Pacific Graphics 2020
E. Eisemann, A. Jacobson, and F.-L Zhang
(Guest Editors)

Volume 39 (2020), Number 7

Automatic Band-Limited Approximation of Shaders Using Mean-Variance Statistics in Clamped Domain

Shi Li[1], Rui Wang[1†], Yuchi Huo[1], Wenting Zheng[1], Wei Hua[2,1], Hujun Bao[1]

[1] State Key Lab of CAD&CG, Zhejiang University, [2]Zhejiang Lab

Abstract

In this paper, we present a new shader smoothing method to improve the quality and generality of band-limiting shader programs. Previous work [YB18] treats intermediate values in the program as random variables, and utilizes mean and variance statistics to smooth shader programs. In this work, we extend such a band-limiting framework by exploring the observation that one intermediate value in the program is usually computed by a complex composition of functions, where the domain and range of composited functions heavily impact the statistics of smoothed programs. Accordingly, we propose three new shader smoothing rules for specific composition of functions by considering the domain and range, enabling better mean and variance statistics of approximations. Aside from continuous functions, the texture, such as color texture or normal map, is treated as a discrete function with limited domain and range, thereby can be processed similarly in the newly proposed framework. Experiments show that compared with previous work, our method is capable of generating better smoothness of shader programs as well as handling a broader set of shader programs.

1. Introduction

GPU shader program plays an important role in the real-time rendering, but it often suffers from aliasing and local flickering due to insufficient samples. An accurate computation of shading value requires an integral over the pixel footprint, which is usually time-consuming and inapplicable to real-time applications. For real-time rendering, one basic antialiasing technique is pre-filtering, which stores precomputed integrals or function parameters beforehand and uses them at runtime. MIP-mapping [Wil83] and summed area tables (SATs) [Cro84] were introduced to linearly pre-filter static textures as lookup tables. However, linear pre-filtering is inaccurate for nonlinear functions, especially when the rendering equation [Kaj86] is a high-dimensional function combining normal, lighting, view, and visibility factors. Numerous researches [Tok05, HSRG07, TLQ*08, OB10, NPS13, DHI*13, KHPL16, XWZB17, WZYR19] have been proposed to filter different components in Rendering Equation [Kaj86]. Among these work, Gaussian or Gaussian-like functions have been widely applied to approximate functions. In some cases, its moments are also used to characterize functions with linear parameters [OB10, DHI*13]. Nevertheless, these researches are mainly designed to approximate a specific equation or distribution. It is generally difficult to apply these approaches to other complex functions or a composition of multiple functions.

One shader is a program with a complex composition of multiple functions. To tackle the challenge of antialiasing shader program, Dorn et al. [DBLW15] and Yang and Barnes [YB18] recently presented an automatic scheme to construct analytical band-limited version of shader and produce smooth results. Technically, it breaks the procedural shader program into sub-part programs and successively smooths each sub-part. Mean-variance statistics are applied to approximate the convolution of the sub-part program with a kernel function using composition rules. However, such an approach has several limitations. First, it individually approximates sub-parts and assumes that each of them is defined in the entire domain. Nevertheless, in the shader, the domain of one sub-part program is usually influenced by other sub-parts. The inaccurate consideration of domain and range of functions in the sub-part program would bring large approximation errors, sometimes wrong results (Figure 2). Second, the kernel function, such as Gaussian function, used to filter shader program is also assumed to be defined in the entire domain, which may bring inaccurate mean or variance and result in large approximation errors as well. Last but not the least, their work only considers analytic functions, excluding textures that store discrete values. Given the popularity of textures in shaders, the lack of support of textures limits the usage of their work in many applications.

In this paper, we extend the shader smoothing framework with mean-variance statistics by taking the range and domain into consideration. We observe that final or intermediate values in the program are usually computed by a complex composition of functions. Therefore, the statistics of smoothed programs could be bet-

† Corresponding author: rwang@cad.zju.edu.cn

ter depicted by considering the domain and range of the composited functions. Accordingly, we introduce three approximation rules to explore such an observation. The first approximation rule is derived from the adaptive Gaussian and compactly supported kernels approximation rules [YB18], but with renormalized kernel functions in the clamped domain. The second approximation rule considers the range of intermediate variables and utilizes Gaussian functions to fit them in the clamped domain. These two rules improve kernel functions to tighten the approximation in the clamped domain. However, not all functions could be analytically smoothed by Gaussian kernels [YB18]. To smooth complex functions, we then propose the third rule that first uses polynomials as basis functions to approximate it in the clamped domain and then filter each basis function obtaining the band-limited output. Besides these approximations on continuous functions, texture, such as color texture and normal map, is regarded as discrete functions with finite ranges and can be specially handled in our framework. Results show that our approach can be integrated into the shader smoothness scheme proposed by Dorn et al. [DBLW15] and Yang and Barnes [YB18], and provide better band-limited shaders.

