Universität des Saarlandes

Rendering and Streaming of Bidirectional Texture Functions

Masterarbeit im Fach Informatik
Master's Thesis in Visual Computing
von / by

Oleksandr Sotnychenko

angefertigt unter der Leitung von / supervised by

Prof. Dr. Philipp Slusallek

betreut von / advised by

M. Sc. Kristian Sons

...

begutachtet von / reviewers

...

...

Saarbrücken, November 2014

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement in Lieu of an Oath

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Sperrvermerk

Blocking Notice

Saarbrücken, November 2014 Oleksandr Sotnychenko

Abstract

Bidirectional Texture Functions (BTF) are 6-dimensional functions that depend on the spatial position on a surface, light and camera directions. Due to changes either of light or camera directions the appearance of real-world materials can severely change. BTF can represent such materials by capturing important material properties for a wide range of illumination changes.

In this work we present a technique to render BTFs at real-time within a web-browser using WebGL. The ever-growing number of mobile devices that use web-browsers imply constraints on the hardware capabilities. Thus, rendering has to be efficient to be possible on such devices and the fact that all data has to be transferred to the client - compression is inevitable. We employ a principal component analysis to compress the data, which allows for rendering of BTFs with interactive frame rates. To provide immediate feedback to the user we provide additionally streaming that allows to progressive enhance the rendering quality while still transferring the remaining data to the client.

Acknowledgements

I would like to express my sincere gratitude ...

To my family.

Contents

List of Figures

Chapter 1

Introduction

One of the main goals in computer graphics is realistic rendering. Even though computer graphics is constantly improving, we are still quite away from reality because material representation in a traditional way lack important realistic properties. A 2-D texture in conjunction with a shading model is a conventional way to represent material appearance in rendering. On the other side, real-world materials surfaces consist of surface meso-structures, i.e. intermediate in size local geometric details. Meso-structures are responsible for fine-scale shadows, self-occlusions, inter-reflection, subsurface scattering and specularities. Also, reflectance of the real-world materials spatially varies.

One of the possible solution to represent such material's attributes is to use sophisticated light functions, for instance a Bidirectional Texture Function (BTF). A BTF is a 6-dimensional function that depends on camera and light directions as well as on spatial texture coordinates. BTFs conceptually extend traditional 2-D texture by the dependence on light and camera directions. This function is usually acquired as a data-set of thousands images that cover varying light and camera directions. Due to enormous size of such data direct rendering on the modern hardware without any compression is impractical. Fortunately, compression methods combined with modern graphics hardware provide fast computational speed, which allows us to employ BTF.

1.1 Outline

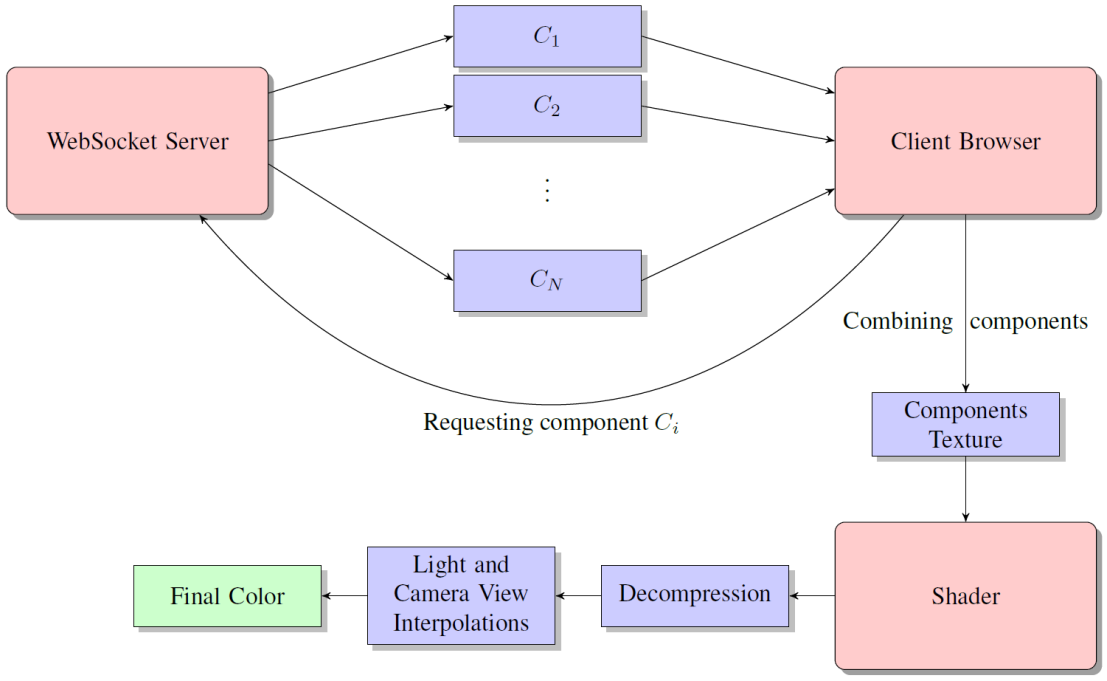

FIGURE 1.1: **Model Overview**

Chapter 2

Related Work

Chapter 3

A hierarchy of scattering functions

In this chapter we will review a hierarchy of scattering functions. Scattering functions describe for a surface how incoming and outgoing directions of the light are related [6]. Such functions are possible to measure for a given object. After obtaining the data, scattering functions can provide all the necessary information to render the material appearance. Due to diversity of material properties, different functions were introduced. In order to chose suitable scattering function for capturing certain scattering effects for a particular type of material, it is important to be aware of the hierarchy of scattering functions.

