An Applied Approach for Real-Time Level-of-Detail Woven Fabric Rendering

Wallace Yuen
The University of Auckland,
New Zealand
wyue013@aucklanduni.ac.nz

Burkhard C. Wünsche
The University of Auckland,
New Zealand
burkhard@cs.auckland.ac.nz

Nathan Holmberg
77-Pieces Ltd,
New Zealand
nathan@77-pieces.com

ABSTRACT

Photorealistic rendering of fabric is essential in many applications ranging from movie special effects to e-commerce and fashion design. Existing techniques usually render the fabric's microscale structure. However, this can result in severe aliasing and is unsuitable for interactive cloth simulation and manipulation. In this paper we describe a novel real-time level-of-detail fabric rendering technique. The algorithm adjusts geometry and texture details with changing viewpoint by using a mipmapping approach, in order to obtain a perceptually consistent representation on the screen. Compared to previous work we also introduce more parameters allowing the simulation of a wider range of fabrics. Our evaluation demonstrates that the presented approach results in realistic renderings, increases the shader's run-time speed, and reduces aliasing artifacts by hiding the underlying yarn geometry.

Keywords: fabric rendering, anisotropic materials, real-time rendering, cloth simulation, anti-aliasing, level-of-detail methods

1 INTRODUCTION

Realistic fabric rendering addresses many different areas and industries in computer games and fashion applications. It is a challenging research field due to the complexity of the underlying fabric structure, textures, and materials, which results in complex light interactions and aliasing problems when using a raster representation. Fabric structures vary depending on the manufacturing process, such as weaving and knitting, and the desired fiber properties. Previous research in this field has explored different aspects of this problem, such as rendering complex weaving and knitting patterns, and developing specialized lighting models that simulate light interaction with the yarn geometry and its microstructure.

The modeling of complex weaving patterns and yarn geometry can result in aliasing when the screen resolution is lower than the perceived color variations on the material (caused by the geometry, lighting and texture). This is particularly problematic when animating the fabric using cloth simulations, which creates conspicuous temporal aliasing artifacts. In recent years, many hardware anti-aliasing techniques have been developed for real-time applications such as computer games, but are mainly used as a post-processing remedy. In this paper, we describe a level-of-detail fabric rendering technique for reducing the aliasing artifacts with minimal impact on computation time. This method can be used in conjunction with post-processing anti-aliasing techniques to further reduce the aliasing artifacts.

We want to create a parameterized fabric shaders for fashion design and e-commerce applications. This fundamentally requires fast interactive speed, high memory efficiency, and high robustness to support for a wide range of woven fabrics. We found that the model proposed by Kang [Kan10] would be the most suitable for our needs with several extensions and modifications to the original algorithm [YW11]. We adopted this model and implemented it in OpenGL Shading Language to support real-time parameterization of weaving pattern and yarn geometry.

Section 2 reviews existing fabric rendering techniques. Section 3 introduces previously presented techniques for modeling light interaction and rendering fabric, which form the foundation of our fabric rendering framework. Section 4 proposes our level-of-detail algorithm and improvements to the existing algorithm for real-time fabric rendering. Section 5 presents an evaluation of our framework and Section 6 draws conclusions and suggests directions for future work.

2 LITERATURE REVIEW

Fabric rendering techniques have been an active area of research since the early 1990s. We classify existing techniques into two categories, example-based and

procedural-based models. Example-based models focus on capturing reflectance information of specific material and use the captured data for rendering virtual fabrics. Procedural-based models are empirical mathematical models that use various parameters to control the appearance of fabrics.

2.1 Example-Based Models

Example-based cloth rendering techniques require the capturing of reflectance information of materials. This usually requires modification of the lightings, sensors, and planar examples of the corresponding materials. The reflectance properties of a material for different viewpoints and light directions can be encoded in a Bidirectional Reflectance Distribution Function (BRDF), which is often obtained with a gonioreflectometer [War92].

Daubert et al. [DLH01] proposed a Spatially-Varying Bi-directional Reflection Distribution Function (SVBRDF) specifically for cloth rendering using the Lafortune reflection model. It is designed to render clothes with repeating patterns such as knitting and weaving patterns. A texture map is used as a look-up table, for storing precomputed parameters of the Lafortune reflection model. This method works for both knitted and woven clothes by modeling the structure explicitly through the generation of new triangle meshes, but the computation consists of several rendering passes. Even though many methods have been proposed to obtain a SVBRDF using only a single view of a material sample [WZT+08], SVBRDF is generally very memory intensive, which makes it impractical for real-time applications where material parameters are interactively changed.

