DTU Compute
Department of Applied Mathematics and Computer Science

Cloth rendering based on weave pattern textures

Master Thesis

Cloth rendering based on weave pattern textures

Master Thesis
Feb, 2023

By
Kengjian Lin

Cover photo: Vibeke Hempler, 2012
Published by: DTU, Department of Applied Mathematics and Computer Science, Brovej, Building 324, 2800 Kgs. Lyngby Denmark
https://www.compute.dtu.dk/
ISSN: [0000-0000] (electronic version)
ISBN: [000-00-0000-000-0] (electronic version)

ISSN: [0000-0000] (printed version)
ISBN: [000-00-0000-000-0] (printed version)

Approval

This thesis has been prepared over five months from August, 2022 to January , 2023 at DTU Compute, Department of Applied Mathematics and Computer Science, at the Technical University of Denmark, for 30 ECTS as part of the degree Master of Science in MSc in Computer Science and Engineering. It is assumed that the reader has knowledge within the areas of computer graphics and rendering.

Kengjian Lin - s202541

.....

Signature

.....

Date

Abstract

There is a growing demand for better rendering quality of cloth for various applications in the industry such as entertainment and fashion design. Traditional texture mapping techniques to represent the detail of cloth geometry require an intensive amount of manual labor from the artists. Focusing on weave pattern and representation of yarn geometry, we investigate methods of yarn modeling to reconstruct the cloth geometry and yarn-light scattering to get a physical-based shading model. We develop a procedural generation approach to synthesize color maps and bump maps from weave patterns. We also implement a shader and integrate the texture maps to improve the quality of the rendering. Our rendering pipeline can produce physically plausible results and its efficiency is suitable for real-time applications and rapid prototyping. Our rendering pipeline is flexible and supports a wide range of weave patterns and various cloth materials to satisfy the artists needs.

Keywords: Cloth rendering, procedural texture synthesis.

Acknowledgements

I am grateful to my supervisors Jeppe Revall Frisvad and Jakob Andreas Bærentzen for their vision, inspiration and guidance.
I'd like to thank my parents and friends for giving me all the support to finish the thesis.

Contents

1 Introduction

Cloth is an important material in daily life and the emergence of digitization also raised the need to transform cloth into the digital world. Modeling cloth appearance has been an active research topic in computer graphics for decades. Simulating the accurate appearance of cloth is challenging and computationally heavy because the cloth has a complex micro-geometry and spatial varying reflectance.

Even though there has been progress in this area with recently proposed new methods and shading models for cloth, many of these are still not applied in practice due to their inherent complexities. In many digital production environments, a common approach for rendering cloth is to use a general-purpose shading model along with textures created by artists. Although this strategy can produce realistic results, it has a few drawbacks. For instance, the quality of rendering depends on the quality of the texture map created by the artist or is limited by the resolution of a real photo. To avoid manual labor of artists or the memory consumption of high-resolution textures, automation of texture generation is considered a more efficient solution.

On the other hand, there are existing industry formats widely used for weave patterns when it comes to textile production. The question is how to generate digital textures for these weave patterns used in industry. However, there is a lack of tools for selecting cloth weave patterns as input to produce rendered textures. Therefore, it is desirable to build a rendering pipeline that incorporates modern shading models, which will allow artists to use existing weave patterns from the industry without having to resort to time-consuming manual labor.

1.1 Problem statement

There is a lack of tools that are capable of synthesizing yarn geometrical structure data from a weave pattern. Subsequently, there have been no attempts to demonstrate that the synthesized texture maps can be used as input for a recent cloth shading model. We list the goals of this thesis as follows:

- Primary goal is to explore a way to analyze weave pattern and use it for texture map synthesis.
- Secondary goal is to validate the effectiveness of synthetic texture maps. We will further implement a cloth shading model, integrate it with the synthetic texture and evaluate the rendering quality.
- Last goal is an optimization for level of detail(LoD). Geometrical data stored in texture maps usually contain high-frequency signal. Thus, there is a need to explore prefiltering techniques to improve the quality of rendering.

1.2 Thesis structure

The outline of the thesis is as follows:

- In chapter 2, we give a summary of prior work relevant to cloth rendering
- In chapter 3, we give a more in-depth explanation about the background and theory in prior work.

- In chapter 4. we integrate some of the theories and techniques from prior work and adapt them into the pipeline.
- In chapter 5, we discuss the workflow and software dependencies.
- In chapter 6, we present the rendering results and performance statistics.
- In chapter 7, we evaluate and discuss the results.
- In chapter 8, we end with the conclusion and discuss future work.

2 Prior work

Focusing on cloth rendering, the book ”Real time rendering” [1] is a starting point that provides a good summary of the existing cloth shading models. However, having only a cloth shading model is generally not enough to get an accurate rendering result. Castillo et al. [2] provided a comprehensive survey on the rendering of fibrous materials. Their work covers many existing techniques involving different crucial aspects of recreating the appearance of fabric, including capturing, modeling, shading and filtering techniques. It serves as a clear guideline for cloth rendering.

Fabric geometry representation
There are 3 main approaches for the representation of fabric geometry:

- Explicit geometry. A massive amount of fiber strands is used to describe a mesh. The work in Zhu et al. [3] achieved a realistic result with this.
- Volumetric models. Fabric is treated as a participating medium with a set of statistical properties. Recent research used CT scans to reconstruct the volumetric representation of fabric. Khungurn et al. [4] presented results of impressive quality.
- Surface-based models. The fabric surface is simplified and represented as a thin sheet. Although this simplification makes it neither comparable to explicit representation nor volumetric representation, surface-based models are light-weight and suitable for polygon meshes that are used extensively in graphics.

The surface-based approach is the cheapest solution and is sufficient to satisfy the need of many production environments. To model fiber geometry, Marschner et al. [5] investigated the light scattering behaviour in human hair and used a rough dielectric cylinders model for fabric representation. Taking into account the similarity between human hair and cloth fiber, Irawan et al. [6] built on the previous model of the cylinder and described a yarn thread as a bent cylinder with coaxial helix fiber.

Rendering
Modeling cloth geometry usually interrelates with the shading algorithm that describes light scattering from the fibers. For volumetric representation, a common practice is using volume ray marching. The basic idea is to accumulate radiance approaching the eye along the rays. There are 2 approaches for surface-based representation:

- The image-based approach that uses a bidirectional texture function (BTF) was first described in the work of Dana et al. [7]. Many visual effects related to fiber geometry such as occlusion, self-shadowing, subsurface scattering are baked into the texture. BTF is data driven which means it can only render the sampled texture once it has been captured and stored in the BTF database. It lacks the flexibility to predict a vast range of cloth structures beyond the sampled data in the BTF database.
- A bidirectional reflectance distribution function (BRDF) is used to integrate the geometry model and light scattering. Based on microfacet theory, the cloth surface is modeled as a collection of small microfacets when viewing at a distance. Only a subset of BRDF models are suited for representing the complex anisotropic reflectance of cloth. The hair fiber shading model from the work of Marschner et al. [5] had an impact on subsequent research.

Understanding light scattering from fibers is essential when physically-based cloth rendering approaches are developed. Combining light scattering and the fabric micro-geometry,

a full representation of a cloth material is fundamental for realistic cloth rendering. Sadeghi et al. [8] presented a yarn-level shading model for woven cloth, and their results are validated against optical measurement. More recent shading models [9] have better rendering results, but are more computationally heavy and have intensive storage requirements.

Level of Detail (LoD)
The fine level detail on the cloth surface usually requires a high frequency signal for accurate representation. A common practice is to use high resolution texture maps such as a displacement map or normal map. Aliasing will occur when viewing from a distance if the high-frequency signal becomes indistinguishable. Normal map pre-filtering techniques such as LEAN mapping [10] might be a generic solution to provide LoD.

Weave pattern and application
Adabala et al. [11] were the first to explore the extraction of a color map from arbitrary woven patterns based on a standard weave information file format (WIF). Later, Irawan et al. [6] developed an analytical model by using a cylindrical model to represent the geometry of the yarn in the weave pattern, and they also developed a new shading model along with geometric modeling. Specific for industry production, Nelson et al. [12] developed an application to predict the appearance of cloth with a given weave pattern based on the Irawan model. On the other hand, for production of graphics in animation, Smith et al. [13] developed a tool to procedurally generate highly customizable patterns of woven fabrics on a mesh. The applications in industry mentioned above have demonstrated that there is great commercial value. Moreover, there are image-based approaches that use images of real fabric to analyze the underlying weave pattern. Schroder et al. [14] demonstrate a practical solution for estimating the yarn parameters and reconstructing yarn structure from images.

3 Theory

3.1 Cloth geometry

In reality, clothes can be made of many materials such as cotton, silk, fur, and leather, each of which has its own unique characteristics. The most common type of cloth produced in the industry is woven fabric, manufactured in a process known as weaving. Yarns are spun onto a device called a loom where it is weaved and interlaced to form the cloth. Yarns can be used to make structures that range from simple to highly complex depending on the weave pattern and weaving technique.

If we observe cloth from a distance only its macro geometry is visible, approximating a rough surface. Taking a closer look makes each yarn discernible and the thread alignment becomes apparent. Going down to the finest level allows us to see the individual fibers on top of yarns. A direct digital representation of cloth without compression includes all levels of detail and usually requires a large amount of data, which makes it too heavy for many production environments. Yarn-level representation could be enough if the cloth is not observed in close-up. This work aims to render results on yarn-level detail, which should give the observer a sense of plausibility from a medium to far viewing distance.

To satisfy the need in industry a light-weight solution like texture mapping is usually more appealing than a method that uses massive volumetric data representation. Therefore, one of our interest is to find a solution that can extract geometric information from weave pattern. More specifically, we are trying to find a way that can synthesise textures and can be exported for rendering.

In the following subsections, we will introduce the weave pattern in section 3.1.1, and relate the yarn cylinder model from the work of Irawan et al. [6] to the weave pattern. Finally we explain the process of texture maps synthesis from weave pattern in section 4.1.

3.1.1 Weave pattern

Weave pattern is an important factor in determining the overall look of cloth. The Weaving Information File (WIF) is a standard file format for weaving drafts [15].

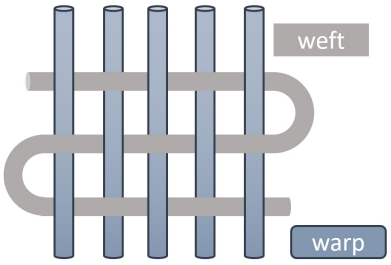


Figure 3.1: Warp (vertical) and weft (horizontal) are interlaced, forming a weave pattern. Warp and weft threads are usually orthogonal to each other.

The weave pattern data can be represented by a 2-dimensional matrix, determining which type of yarn is visible from above at each crossing point. A schematic representation of different weave patterns can be seen in fig. 3.2.

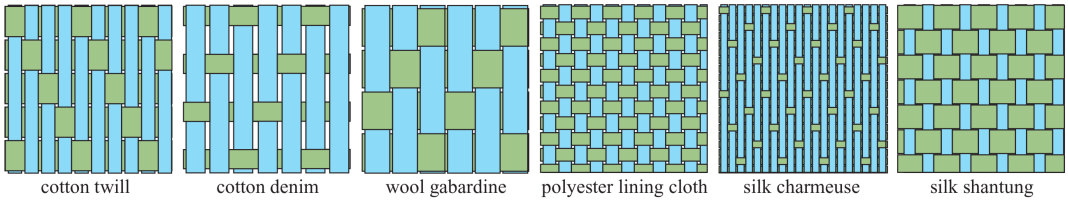


Figure 3.2: Common weave pattern for textiles. (Image from [6])

To reproduce the appearance of woven fabric in the digital world, the weave pattern is one of the deterministic factors that we should consider. A benefit of using WIF is making design in digital environments interchangeable to the look of the product in reality. The work of design can be done digitally and can be used for easy visualization. Based on this idea, Adabala et al. [11] demonstrated that it is feasible to extract diffuse textures from weave pattern files and use them for rendering. However, in their work they only considered a normal distribution function from microfacet theory to represent the micro-geometry. This implies they treated the cloth surface as a rough surface and neglected complex geometry of the weave pattern. The geometric structure of the weave pattern can play an important role in the rendering of the cloth.

3.1.2 Thread modeling

The reflection characteristics of a fabric have a strong correlation with its microgeometry and having a good geometric model that is as accurate as possible is crucial. Figure 3.3 shows the very complex micro structure of a fabric surface, and it is a challenging task to model such a fabric surface. Using data-driven approaches to model each yarn on a weave pattern can require remarkable storage and computational resources. For example, Zhao et al. [16] used a volumetric representation approach to reconstruct the yarn orientation field from a CT scan. In their work, the space to store the scan data was up to hundreds of MB, which is quite expensive.