3.1 Light material interaction

Before defining any scattering functions of light the basics light-material interaction. Generally speaking, when light hits a material's surface, a sophisticated light-matter process happens. Such process depends on physical properties of the material as well as on physical properties of light[24]. For instance, an opaque surface such as wool will reflect light differently than a smooth surface with high specularities such as metal.

When light makes a contact with a material, three types of interactions may occur: light *reflection*, light *absorption* and light *transmittance*. Light *reflection* is the change in direction of light at an interface between two different media so that light returns into the medium from which it originated. Light *absorption* is a process when a light is being taken up by a material and transformed into internal energy of the material, for instance thermal energy. When a material is transparent, light *transmittance* can occur.

It means, that the light travels through the material and exits on the opposite side of the object. Figure 2 demonstrates these 3 types of interactions.

Because light is a form of energy, conservation of energy says that [24]

incident light at a surface = light reflected + light absorbed + light transmitted

3.2 General Scattering Function

To define the general scattering function(GSF) imagine the light-wave hitting the surface at time t_i and position x_i and with wavelength λ_i[15]. With a given local coordinate system at a surface point, the incoming direction of light can be defined as (θ_i, ϕ_i). Light travels inside the material and exits the surface at position x_o and time t_o, with possibly changed wavelength λ_o in the outgoing direction (θ_o, ϕ_o).

According to the description we get a GSF

$$GSF(t_i, t_o, x_i, x_{0,}\theta_i, \phi_i, \theta_o, \phi_o, \lambda_i, \lambda_o)$$

in which spatial positions $x_{i,o}$ are 2-D variables. This function describes light interaction for each surface point for any incoming light and outgoing direction at certain time. The function has 12 parameters! Note that we neglected light transmittance, which would even further complicate the function.

3.3 Bidirectional Scattering-Surface Reflectance Distribution Function

Since the measurement, modeling and rendering of a 12-D GSF function is currently not practical, additional assumptions have to be made to simplify the function. Usually such assumption are made [15]:

- light interaction takes zero time ($t_i = t_o$)
- wavelength is separated into the three color bands red, green and blue ($\lambda_{r,g,b}$)
- interaction does not change wavelength ($\lambda_i = \lambda_0$)

After mentioned assumptions we get a 8-D bidirectional scattering-surface reflectance distribution function (BSSRDF)

$$BSSRDF(x_i, x_o\theta_i, \phi_i\theta_o, \phi_o)$$

BSSRDF describes various light interactions for heterogeneous both translucent and opaque materials. That is why BSSRDF can be used for rendering materials such as skin, marble, milk and other objects which do not look realistic without subsurface scattering. Subsurface scattering is a process when light penetrates an object at an incident point, travels inside the object and exists at a different point of the object.

3.4 Bidirectional Texture Function

If we simplify further and assume that

- light entering a material exits at the same point $x_i = x_o$, while internal subsurface scattering is still present

we will get a 6-D bidirectional texture function (BTF).

Subsurface scattering, self-occlusion, self-shadowing are still present, now it just comes pre-integrated, i.e. it can be defined through BSSRDF [15]:

$$BTF(x, \theta_i, \phi_i, \theta_o, \phi_o) = \int_S BSSRDF(x_i, x, \theta_i, \phi_i, \theta_o, \phi_o)dx_i$$

The assumption that $x_i = x_o$ simplifies measuring, modeling and rendering of the scattering function. As we can see the BTF integrates subsurface scattering from neighbouring surface locations.

3.5 Bidirectional Subsurface Scattering Distribution Function

Another possible reduction of the 8-D BSSRDF is to assume that we deal with a homogeneous surface [6], i.e.

- subsurface scattering depends only on relative surface positions of incoming and outgoing light $(x_i - x_o)$

Simply saying it means that scattering do not vary over a surface. With such assumption we get a 6D function that known as a bidirectional subsurface scattering distribution function (BSSDF).

$$BSSDF(x_i - x_o, \theta_i, \phi_i, \theta_o, \phi_o)$$

BSSDF represents homogeneous materials for which subsurface scattering is a significant feature of their overall appearance. For instance, BSSDF accounts for objects such as water, milk, human skin, and marble.

3.6 Bidirectional Reflectance Distribution Function

If we assume the following for a BSSDF that

- there is no spatial variation
- no self-shadowing
- no self-occlusion
- no inter-reflections
- no subsurface scattering
- energy conservation
- reciprocity $BRDF(\theta_i, \phi_i, \theta_o, \phi_o) = BRDF(\theta_o, \phi_o, \theta_i, \phi_i)$.

we get a 4-D bidirectional reflectance distribution function (BRDF)

$$BRDF(\theta_i, \phi_i, \theta_o, \phi_o).$$

Nicodemus et al. [18] was the one who proposed the BRDF. Two principal properties of the BRDF were introduced, i.e. *energy conservation* and *reciprocity*. *Energy conservation* law states that the total amount of outgoing light from a surface cannot exceed the original amount of light that arrives at the surface [24]. *Reciprocity* says that if we swap incoming and outgoing directions BRDF stays the same. If either of these conditions are not satisfied then such BRDF is called *apparent* BRDF (ABRDF) [23].