Another popular example-based model is the Bidirectional Texture Function (BTF), which captures a material's light interaction for varying light source and camera positions. The BTF captures more effects than the SVBRDF including self-shadowing, occlusion, and inter-reflection effects, and is used for many surface reflectance data measurements. The actual textures of the cloth samples are used and stored as texture maps, and they are used at render time with different parameters such as the illumination, camera position, and the texture coordinates of the object. Due to the higher number of parameters, the BTF suffers from high memory requirements and acquisition costs. Kautz [Kau05] introduced a compression method for the BTFs. He acquires a lower number of images and interpolates between them. Despite these compression approaches, example-based methods still require an over-abundant storage capacity for practical use, and they do not offer enough flexibility for rendering different types of clothes with different weaving patterns.

A volumetric approach that uses the microflake model was proposed by Zhao et al. [ZJMB11]. The approach acquires a volume model of the material that needs to be rendered using a X-ray computed tomography (CT) scanner. The volumetric data acquired is post-processed for orientation extraction and noise removal, and is matched to a photograph of the same material captured to obtain the optical properties for fabric rendering [ZJMB11]. This approach suffers from high memory requirements, due to the size of volumetric data, where each fabric sample takes approximately 7.26GB [ZJMB11]. It is also difficult to acquire equipments for this approach, due to the cost of CT scanners, thus making it difficult to capture different fabrics.

2.2 Procedural-Based Models

Procedural-based cloth rendering techniques are models that are designed based on the analysis of fabric structure. Yasuda et al. [YYTI92] developed shading models for woven cloth by analyzing fiber properties and weaving patterns. The authors proposed a tiny facet model for fabric materials taking into consideration reflection and refraction of multiple fabric layers. Using a multiple layer model, they simulated the scattering effects of fabrics by calculating the light refraction at different layers [YYTI92]. The reflection model assumes a simple percentage of warp and weft yarns in woven clothes and used a non yarn-based reflection. The light interaction with a small area of fabric is calculated by obtaining the total reflections [YYTI92]. Hence, this approach does not explicitly model the weaving patterns, but simulates the appearance of the fabric at a higher level where the weaving patterns are not visible.

Ashikhmin et al. [AS00] developed a microfacet-based anisotropic model that can be used for general materials, and was tested by simulating satin and velvet. The authors take into account the weaving pattern of satin and velvet. For example, satin is modeled by weighting the BRDF values of weft and warp yarns [AS00]. Due to a lack of self-shadowing and light interaction at the yarn-level, this microfacet anisotropic model is too generic to be used directly for fabric rendering, but it formed the foundation for many subsequently developed techniques.

Adabala et al. [AMTF03] use a weaving pattern input defined by the user, and generate a corresponding Anisotropic BRDF, texture, and horizon map for the clothing material. The authors render the fabric based on the weaving pattern input provided by the user to generate the overall appearance of the cloth [AMTF03]. This approach extends previous fabric models and allows more complicated weaving patterns to be defined. However, the authors applied the Ashikhmin-Shirley BRDF [AS00] on the object-level rather than the yarn-level, thus the modeling of light interaction with the fabric lacks realism compared to techniques which calculate light interaction based on yarn material and weaving patterns.

Kang [Kan10] proposed a procedural method that models the reflectance properties of woven fabric using alternating anisotropy and deformed microfacet distribution function. The proposed method is based on the microfacet distribution function (MDF) along with the Ashikhmin-Shirley [AS00] anisotropic shading model. Each sample point on the cloth is classified as a weft or warp yarn, and a corresponding distribution function is used accordingly to calculate the reflectance of that point [Kan10]. The alternating anisotropy approach allows the lighting to be defined for weft and warp thread by rotating the microfacet distribution function. Further deformation of the MDF enables the elaboration of yarn geometries and twisted appearances on the surface of each yarn. This approach enables not only the rendering of anisotropic clothes, but also the rendering of bumpy surfaces created by the weaving structure of the fabrics [Kan10].

Irawan [Ira08] developed a reflectance model and a texture model for rendering fabric suitable for distant and close-up views. The reflectance model is defined by the scattering effect of the fabric, while the texture model incorporates highlights calculated from the reflectance model. The texture model (BTF) is generated on the fly using parameters to control the types of yarn (i.e. staple or filament), and the appropriate weave pattern, and the yarn geometry is captured by using the fiber twisting angle to render the highlight on each yarn. A downside of this approach is the lack of shadowing and the masking effects to render some types of fabrics realistically, such as denim. However, the results look convincing and the approach is fully procedural, with intuitive variables at the fiber level, the yarn level, and the weaving pattern level.

3 FABRIC RENDERING MODEL

This section explains two techniques adopted by us in more detail: Kang's fabric rendering model [Kan10] and the Ashikhmin-Shirley anisotropic shading model [AS00] for capturing anisotropic reflections.

