationary background quad, that is used to fill the remaining screen space with the examined shader. This ensures that every pixel rendered in the scene is using the shader in question and confirms that the different runs with varied parameters are comparable. The data files from the experiments are available in the application repository and provide a more detailed information of these specific experiment runs (Kinnunen 2018).

All measurements were done using a NVIDIA GeForce GTX 765M GPU, 800 x 600 window resolution, and vertical synchronization turned off. As some shader setups have varying performance times depending on viewing angles, a slowly rotating object was used. The frame times from the performance tests are reported in milliseconds and are averaged over a

test run of 30 seconds.

6.2.1 3D objects

First, we examine different 3D objects and their effect on the performance. Also, a baseline for our investigation is set by varying the techniques used by the shader and include the basic methods commonly used in production systems. These techniques include basic normal, parallax and parallax occlusion mapping as meso-scale appearance techniques. We compare these to the proposed small-scale surface modeling techniques presented in this study. These include small-scale normal mapping, small-scale parallax mapping and small-scale parallax occlusion mapping.

During the investigation, a curious implementation detail arose. In trying to find a comparable baseline for parallax mapping and parallax occlusion mapping, it was noted that these heavier operations of parallax mapping were in fact aprroximately 2 times faster than a simple normal mapping implementation generally used. Careful examination of the subject revealed that a single conditional in the parallax mapping function produced this speed-up. This conditional was always false for these runs and thus the code contained was never executed. The cause of the speed-up might be due to heavy optimization done to the compiled shaders or a bordercase in the graphics card drivers.

To establish a more suitable baseline for the comparison, this false conditional was implemented exactly the same way in the basic normal mapping case, which resulted in a similar speed-up and comparable results. Finding the root cause of this curious occurence with this specific test setup was out of the scope of this thesis and this faster normal mapping case is presented as the baseline for other techniques. The code regarding this issue is provided in the application repository (Kinnunen 2018). Both of the normal mapping cases and their runtime results are documented in Table 1 and Table 2.

In this experiment, three different 3D objects were used with the techniques reported above. The models in question were a simple rectangular quad with two triangles, a cube with 12 triangles, and the Stanford bunny which is a usual choise for benchmarking in graphics reasearch since it's development in 1994. The bunny consists of 69,451 triangles and even

Meso-scale	Quad	Cube	Bunny
Normal Mapping	0.64	0.67	2.30
Parallax Mapping	0.65	0.69	2.33
Parallax Occlusion Mapping	0.97	1.04	3.06
Slow Normal Mapping	1.23	1.32	3.46

Table 1. Baseline measurements of frametimes in milliseconds varying different 3D objects with meso-scale application of the texturing techniques.

Small-scale	Quad	Cube	Bunny
Normal Mapping	20.95	22.55	54.59
Parallax Mapping	20.95	20.78	52.84
Parallax Occlusion Mapping	34.23	34.17	95.19
Slow Normal Mapping	33.69	48.78	122.47

Table 2. Performance of small-scale texture mapping techniques with 36 samples on different 3D objects. Average frame times are reported in milliseconds.

though it is a simple model by today's standards, it can act as a considerable increase in the number of triangles and complexity in comparison to the other models. Since we are measuring the shader performance and not polygonal techniques, the bunny will act as our benchmark for an actual model in regards to geometric complexity.

In the experiment, the same configurations were used in the shader and scene. Only variables were the 3D object and the technique. Texture was a 32x32 pyramid texture and BRDF was the Cook-Torrance BRDF. For the small-scale techniques, sampling was set to 36 samples and self-shadowing effect was turned off for parallax and parallax occlusion mapping.

As we can see from these figures, the increases in frame times are quite similar in different models. Baseline figures increase for the bunny as expected because of the more complex polygonal structure, but the increases between the different techniques are similar in scale. Normal mapping and parallax mapping are quite similar as their underlying complexity is quite similar. There is a significant performance cost to using parallax occlusion mapping as it employs a heavier and a more accurate algorithm to calculate the parallax effect.

Small-scale	12	36	64	100	144
Normal Mapping	10.76	22.54	38.91	59.94	85.61
Parallax Mapping	9.97	20.78	35.67	54.74	78.04
Parallax Mapping with Shadows	29.76	62.38	106.79	163.18	233.96
Parallax Occlusion Mapping	16.30	34.18	58.72	89.95	128.01
Parallax Occlusion Mapping with Shadows	91.94	202.93	355.13	548.23	784.34
Slow Normal Mapping	22.30	48.79	85.74	133.15	191.06

Table 3. Performance of small-scale texture mapping techniques with varying sampling rates. Average frame times are reported in milliseconds.