It is quite difficult to create a mathematical model for a BRDF that satisfies reciprocity, energy conservation and the same time produces realistic images. However, most BRDF's models do not satisfy these conditions and still get plausible rendering results. For instance, Phong model is the most well-known shading model in computer graphics. The traditional Phong model satisfy neither energy conservation nor reciprocity, but can still render many materials realistically plausible. Usually, such materials are of opaque

and flat nature, for example plastic materials. The Phong model is an empirical model and is designed to fit the original function, often based on simple formulas which were derived from observations.

3.7 Spatially Varying Bidirectional Reflectance Distribution Function

If spatial dependence for BRDF takes place, we get a 6-D spatially varying BRDF (SVBRDF)

$$SVBRDF(x, \theta_i, \phi_i, \theta_o, \phi_o).$$

Assumptions are the same as for the BRDF, except now spatial dependence is present.

A SVBRDF is closely related to a BTF. The SVBRDF and the BTF almost the same scattering function, the difference is in scattering process. Changes in scattering at local position x for the BTF are influenced from neighbouring 3D surface geometry, as a result the self-shadowing, masking and inter-reflections are captured by the BTF. On the other side, the spatial dependence of a SVBRDF describes variations in the optical properties of a surface [8].

A SVBRDF represents structures at micro-scale level, which corresponds to near flat opaque materials. On the other side, a BTF capture structure both at macro and micro scales. That means that the BTF takes into account influences from local neighbourhood structures. Even though measurement, compression, rendering are more efficient for the SVBRDF, the BTF can produce better visual results. [8]

3.8 Summary of scattering functions

In practice, an advantage of simpler scattering function is a computation efficiency, while a disadvantage is a reduction of visual quality. But, development of the graphics hardware is always improving and as a result this encourages to use sophisticated scattering functions, which provide improvement in realistic material rendering.

However, a complex material representation introduces additional constraints on the data measurement as well as modeling becomes more challenging. For instance, till now a GRF has not been measured and still stays as a state-of-the-art problem. In

practice, the appropriate scattering function depends on the specific application. For instance, a scene with various textures can be rendered with different scattering functions. Simpler materials that do not have complex scattering features can be rendered with a 2-D textures in conjunction with BRDFs model. If the material cannot be presented realistic without subsurface scattering, masking, self-reflections then typically such materials require high quality representations (BTF,SVBRDF). However, these advanced material representations are very complex. In practice a tradeoff between visual quality, measurement, data size and rendering cost is inevitable. Ideally high visual quality is preferred with low rendering computational cost.

Chapter 4

BTF acquisition

BTF acquisition is not an easy task as it requires time for acquiring and post-processing the data, and also resources are needed to create acquisition setup. There are only a few measurement systems [15, 21, 5, 10, 12, 16] exist, but as the interest in the realistic material rendering using BTF is growing, measurement systems are developing. In this chapter we will review how in general BTF data acquisition is made, which post-processing steps are made, and pros and cons of existing measurement systems. Also, we will take a look at publicly available BTF datasets.

4.1 General acquisition methods

All the mentioned BTF acquisition systems share the same idea in the data acquisition, i.e. capturing the appearance of a flat square slice of the material surface under varying light and camera directions. The material surface is usually sampled over a hemisphere above the material slice as shown in Figure 4.1. Depending on the material reflectance properties sampling distribution may vary, e.g. sampling distribution can be dense in regions where specular peaks in light reflection occur. Then, if needed uniform distribution can be made in a post-processing step with a help of interpolation [8].

Digital cameras are used as capturing devices of the material appearance. Depending on the setup the number of cameras can vary. If there is only one camera [15, 16, 5], it usually moves around over the hemisphere above the sample with a help of robotic arm or the rail-trail system [15]. The advantage of such approach is that it is less expensive and can suit for low-budget applications. But, the disadvantage is the positioning errors that can arise, which influence the overall measurement error. Depending on the application light sources can be fixed or moveable.

There are approaches which do not involve camera and light source movement at all. Schwartz et al. [21] developed a novel measurement system which uses a dense array of 151 camera, which are uniformly placed on hemispherical structure. Flashes of the cameras are used as light sources. Such setup provide high angular and spatial resolutions.

Also, Ngan et al. [16] made a setup which does not involve camera movement by placing the planar slices of the material in a form of the pyramid. Thus, such setup captures the material appearance for 13 different camera views at once. Light directions are sampled by hand-moved electronic flash. The disadvantage of such setup is that it provides sparse angular resolutions, but depending on the application such approach can be plausible.

The material sample is commonly flat and squared slice, which is placed the holder plate. To conduct automatic post-processing borderline markers are placed on the holder. Those markers provide the important information for further post-processing steps such as image rectification and registration.