3.1 Ashikhmin-Shirley BRDF

The Ashikhmin-Shirley BRDF [AS00] is given by the following equation:

$$\rho(k_1,k_2) = \frac{p(h)P(k_1,k_2,h)F(k_1h)}{4(k_1n)(k_2n)(nh)} \qquad (1)$$

This equation represents the illumination of a point with the incoming light vector k_1 and outgoing light vector k_2 where additional functions explained below describe the effect of the microfacet structure. The vector n represents the surface normal at a point, and the vector h describes the half vector obtained from the incoming and outgoing light vector. The function $P(k_1,k_2,h)F((kh))$ captures the shadowing effects caused by microfacets. The function $F(kh)$ is the Fresnel reflectance that describes the amount of incoming light that is reflected off the surface specularly. The function $p(h)$ is the MDF given by Equation 2. It describes the illumination effects of weaving patterns in Kang's model [Kan10].

3.2 Microfacet Distribution Function

The microfacet distribution function characterizes a surface's distribution of microfacets, by encoding their normal direction relative to the underlying surface.

$$p(h) = \frac{\sqrt{(x+1)(y+1)}}{2\pi}(h\cdot n)^{xcos^2\phi+ysin^2\phi} \qquad (2)$$

The microfacet distribution function in Equation 2 is used to generate the BRDF. The function captures the visual effects of microgeometry, where the reflected specular light on the surface is proportional to the probability of the microfacet surface normals that are aligned to the half vector. Variables x and y controls the shape of the specular highlight and the intensity in anisotropic reflection.

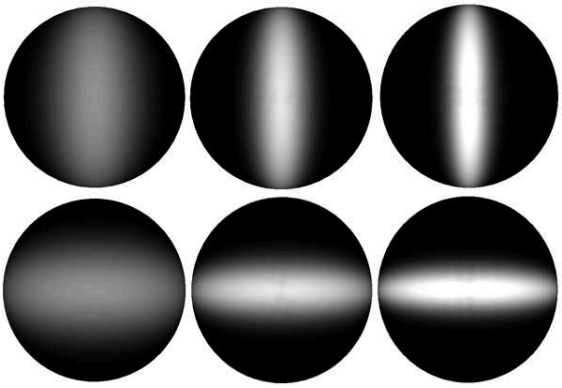

Figure 1: Our generated images using the Ashikhmin-Shirley BRDF [AS00] for visualizing the difference in specular highlights with varying parameters x and y. Top row: x = 10, 30, 50, while y stays constant equal to 1. Bottom row: y = 10, 30, 50, while x stays constant equal to 1.

A visualization of the microfacet distribution is shown in Figure 1. When the x and y values are close to each other, then the distribution of microfacets aligning to h is spread more evenly across the surface. The top row of the diagram demonstrates that if the x-value in the MDF increases from 10 to 50, then the distribution of microfacets becomes denser in the center, thus resulting in a less spread specular lobe on the object surface. This results in an increasingly narrow highlight stretched in y-direction.

3.3 Weaving Pattern Rendering

Kang [Kan10] proposed an alternating anisotropy solution to render weaving patterns by using Equation 2

to generate the specular highlight that can be seen on weaving pattern [Kan10]. Using the fact that weaving patterns are only made of weft and warp yarns, the specular highlight of weft yarns must therefore be obtained by rotating the microfacet distribution by 90°. This is again shown in Figure 1. The rotated microfacet distribution is seen on the second row, and is done by exchanging the values of x and y to create such rotation.

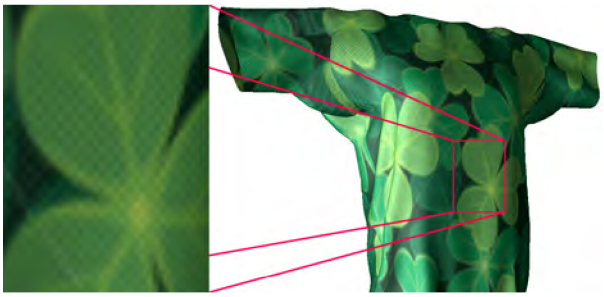

Figure 2: Weave pattern generated by using the alternating anisotropy method showing the closer view (left) and distant view (right).

An example of a weaving pattern generated using the alternating anisotropy method is shown in Figure 2. Without the yarn geometry the weaving pattern looks like a checkerboard pattern, thus the yarn geometry has to be defined for close-up viewpoints.

3.4 Yarn Geometry

The yarn geometry is generated after the weaving pattern by normal perturbation at run-time. Kang [Kan10] proposed that the yarn geometry can be specified by several parameters including: number of threads for each strand of yarn N_ξ, fiber curvature of each strand c_f, and the yarn curvature c_y. Therefore, the normal of each point is altered using these parameters and its sampling position on the weft or warp yarn [Kan10].

$$\text{weft:}(\delta x, \delta y) = (c_f(2\text{fract}(N_\xi(u_w - s_f\sigma^u)) - 1, c_y\sigma^v) \quad (3)$$

$$\text{warp:}(\delta x, \delta y) = (c_y\sigma^u, c_f(2\text{fract}(N_\xi(v_w - s_f\sigma^v)) - 1) \quad (4)$$

Equations 3 and 4 show the changes made to the x- and y-coordinates of the normal. The z-coordinate of the perturbed normal is generated by calculating $\sqrt{1.0 - \delta x - \delta y}$. This achieves yarn based lighting by taking into account different user-defined parameters including: fiber curvature (c_f), yarn curvature (c_y), slope of fibers (s_f), offsets in yarn space (σ^u and σ^v), and number of fiber strands (N_ξ) used to make up each yarn. The variables u_v and v_w are texture coordinates of the model, and the function fract() calculates the fraction component of a floating point.