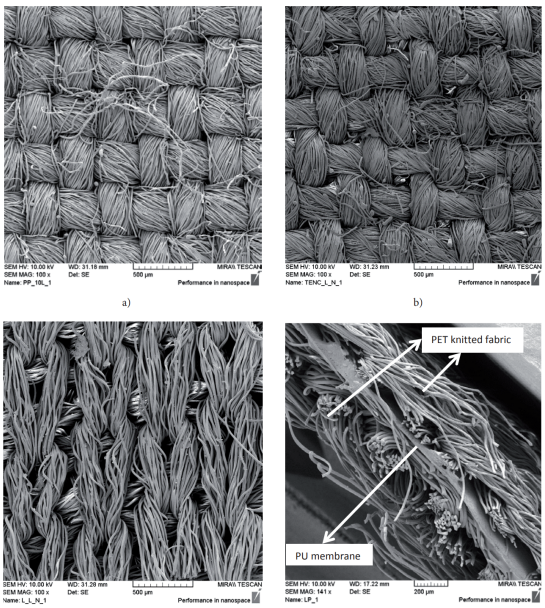


Figure 3.3: Microscopic image of the a fabric surface. Weave pattern can be seen as a grid of yarns. Each yarn is consist of numerous fibers that have irregular arrangement. (Image from [17])

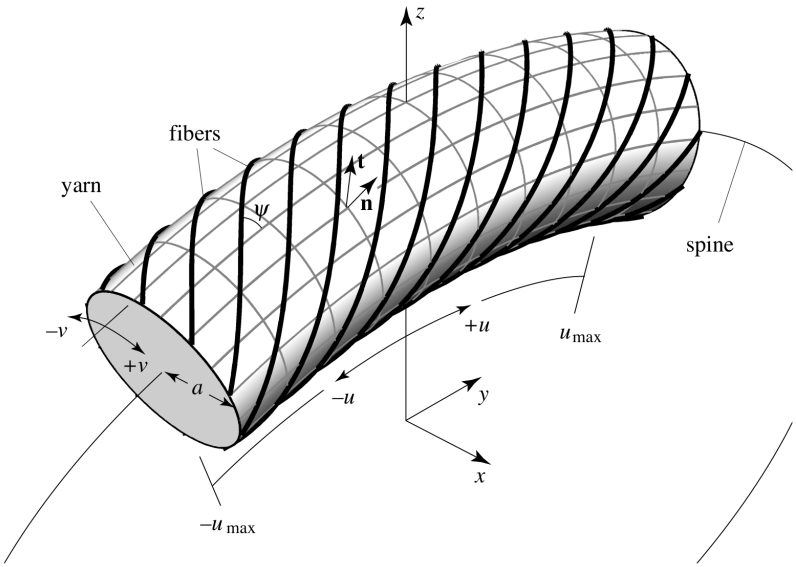


Figure 3.4: Microcylinder model. u_{max} is the maximum angle of inclination , $u \in [-u_{max}, u_{max}]$ describe the angle deviation from the center along the yarn longitudinal axis. $v \in [-\pi, \pi]$ describe the angle deviation from the center along the lateral axis of the yarn. $\psi \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ is the fiber twist angle. Image from [6]

Irawan et al. [6] developed an analytical model to model each yarn thread as a bent cylinder with a circular cross section. In their micro-cylinder model, a weave pattern can be seen as a collection of rectangular segments. Each of the rectangle segments represents a yarn thread and can be unwrapped as a bent cylinder. Conversely, the bent cylinder can be flattened to a rectangle, which implies that the projection of the cylinder fits into the rectangular area. The yarn cylinder model is described in a local coordinate system as shown in Figure 3.4 . The z axis align with the macro normal of the fabric surface, the y axis aligns with the longitudinal axis of the yarn spine and x completes the orthogonal basis of the right hand. The angular parameters u and v can be used to described a point's coordinates on the cylinder surface. The normal direction to the yarn surface and the tangent of the fiber are given as [6]:

$$\begin{aligned} n(u,v) = \begin{bmatrix} x \\ y \\ z \end{bmatrix} &= \begin{bmatrix} \sin(v) \\ \sin(u)\cos(v) \\ \cos(u)\sin(v) \end{bmatrix} \\ t(u,v,\psi) = \begin{bmatrix} x \\ y \\ z \end{bmatrix} &= \begin{bmatrix} -\cos(v)\sin(\psi) \\ \cos(u)\cos(\psi) + \sin(u)\sin(v)\sin(\psi) \\ -\sin(u)\cos(\psi) + \cos(u)\sin(v)\sin(\psi) \end{bmatrix} \end{aligned} \tag{3.1}$$

The Irawan model has well defined the normal and tangent in a yarn segment. We will explain the detail of how to utilize these fiber orientation vectors to generate texture maps in Section 4.1.

3.2 Light scattering model

The reflection characteristics of a fabric surface are described as a bidirectional reflectance distribution function (BRDF). Sadeghi et al. [8] developed a robust empirical shading model for woven cloth based on measurements of light scattering from individual threads. Their optical measurements were performed on different material yarn threads and are based on yarn-level modeling. The Sadeghi model is a far-field model, assuming that the width of the fiber is insignificant at the observation distance. This assumption is generally valid considering that most use cases of this work are viewing the cloth from a distance. A benefit of using a far-field model is avoiding aliasing, as it abstracts the complex geometry. Their BSDF is built on a classical model from Marschner et al. [5] and it inherits the same local coordinate system (see Figure 3.5). The notation of the fiber coordinate system is described in Table 3.1.

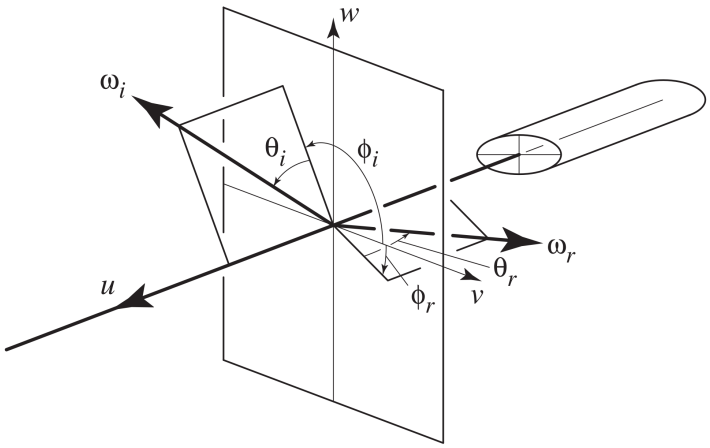


Figure 3.5: Light scattering coordinate system. For notation, see Table 3.1.
Image from [5]

Table 3.1: Symbols used for local coordinate system

Symbols	Description
u	fiber tangent direction
$v-w$	normal plane
ω_i	Light incident direction
ω_r	Light scatter direction
θ_i, θ_r	Longitudinal angle of ω_i, ω_r are computed with respect to normal plane. $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$
$\phi_i, \phi_r,$	Azimuthal angle of ω_i, ω_r in the normal plane. $\phi \in [0, 2\pi]$
θ_d	Longitudinal difference angle $\theta_d = (\theta_i - \theta_r)/2$
θ_h	Longitudinal half angle $\theta_h = (\theta_i + \theta_r)/2$
ϕ_d	Azimuthal difference angle $\phi_d = \theta_i - \theta_r$

It is easier to describe angles in a spherical coordinate system, but many of vector computations are done in a Cartesian coordinate system. Note that the fiber tangent u was chosen to align with the x axis in the coordinate space in Figure 3.5. The Cartesian coor-

dinates can be retrieved from the spherical coordinates by using:

$$\omega = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \sin(\theta) \\ \cos(\theta)\cos(\phi) \\ \cos(\theta)\sin(\phi) \end{bmatrix} \tag{3.2}$$

We can compute the Jacobian for a Cartesian to spherical coordinate transformation and the result is $\cos\theta$.

The radiance integral in Sadeghi's model is given as follows:

$$L_r = \int f_s(t, \omega_i, \omega_r) L_i(\omega_i) \cos(\theta_i) d\omega_i \tag{3.3}$$

Because their BSDF is measured and analyzed in spherical coordinates, it is more suitable to use a longitudinal-azimuthal angle (θ, ϕ) for modeling. Thus, we can rewrite eq. (3.3) as follows:

$$L_r = \int_0^{2\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} f_s(t, \theta_i, \theta_r, \phi_i, \phi_r) L_i(\theta_i, \phi_i) \frac{\cos^2(\theta_i)}{\cos^2(\theta_d)} d\theta_i d\phi_i \tag{3.4}$$

where f_s is the scattering function, L_i and L_r are the incoming radiance from direction ω_i and the outgoing radiance to direction ω_r, respectively. The extra $\cos\theta_i$ term comes from the jacobian transformation of the coordinate system. The term $\frac{1}{\cos^2\theta_d}$ is derived from Marschner's model to account for the solid angle attenuation of the specular cone. The scattering function f_s is separated into a surface reflectance component $f_{r,s}$ and a volume scattering component $f_{r,v}$. The notation of

$$\begin{aligned} f_s(t, \theta_i, \theta_r, \phi_i, \phi_r) &= f_{r,s}(t, \theta_i, \theta_r, \phi_i, \phi_r) + f_{r,v}(t, \theta_i, \theta_r, \phi_i, \phi_r) \\ f_{r,s}(t, \theta_i, \theta_r, \phi_i, \phi_r) &= F_r(\eta, \omega_i) \cos(\phi_d/2) g(\gamma_s, \theta_h) \\ f_{r,v}(t, \theta_i, \theta_r, \phi_i, \phi_r) &= F_t \frac{(1 - k_d) g(\gamma_v, \theta_h) + k_d}{\cos(\theta_i) + \cos(\theta_r)} A \end{aligned} \tag{3.5}$$

Table 3.2: Symbols used in the light scattering function eq. (3.5), from [8]

Symbols	Description
η	Index of refraction
F_r	Fresnel reflectance
F_t	Fresnel transmittance
g	Gaussian distribution
γ_s	Surface reflectance Gaussian width
γ_v	Volume scattering Gaussian width
k_d	Isotropic scattering coefficient
A	Colored albedo coefficient

$f_{r,s}$ is derived from the R lobe, whereas $f_{r,v}$ is derived from the TT lobe in the Marschner model. The $f_{r,s}$ term models light reflected directly off fiber surfaces on a cone around the thread tangent. The $f_{r,v}$ term models light transmitted forward through the fibers and forms a single forward scattering lobe (see Figure 3.6).

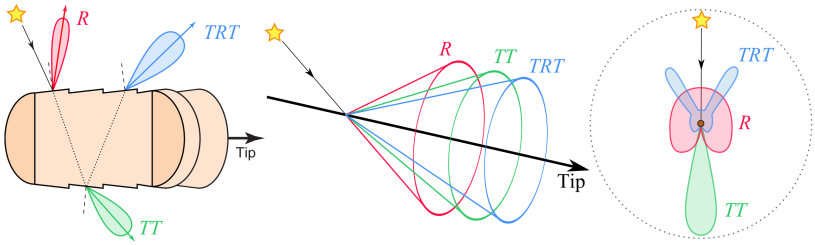


Figure 3.6: Light scattering behavior in fibers, (image from [18])

3.3 Importance sampling

Many ray tracers are built upon a Monte Carlo algorithm. One way to efficiently reduce variance in Monte Carlo integration is to draw samples from a distribution whose probability distribution function (PDF) is proportional to the rendering function we are integrating. A practical way is to use the inverse transform sampling method by calculating the cumulative distribution function (CDF) of the distribution and then inverting that function.

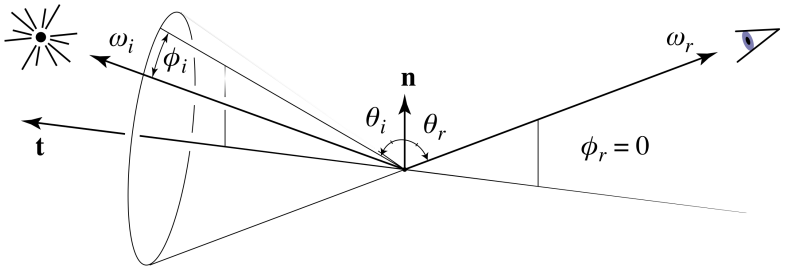


Figure 3.7: Geometry of specular reflectance, image from [6]

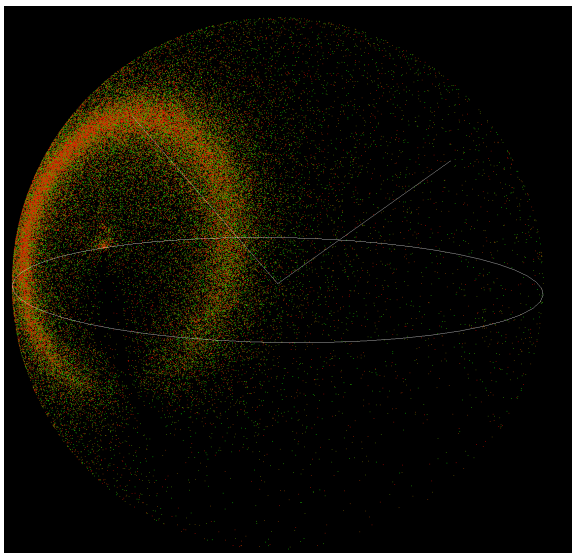
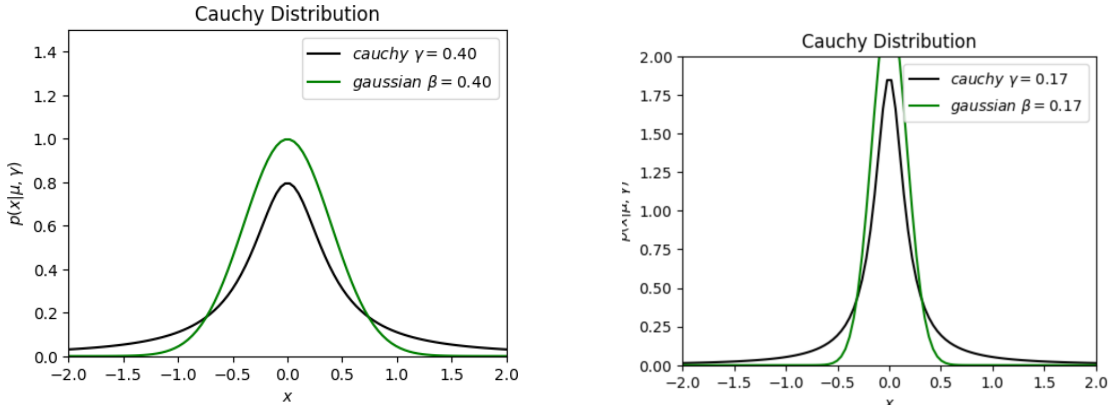


Figure 3.8: Importance sampling of a cone

Taking the cone geometry into account, we should draw more samples within the cone

over the sphere, because samples within the cone have a higher contribution to Monte Carlo integration. (Figure 3.8). The Gaussian distribution function used in Equation (3.5) cannot be analytically integrated and inverted; therefore, we need to find another way to overcome this problem. A different approach to sample the Gaussian distribution function is to use a method called the Box-Muller transform [19]. There are studies performing importance sampling for fabric rendering based on the Box-Muller transform method [20]. Taking this one step further, it is also possible to approximate the Gaussian distribution function using a different Gaussian-like distribution function. The reason is the other Gaussian-like distributions may have an antiderivative form of its CDF, allowing us to apply the inversion method. Wang et al. [21] proposed a practical importance sampling strategy, which replaces the Gaussian distribution function with a Cauchy distribution function.