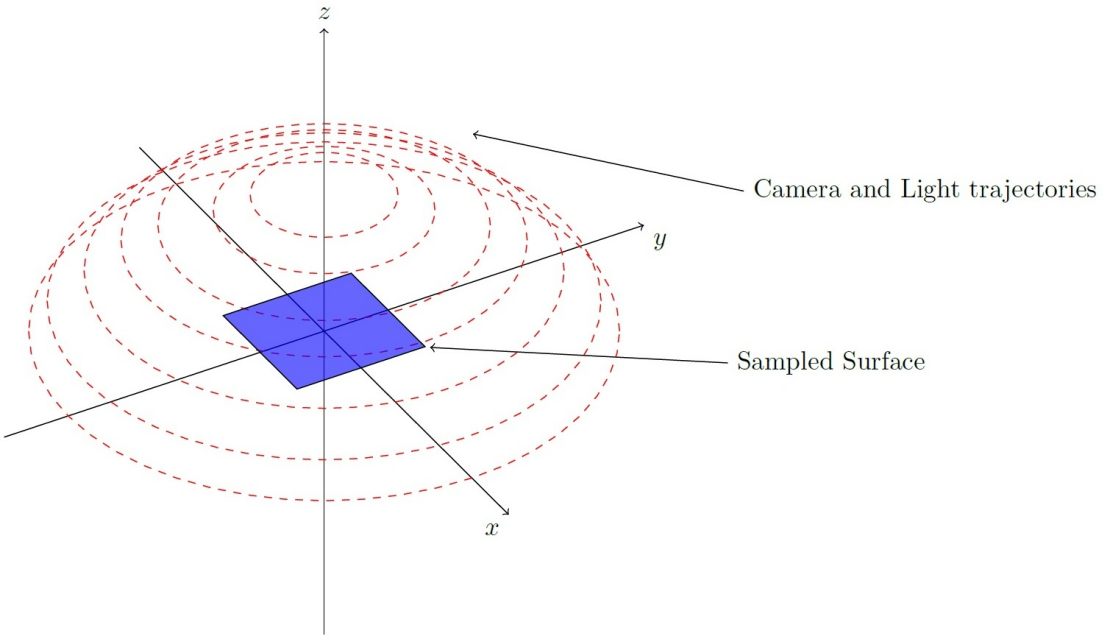

FIGURE 4.1: **Example of BTF measurement.**
Camera and light positions share the same trajectories. Red dashed circles are the sample positions on the hemisphere.

4.2 Post-processing steps

After the measurement is done, the raw data has to be further post-processed, because typically such such data is not ready for further modeling/compression or rendering.

Raw data is a set of images that are not aligned with each other and images are not mutually registered.

When the raw images are obtained under different camera angles (θ, ϕ) they are perspectively distorted [19]. Thus, sample image have to be aligned with each other and spatially registered to be further exploited. Firstly, borderline markers that were placed around the material sample on the holder plate aid the automatic detection of the material slice. Then, after the material slices are detected and cropped, they are ready to for mutual alignment. This process is called *rectification*. *Rectification* is a process which involves projecting all sample images onto the plane which is defined by the frontal view, e.g. $(\theta_o = 0, \phi_o = 0)$. In other words, all normals of sample images have to be aligned with their corresponding camera directions, i.e. as if all sample images were taken from frontal view $(\theta = 0, \phi = 0)$. The last step is image *registration*, a process of getting pixel-to-pixel correspondence between the images. As, all transformation were done, it is only enough to rescale all images to some equal resolution.

Even after the proper rectification and registering of the measured data, registration errors can be still present between individual camera directions [8]. This happens due to structural occlusions of the material surface. Because, of such self-occlusions some geometry structures are not captured by certain camera directions, but can be captured with other camera directions. That is why even after rectication images captured from completely different directions are not correctly mutually align. Also, registration errors can be caused both by inaccurate camera and material sample positions happened during the measurement processes. One way to avoid artifacts in rendering caused by registration errors is to employ a compression step separately for each fixed camera positions, i.e. subsets of BTF data. For instance, Sattler et. al. [19] has done this approach.

If needed, further processing steps can be done, for instance *linear edger blending* to reduce tiling artifacts [19]. Also, typical image processing steps may be employed, e.g. noise reduction filters.

4.3 Publicly available BTF datasets

The accurate rendering of the material surface is highly depended on the quality of acquired data, especially for BTF. There are several properties that are vital for reproducing quality rendering results. The BTF datasets can be distinguished by how well image post-processing were done and how good spatial and angular resolutions are. So,