Figure 3 shows a fabric model with weaving pattern and yarn geometry. The image on the left of Figure 3 shows a rendering of the fabric seen from a distance, with the microscale details well below an individual pixel size. The results are not significantly different to the version of rendering without yarn geometry as shown in Figure 2. The difference between the two fabric models becomes more visible when rendering a close-up view. The image on the right of Figure 3 illustrates that each yarn is constructed by several threads, as specified by the variable N_ξ, with a high yarn curvature resulting in large shaded areas on the sides of the yarns.

4 DESIGN

Fabric rendering techniques often suffer from strong aliasing effects if the resolution of the fabric microstructure is higher than the screen resolution it is displayed on. While post-processing can soften these effects, the solution is inefficient and artifacts can remain, especially for interactive applications such as cloth simulations. Some solutions use large weave sizes to reduce aliasing effects, but this is not suitable for fashion applications where realism is essential. We analyzed existing cloth rendering techniques and found that the method from Kang [Kan10] is the most promising one [YW11]. A major advantage of this algorithm is its speed, which was shown to be only 1.846 time more expensive than Gouraud shading [Kan10]. The approach also displays a high level of realism, as the results of rendered woven fabrics look realistic in close-up. However, this approach lacks a coloring scheme at the yarn level to render fabrics such as denim, and it also displays blatant aliasing artifacts on the fabric surface.

4.1 Level-of-detail Fabric Rendering

We propose a level-of-detail design for the fabric model proposed by Kang [Kan10], which removes unnecessary detail, and only renders the detailed fabric structure in close-up views.

Our design consists of two levels of visually perceivable fabric details: weave structure and yarn geometry. The visibility of yarn geometry is determined by the mipmap level, which is calculated with the use of derivative functions on texture coordinates. We explicitly render two mipmap levels for the general weave structure and the underlying yarn geometry. We limit the mipmap levels to between 0 and 1, and uses it as an alpha value for blending between the two layers of mipmap. This concept is shown in Figure 4, for those fragments that are highlighted in lighter colors, they are rendered with the general weave structure, whereas for those that are highlighted in darker colors, they are rendered with more detailed yarn geometry. This means that if the texture coordinates between neighboring fragments are changing quickly, then a higher level

Figure 3: Rendering of weave pattern with yarn geometry seen from distance (left), and from close-up (right).

mipmap (less detail) is used, thus avoiding the rendering of unnecessary detail when the fabric is observed in a larger distance.

Figure 4: Visualization of mipmap level selection from far view point to close view point (left to right).

In essence, two MDFs are calculated, one using the pre-perturbed normals for weaving pattern rendering, and the other using the perturbed normals for yarn geometry rendering. The yarn geometry is obtained from the normal perturbation using Equations 3 and 4, which is then used as an input to the MDF. In practice, however, only the dot product between the halfway vector h and the normal vector n has to be recalculated two times for each values, with the rest of the calculations in Equation 2 only calculated once. Equation 5 shows the calculation of the final MDF, which is done by using the two previously mentioned MDFs, and weighting them with the α value obtained from the mipmap calculation that determines which level of mipmap should be used for rendering.

$$p(h) = (1.0 - \alpha)p(h_1) + \alpha p(h_2) \tag{5}$$

4.2 Extensions for Real-time applications

We also extend the model developed by Kang [Kan10] to support more types of fabrics.

4.2.1 *Extended Fabric Coloring Scheme*

For our system, we require the visualization of fabrics such as denim. Denim is often constructed from fiber using the twill-weaved weaving pattern. In contrast to ordinary cotton twill, denim uses different colors for the weft and warp yarn, with the most popular combination being white for the warp yarn and blue for the weft yarn.

We define extra parameters to specify the base color of individual weft and warp yarns, both in terms of diffuse and specular colors. Using these base colors, we apply Kang's algorithm [Kan10] to generate the procedural textures with weaving patterns and yarns to simulate the virtual fabrics.

4.2.2 *Ambient Occlusion*

The original method by Kang [Kan10] introduced an ambient occlusion term defined by the z value of the perturbed normal. Since the perturbed normal is generated to define the yarn geometry at the micro-level, the ambient occlusion term only works for scaling the anisotropic reflection at the yarn level to create a shadow effect on the reflection.