As we move from single sample techniques to multiple sampling of the small-scale texture mapping techniques, the average frame times increase significantly. This is due to the fact that the operations retrieving the texture value, doing the parallax calculation, and the shading calculation is done on every sample. This is a heavier operation and the decreases in performance are seen above.

The magnitude of the increase in the frame times is seen more clearly when using the bunny object. This is because, even though the rendering resolution and the amount of pixels is the same across the different experiment configurations, no culling is done in the renderer. This means that even though the amount of pixels is the same, the pixel shader operations in the shader are done for all overlapping triangles of the scene. This can be dealt with by implementing the Z-culling technique to the renderer. This would result in computing the pixel shader operations only for the visible triangles. However, this was out of the scope of this thesis but should be noted when inspecting the frametimes of the more complex bunny object.

6.2.2 Samples

Next, we inspect the performance values of varying sample rates in the small-scale texture mapping techniques presented in this thesis. This experiment used the same scene as above, but only the cube object was used. Varying factors are the small-scale texture mapping technique and the sample rate.

As we can see from the frame times in Table 3, the sample rate has a major effect on the performance of all small-scale texture mapping techniques. As the sampling rate is increased, the amount of operations done in the pixel shader increases considerably. The basic operations for a single sample are the possible parallax operations that determine the texture coordinates, the texture sampling itself and the shading calculation for that specific sample. These calculations are done for each sample.

This is made more apparent in the case of self-shadowing. For each sample, the self-shadowing calculations are done for all light sources using the corresponding parallax technique. The self-shadowing operations are done in each sample and consist of the corresponding parallax calculation to determine if the light is visible in that position. For example in the case of a 100 pixels, 10 samples and 10 lights, the calculation for each pixel needs 10 original sample shading calculations and each sample needs 10 self-shadowing calculations. This case would result in 1,000 original sample shading calculations and 10,000 self-shadowing calculations.

Here we can see that the sampling rate and inclusion of self-shadowing are the most significant factors in the performance in the proposed shader.

6.2.3 Textures

Another possible factor for the performance of the algorithm might be the given texture. This is particularly relevant with parallax occlusion mapping. In parallax occlusion mapping, the amount of calculations for the intersection point increases as the algorithm calculates deeper into the depth map. In Table 4, we present findings from experiments with different textures. A flat texture was used with the depth map set to the lowest or the highest setting in the corresponding experiment runs. The amount of intersection calculations are smallest in the top setting of the flat texture. The pyramid texture performance from previous experiment is given as a reference. Other variables in the experiment were the same as in the previous experiment and sample rate was set to 36.

As expected, the choice of texture has no noticeable performance difference in the normal mapping or parallax mapping cases. The parallax mapping algorithm does not do extra depth

Small-scale	Flat (Top)	Flat (Bottom)	Pyramid
Normal Mapping	22.63	22.62	22.54
Parallax Mapping	20.78	20.78	20.78
Parallax Mapping with Shadows	66.97	67.05	62.38
Parallax Occlusion Mapping	22.38	38.14	34.18
Parallax Occlusion Mapping with Shadows	57.95	208.39	202.93

Table 4. Performance of small-scale texture mapping techniques with varying textures. Average frame times are reported in milliseconds.

testing. Instead, it approximates the new vector coordinates in a single step using the first depth map value it encounters and the current viewing angle.

In the case of parallax occlusion mapping however, the depth map itself has an effect on performance. The algorithm tests multiple texture samples along the view vector when determining if the vector is above or below the depthfield. In the case of the bottom depth map, the algorithm checks a longer path along the view vector sampling the texture at regular intervals. This interval is defined by dividing the depth into layers. The depth map is then sampled at each layer intersection and the amount of layers is increased as the viewing angle approaches a grazing angle. In this experiment the minimum layer amount was set to 15 layers (used at perpendicular viewing angles) and maximum to 30 layers (used at grazing angles). The effect of this sampling on perfomance is seen most clearly in the difference between top and bottom depth textures in parallax occlusion mapping cases.

7 Discussion

This thesis presents a texture mapping method for modeling small-scale surface structures. This approach is suitable for a large array of materials featuring small-scale geometry variation. This small-scale texture mapping approach allows using the same implementation for multiple kinds of materials with different small-scale surface structures. The implementation of the technique works as a proof of concept that can function as the first step towards a small-scale texture mapping system usable in real-time graphics applications. There are still issues to be solved until this goal of an applicaple system can be achieved.

Small-scale texture mapping is an interesting approach for modeling the small-scale surface structure as it implements well with regular texturing workflow. This also enables it's use for a wide variety of different repeating textures already developed for these kinds of applications. The obvious downside in all the presented techniques is the approach of sampling the texture multiple times to find a mean shading for each pixel. This approach makes the shader implementation too expensive for actual use in real-time graphics applications. It also introduces wide variety of problems including noise that appears from insufficent sampling and artefacts resulting from textures of low quality. In the implementation of the method in this thesis, the texture sizes were very small to achieve frame times that offer even limited interactability. To achieve high quality results, it is clear that larger textures need to be used. This cannot be achieved with the sampling based approach used here. This indicates that some sort of texture preprocessing is needed to form a faster calculation of the shading.