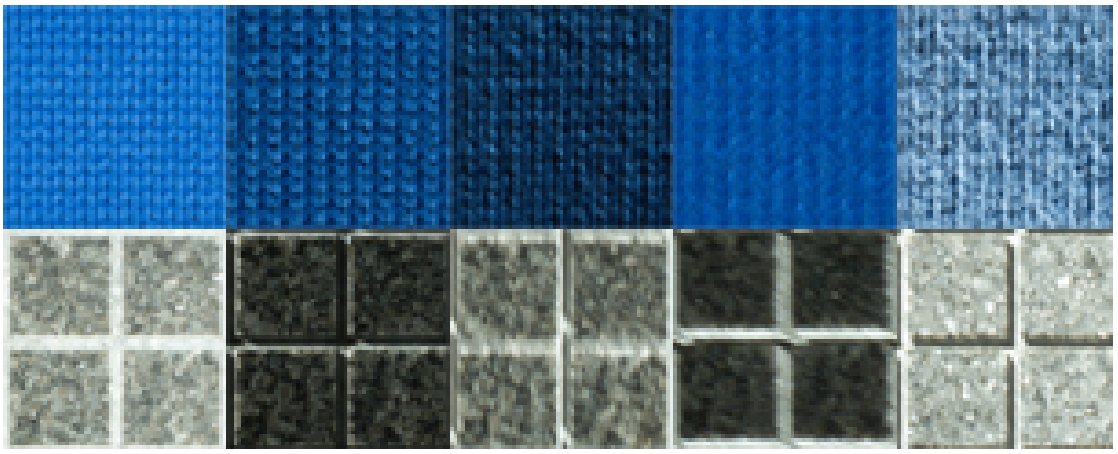

FIGURE 4.2: **BTF example of Bonn Database [1].**
Example how BTF catches rich appearance of the material due to dependencies on light and camera directions. Upper row is a knitted wool, lower is a graved granite stone.

depending on the application trade-off between high and low spatial or angular resolutions is done. For example, some materials with a low range of reflectance may benefit from sparse resolutions, e.g. wool, plastic, etc.

A pioneer in the BTF acquisition was Dana et. al. [2], who measured 61 materials with fixed light and moving camera aided by a robotic arm. Such procedure resulted in a set of images, which can be regarded as a subset of BTF, which is called surface light field (SLF)

$$SLF(x, y, \theta_i, \phi_i, \theta_o, \phi_o)$$

where (x, y) is a surface point of a flat sampled material, (θ_i, ϕ_i) incoming light direction (light direction) and (θ_o, ϕ_o) outgoing light direction (camera direction).

Data et. al. *CUReT* database is publicly available [2]. For each measured surface Dana et. al. used 205 different combinations of camera and light directions, which resulted in relatively sparse angular resolution. Dana's et. al. BTF database are not rectified, but the authors provide image coordinates to allow their further rectification. Because, of this limitations such BTF dataset usually used for computer vision purposes, i.e. texture classification [8].

Based on Dana et. al. BTF measurement system, Sattler from Bonn University made his own measuring system [19]. The main difference in that system is that a camera moves on a semi-circle rail around material sample. Such setup provides spatially rectificated and mutually registered data, with reasonable angular and spatial resolutions. Datasets of Bonn University [1] are publicly available and were used in this thesis.

Consider Figure 4.2, which illustrates one of the sampled materials of Bonn database. The measured surface is being fixed all the time on the sampler holder as shown in Figure

4.1. For each light position, a camera takes a shot of the material while moving from point to point of the hemisphere. Bonn database has the same trajectory for camera and light positions, i.e. 81 positions on the hemisphere, which resulted in $81 \times 81 = 6561$ total number of acquired images. After that the sample images were rectificated and registered, resulting in a set of images with spatial resolution 256×256. Typically, the size of one uncompressed BTF is around 1.2 Gb.

Chapter 5

BTF data representations

Before doing any compression step it is important to chose a suitable data representation for BTF data. Suitable presentation can enormously influence the final result of BTF rendering. Basically, there are two common arrangements for BTF, i.e. as a set of original rectified and registered textures (*image-wise* representation) and a set of ABRDFs (*pixel-wise* representation). ABRDF was defined in chapter 3.6. In this case it is called *apparent*, because BTF includes effects such self-occlusions, sub-surface scattering and other complex effects which violate two basic properties of BRDF. Both representations can be mathematically expressed as

$$BTF_{Texture} = \left\{ I_{(\theta_i,\phi_i,\theta_o,\phi_o)} \mid (\theta_i, \phi_i, \theta_o, \phi_o) \in M \right\}$$
$$BTF_{ABRDF} = \left\{ P_{(x)} \quad \mid (x) \in I_{(\theta_i,\phi_i,\theta_o,\phi_o)} \subset \mathbb{N}^2 \right\}$$

where M denotes a set of images $I_{(\theta_i,\phi_i,\theta_o,\phi_o)}$ measured for different light and camera directions (i, o) accordingly. BTF_{ABRDF} denotes a set of $P_{(x)}$ images, where each of them stores light intensity for a fixed spatial position x for all possible light and camera directions, i.e. a domain for a $P_{(x)}$ image is $(n_i \times n_o) \subset \mathbb{N}^2$, where n_i and n_o are number of light and camera directions accordingly. In our case with Bonn datasets the images are given with 256×256 resolution for 81×81 directions, then the size of one single image for both representations are $|I| = 3 \times 256 \times 256$ and $|P| = 3 \times 81 \times 81$, where 3 stands for RGB channels.

Basically, one representation, i.e. $BTF_{Texture}$ is used for compression methods that do analysis on the whole sample plane, while BTF_{ABRDF} representation allows better pixel-to-pixel comparisons, which can give a big advantage for methods that employ pixel-wise compression, e.g. BRDF based models. Also, such arrangement provides images with lower variance [9], which can allow better compression results in certain scenarios. But, anyway both representations posses the same information, and any compression method can use either of them.

Müller et. al. [14, 9] claim that ABRDF representation works slightly better in comparison to image-wise representation. The advantage of ABRDF is that a compression ratio can be 10 times better and reconstruction error is slightly smaller than the image-wise representation. Also, Borshukov et. al. [17, Ch. 15] chose ABRDF representation, claiming that it provides better compression ratio. The reason why ABRDF provides better compression ratio is because it provides better pixel-to-pixel comparison than image-wise. Each variable vector of ABRDF arrangement depends only on a surface complexity [14], i.e. on a variation of reflection properties at a spatial point of the surface. On the contrary, image-wise variable depends on the whole measured image plane, thus it is logical that it may provide lower compression rate due to stronger variations. After all, we have chosen to employ ABRDF data representation based on above reasons, which allowed us to achieve $1 : 100$ compression ratio.