The self-shadowing effects at a higher level are not captured due to the lack of indirect lighting involved in calculating the overall reflectance of the fabric. However, self-shadowing is common in practice, e.g. when cloth is folded. Hence, we use Screen-Space Ambient Occlusion (SSAO) [Mit07] as a real-time ambient occlusion method to introduce self-shadowing effects to the existing fabric rendering method.

In the initial pass, we render our fabric at the same time as normal buffer and position buffer to avoid rendering the same object in multiple passes. We store fabric rendering results in a color buffer, and the color buffer is referred to for scaling its values using the calculated

ambient occlusion from the normal buffer and position buffer.

$$ao = max(0.0, \frac{dot(N,V)}{1.0+d}) \quad (6)$$

Equation 6 describes the calculation of the ambient occlusion of a pixel. A point (occluder) occludes another point (occludee) if the dot product between the normal of the ocludee and the vector from the ocludee to occluder is greater than zero, i.e if the point is located at the front face of the occludee, then it contributes some amount of occlusion scaled by the dot product and the distance between two points. In order to calculate the ambient occlusion at each pixel of the screen, neighboring pixels are sampled randomly using a noise function and the ambient occlusion value is averaged according to the number of sample points [Mit07].

4.2.3 Anti-Aliasing

The original implementation of the renderer produced clearly visible aliasing effects and moire patterns due to high frequency textures. These artifacts are caused by the microscopic details of the procedural textures generated in real-time. When the object is in motion, the aliasing artifacts become even more visible to temporal aliasing.

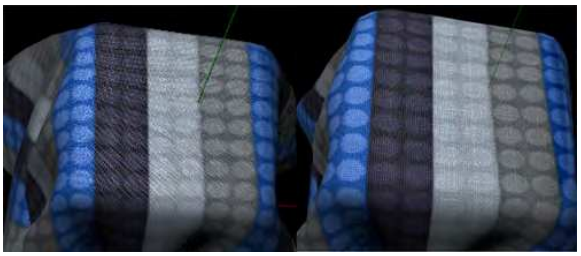

Figure 5: Aliasing of weave patterns. Comparison of fabric being viewed from a distance (left) and from close-up (right). The fabric was rendered with the original implementation by Kang [Kan10] without any anti-aliasing or level-of-detail algorithm

The left image of Figure 5 displays the distortion of weaving patterns when the viewpoint is located far away from the object. The moire patterns on the surface of the fabric are clearly visible as distorted curve lines. The fabric on the right in Figure 5 shows the weaving pattern when the viewpoint is at a closer distance to the object - no moire pattern is visible. Another example is given by Figure 6, which shows a denim fabric rendered without any underlying textures, but using blue colored weft yarns and white colored warp yarns. When the fabric is viewed from a distance (left image of Figure 6), the aliasing artifacts are more visible with this fabric due to the highly contrasted yarn colors between weft and warp yarns, also causing moire patterns to appear on the surface of the fabric. Furthermore, the twill-weave on the denim fabric is clearly distorted and unrecognizable from this distance. The aliasing artifacts are significantly reduced on the right of Figure 6, but still exist in high frequency areas such as regions where the fabric is bent around the underlying cuboid object.

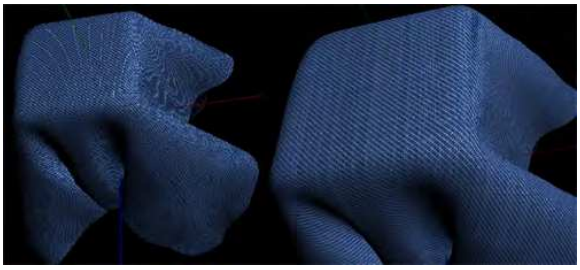

Figure 6: Aliasing of weave patterns of denim fabric. Comparison of distant viewpoint in magnified view (left) and close viewpoint (right)

An anti-aliasing method is required to reduce the moire effects on the surface of the fabric. While a level-of-detail scheme was proposed in Section 4.1, the size of the weaving patterns still introduces a high level of texture frequency on the fabric surface.

Traditionally, oversampling methods such as supersampling anti-aliasing (SSAA) and its variation, multisampling anti-aliasing (MSAA) were used to handle aliasing for graphics applications and computer games. SSAA is known to incur large bandwidth and shading cost, due to multiple numbers of samples being taken inside each pixel [HA90, Mam89], while MSAA improves on SSAA's performance by only evaluating each fragment value once and only supersampling depth and stencil values [Ake93].

Recently, post-processing anti-aliasing methods have become more popular for reducing aliasing artifacts. Methods such as morphological anti-aliasing (MLAA) [Res09] and fast approximation anti-aliasing (FXAA) [Lot09] have the advantage that they are independent to the rendering pipeline. The methods are applied at the last stage as an image-based post-processing technique. While MLAA resolves many edge aliasing problems, it is unable to handle pixel-sized features, and fails to reduce the moire-effects from high frequency textures [Res09]. FXAA employs a similar anti-aliasing procedure, where the image is used to detect edges using highly contrasting areas of each pixel, with an additional sub-pixel offset from the detected edge for low-pass filtering to achieve anti-aliasing in the sub-pixel level [Lot09]. Therefore, FXAA can be considered as a sub-pixel anti-aliasing technique and is hence potentially useful for our fabric shader. However, MSAA is the preferred choice due to its proven anti-aliasing performance over the entire object surface.