Fortunately, there are interesting avenues of future research in this regard. For normal mapping, there are promising results in the challenging problem of normal map filtering. Texture map filtering is required when a texture is viewed from a distance and averaging of the contributing texture values is needed. For color mapping, usual approaches for filtering are bilinear, trilinear, or anisotropic filtering. Although normal mapping is an analogue for color mapping applied to surface normals, the filtering methods in normal mapping do not apply as the shading is not linear. This means that as we zoom out, averaging the normal map values into a single pixel value would average the complex surface normals into a flat surface.

In regards to the technique of microstructure normal mapping, an interesting approach would be to apply frequency domain normal map filtering presented by Han et al. (2007). In their paper, they derive an analytic formula, which shows that filtering can be written as a spherical convolution of the material's BRDF and the normal distribution function (NDF). The mathematical form holds for a large class of common BRDFs (Lambertian, Blinn-Phong, Torrance-Sparrow and many measured BRDFs). Spherical harmonics are used for low-frequency functions. Spherical harmonics are the frequency domain analog to Fourier series on the unit sphere. The normals are represented as spherical delta distributions in the NDF which is convoluted with the BRDF to gain the final effective BRDF used in shading. For higher frequency functions, Han et al. (2007) use von Mises-Fisher distributions, which model Gaussian-like distributions on the unit sphere. Both of their approaches preserve the original normals and provides a full BRDF with all the original surface normals.

Frequency domain normal map filtering approach could introduce significant performance gains in the small-scale normal mapping technique. Filtering the full small-scale normal map to a single effective BRDF would help to overcome the problematic sampling approach and allow to calculate a single BRDF from the small-scale texture maps. This would quite probably introduce high performance gains for the technique presented in this thesis.

Han et al. (2007) acknowledge that their approach only addresses the filtering of normal maps and state that a critical direction of future research is filtering of displacement maps and geometry. This would also be needed for the small-scale parallax mapping technique. Wu, Dorsey, and Rushmeier (2009) use the charasteristic point maps for this. They calculate a bidirectional visible normal distribution function (BVNDF) to describe the NDF of small-scale geometry with masking and shadowing and get the effective BRDF by integrating the product of the BVNDF and small-scale BRDFs. Unfortunately, this approach requires a large volume of precomputed data and significant precomputation time for the BVNDF and BRDFs, making it infeasible for the proposed techniques.

Iwasaki, Dobashi, and Nishita (2012) present a different technique for the same problem. They represent the BVNDF as a sum of spherical gaussians (SG). The von Mises-Fisher distributions used by Han et al. (2007) are similar as they are the same as the normalized SGs. Iwasaki, Dobashi, and Nishita (2012) state that this SG representation of the BVNDF

can be presented in a simple form for the convolution of SGs with a memory footprint that is small enough to fit in GPU memory. Their SG representation of the BVNDF, small-scale BRDFs, and effective BRDFs is stated to achieve real-time rendering in all-frequency environment lighting.

The SG approach is an interesting avenue of future research for the small-scale parallax mapping techniques as well. Iwasaki, Dobashi, and Nishita (2012) calculate their SG based effective BRDF from small-scale 3D models as opposed to the parallax mapping approach of this study. The parallax microstructure maps could possibly be computed into the SG form similarly as Iwasaki, Dobashi, and Nishita (2012) compute them from the 3D models.

The artefact and aliasing problem encountered with the parallax mapping techniques also needs to be addressed. As stated before, in this study the microstructure texture maps used were very small compared to the textures used in regular parallax and parallax occlusion mapping. This was done due to the expensive sampling based approach. If the SG based effective BRDF could be calculated the parallax mapping textures, the size of the textures could be increased with computational cost landing mainly on the texture preprocessing phase, while not affecting the runtime performance so dramatically. These larger textures with a higher layer count could reduce the aliasing of the parallax occlusion maps significantly.

Unfortunately, the larger textures would not help the parallax mapping case because of it's simple approach to determine masking. The simplistic approach would result in similar aliasing and incorrect masking calculation even if larger textures were used. For this reason, using the regular parallax mapping technique and it's applicability for accurate small-scale texture mapping seems unlikely. The use of more sophisticated parallax mapping techniques is recommended in further research into the area.