Chapter 6

BTF compression methods

BTF data consists of thousands of images, which means single BTF requires lots of storage space. Bonn Database [1] consists of 8-bit PNG images with a resolution of 256×256 sampled for 81×81 different camera and light directions. The uncompressed data occupies approx. 1,2 GB of space. And that is only for one BTF material. To render the scene with several BTFs and to achieve acceptable frame-rates becomes practically impossible task, especially for low-end hardware. Also, as we intent to render BTF in a web-browser, BTF data would have to be transferred from a server to a client, which means the compact representation of BTF is inevitable in such case. For our scenario it is important to chose the right compression method, i.e. the one that can allows real-time decompression, high compression rate while preserving good quality, and separability of compressed BTF data. Separability is needed for real-time streaming in a web-browser. As the data is transferred in small chunks during the stream and at some point the rendering has to be refreshed, thus it is inevitable to have the ability to decompress full BTF sample having only partial data.

There are different types of methods applied for compressing the BTF. Those methods can be categorized the following way:

- *Analytic methods* group, where BTF is represented by analytic BRDF models. Analytic BRDF models are the functions which fit separately each texel of BTF. Such functions store few parameters, thus high real-time performance is easily achieved. However, these group of methods can suffer from decreased quality [9]. Also, it is hard to change parameters in order to control the visual quality. Chapter 6.1.

- *Statistic methods*, which belong to methods that reduce dimensionality based on statistics, for instance based linear basis decompression method such as PCA (Principal component analysis). PCA based methods are frequently used, because its parameters directly correspond to the trade-off between compression ratio and reconstruction quality. Also, PCA is frequently a basis point for some more sophisticated methods [20]. Chapter 6.2.

- *Probabilistic models*, which can spatially enlarge BTF to any arbitrary size without visible discontinuities and extremely compress the original BTF. However, the resulted quality usually suits for flat surfaces and there are problems of implementing in GPU for real-time rendering. Chapter 6.3.

6.1 Analytic methods

This group of methods take advantage of ABRDF representation of BTF. There is a large number of techniques which allow compactly represent BRDF, which in essence can be also applied for ABRDF. Each texel of BTF, i.e. spatial position of surface are represented as ABRDF. Each of these ABRDFs can be modeled and compressed by any BRDF model.

One of the possible ways to model ABRDF is to use *Polynomial Texture Mapping* (PTM) approach. Malzbender et. al. [13] used this approach which allows for high compression rates and generally good quality. However, PTM requires to compute specular and diffuse effects separately. PTM model assumes that the input surfaces has to be either diffuse or their specular contribution is separated beforehand. For BTF it can be quite difficult to separate specular highlights [9].

Haindl et. al. [9] applied PTM for a fixed camera positions of BTF, i.e. for reflectance fields (RF). General formula looks the following way (PTM RF):

$$R_o(r,i) \approx a_0(r)u_x^2 + a_1(r)u_y^2 + a_2(r)u_x u_y + a_3(r)u_x + a_4(r)u_y + a_5(r)$$

where R_o is approximated RF for fixed camera direction o and u_x, u_y are projections of the normalized light vector into the local coordinate system $r = (x, y)$. Set of all possible R_o is the number of all camera positions, i.e. n_o. Coefficients a_p are fitted by the use of *singular value decomposition* SVD for each R_o and parameters stored as a spatial map referred to as a PTM.

However, Malzbender et. al. [13] claims that this method produce considerable errors for high grazing angles. But, nevertheless this method enables fast rendering and generally suited for smooth material surfaces.

Another model which produces slightly better visual quality is the polynomial extension of one-lobe Lafortune model (PLM). One-lobe Lafortune model (LM) looks the following way [7]:

$$Y_o(r,i) = \rho_{o,r}(C_{o,r,x}u_x + C_{o,r,y}u_y + C_{o,r,z}u_z)^{n_{o,r}}$$

where $w_i(\theta_i, \phi_i) = [u_1, u_2, u_3]^T$ is a unit vector pointing to light position. Parameters ρ, C_x, C_y, C_z, n can be computed with a Levenberg-Marquardt non-linear optimisation algorithm [7]. During testing of this model Filip and Haindl [7] claim that LM produce unsatisfactory results for complex ABRDFs. Thus, the polynomial extension of one-lobe Lafortune was introduced (PLM RF):

$$R_o(r,i) \approx \sum_{j=0}^{n} a_{r,o,i,j} Y_o(r,i)^j$$

PLM RF solves the problem of bad quality for grazing angles and improves the rendering quality compared to PTM RF. However, statistic based methods produce even better quality compared to above methods but with lower compression rates.

6.2 Statistic methods

6.3 Probabilistic models

Chapter 7

Viewing and Illumination Angle Interpolation

Chapter 8

BTF streaming

Chapter 9

Implementation

9.1 Compression

To compress the BTF data we used Java programming language. In our case, the main problem of PCA lies in implementation of singular value decomposition (SVD). To solve this problem, we used a fast linear library *jblas* [3] developed by Mikio Braun. The *jblas* library is gaining popularity in scientific computing. This library is one of the most fastest library for the Java programming that can solve various linear algebra problems.