Despite the popularity of these methods, we found that they did not alleviate the high frequency aliasing prob-

lem we faced with texture aliasing from our implementations. Therefore, we decided to simply use an adaptive prefiltering approach [Ros05] inside our GLSL shader for averaging the pixel with its neighbors. The filter is adaptive such that the number of surrounding colors it calculates depends on the degree of similarity in each iteration. This algorithm is shown in Algorithm 1, which shows an overview of the algorithm for each fragment in calculating its final color. Essentially, the final color is iterated until its difference with other colors between neighboring fragments is less than a threshold, defined by the inverse distance of the fragment from the view point.

Algorithm 1 Prefiltering for fabric shader

```
count ← 1
ddx ← dFdx(worldPos) * 0.5
ddy ← dFdy(worldPos) * 0.5
while true do
    lastColor ← color
    color ← color + calcFragmentColor(worldPos
+ ddx * rand() + ddy * rand())
    count ← count + 1
    if count > 5 then
        δcolor ← lastColor − color
        if length(δcolor) < (1 / viewDistance) then
            break
        end if
    end if
end while
finalColor ← color/count
```

5 RESULTS

This section evaluates the effectiveness and efficiency of the improvements proposed by us. The following tests were performed:

- LOD fabric rendering quality test
- LOD fabric rendering performance test
- Denim Rendering

With rendering quality, we compare and contrast the quality of several images with real fabrics. To compare the effects of using level-of-detail rendering, we compare rendering results with and without the improvements, and we also compare their effects on aliasing artifacts. For rendering performance we compare the frame rates achieved with the different implementations. All tests were performed on a PC with Intel Core i7 2600k,12 GB memory at 1600 MHz with an AMD Radeon HD 6950 graphics card with 2GByte GPU memory.

5.1 Level-of-Detail Rendering

5.1.1 Rendering Quality

The level-of-detail rendering of woven fabric was tested by comparing the rendering quality of fabrics with and without our level-of-detail fabric rendering algorithm. Figure 7 shows that without level-of-detail rendering (left) many aliasing artifacts are seen, which they are not visible using level-of-detail rendering (right). The observations are confirmed by a second example shown in Figure 8, which represents a red twill-weaved woven fabric.

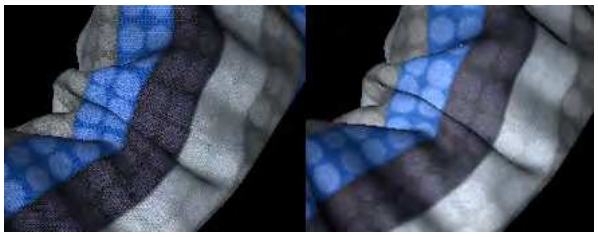

Figure 7: Level-of-detail rendering, without LOD (left) and with LOD (right).

Figure 8: Level-of-detail rendering, without LOD (top) and with LOD (bottom).

The denim fabric still displays some aliasing artifacts with high frequency weaving pattern that is rendered on polygons facing away from the screen. This is due to the high contrast in color in the underlying weaving construct, coupled with the projection of weaving pattern to a much smaller projected area, thus making it difficult to smooth the high frequency changes in color at the micro-level. An example of this problem is shown in Figure 9, where the left image is a magnification of the small rectangle section in the right image. The left image illustrates some aliasing artifacts close to the edge of the fabric, with white lines going against the flow of the twill-weaved weaving pattern. These artifacts are not clearly noticeable in static images, but they

Implementation	Performance (ms)	Std. dev (ms)
Original	5.9302	0.0346
LOD	4.4037	0.04115

Table 1: Table comparing performance and standard deviation of the two algorithms in milliseconds per frame.

become very conspicuous in temporal aliasing when the fabric is in motion.

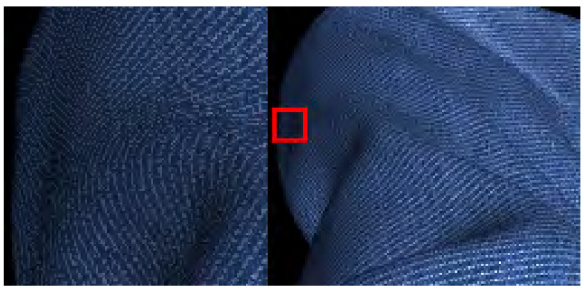

Figure 9: An example of aliasing problem due to texture projection. The left image is a magnification of the red square in the right image.