To further improve the accuracy of the shading computations, we should also consult the improvements in parallax mapping techniques. Parallax occlusion mapping is only one of the many approaches to displacement mapping and was selected for this study as an example of a more accurate parallax mapping technique. Other promising techniques include steep parallax mapping (McGuire and McGuire 2005), relief mapping (Policarpo, Oliveira, and

Comba 2005), view dependent displacement mapping (Wang et al. 2003), and interval mapping (Risser, Shah, and Pattanaik 2005). All of these methods allow for self-occlusion and self-shadowing. A comprehensive comparison of these techniques should be conducted to determine their suitability for small-scale texture mapping.

A combination of larger texture sizes, an accurate aliasing free displacement mapping technique, and an effective filtering to combine the small-scale shading into a single BRDF could make the texture mapping of small-scale surfaces an effective method for graphics applications. This would allow these applications to make use of modeled small-scale surface structures and add a new degree of freedom in modeling complex surface materials.

Bibliography

Walter, Bruce, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. 2007. "Microfacet Models for Refraction Through Rough Surfaces". In *Proceedings of the 18th Eurographics Conference on Rendering Techniques,* 195–206.

Wang, Lifeng, Xi Wang, Xin Tong, Stephen Lin, Shimin Hu, Baining Guo, and Heung-Yeung Shum. 2003. "View-dependent Displacement Mapping". *ACM Transactions on Graphics* 22 (3): 334–339.

Welsh, Terry. 2004. *Parallax Mapping with Offset Limiting: A Per-Pixel Approximation of Uneven Surfaces.* Technical report. Infiscape Corporation.

Westin, Stephen H., James R. Arvo, and Kenneth E. Torrance. 1992. "Predicting Reflectance Functions from Complex Surfaces". In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques,* 255–264. SIGGRAPH '92. ACM.

Wu, Hongzhi, Julie Dorsey, and Holly Rushmeier. 2009. "Characteristic Point Maps". *Computer Graphics Forum* 28 (4): 1227–1236.

———. 2011. "Physically-based Interactive Bi-scale Material Design". *ACM Transactions on Graphics* 30 (6): 145:1–145:10.

———. 2013. "Inverse Bi-scale Material Design". *ACM Transactions on Graphics* 32 (6): 163:1–163:10.

Akenine-Moller, Tomas, Eric Haines, and Naty Hoffman. 2008. *Real-Time Rendering.* 3rd. Natick, MA, USA: A. K. Peters, Ltd.

Blinn, James F. 1977. "Models of Light Reflection for Computer Synthesized Pictures". *SIGGRAPH Comput. Graph.* 11 (2): 192–198.

Cook, Robert L., and Kenneth E. Torrance. 1982. "A Reflectance Model for Computer Graphics". *ACM Transactions on Graphics* 1 (1): 7–24.

Dorsey, Julie, Holly Rushmeier, and Franois Sillion. 2008. *Digital Modeling of Material Appearance.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Glassner, Andrew S. 1994. *Principles of Digital Image Synthesis.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Han, Charles, Bo Sun, Ravi Ramamoorthi, and Eitan Grinspun. 2007. "Frequency Domain Normal Map Filtering". *ACM Transactions on Graphics* 26 (3): 28:1–28:12.

Hoffman, Naty. 2013. *Background: Physics and Math of Shading.* Technical report. SIGGRAPH.

Iwasaki, Kei, Yoshinori Dobashi, and Tomoyuki Nishita. 2012. "Interactive Bi-scale Editing of Highly Glossy Materials". *ACM Transactions on Graphics* 31 (6): 144:1–144:7.

Karis, Brian. 2013. *Real Shading in Unreal Engine 4.* Technical report. SIGGRAPH.

Kinnunen, Matti. 2018. "Application Source Code for the Master's Thesis". `https://yousource.it.jyu.fi/pro-gradu-application`.

McGuire, Morgan, and Max McGuire. 2005. "Steep Parallax Mapping". *Poster at ACM Symposium on Interactive 3D Graphics and Games.* `http://www.cs.brown.edu/research/graphics/games/SteepParallax/index.html`.

Phong, Bui Tuong. 1975. "Illumination for Computer Generated Pictures". *Communications of the ACM* 18 (6): 311–317.

Policarpo, Fábio, Manuel M. Oliveira, and João L. D. Comba. 2005. "Real-time Relief Mapping on Arbitrary Polygonal Surfaces". In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games,* 155–162. I3D '05. ACM.

Risser, Eric A, Musawir A Shah, and Sumanta Pattanaik. 2005. *Interval mapping.* Technical report. University of Central Florida.

Schlick, Christophe. 1994. "An Inexpensive BRDF Model for Physically-based Rendering". In *Computer Graphics Forum,* 13:233–246. 3.

Torrance, K. E., and E. M. Sparrow. 1967. "Theory for off-specular reflection from rough surfaces". *Journal of the Optical Society of America* 57 (9): 1105–1114.