Compressed data has to be sent to a shader. The best way to send matrices to a shader, is to send them as textures. So, after we perform SVD, we save our resulted matrices U, Σ, V as textures. Matrices U and V are in range $[-1; 1]$, so we map the data into image domain $[0; 1]$. We store each component of matrix U separately as PNG images. We store each component separately as we would need to stream the data. Consider an example shown in Figure 9.1, which shows first components of some materials. Matrices Σ and V are stored together in one texture as they are small enough. Note that the values of matrix Σ are not in range $[-1; 1]$, but we are still able to map them and store in the texture. We will not go in details here, as it is a trivial task.

Also, one practical consideration when using *jblas* is to scale the data for better reconstructing quality in the shader. We found out that tenths of values of U and V matrices are zeros. So, basically we can scale the data by multiplying it with 10 and at the same time matrix Σ by 0.1. This way the reconstruced resulted will the same. But, with scaling we improves precision of the data when we map it.

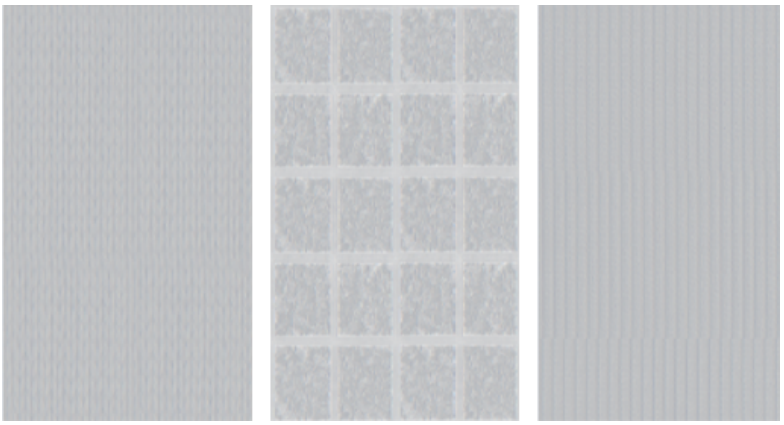

FIGURE 9.1: **Example of Principal Components**
From left to right: wool, impalla, corduroy.

9.2 Rendering

The aim of this thesis was to implement efficient BTF-shader for XML3D [22]. XML3D platform was implemented to deploy 3D graphics in web browsers. This technology is based on WebGL and JavaScript. We also use Xflow [11] to combine principal component textures in one texture, which further is needed for BTF-shader. The shader is written in OpenGL Shading Language (GLSL). The rendering process of the shader is depicted in figure 9.2.

The compressed BTF data is stored in two textures. One texture L stores principal components, the other R stores PCA weights, which determine how the components have to be summed up. The other inputs are texture coordinates, eye and light positions. Eye and light positions are transformed to spherical coordinates. Then, we lookup the three closest views from the measured BTF data, which are will be needed for interpolation purpose. This lookup process is fairly simple as we have a static array in the shader, which stores sample intervals of the measured BTF data.

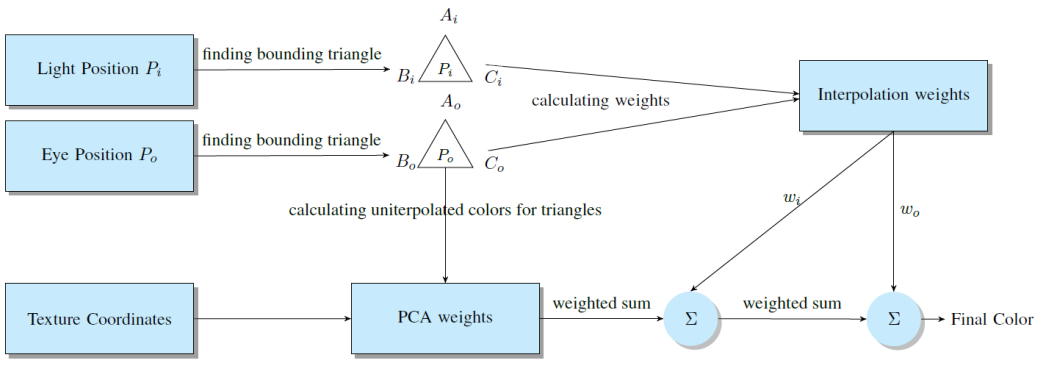

FIGURE 9.2: **Shader Design**

9.3 Streaming

Xflow is a part of XML3D implementation, which allows to process the data on flow, i.e. in runtime. At start, on a client side we create array *texData* of RGB colors, which further will be fulfilled with newly arrived principal components. As it was described in chapter **??**, we stream principal components one by one. Each component C_i arrives as a PNG image. Then, on a client side we read PNG image using PNG decoder written in JavaScript by Arian Stolwijk [4]. PNG images are decoded to pure array of RGB colors. This new data of component C_i is then placed to its place in *texData* array. Finally, using Xflow we create from *texData* array a texture with which we update our BTF-shader.