5.1.2 Performance Analysis

The performance of the two algorithms (the original algorithm and the LOD algorithm) was analyzed using an identical camera position and view, cloth model, texture, and input parameters for the rendering model. Both algorithms were tested along with the prefiltering approach, as we have decided to incorporate it to reduce the procedural texture aliasing.

Table 1 shows the results of the two algorithms, performed with a 3D model as shown in Figures 7 and 8, which contains 18892 vertices and 37566 triangles. Our level-of-detail fabric rendering algorithm is faster than the original algorithm. Given the improved rendering quality and reduced artifacts our new approach is hence preferable.

5.2 Denim Rendering

By extending the model to allow the specification of coloring of yarns, we managed to procedurally generate fabrics such as denim using the twill-weaved weaving pattern coupled with blue colored weft yarns and white colored warp yarns. Figure 10 provides a comparison between our results and a close-up of real worn denim fabric. The rendering results look realistic in terms of fabric structure, but more randomness and tear and wear needs to be incorporated into our model in the future.

We adopted the noise function proposed by Kang [Kan10]. The noise function generates random illumination at different yarns, which enhances the realism of our rendering. Note that some white warp yarns look brighter than others as is the case for real denim. So far our framework does not simulate washed denim fabric

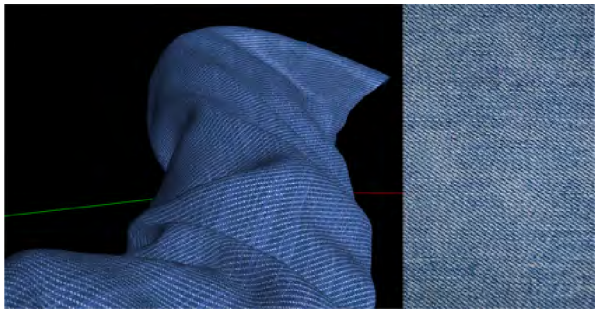

Figure 10: Denim fabric rendered with our framework (left) and a photo of real denim (right).

and hence we cannot replicate the random whitish patches in the image of real denim fabric.

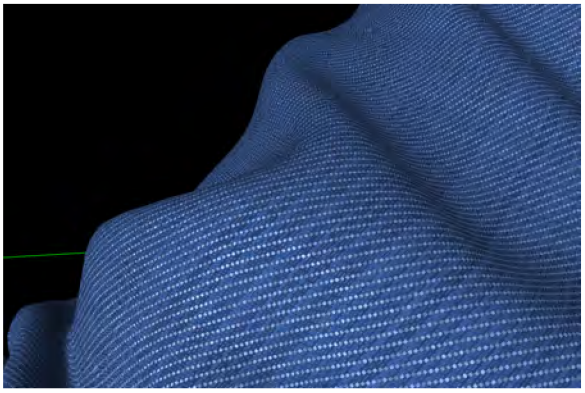

Figure 11: Close-up view of denim fabric rendered with our framework.

Figure 11 shows a close-up view of our rendering results. Our model renders the twill-weave too uniformly, lacking the natural imperfection found in real denim. When the denim is being viewed from a larger distance, the yarn geometry and weaving pattern gets aggregated and only the dominating color in the weaving pattern is visible to the user.

Figure 12 shows another close-up appearance of the denim fabric using a model of a pair of jeans.. The weaving pattern is defined to closely simulate the structure of a pair of real jeans. In order to render jeans realistically, we modified our fabric shader so that it supports the input of wash maps. Wash maps are used to define the patterns of washes for a pair of jeans, where washes are patterns of white patches on the jeans as shown in Figure 12. In Figure 12, a pair of rendered jeans is shown on the left with a pair of real jeans on the right. This close-up view demonstrates that the weaving pattern of the rendered jeans closely resembles the weaving pattern of the real jeans, as they are both similar in structure and size relative to the jeans model. Furthermore, at this viewing distance, the appearance of both jeans is very similar to each other.

A comparison of an entire pair of real jeans and our rendered jeans using a distance view is shown in Figure 13.

Figure 12: Jeans comparison when viewed from close-up: rendered jeans (left) and real jeans (right).

The rendered jeans (left) in Figure 13 closely resembles the real jeans (right) in Figure 13. From this distance, the weaving pattern is completely invisible to the observer, and aliasing artifacts are also unrecognizable on the fabric surface with the use of our LOD algorithm and prefiltering approach.

Figure 13: Jeans comparison when viewed from a distance: rendered jeans (left) and real jeans (right).

In our results, we found that that the use of direct lighting often makes the resulting rendered object too bright in areas that are occluded by the object itself. An example is shown in Figure 14, where the left image is the rendered jeans without ambient occlusion, and the right image is the rendered jeans with ambient occlusion. In this scene, the light is positioned to the left of the jeans, hence the inner thigh area should be dark as it is not directly illuminated by the light source. However, without ambient occlusion the rendered jeans still seems to be too bright around this area, and we found that the SSAO approach results in a more natural appearance of occluded areas.