Our contributions are summarized as follows:

- An extended mean-variance statistics framework, which specifically considers the domain and range of functions, thereby provides better approximations of the smoothed shading values over the pixel footprint (Section 3.1);
- Three new approximation rules to reduce approximation errors at smoothing shaders (Section 3.2-3.4);
- The support of textures in the band-limited shaders (Section 4.3).

2. Related Work

Antialiasing techniques play an important role in rendering details accurately and efficiently in computer graphics. Super-sampling [AGB00] and stochastic sampling [Cro77] are two common ways to achieve this, but with high costs. Specific antialiasing techniques are then proposed for different types of functions in rendering for better efficiency.

Texture filtering. MIP-mapping [Wil83] uses static footprint given by a quadtree for static texture. SATs [Cro84] stores the sum of all the values in the rectangles from the bottom left corner to the current texel, which is more flexible than MIP-mapping but needs extra storage for better precision. Heitz et al. [HNPN13] utilized Gaussian functions to estimate height/slope-correlated color to obtain smooth color mapped surface. However, these traditional filtering techniques only target to linear filtering. Belcour et al. [BSS*13] traced 4D Gaussians in the path space to render effects that require costly 5D integrals, such as motion blur and depth-of-field.

BRDF/Normal map filtering. Since BRDF/normal map has great significance for enhancing realism, the filtering algorithms for BRDF/normal maps have always been an important research topic. Normal map filtering algorithms [Tok05, HSRG07, TLQ*08] express normal distribution function (NDF) as a linearly interpolable representation as the probability distribution of the normal in the footprint. Marc Olano and Dan Baker [OB10] adopted a 2D

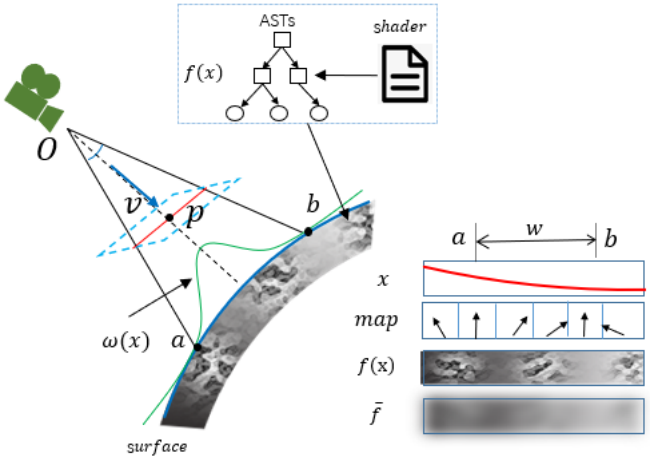


Figure 1: *The band-limited shading problem.*

Gaussian in a tangent plane and linearly MIP-mapped and interpolated its first and second moments. Jonathan Dupuy et al. [DHI*13] proposed a technique adapted to the reflectance of displacement maps. They modeled physically-based microfacet BRDFs by non-centered Beckmann distributions and derived an analytical filtering solution under directional/point lighting. Recently, Jakob et al. [JHY*14] and Yan et al. [YHJ*14, YHMR16] introduced two different approaches for rendering glints on discrete and spatially varying mesoscale NDFs. Xu et al. [XWZB17] MIP-mapped the von Mises-Fisher (vMF) lobes in an unnormalized vector space to achieve the linear interpolation of BRDFs. Wu et al. [WZYR19] presented a prefiltering technique to preserve appearance details for rendering displacement-mapped surfaces.

However, all aforementioned methods focus on filtering specific distribution functions, while our approach addresses shader programs, which are more complex and usually with general functions.

Shader filtering. Velazquez-Armendariz et al. [VAZH*09] adopted interval arithmetic to automatically generate interval versions of programmable shaders that can be used to bounding query and guaranteed quality importance sampling. Dorn et al. [DBLW15] and Yang and Barnes [YB18] extended the automatic shader generation idea to automatic filter procedural shader. The main idea is to break the computation into different subparts and compute approximate mean and variance statistics for each sub-part. With three approximation rules, optimal smoothed shaders are explored to reach the best trade-off between running time and error.