Chapter 10

Conclusions and Future Work

10.1 Summary

10.2 Future work

Bibliography

[1] Btf database bonn 2003. http://cg.cs.uni-bonn.de/en/projects/btfdbb/download/ubo2003/. Accessed on November 2014.

[2] Curret btf database. http://www1.cs.columbia.edu/CAVE/software/curet/index.php. Accessed on November 2014.

[3] Linear algebra for java - jblas. http://mikiobraun.github.io/jblas/. Accessed on November 2014.

[4] Pure javascript png decoder. https://github.com/arian/pngjs. Accessed on November 2014.

[5] K.J. Dana, B. Van-Ginneken, S.K. Nayar, and J.J. Koenderink. Reflectance and Texture of Real World Surfaces. *ACM Transactions on Graphics (TOG)*, 18(1):1–34, Jan 1999.

[6] Y. Dong, S. Lin, and B. Guo. *Material Appearance Modeling: A Data-Coherent Approach: A Data-coherent Approach*. SpringerLink : Bücher. Springer, 2013.

[7] J. Filip and Michal Haindl. Non-linear reflectance model for bidirectional texture function synthesis. In J. Kittler, M. Petrou, and M. Nixon, editors, *Proceedings of the 17th IAPR International Conference on Pattern Recognition*, pages 80–83, Los Alamitos, August 2004. IEEE, IEEE.

[8] M. Haindl and J. Filip. *Visual Texture*. Advances in Computer Vision and Pattern Recognition. Springer-Verlag, London, 2013.

[9] Michal Haindl and Jiri Filip. Advanced textural representation of materials appearance. In *SIGGRAPH Asia 2011 Courses*, SA '11, pages 1:1–1:84, New York, NY, USA, 2011. ACM.

[10] Jefferson Y. Han and Ken Perlin. Measuring bidirectional texture reflectance with a kaleidoscope. *ACM Trans. Graph.*, 22(3):741–748, July 2003.

[11] Felix Klein, Kristian Sons, Dmitri Rubinstein, and Philipp Slusallek. Xml3d and xflow: Combining declarative 3d for the web with generic data flows. *Computer Graphics and Applications, IEEE*, 33(5):38–47, 2013.

[12] Melissa L. Koudelka, Sebastian Magda, Peter N. Belhumeur, and David J. Kriegman. Acquisition, compression, and synthesis of bidirectional texture functions. In *In ICCV 03 Workshop on Texture Analysis and Synthesis*, 2003.

[13] Tom Malzbender, Tom Malzbender, Dan Gelb, Dan Gelb, Hans Wolters, and Hans Wolters. Polynomial texture maps. In *In Computer Graphics, SIGGRAPH 2001 Proceedings*, pages 519–528, 2001.

[14] Gero Müller, Jan Meseth, and Reinhard Klein. Compression and real-time rendering of measured btfs using local pca. In T. Ertl, B. Girod, G. Greiner, H. Niemann, H.-P. Seidel, E. Steinbach, and R. Westermann, editors, *Vision, Modeling and Visualisation 2003*, pages 271–280. Akademische Verlagsgesellschaft Aka GmbH, Berlin, November 2003.

[15] Gero Müller, Jan Meseth, Mirko Sattler, Ralf Sarlette, and Reinhard Klein. Acquisition, synthesis and rendering of bidirectional texture functions. In Christophe Schlick and Werner Purgathofer, editors, *Eurographics 2004, State of the Art Reports*, pages 69–94. INRIA and Eurographics Association, September 2004.

[16] Addy Ngan and Frdo Durand. Statistical Acquisition of Texture Appearance. pages 31–40.

[17] Hubert Nguyen. *GPU Gems 3*. Addison-Wesley Professional, August 2007.

[18] F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, and T. Limperis. Radiometry. chapter Geometrical Considerations and Nomenclature for Reflectance, pages 94–145. Jones and Bartlett Publishers, Inc., USA, 1992.

[19] Mirko Sattler, Ralf Sarlette, and Reinhard Klein. Efficient and realistic visualization of cloth. In *Eurographics Symposium on Rendering 2003*, June 2003.

[20] Christopher Schwartz, Roland Ruiters, Michael Weinmann, and Reinhard Klein. Webgl-based streaming and presentation of objects with bidirectional texture functions. *Journal on Computing and Cultural Heritage (JOCCH)*, 6(3):11:1–11:21, July 2013.

[21] Christopher Schwartz, Ralf Sarlette, Michael Weinmann, and Reinhard Klein. Dome ii: A parallelized btf acquisition system. In Holly Rushmeier and Reinhard Klein, editors, *Eurographics Workshop on Material Appearance Modeling: Issues and Acquisition*, pages 25–31. Eurographics Association, June 2013.

[22] Kristian Sons, Felix Klein, Dmitri Rubinstein, Sergiy Byelozyorov, and Philipp Slusallek. Xml3d: interactive 3d graphics for the web. In *Web3D '10: Proceedings of the 15th International Conference on Web 3D Technology*, pages 175–184, New York, NY, USA, 2010. ACM.

[23] Frank Suykens, Karl vom Berge, Ares Lagae, and Philip Dutr. Interactive rendering with bidirectional texture functions. *Comput. Graph. Forum*, 22(3):463–472, 2003.

[24] Chris Wynn. An introduction to brdf-based lighting. *NVIDIA Corporation*.