Figure 14: Effect of ambient occlusion, before ambient occlusion (left), and after ambient occlusion (right)

6 CONCLUSION

In this paper, we analyzed several existing fabric rendering techniques. We chose to use the method proposed by Kang [Kan10] as a basis for our fabric shader, due to its rendering quality and performance. Several extensions were proposed to improve the robustness of the model and for supporting fabrics such as denim, and

ambient occlusion for enhancing the realism of self-occlusion of the cloth model. Furthermore, we proposed a level-of-detail approach in visualizing the aggregation of details with the use of a mipmap LOD selection mechanism, to help reduce aliasing artifacts resulting from high frequency textures. Overall, our extension to the model enabled us to successfully render denim fabric with an unwashed look and it significantly reduced aliasing problems. With the incorporation of wash maps to specify areas of washes, we have successfully replicated the overall and close-up appearance of actual jeans.

7 FUTURE WORK

The weave-based level-of-detail algorithm only reduces parts of the aliasing caused by high frequency textures. It still suffers from aliasing from small scaled weaving pattern and highly contrasting weft and warp yarns' colors, such as for denim fabric, depending on the size of weft and warp segments. Our approach rectified the aliasing problem that is often seen in weave-based fabric rendering approaches, but a better algorithm can be investigated in the future to reconstruct the appearance of high-detail level from lower levels, rather than filtering these details away.

8 REFERENCES

[Ake93] Kurt Akeley. Reality engine graphics. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '93, pages 109–116, New York, NY, USA, 1993. ACM.

[AMTF03] N. Adabala, N. Magnenat-Thalmann, and G. Fei. Real-time rendering of woven clothes. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 41–47. ACM, 2003.

[AS00] M. Ashikhmin and P. Shirley. An anisotropic phong BRDF model. *Journal of graphics tools*, 5(2):25–32, 2000.

[DLH01] K. Daubert, H.P.A. Lensch, and W. Heidrich. Efficient cloth modeling and rendering. In *Rendering techniques 2001: proceedings of the Eurographics workshop in London, United Kingdom, June 25-27, 2001*, page 63. Springer Verlag Wien, 2001.

[HA90] Paul Haeberli and Kurt Akeley. The accumulation buffer: hardware support for high-quality rendering. *SIGGRAPH Comput. Graph.*, 24(4):309–318, September 1990.

[Ira08] Piti Irawan. *Appearance of woven cloth*. PhD thesis, Ithaca, NY, USA, 2008. AAI3295837.

[Kan10] Y.M. Kang. Realtime rendering of realistic fabric with alternation of deformed anisotropy. *Motion in Games*, pages 301–312, 2010.

[Kau05] J. Kautz. Approximate bidirectional texture functions. *GPU Gems*, 2:177–187, 2005.

[Lot09] T. Lottes. FXAA. 2009. Also available as `http://developer.download.nvidia.com/assets/gamedev/files/sdk/11/FXAA_WhitePaper.pdf`.

[Mam89] Abraham Mammen. Transparency and antialiasing algorithms implemented with the virtual pixel maps technique. *IEEE Comput. Graph. Appl.*, 9(4):43–55, July 1989.

[Mit07] Martin Mittring. Finding next gen: Cryengine 2. In *ACM SIGGRAPH 2007 courses*, SIGGRAPH '07, pages 97–121, New York, NY, USA, 2007. ACM.

[Res09] A. Reshetov. Morphological antialiasing. In *Proceedings of the Conference on High Performance Graphics 2009*, pages 109–116. ACM, 2009.

[Ros05] R.J. Rost. *OpenGL (R) shading language*. Addison-Wesley Professional, 2005.

[War92] G.J. Ward. Measuring and modeling anisotropic reflection. *ACM SIGGRAPH Computer Graphics*, 26(2):265–272, 1992.

[WZT+08] J. Wang, S. Zhao, X. Tong, J. Snyder, and B. Guo. Modeling anisotropic surface reflectance with example-based microfacet synthesis. In *ACM SIGGRAPH 2008 papers*, pages 1–9. ACM, 2008.

[YW11] W. Yuen and B. Wünsche. An evaluation on woven cloth rendering techniques. In *Proceedings of the 26th International Image and Vision Computing New Zealand Conference (IVCNZ 2011)*, pages 7–12, Auckland, New Zealand, November 2011. Also available as `http://www.cs.auckland.ac.nz/~burkhard/Publications/IVCNZ2011_YuenWuensche.pdf`.

[YYTI92] T. Yasuda, S. Yokoi, J. Toriwaki, and K. Inagaki. A shading model for cloth objects. *Computer Graphics and Applications, IEEE*, 12(6):15–24, 1992.

[ZJMB11] S. Zhao, W. Jakob, S. Marschner, and K. Bala. Building volumetric appearance models of fabric using micro ct imaging. *ACM Trans. Graph*, 30(44):1–44, 2011.