(a) Longitudinal width (standard deviation): **0.4(radiance)**

(b) Longitudinal width (standard deviation): **0.17(radiance)**

Figure 3.9: Comparison between Cauchy distribution and Gaussian distribution using cloth measurement data.
With an offset(mean) set to 0, (a) approximates 24° and (b) approximates 10° used in Sadeghi's cloth measurement for describing surface roughness.

The Gaussian distribution has a taller central peak while the Cauchy distribution is flatter in the center and heavier in the tails. Therefore, minor numerical differences between the Gaussian distribution and the Cauchy distribution will cause a difference in the visual appearance of a rendered image. The Gaussian central peak will contribute more energy and it will look much brighter when the observation direction is near the perfect reflectance angle. On the other hand, the Cauchy distribution gives us the benefit of using the simpler inversion transform sampling technique. Beyond cloth shading, a Gaussian-like distribution logistic distribution used for fur rendering can also be used with the inverse method [22]. A re-parameterized logistic distribution can also closely resemble a Gaussian distribution function. The choice of Cauchy distribution in this work is only considered from the perspective of performance. The convenience in sampling will make the results converge faster and reduce noise. Replacement of the Gaussian distribution lacks support from physical measurement data, and there are no studies comparing which of the Gaussian-like distributions numerically approximates the appearance of cloth in reality most well. The original Gaussian distribution might be a better choice to match the original measurement data if the application has a stricter quantitative assessment of the resulting renderings.

The definition of the Cauchy distribution function is as follows:

$$f(\gamma, x - x_0) = \frac{1}{\pi} \frac{\gamma}{(x - x_0)^2 + \gamma^2} \tag{3.6}$$

and the Cauchy distribution has an analytic antiderivative form:

$$F(x) = \frac{1}{\pi} \tan^{-1}(\frac{x - x_0}{\gamma}) \tag{3.7}$$

The surface reflectance component f_s and the volume scattering component f_v are sampled separately. Because the longitudinal angle θ in eq. (3.5) is independent of azimuth, we first sample θ and then sample the azimuthal angle ϕ.

Given the direction of observation ω_r, we transform the direction into a spherical coordinate representation (θ_r, ϕ_r). Then we sample θ_h and ϕ_d, to compute θ_i and ϕ_i. The goal is to get the incident angle (θ_i, ϕ_i) and compute the corresponding probability density function (***pdf***) $p(\theta_i)$ and $p(\phi_i)$. With (θ_i, ϕ_i) we can transform back to a Cartesian coordinate ω_i with eq. (3.2). The ***pdf*** of the sampled direction ω_i is $p(\omega_i) = p(\theta_i)p(\phi_i)$.

The calculation of ***pdf***, sampling of θ_i and ϕ_i in the folloing parts is from Wang et al. [21].

Sampling surface reflectance $f_{r,s}$
Given a uniform random variable $\xi \in [0, 1)$, we want sample θ_i proportional to the Cauchy distribution:

$$p(\theta_i) \propto \frac{\gamma_s}{\theta_h^2 + \gamma_s^2} \frac{1}{\cos(\theta_i)}$$

The integral should be normalized to 1 as following:

$$\begin{aligned} c \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \frac{\gamma_s}{(\frac{\theta_i + \theta_r}{2})^2 + \gamma_s^2} \frac{1}{\cos(\theta_i)} \cos(\theta_i) d\theta_i = 1 \\ 2c \tan^{-1}(\frac{\theta_i}{\gamma_s}) \Big|_{\frac{\theta_r - \pi/2}{2}}^{\frac{\theta_r + \pi/2}{2}} = 1 \end{aligned} \tag{3.8}$$

Let $A = \tan^{-1}(\frac{\theta_r + \pi/2}{2\gamma_s})$ and $B = \tan^{-1}(\frac{\theta_r - \pi/2}{2\gamma_s})$, therefore $c = \frac{1}{2(A-B)}$. The ***pdf*** of the longitudinal angle is $p(\theta_i) = \frac{1}{2\cos(\theta_i)(A-B)} \frac{\gamma_s}{\theta_h^2 + \gamma_s^2}$. The ***cdf*** can be computed by integrating the ***pdf***:

$$P(\theta_i) = \int_{-\frac{\pi}{2}}^{\theta_i} p(\theta_i) \cos(\theta_i) d\theta_i = \frac{\tan^{-1}(\frac{\frac{\theta_i + \theta_r}{2}}{\gamma_s}) - B}{A - B} \tag{3.9}$$

The inversion of ***cdf*** gives:

$$\theta_i = 2\gamma_s \tan(\xi(A - B) + B) - \theta_r \tag{3.10}$$

To sample the azimuthal part ϕ:

$$p(\phi) \propto \cos(\frac{\phi}{2})$$

$$c \int_{-\pi}^{\pi} \cos(\frac{\phi}{2}) d\phi = 1 \tag{3.11}$$

we can compute $c = \frac{1}{4}$ and $p(\phi) = \frac{1}{4}\cos(\frac{\phi}{2})$. The *cdf* can be computed by integrating the *pdf*.

$$\int_{-\pi}^{\phi} \frac{1}{4}\cos(\frac{\phi}{2}) = \frac{1}{2}\sin(\frac{\phi}{2} + 1) \tag{3.12}$$

The inversion of *cdf* gives $\phi_d = 2\sin^{-1}(2\xi - 1)$, $\phi_i = \phi_r - \phi_d$

Sampling volume scatter $f_{r,v}$

The $f_{r,v}$ sampling can be divided into two cases. If the scattering is anisotropic, then we sample it in the same way as in $f_{r,s}$. Otherwise, if it is the isotropic, we use a cosine-weighted distribution to sample. This gives the following:

$$\theta_i = \begin{cases} 2\gamma_v \tan(\xi(A-B)+B) - \theta_r & \xi \leq (1-k_d) \\ \sin^{-1}(2\xi - 1) & otherwise \end{cases} \tag{3.13}$$

where $A = \tan^{-1}(\frac{\theta_r + \pi/2}{2\gamma_v})$ and $B = \tan^{-1}(\frac{\theta_r - \pi/2}{2\gamma_v})$. The combined *pdf* is $p(\theta_i) = (1 - k_d)\frac{1}{2\cos(\theta_i)(A-B)}\frac{\gamma_v}{\theta_h^2 + \gamma_v^2} + k_d\frac{cos(\theta_i)}{2}$

3.4 Multiscale representation

Moiré pattern phenomena can be observed in reality, especially with highly anisotropic materials such as silk. When we use a digital camera to capture images of very glossy fabric, we may get a moiré pattern image. Because we mainly use texture maps to represent the cloth surface geometry, we put our focus on texture map filtering and related antialiasing solutions. There are different techniques for antialiasing, and they depend on our implementation for rendering. First, a ray tracer can render an image with more samples per pixel to get a more accurate and less noisy result. Increasing the number of samples also means yielding a higher performance cost, which is an important factor to consider. Texture filtering and multi-sampling are employed in a rasterization pipeline.

3.4.1 Problem description

When viewing from a distance, many texels from texture maps squeeze into one pixel on the screen. Sampling the diffuse texture map is an easy task using traditional filtering techniques, such as bilinear filtering. Final shading is linearly correlated to the texture color and linear filtering will not cause problems. On the other hand, the normal and tangent information used to represent the microgeometry is also used in the lighting calculation. Final shading has a non-linear correlation with the normal in lighting calculation when calculating angles. Using the same linear filtering strategy will cause the final shading to become inaccurate. Our rendering approach is highly dependent on the normal and tangent maps extracted from the weave pattern to represent geometry details. To address this problem, we have to look for an alternative textile filtering solution. Bruneton et al. [23] provides a comprehensive survey of the existing techniques for pre-filtering complex surfaces.

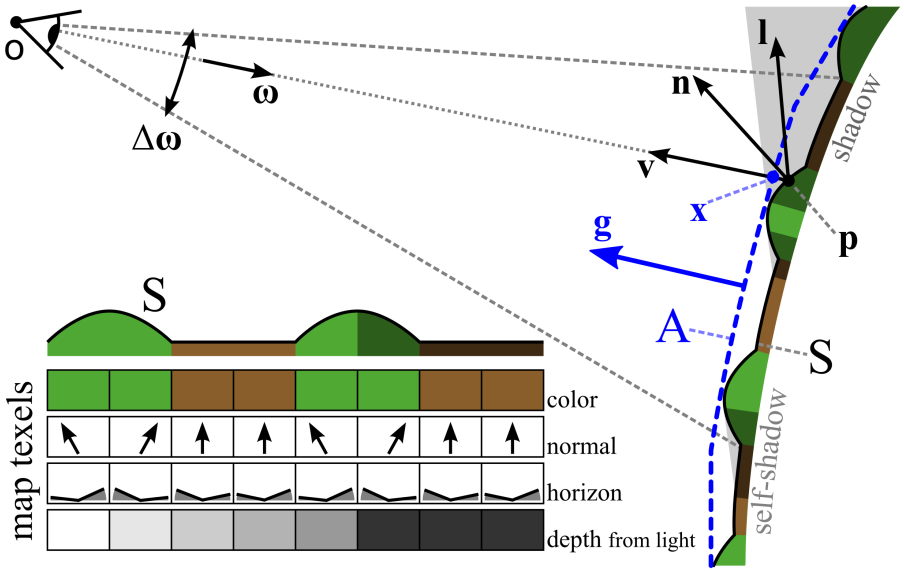


Figure 3.10: A pixel cover a complex surface and multiple texels. Accurate rendering requires proper weighted sampling over the texels. Image from [23]

In their survey, it was pointed out that a normal map cannot be directly pre-filtered, and using a linear combination of microgeometry and normal orientation to represent a larger-scale normal is inaccurate, as shown in Figure 3.11. Instead, using a normal distribution function (NDF) to describe the micro-geometry normals is more appropriate. The filtering problem is transformed into a problem of finding a good NDF to approximate the underlying normal orientation. The normal map filtering techniques in the survey can be divided into two groups, to use either a single lobe to represent the NDF, or to use multiple lobes.

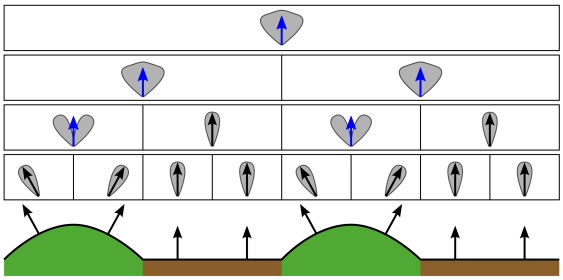


Figure 3.11: An illustration of a normal filtering strategy. Linear filtering is inaccurate(blue arrow). An alternative approach is to pre-filter normal distribution functions(gray lobes). The problem is then to find an accurate and linearly interpolable NDF representation. Image from [23].

3.4.2 Existing methods

LEAN mapping [24] and LEADR mapping [25] are popular methods when using a single lobe to represent the NDF. However, the limitations of LEAN mapping make it incompatible with our strategy. LEAN mapping works very effectively and efficiently for Beckmann distribution-based shading models, but it can only capture one direction of anistropy. It has difficulty capturing multiple directions of anistropy aligned to the threads of woven materials. These models assume that normals of the input surfaces follow a Beckmann distribution, but this does not fit into the cloth cylinder model assumption.

Han et al. [10] developed a different normal map filtering technique, which uses multiple lobes to represent the NDF. This method does not enforce any restrictions on the geometry of the input mesh surface, which is more suitable when representing a wider range of complex surfaces such as fabric. Han et al. [10] proposed that normal map filtering can be formalized as a spherical convolution of the NDF and BRDF. A normal in a texel in the normal map can be approximated as an NDF via a spherical expectation maximization algorithm. The estimated NDF is represented as multiple von Mises distribution (vMF) lobes, and the parameters of vMF lobes are stored in another texture map that is capable of supporting hardware mip mapping. Wu et al. [26] further extend this prefiltering method. Texture maps and BRDF are downsampled jointly to construct a spatially varying BRDF at reduced resolution to form a mipmap. To summarize, the goal of these approaches is to simplify the complex surface details in geometry and adjust the resulting BRDFs to keep the overall appearance unchanged. These prefiltering techniques are quite effective for a large class of common BRDFs such as Blinn-Phong and Torrance-Sparrow, and these BRDFs are usually single-lobe. However, there is a limitation in these strategies; they only use a half-vector or normal vector for lighting computation, and the tangent direction for the local shading frame is neglected. We want to test these prefiltering techniques and validate the effectiveness of filtering to determine whether they are compatible with our rendering pipeline.

4 Method

In this chapter, we will explain about the process of texture maps synthesis in Section 4.1. Then we mention the modification to the shading model to adapt the pipeline in Section 4.2. Finally we mention the attempt to do multiscale representation in Section 4.3.

4.1 Texture maps synthesis

The modelling of yarn thread as a cylinder is demonstrated in Section 3.1.2. The next step is to relate the indexing of the weave pattern matrix to the coordinates on a rectangle segment (Figure 4.1) and the geometric data described in the cylinder model.

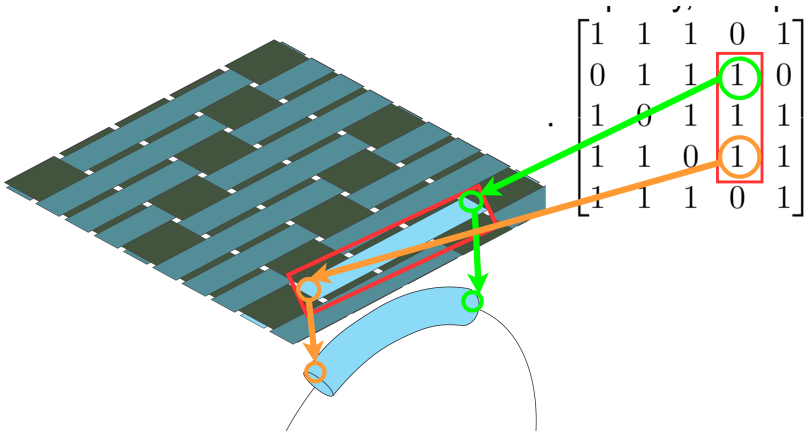


Figure 4.1: A patch of weave pattern. The red box area in the matrix corresponds to the highlighted yarn. The green and orange arrow representing the mapping relationship. Green and orange circles intend to mark middle of two end of the yarn(Image from [6])

In reality, the curvature of the spine can be affected by many factors such as the material of the yarn and the yarn interwoven tension. These variables can make the yarn more curved at the center or the end of the yarn. In the Irawan model, the curvature of the yarn spine is described as different types of conic sections and is controlled by a series of empirical parameters. For the sake of simplicity, the spine is modeled as a segment of a circle as a starting point for this project. Now, we have a simplified version of Irawan cylindrical model, and its geometry is controlled by parameter u_{max} and ψ. In the Irawan model, all the fibers that twist around the yarn thread are parallel to each other. In eq. (3.1), ψ is a user-defined variable to control the fiber arrangement, but it is a fixed value in the original model. However, this will not capture the irregularity of the fiber arrangement as seen in Figure 3.3. Heuristically, ψ can be modeled with some Gaussian-like functions to add randomness to displace fibers and change their arrangement, representing the irregularity. We let $\psi \sim \mathcal{N}(\mu, \sigma^2)$, μ be a fibers twist angle on average, and σ be the added randomness.

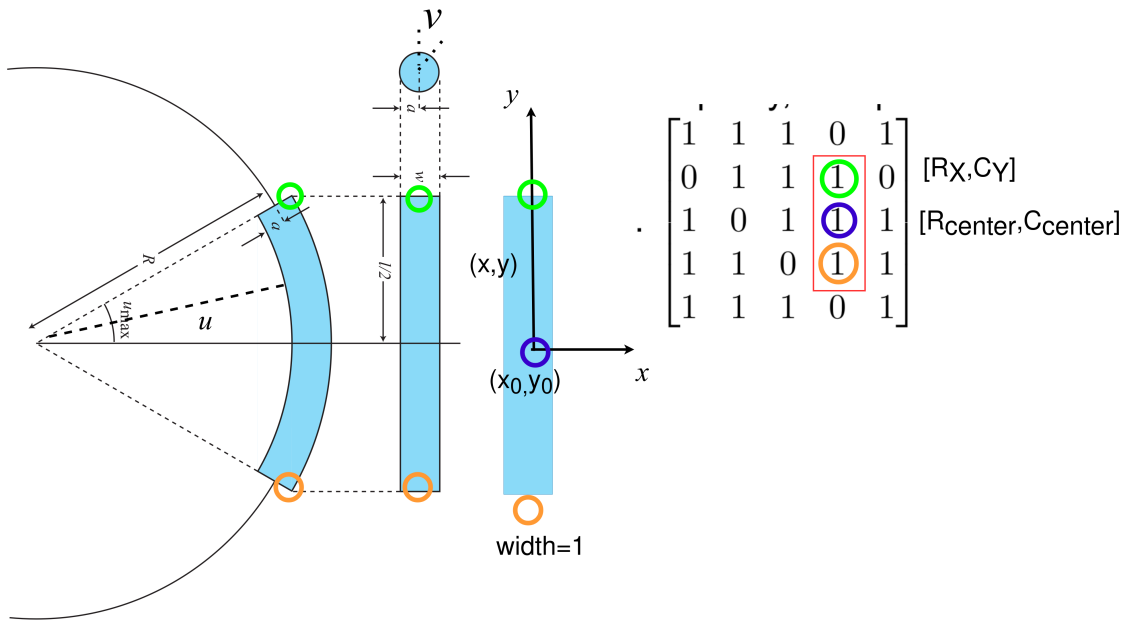


Figure 4.2: Local coordinate mapping. u_{max} is maximum inclination angle. w, l represents width and length of the rectangular segment of the yarn. A local 2-D coordinate system is used in the rectangle segment and (x_0, y_0) denotes the center. Green and orange circles. (R_{center}, C_{center}) represents the index of the center of a yarn segment.

Analyzing the projection of a cylinder model on the rectangle segment, this gives the following:

$$\begin{aligned} \sin(v) &= x - x_0 \\ \sin(u) &= y - y_0 \end{aligned} \tag{4.1}$$

When u becomes u_{max}, or $v = \frac{\pi}{2}$, this gives:

$$\begin{aligned} \sin(\frac{\pi}{2}) &= \frac{w}{2} \\ \sin(u_{max}) &= \frac{l}{2} \end{aligned} \tag{4.2}$$

Putting eq. (4.2) and eq. (4.1) together, now we have the mapping from the rectangle segment to yarn surface coordinate.

$$\begin{aligned} \frac{\sin(u_{max})}{\sin(u)} &= \frac{\frac{l}{2}}{y - y_0} \\ \frac{1}{\sin(v)} &= \frac{\frac{w}{2}}{x - x_0} \\ u &= \sin^{-1}(2\frac{y - y_0}{l}\sin(u_{max})) \\ v &= \sin^{-1}(2\frac{x - x_0}{w}) \end{aligned} \tag{4.3}$$

By parsing the data in the weave pattern matrix, we can know that the index (R_{center}, C_{center}) is the center of a rectangle segment (Figure 4.2). We can also count the number of rows

and columns in the matrix to get the aspect ratio. Given an index R_x, C_y belonging to the same rectangle segment as (R_{center}, C_{center}), it can be linearly mapped to the rectangle segment coordinates. This being said, we can use an index in the matrix to compute the u, v parameters by using the mapping mentioned above. Finally, the texture coordinates can be linearly mapped to the indices of an element in the matrix. With the calculated u, v variables from the yarn surface coordinate system, we can further compute the corresponding tangent and normal. We use eq. (3.1) to calculate the corresponding normal and tangent vectors for each element in the weave pattern matrix. Note that the normal and tangent vectors we compute are in tangent space because they are defined in a local coordinate system. Typically, the normal vector in tangent space can be stored in RGB channel in one pixel. In the later rendering stage, we will use the conventional normal mapping technique to retrieve normal vectors from the normal map to do lighting calculations.

Note that apart from the normal map, a tangent map is also generated to be used later for the shading algorithm, as for many cloth anisotropic shading models you need to specify the fiber tangent direction to construct a local shading frame. Using an arbitrary tangent direction and yarn surface normal may not describe the fiber arrangement well. This arrangement contributes to the overall light calculation and should not be neglected.

4.1.1 Challenges

We model the cloth as a thin surface with a single layer of fabric. We can change physical parameters such as the maximum angle of inclination u_{max} and the twist angle of the fibers ψ to model different types of weave patterns. One common limitation of surface based modelling is that it neglects the thickness of the weave pattern. A thicker textile surface usually absorbs more light during transmission through the surface and this causes it to have a darker look. However, we do not have any parameters to control the thickness characteristic. Another issue could be the loss of precision during the export of texture generation. The mathematical model to describe the surface orientation is continuous in the Irawan model; however, the data of normal and tangent vectors become discrete after being stored in a texture. Loss of precision can be mitigated by interpolation during sampling. In general, it is not recommended to set the resolution too low to avoid this precision problem.

4.2 BSDF adaptation

We use a large part of Sadeghi's BSDF model as our shading model, but there are two things that need to be changed to fit our needs. First is to replace the Gaussian distribution for sampling, The second is yarn tangent distribution.

As discussed in Section 3.3, the Gaussian distribution term g used in Equation (3.5) is less friendly for importance sampling. Therefore, in order to follow the importance sampling scheme, it is decided to replace all the Gaussian distribution occur in the BSDF in Equation (3.5) with a Cauchy distribution using the same offset as parameters.

The Sadeghi BSDF model only describes the light scattering on the individual yarn thread. For weave pattern, as a larger scale to a yarn, they evaluate the outgoing radiance from a smallest patch of a weave pattern and weigh their radiance contribution based on yarn length. Figure 4.3. One benefit of using controlling points is that it gives greater flexibility to control yarn curvature than using a simple mathematical curve. In practice, for a user who has little knowledge of fabric, using a series of control points to infer yarn curvature is neither intuitive nor easy. Using controlling points from empirical experience is not better than using a mathematically parameterized curve. Last but no least, manually editing a

set of control points might be easy for a simple weave patterns, but it is impractical to set a groups of control points for each arbitrary weave pattern.

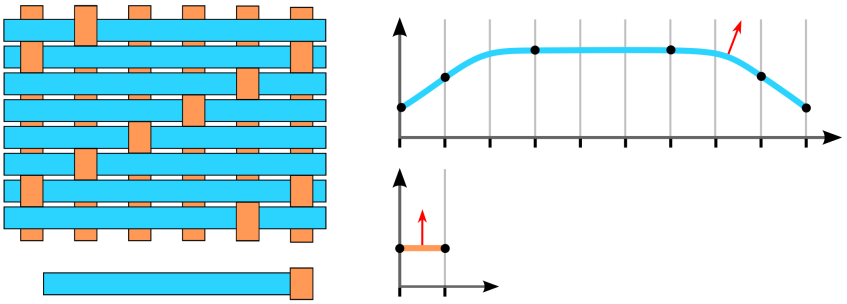


Figure 4.3: A smallest patch of weave pattern at the bottom left. On the right is tangent curve with control points that describe the angle of bending. (Image from [8])

To avoid the manual specified control points that brings a huge learning cost to users , we sample the tangent direction directly from the synthetic textures acquired from Irawan's model to achieve the goal of tangent sampling in a similar manner.

4.3 Pre-filtering Strategy

It is decided to adopt Han et al. [10]'s approach to formulate our prefiltering strategy, considering that a multilobe NDF approximation could be more suited for a complex surface. Han et al. [10] purpose to solve the pre-filtering problem as a convolutional problem to compute a prefiltered BRDF ρ^{eff}. In the case where the base BRDF is a microfacet model, a microfacet normal distribution for ρ^{eff} can be estimated by convolving the microfacet distribution of the base BRDF with the normal distribution.

$$\rho^{eff}(\omega_i, \omega_o; D) = \int_{S^2} \rho(R_\omega(\omega_i), R_\omega(\omega_o))D(\omega)d\omega \tag{4.4}$$

where S^2 denotes the integral on the sphere, R_ω is a rotation operator that rotates the world space ω_i and ω_o to a local shading frame for lighting calculation. D represents the normal distribution function and can be approximated by a mixture of the von Mises-Fisher distribution (vMF) denoted as γ:

$$D(\omega) \approx \sum_{i=1}^{m} \alpha_i \gamma(\omega; \kappa_i, \mu_i)$$

where α_i represents the amplitude of the vMF lobe. Each vMF lobe has a central direction μ and width κ.

We break down the pre-filtering strategy into two steps. The first step is to convert the normal map to its downsampled NDF representation. It is achieved using the EM algorithm mentioned in the work of Han et al. [10]. The EM algorithm takes a high resolution normal map as input, the output is a low resolution texture store vMF parameters α, κ, μ and we call it vMF maps.

The second step is rendering with vMF maps. In vMF maps, it store μ as the central direction and κ as the width of vMF lobe. It requires one extra step to sample a micro normal direction from a vMF lobe representation. Sampling a vMF distribution may have

severe numerical problems but there is a practical solution purposed by Jakob [27]. We then use the sampled normal for lighting calculation.

4.3.1 Challenges

With this normal filtering technique, it is not sufficient for multiscale representation. Note that our model needs both normal and tangent inputs to reconstruct the fabric geometry. The remaining tangent is not filtered. In this case, we also want to test if it is possible that the tangent can be replaced by a polygon mesh tangent for lighting calculation at coarse scale. Moreover, the normal and tangent are orthogonal before filtering; neglecting their correlation could cause the basis of the shading frame to not be orthogonal, consequently making the result inaccurate. We summarize the challenges of multiscale representation as follows:

1. Base BSDF has a multi-lobe representation, including surface reflectance and volumetric transmittance.
2. Base BSDF rely on a local shading frame for lighting computation, and the local shading frame needs to know the fiber tangent direction.

Note that we use a procedural generation approach to synthesize a normal map and we can specify the output texture map resolution. In this way, we can reduce the output resolution by a factor of 2 to form a mipmap during texture map synthesis. However, different rendering software has different implementations for texture mapping. In most scenarios, it is up to the renderer to decide how to generate mip-map. Therefore, it is impractical and not flexible to form a mipmap in this approach. In an extreme case, to consider a 2×2 plain weave pattern, we may use very low-resolution textures to represent a simple pattern. Arguably, it may not be ideal to mip-map if the resolution is already low enough. Filtering fabrics is challenging because of their inherent anisotropy due to the different multi-scale structures that form them.

5 Implementation

The workflow of this project is broken down into the following parts:

1. Raw data acquisition
2. Weave pattern texture synthesis
3. Shader implementation
4. Scene setup and rendering configuration

An overview of the general workflow can be seen in Figure 5.1

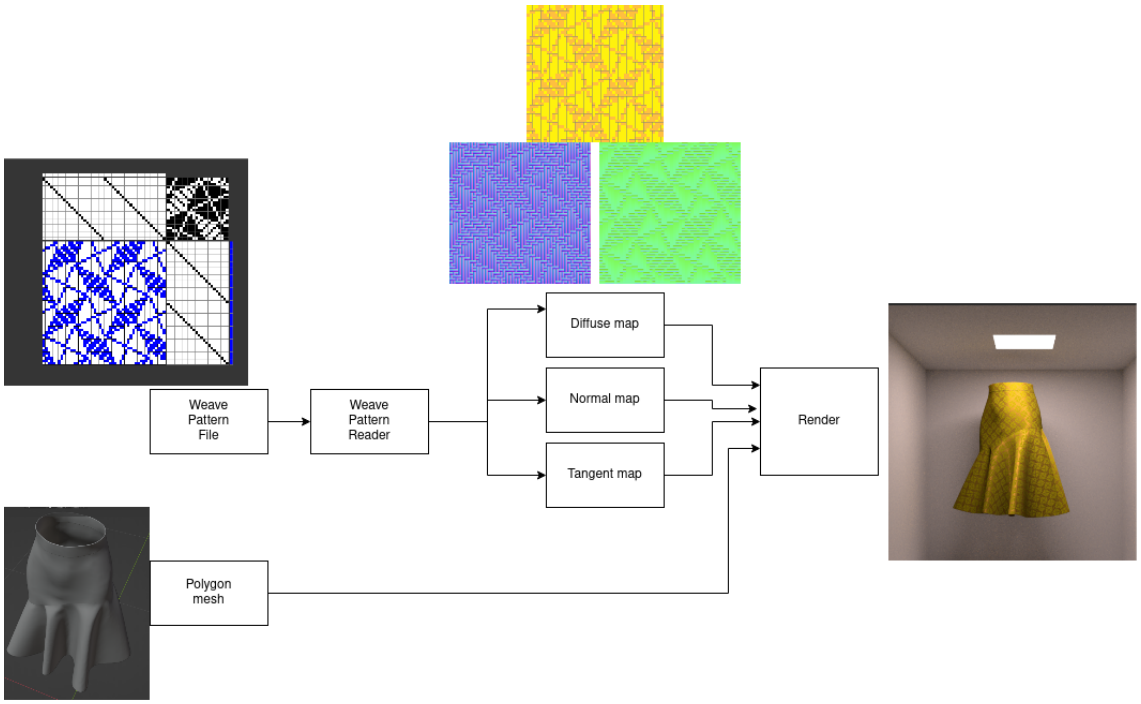


Figure 5.1: Project workflow

Raw data
Polygon meshes are the most widely used representation format in the industry, and there are many polygon mesh resources for cloth available on the Internet. WIF files are publicly available and can be created using some WIF design software.

Texture synthesis
Once we have extracted the weave pattern raw data, we can process the file and generate a weave pattern from it. Then the weave pattern is parsed and stored in a 2D matrix data structure. A minor issue of the matrix representation is that it generally neglects the gap in between the weave pattern. A possible fix would be to set the value in the gap to a specific value and handle it differently. Usually weave patterns are dense, and here we do not address this potential problem. There is many software available for reading WIF format such as a Python library Pyweave [28], which core function is parsing WIF files. However, to model the cloth microgeometry we still need the normal and tangent information. Fortunately, the source code of this library is available and modifiable. We

can use the method in section 4.1 to process the weave pattern binary matrix and to generate and export the synthesis texture maps. In addition, many different software programs can be used for the design and editing of WIF files, but this is not the main concern of this study.

Renderer selection

The next step of the pipeline is to render the polygon meshes. We want to implement the shader in a renderer with multi-platform support and easy to use. OpenGL is a widely used rendering software and is known to have wide platform support and is hardware friendly, allowing use on a variety of mid- to low-end hardware and making it more hardware-friendly. However, some advanced lighting features such as indirect lighting and shadow are not easy to implement in the OpenGL framework. However, a ray tracer usually has a better support for these advanced lighting features. The implementation in OpenGL uses the C++ programming language and OpenGL shading language (GLSL). A framework provided by LearnOpenGL is used to facilitate development. [29].

Mitsuba is an open source ray-tracing renderer [30] implemented in C++, with the aim of being easy to extend with new functionalities. Mitsuba has a retargetable feature that supports compiling the same code on either CPU or GPU, which is a huge benefit for developing.

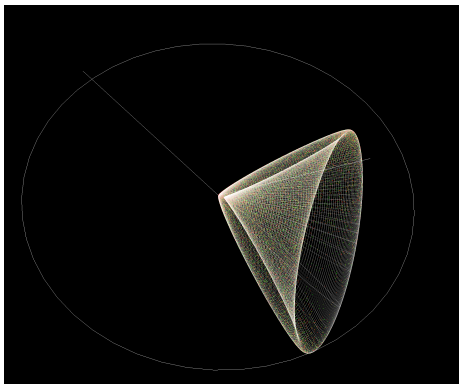
Source code and detail implementation in https://github.com/gniknyh/cloth_2023.

6 Result

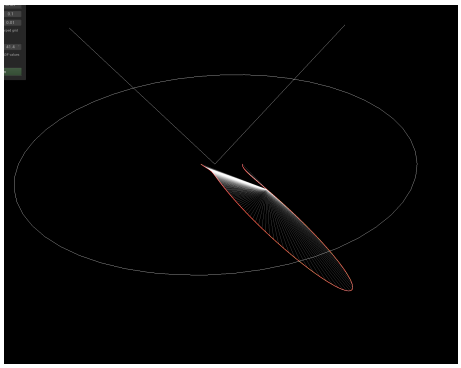
The results presented in this section were created using a desktop computer with an AMD Ryzen 5600g processor, 32 GB RAM and an NVIDIA GeForce RTX 3060 desktop GPU running on Ubuntu environment.

6.1 BSDF validation

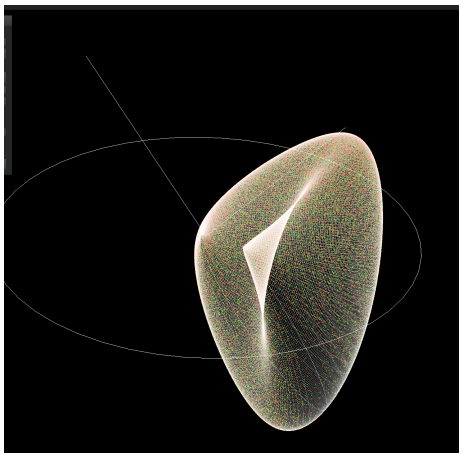
Given the complex illumination environments and textures, it is difficult to verify the correctness of our BSDF directly based on the final rendering result. Therefore, it is a good idea to take an extra step and plot our BSDF value before starting the rendering process. Before we see the rendered result, we first validate our BSDF output using a visualization tool from Nori. We can plot the shape of the lobe from our modified BSDF model. We decompose the components from the shading model and plot the shape individually in fig. 6.1. We also plot the scattering lobes using different thread parameters in fig. 6.2. The shape of the lobe in polyester is much narrower and sharper than linen, and the reflected energy is more concentrated on the lobe center and will yield a more shiny appearance.



(a) Reflectance component $f_{r,s}$

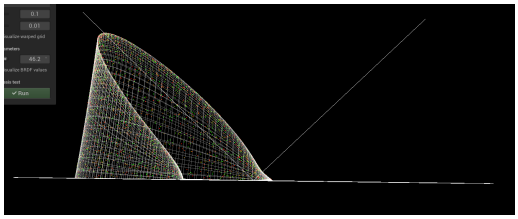


(b) Volume scattering component $f_{r,v}$

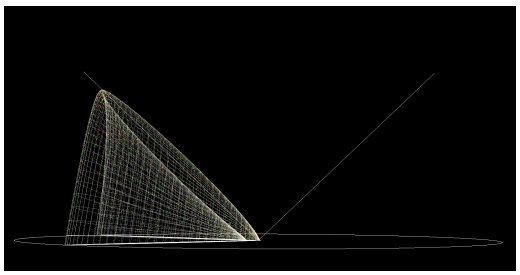


(c) Mixture of scattering components

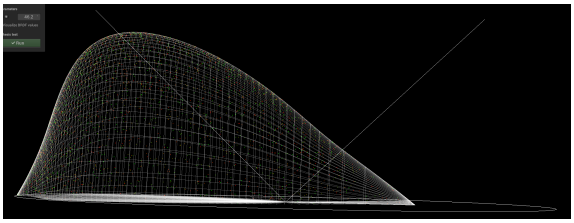
Figure 6.1: Light scatter components visualization



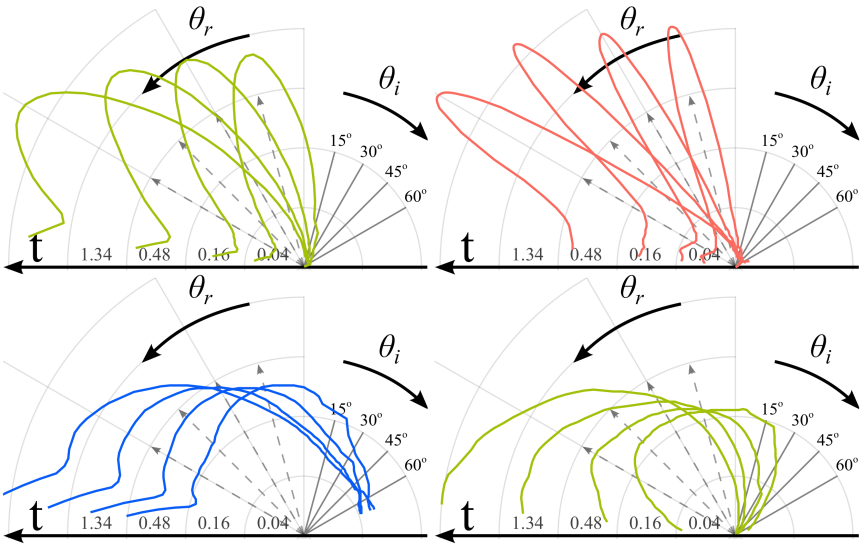
(a) Silk



(b) Polyester



(c) Linen



(d) Reference BRDF plot

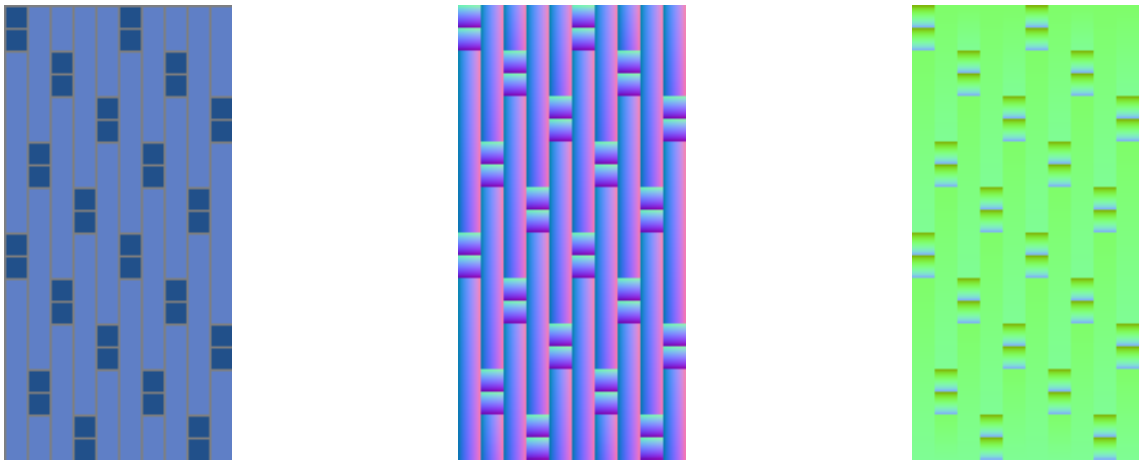
Figure 6.2: BRDF visualization using different thread parameters. θ_i is set to near $45°$ in (a)(b)(c). (d) is BSDF measurement in Sadeghi model as reference. Starting at the top row, and moving left to right, there are the measurement of flat silk thread, flat polyester thread, twisted linen thread and twisted silk thread.(image from [8])

From the visualizations in Figure 6.1c, we can see that even when a new sampling strategy is adopted, the sampling result is still approximately a cone. Using the fabric parameters of the Sadeghi model, we further visualized the modified BRDF results and compared them with their measurements. The Cauchy distribution will cause numerical differences in the shading calculation and this could lead to a subtle difference in visual appearances. Arguably, such subtle differences will not significantly harm the physical plausibility of the light scattering model, as long as the render results can still reproduce the anisotropic specular reflectance. The aim of this project is to qualitatively reproduce the anisotropic behaviour. Quantitatively matching rendering appearance to reality is out of the scope of

this project due to a lack of measurement data.

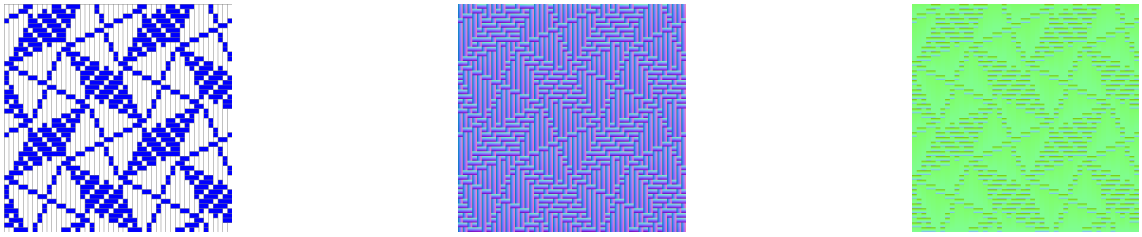
6.2 Weave pattern texture synthesis

We select the most representative weave pattern to begin, showing the result from a simple pattern to a complex one.



(a) Diffuse texture (b) Normal map (c) Tangent map

Figure 6.3: Synthesis texture of the twill pattern



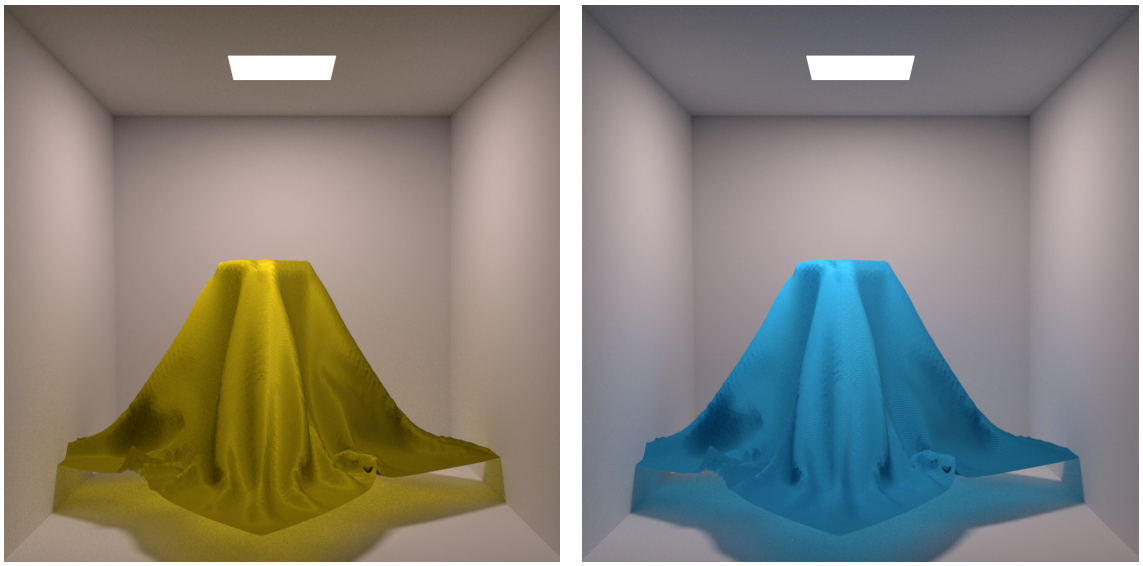
(a) Diffuse texture (b) Normal map (c) Tangent map

Figure 6.4: Fancy weave pattern

Each weave pattern generates three texture maps as an output. Taking the normal map as example, we can see that the normal direction changes sharply in between warp and weft. The resolution of the texture maps can vary due to the treading used in the weave pattern configuration. In practice, the total space to store three texture maps with a resolution lower than 1024x1024 is usually less than 1 MB. Once we generate the geometrical data from the weave pattern, we can proceed to the rendering stage.

6.3 Rendering quality

6.3.1 Raytracing result

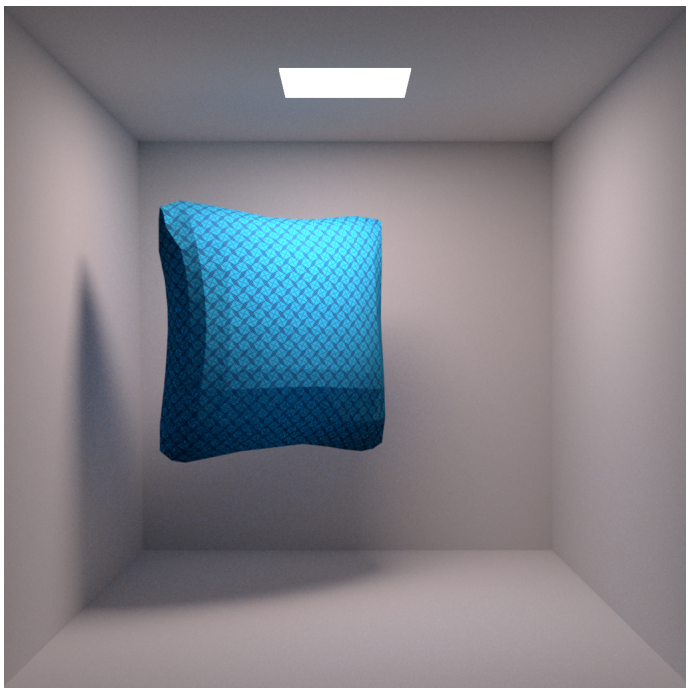


(a) Silk

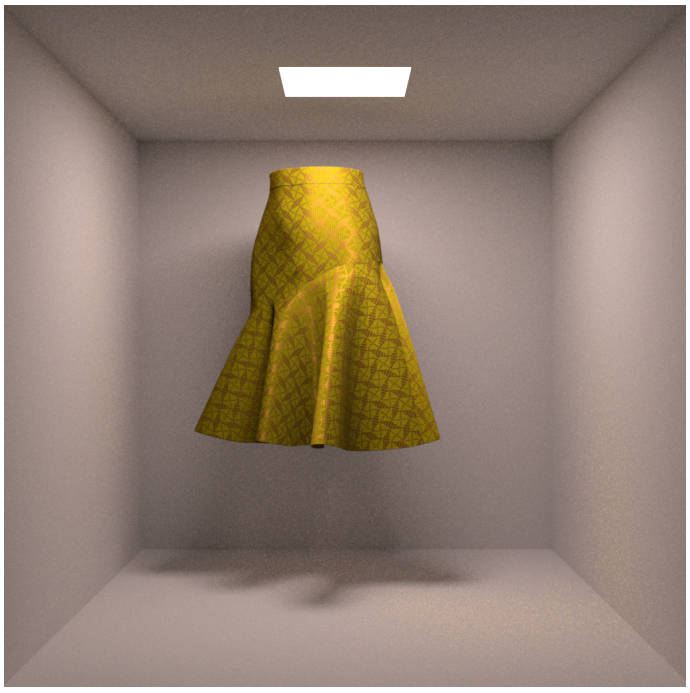
(b) Linen

(c) Polyester

Figure 6.5: Render result of a cloth polygon mesh using the simple weave pattern from fig. 6.3. Rendered using Mitsuba with 256 SPP under area light.

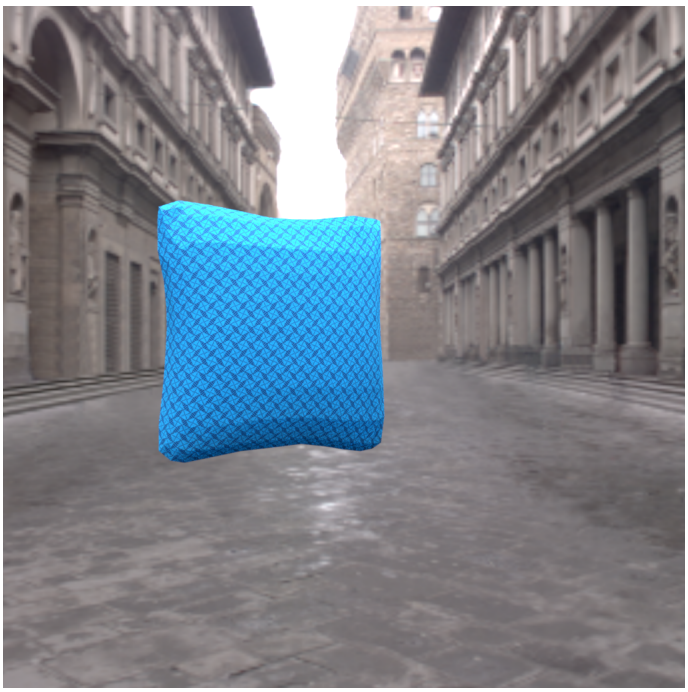


(a) Linen pillow

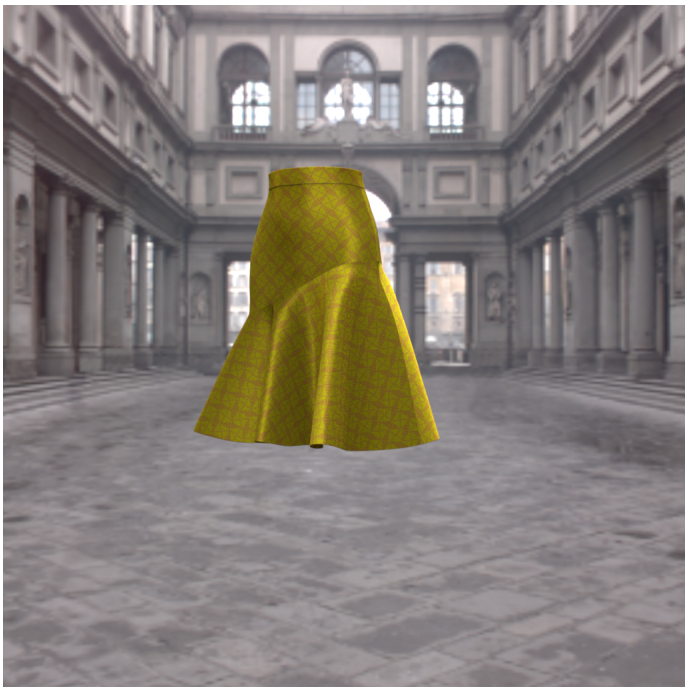


(b) Silk skirt

Figure 6.6: A Cornell box scene using area light illumination rendered in Mistuba. Used weave pattern can be seen in Figure 6.4. The silk skirt shows a stronger specular reflectance than a linen pillow.



(a) Linen pillow

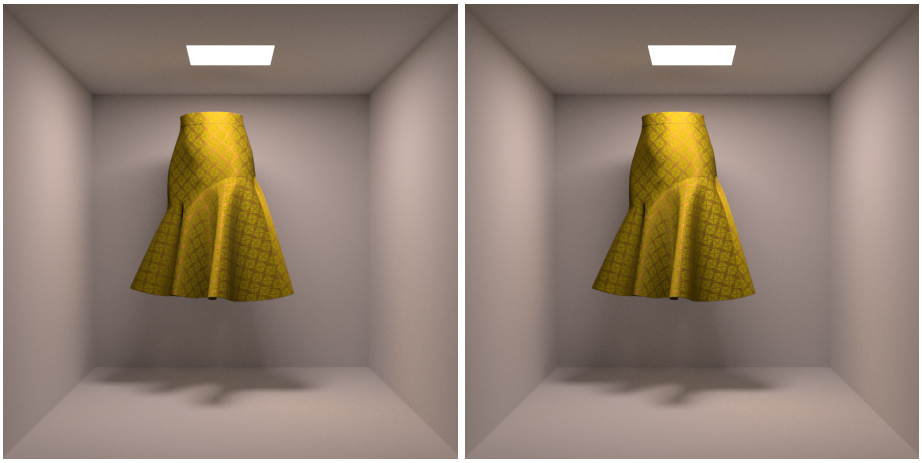


(b) Silk skirt

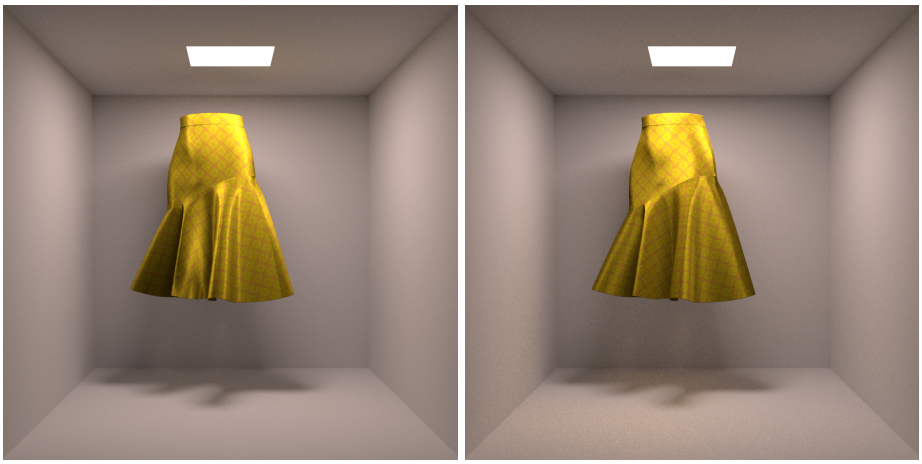
Figure 6.7: A light probe image ”Uffizi Gallery” used as environmental light illumination rendered in Mitsuba.

Note that the UV texture coordinates of a polygon mesh will affect the rendering result. These UV texture coordinates are important vertex attributes, as they contain mapping information about how to sample the texture image over the surface. If the UV coordinates are absent, we cannot use texture maps generated from a weave pattern to represent its micro-appearance.

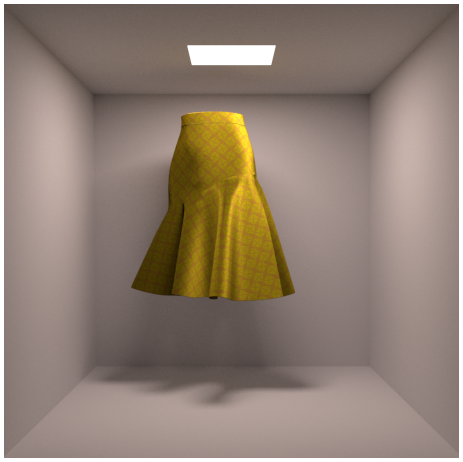
In the Sadeghi model, empirical parameters were used to estimate the distribution of threads' tangent direction, while we model the weave pattern tangent direction differently. Moreover, the difference in sampling, weave pattern, illumination environment, and chosen polygon mesh also causes minor visual differences in appearance. Regardless of the alteration of these factors, our modified shading model is able to reproduce the anisotropic reflectance behaviour in a similar manner as the Sadeghi model.



(a) Reference (b) Low resolution synthesis textures



(c) prefiltering normal map (d) prefiltering tangent map



(e) prefiltering tangent and normal map

Figure 6.8: Level of detail. (a) is rendered using a resolution of 800×800 pixels synthesis textures. (b) is using a lower resolution of 200×200 pixels synthesis textures and has subtle difference to reference. (c,d,e) are using a resolution of 200×200 pixels vMF maps by downsampling the synthesis texture maps. Down-sampled results couldn't resemble the reference and have visually noticeable difference, more specifically, a stronger specular reflectance.

An attempt was made to validate the normal filtering techniques to perform multi-scale representations and the results are shown in Figure 6.8. However, the visual difference in the result shows that the attempt has failed and the prefiltering strategy is incomplete. Ideally, the rendering result should preserve its original appearance. However, the specular reflectance appears to be displaced, and the shadow does not appear to be correct. There are not many studies about filtering on an anisotropic BSDF, the normal and tangent jointly. Therefore, it might be better to look for alternative strategies and develop a better filtering strategy.

6.3.2 OpenGL rendering results

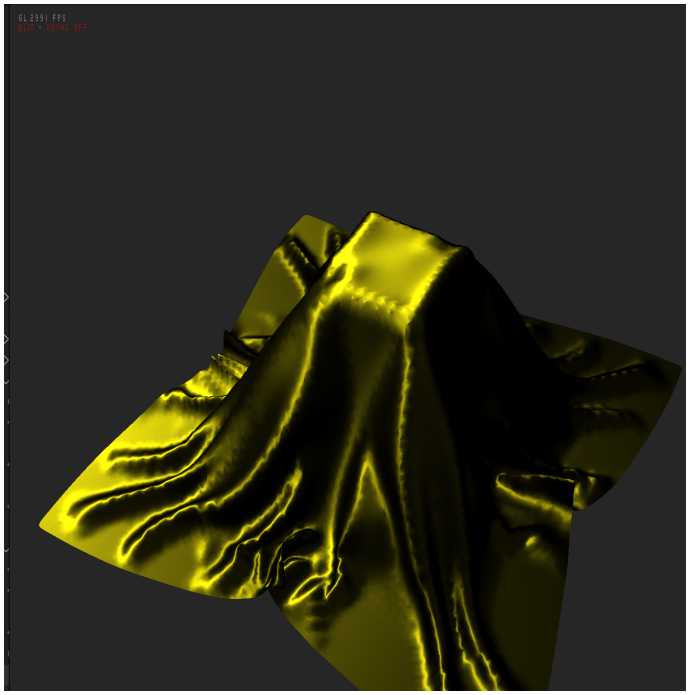
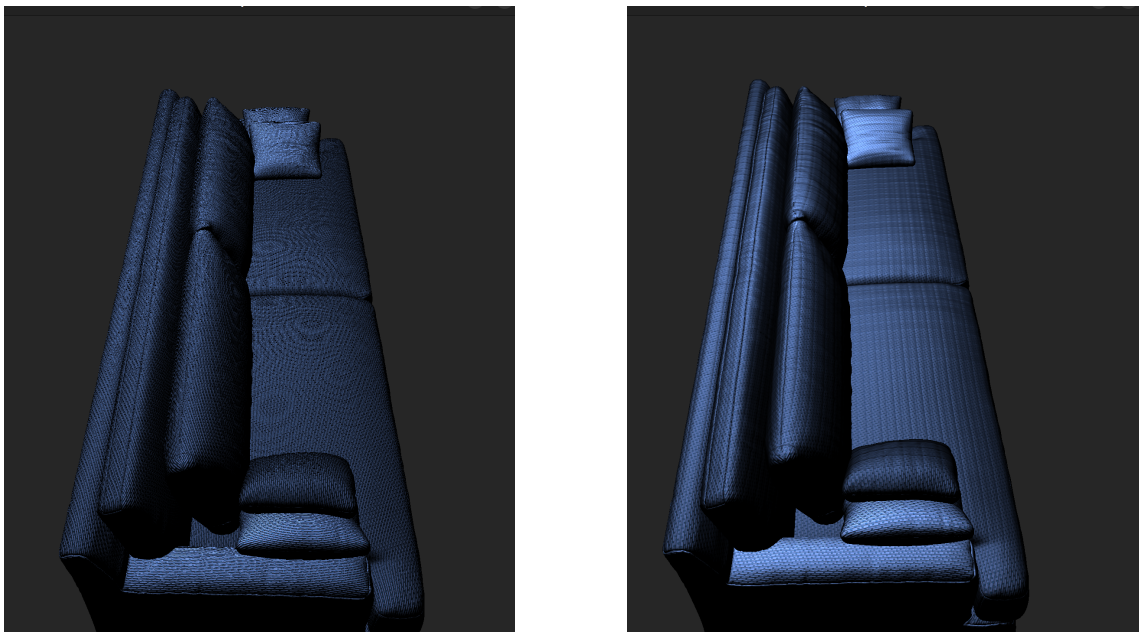


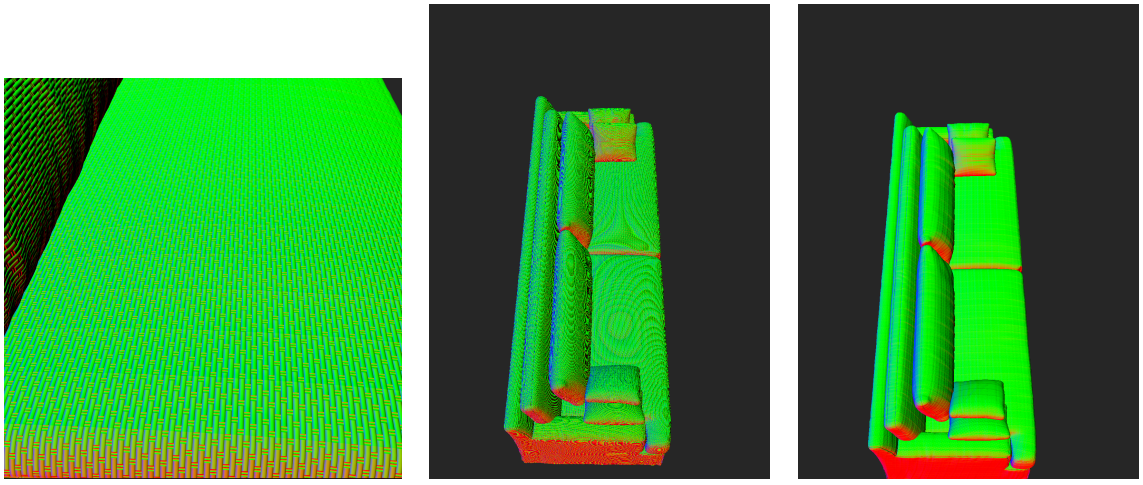
Figure 6.9: Cloth rendering under direct lighting in OpenGL using polygon mesh macro geometry. FPS in topleft is 2991

Before applying texture map to represent the geometrical detail, this far-field shading model does not need to employ aliasing and can have a smooth look (See fig. 6.9). However, the aliasing problem arises once the texture maps are applied. The pre-filtering strategy is only tested in the ray tracer but not validated in OpenGL, therefore we use the default filtering function from OpenGL.



(a) Mipmap disabled (b) Mipmap enabled

Figure 6.10: A sofa rendered in OpenGL. Severe aliasing in (a)



(a) close up (b) mipmap disabled (c) mipmap enabled

Figure 6.11: Result of world space normal. Aliasing in normal mapping due to minification in (b). High-frequency details are filtered out in (c), leaving a smooth and flat surface.

The texture parameter of minification is set to **GL_LINEAR_MIPMAP_LINEAR** to enable mip-mapping in OpenGL. Using a simple linear filter can perceptually reduce the moire pattern and alleviate the aliasing problem. However, the approximation of normal and tangent vectors is not very effective, bringing an error to the anisotropic specular, which causes the result to be either brighter or darker than it should be.

6.4 Performance

Texture maps synthesis
The rendering time is separated from the time it takes to create texture maps. The processing time for generating textures is quick; it usually takes a few seconds, varying on the size of the weave pattern and the output texture's resolution. It means whenever an artists get interested into one weave pattern and want to preview its look on a cloth, it only takes a few seconds to generate textures. The texture maps can be used in the rendering pipeline or external renderer. The whole process cost very little time and very beneficial to rapid prototyping. Because the textures are already computed before rendering, there is no need to compute the micro geometry at run-time, which improves rendering performance.

Rendering

Table 6.1: Performance test on Mitsuba. A Cornell box scene takes 5.6 seconds to render on GPU at 1024 $\times$ 1024 with resolution with 256 samples per pixel(SPP).

	Resolution	triangles number		SPP	time(second)
Mitsuba 3 (GPU)	1024$\times$1024	247	K	256	8.76
Mitsuba 3 (GPU)	1024$\times$1024	247	K	32	1.16
Mitsuba 3 (GPU)	512$\times$512	247	K	256	2.456
Mitsuba 3 (GPU)	1024$\times$1024	500		256	6.63
Mitsuba 3 (CPU)	512$\times$512	247	K	32	54.1

With the performance result in table 6.1, using the Cornell box scene as a baseline to compare, we can see that this cloth shader can run very efficiently in Mitsuba. With Ray tracing techniques gaining popularity nowadays, light-weight cloth shading models are becoming more affordable to be used in production.

On the other hand, a scene with simple polygon meshes and direct illumination in OpenGL usually runs in above 2000 FPS. It turns out that the traditional rasterization pipeline with this shading model can have a satisfactory performance, and can be used for real-time rendering reliably. More anti-aliasing solutions to improve rendering quality should be affordable.

7 Discussion

Storage requirements
The raw input of our model is a weave pattern file, which takes very little storage space. The space to store synthetic textures ranging from 300 KB to 2MB for different resolutions, which is affordable in most production environment.

Efficiency and rendering time
The time spent for texture maps synthesis is insignificant in production and it is satisfactory for rapid prototyping. The implementation of OpenGL clearly shows that this approach can be rendered in real-time, which is a major advantage. For some environments that have a restriction on performance like VR, a light-weight shading model could be more attractive than those with heavy computation cost. Monte Carlo path tracing has seen remarkable improvement with recent hardware advancements. However, the results still have a long way to go before they can be used for real-time rendering. Reducing the sample per pixel will bring it closer to real-time ray tracing. It will be interesting to see the rendering results when combined with more advanced denoising techniques.

Editing capabilities
We use a procedural generation approach to synthesize texture maps representing geometric details. The yarn geometry is generated based on the input weave pattern. The weave pattern file is editable and users can alter the parameters during texture generation. The current texture synthesis solution can only modify a few parameters like the orientation of fibers on the yarn. In reality, some techniques used for cloth production such as dyeing and washing will add change the cloth color. One easy way to simulate such effects could be to add noise. This shading model uses several parameters, such as the cloth surface roughness, to model the specular lobe. Users can change such parameters to get a desired visual effect. In order to maintain shading plausibility, the valid range of parameters still needs to be tested to satisfy physical constraints. Extending on usability, our current implementation only uses simple heuristic parameters for weave pattern synthesis. However, users might not be familiar with those heuristic physical parameters of the analytical model. For example, the level of thread curvature could be difficult to predict if users have no knowledge of cloth manufacturing. It will be beneficial to generalize some pre-defined settings of cloth material parameters to make the work more automatic.

Accuracy
The accuracy of this work depends greatly on the underlying shading model and the cylindrical model that describes the weave pattern, thus inheriting their limitations. Regarding the geometry part, we only model the fabric at the yarn level as a cylinder, disregarding all the sub-yarn level detail. Regarding the light scattering model, it only uses a surface reflectance component and a transmittance component to simplify the complex light-fiber interaction in reality. The efficiency of this shading model and the plausibility of rendered results can be considered as a good trade-off. Last but not least, although the current shading model is based on yarn-level, it is compatible to add sub-yarn level detail and procedurally generate more fiber details as demonstrated by Luan et al. [31]. Additional sub-yarn level representation can bring the rendered result to the next level. For the current version of the work, it is more favorable to use this pipeline in an environment that does not require photorealistic standards.

Aliasing
For ray tracing, noise reduction is of much more concern; therefore, we apply the importance sampling strategy to improve the rendering quality. Importance sampling makes the

render result converge faster and reduces noise. The attempt of downsampling texture to achieve LoD does not work as expected, thus we still need to find new feasible offline rendering solutions for LoD in the future.

On the other hand, for real-time rendering, the results encounter more problems than ray tracing specific on antialiasing due to a lack of valid pre-filtering solution. Although simple filtering could be a valid option to reduce the moiré pattern phenomenon, this makes the results more prone to be less accurate. The state-of-the-art work proposed a filtering technique that jointly prefilters tangent and normal maps as well as BSDF parameters during runtime [32]. It could be a promising solution for the multiscale representation.

8 Conclusion and future work

An approach for synthesis of textures from weave patterns is presented. The rendering is implemented with a modified Sadeghi cloth shading model. The rendering result is able to produce the complex anisotropic highlights and give a plausible appearance of cloth with given weave pattern.

The aim of the project is to reduce the extensive manual labor of artists to simulate the visual effect of cloth, and it is achieved by procedurally generated texture maps. The shading model of the project is able to produce physically plausible rendering results. Unfortunately, this work does not have a good multi-scale representation solution. The aliasing problem can be alleviated by mipmapping when viewing from a distance. However, the simple prefiltering strategy cannot handle well the nonlinearity of a normal map and brings inaccuracies.

For rendering quality improvement, one interesting direction is to study the sub-yarn level geometry and adopt the procedural fiber generation technique into the pipeline, which is necessary to add the sub-yarn detail and bring a realistic render result. Last but not least, more solutions need to be explored to develop a better strategy for multiscale representation and antialiasing. Reproducing the appearance of fabrics in an accurate and efficient manner is a very challenging problem that remains open to be solved.

Bibliography

Akenine-Mller, Tomas, Eric Haines, and Naty Hoffman (2018). *Real-Time Rendering, Fourth Edition*. 4th. USA: A. K. Peters, Ltd. ISBN: 0134997832.

Castillo, Carlos, Jorge López-Moreno, and Carlos Aliaga (Nov. 2019). "Recent Advances in Fabric Appearance Reproduction". In: *Comput. Graph.* 84.C, pp. 103–121. ISSN: 0097-8493. DOI: 10.1016/j.cag.2019.07.007. URL: https://doi.org/10.1016/j.cag.2019.07.007.

Zhu, Junqiu et al. (July 2022). "Practical Level-of-Detail Aggregation of Fur Appearance". In: *ACM Trans. Graph.* 41.4. ISSN: 0730-0301. DOI: 10.1145/3528223.3530105. URL: https://doi.org/10.1145/3528223.3530105.

Khungurn, Pramook et al. (Dec. 2016). "Matching Real Fabrics with Micro-Appearance Models". In: *ACM Trans. Graph.* 35.1. ISSN: 0730-0301. DOI: 10.1145/2818648. URL: https://doi.org/10.1145/2818648.

Marschner, Stephen R. et al. (July 2003). "Light Scattering from Human Hair Fibers". In: *ACM Trans. Graph.* 22.3, pp. 780–791. ISSN: 0730-0301. DOI: 10.1145/882262.882345. URL: https://doi.org/10.1145/882262.882345.

Irawan, Piti and Steve Marschner (Feb. 2012). "Specular Reflection from Woven Cloth". In: *ACM Trans. Graph.* 31.1. ISSN: 0730-0301. DOI: 10.1145/2077341.2077352. URL: https://doi.org/10.1145/2077341.2077352.

Dana, Kristin J. et al. (Jan. 1999). "Reflectance and Texture of Real-World Surfaces". In: *ACM Trans. Graph.* 18.1, pp. 1–34. ISSN: 0730-0301. DOI: 10.1145/300776.300778. URL: https://doi.org/10.1145/300776.300778.

Sadeghi, Iman et al. (Apr. 2013). "A Practical Microcylinder Appearance Model for Cloth Rendering". In: *ACM Trans. Graph.* 32.2. ISSN: 0730-0301. DOI: 10.1145/2451236.2451240. URL: https://doi.org/10.1145/2451236.2451240.

Aliaga, Carlos et al. (July 2017). "An Appearance Model for Textile Fibers". In: *Comput. Graph. Forum* 36.4, pp. 35–45. ISSN: 0167-7055. DOI: 10.1111/cgf.13222. URL: https://doi.org/10.1111/cgf.13222.

Han, Charles et al. (July 2007). "Frequency Domain Normal Map Filtering". In: *ACM Trans. Graph.* 26.3, 28–es. ISSN: 0730-0301. DOI: 10.1145/1276377.1276412. URL: https://doi.org/10.1145/1276377.1276412.

Adabala, Neeharika, Nadia Magnenat-Thalmann, and Guangzheng Fei (2003). "Visualization of Woven Cloth". In: *Proceedings of the 14th Eurographics Workshop on Rendering*. EGRW '03. Leuven, Belgium: Eurographics Association, pp. 178–185. ISBN: 3905673037.

Nelson, Vidar, Peter M. McEvoy, and Marco Fratarcangeli (2016). "Practical Offline Rendering of Woven Cloth". In: *Proceedings of the Conference on Smart Tools and Applications in Computer Graphics*. STAG '16. Genova, Italy: Eurographics Association, pp. 63–70. ISBN: 9783038680260.

Smith, Bryan et al. (2018). "Simulating Woven Fabrics with Weave". In: *ACM SIGGRAPH 2018 Talks*. SIGGRAPH '18. Vancouver, British Columbia, Canada: Association for Computing Machinery. ISBN: 9781450358200. DOI: 10.1145/3214745.3214781. URL: https://doi.org/10.1145/3214745.3214781.

Schroder, Kai, Arno Zinke, and Reinhard Klein (Feb. 2015). "Image-Based Reverse Engineering and Visual Prototyping of Woven Cloth". In: *IEEE Transactions on Visualization and Computer Graphics* 21.2, pp. 188–200. ISSN: 1077-2626. DOI: 10.1109/TVCG.2014.2339831. URL: https://doi.org/10.1109/TVCG.2014.2339831.

Nielsen, R et al. (1997). *Weaving Information File Version 1.1*.
Zhao, Shuang et al. (July 2012). “Structure-Aware Synthesis for Predictive Woven Fabric Appearance”. In: *ACM Trans. Graph.* 31.4. ISSN: 0730-0301. DOI: 10.1145/2185520.2185571. URL: https://doi.org/10.1145/2185520.2185571.
Rogina-Car, Beti, Dragana Kopitar, and Ivana Schwarz (2018). “Protective Properties of Health Care Materials Influenced by the Application Conditions”. In: *Textile & Leather Review* 1.1, pp. 18–28.
Sadeghi, Iman et al. (2010). “An Artist Friendly Hair Shading System”. In: *ACM SIGGRAPH 2010 Papers*. SIGGRAPH ’10. Los Angeles, California: Association for Computing Machinery. ISBN: 9781450302104. DOI: 10.1145/1833349.1778793. URL: https://doi.org/10.1145/1833349.1778793.
Box, George EP (1958). “A note on the generation of random normal deviates”. In: *Ann. Math. Statist.* 29, pp. 610–611.
Hery, Christophe and Ravi Ramamoorthi (2012). “Importance sampling of reflection from hair fibers”. In: *Journal of Computer Graphics Techniques (JCGT)* 1.1, pp. 1–17.
Wang, Chunpo, Feng Xie, and Parashar Krishnamachari (2014). “Importance Sampling for a Microcylinder Based Cloth Bsdf”. In: *ACM SIGGRAPH 2014 Talks*. SIGGRAPH ’14. Vancouver, Canada: Association for Computing Machinery. ISBN: 9781450329606. DOI: 10.1145/2614106.2614134. URL: https://doi.org/10.1145/2614106.2614134.
Chiang, Matt Jen-Yuan et al. (2015). “A Practical and Controllable Hair and Fur Model for Production Path Tracing”. In: *ACM SIGGRAPH 2015 Talks*. SIGGRAPH ’15. Los Angeles, California: Association for Computing Machinery. ISBN: 9781450336369. DOI: 10.1145/2775280.2792559. URL: https://doi.org/10.1145/2775280.2792559.
Bruneton, Eric and Fabrice Neyret (Feb. 2012). “A Survey of Non-linear Pre-filtering Methods for Efficient and Accurate Surface Shading”. In: *IEEE Transactions on Visualization and Computer Graphics* 18.2, pp. 242–260. DOI: 10.1109/TVCG.2011.81. URL: https://hal.inria.fr/inria-00589940.
Olano, Marc and Dan Baker (2010). “LEAN Mapping”. In: *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D ’10. Washington, D.C.: Association for Computing Machinery, pp. 181–188. ISBN: 9781605589398. DOI: 10.1145/1730804.1730834. URL: https://doi.org/10.1145/1730804.1730834.
Dupuy, Jonathan et al. (Nov. 2013). “Linear Efficient Antialiased Displacement and Reflectance Mapping”. In: *ACM Trans. Graph.* 32.6. ISSN: 0730-0301. DOI: 10.1145/2508363.2508422. URL: https://doi.org/10.1145/2508363.2508422.
Wu, Lifan et al. (July 2019). “Accurate Appearance Preserving Prefiltering for Rendering Displacement-Mapped Surfaces”. In: *ACM Trans. Graph.* 38.4. ISSN: 0730-0301. DOI: 10.1145/3306346.3322936. URL: https://doi.org/10.1145/3306346.3322936.
Jakob, Wenzel (2012). “Numerically stable sampling of the von Mises-Fisher distribution on S^2 (and other tricks)”. In: *Interactive Geometry Lab, ETH Zürich, Tech. Rep*, p. 6.
storborg (n.d.). *GitHub - storborg/pyweaving: Python Weaving Tools*. URL: https://github.com/storborg/pyweaving.
Vries, J. de (2020). *Learn OpenGL: Learn Modern OpenGL Graphics Programming in a Step-by-step Fashion.* Kendall & Welling. ISBN: 9789090332567. URL: https://books.google.dk/books?id=koWWzQEACAAJ.
Jakob, Wenzel et al. (2022). *Mitsuba 3 renderer*. Version 3.0.1. https://mitsuba-renderer.org.
Luan, Fujun, Shuang Zhao, and Kavita Bala (July 2017). “Fiber-Level On-the-Fly Procedural Textiles”. In: *Comput. Graph. Forum* 36.4, pp. 123–135. ISSN: 0167-7055. DOI: 10.1111/cgf.13230. URL: https://doi.org/10.1111/cgf.13230.

Xu, Chao et al. (2021). “Multi-Scale Hybrid Micro-Appearance Modeling and Realtime Rendering of Thin Fabrics”. In: *IEEE Transactions on Visualization and Computer Graphics* 27.4, pp. 2409–2420. DOI: 10.1109/TVCG.2019.2949406.

Technical
University of
Denmark

Brovej, Building 324
2800 Kgs. Lyngby
Tlf. 4525 1700

https://www.compute.dtu.dk/