Nevertheless, there are remaining problems that need to be solved. Firstly, since the domain of a sub-part function will be influenced by the value range of its inner sub-part functions, the approximation results may fall out of the range of the function composition. Secondly, their work only considers continuous functions, excluding discrete shading values stored in textures. In this work, we present a new method to tackle these problems.

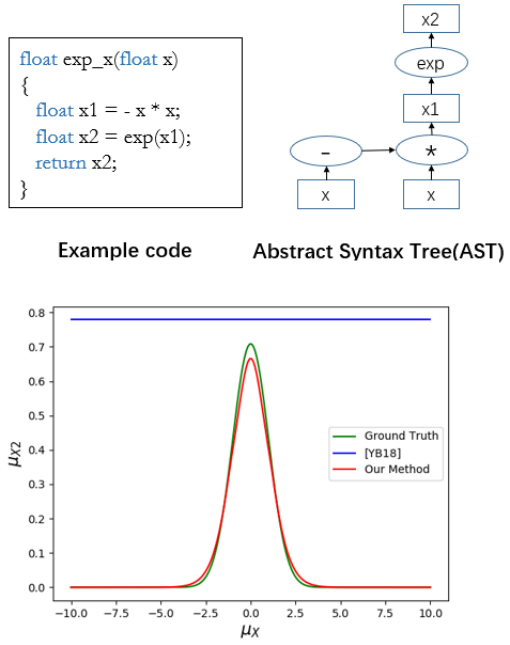


Figure 2: *An example code, the abstract syntax tree (AST) and the band-limited result of x2 when* $\sigma_X^2 = 0.5$.

3. Mean-Variance Statistics in Clamped Domain

In this section, we present the problem and then introduce novel approximations proposed in this paper.

3.1. Problem Statement

This work pursues the band-limited version of an arbitrary program (function) $y = f(x)$. For real-time rendering, the target program is the shader program, $f_s(x)$, which provides fine shading details of the surface. Without loss of generality, we use $f(x)$ to represent an arbitrary program and discuss the general approach to smooth it.

Figure 1 illustrates a one-dimension example of the problem, where O is the camera viewpoint, v is the view direction, p is a pixel in screen space, x is the surface signals (e.g., normals) that can be represented by continuous geometry shapes or discrete texture maps, the region from point a to b is the footprint of pixel p on the surface, and $f(x)$ is the function to be smoothed.

The output of band-limited (smoothed) function on pixel p is expressed by $\overline{f}$ and computed as the weighted average of $f(x)$ over the domain $\overline{ab}$ on the surface:

$$\overline{f} = \frac{\int f(u)\omega(u)du}{\int \omega(u)du}, \tag{1}$$

where $\omega(x)$ is the weight function.

However, such an average is not easy to compute, especially with a complex function $y = f(x)$. Yang and Barnes [YB18] observed that while the input values of x could be depicted by mean-variance statistics, the output values y is also a distribution with mean and variance. Let us use lower-case letters such as x and y to represent real values in the original function and use higher-case letters such as X and Y to represent random variables with distributions. The mean and variance of the distribution of X is denoted as μ_X and σ_X^2, respectively. In general, the function $f(x)$ takes input random variable X with distribution f_X and outputs random variable Y with distribution f_Y. As such, the mean μ_Y can be regarded as the smoothed output. The smoothed output $\overline{f}$ of a function $y = f(x)$ can be computed as:

$$\overline{f} = \mu_Y = \int_{-\infty}^{\infty} f(u) f_X(u;\mu_X,\sigma_X^2)du. \tag{2}$$

Such an integral usually does not have a closed-form solution for a complex shader program, therefore Yang and Barnes [YB18] used composition rules to decompose the complex shader program into an AST of sub-parts then approximate each sub-part's input and output by mean-variance parameterized distribution.

Figure 2 shows an example of such a decomposition, where x, $x1$, and $x2$ are three variables, with Gaussian distributions, $X \sim N(\mu_X,\sigma_X^2)$, $X1 \sim N(\mu_{X1},\sigma_{X1}^2)$, and $X2 \sim N(\mu_{X2},\sigma_{X2}^2)$, respectively. By applying the adaptive Gaussian rule [YB18] to approximate $x2$, the smoothed value of $x2$ could be calculated as $\mu_{X2} = \exp(-\mu_X^2 - \sigma_X^2 + 2\mu_X^2\sigma_X^2 + \sigma_X^4)$.† However, Yang and Barnes' approximation [YB18] has a problem. When $\sigma_X^2 = 0.5$, no matter what the value of μ_X is, the mean of $X2$ is a constant $\mu_{X2} = \exp(-0.25)$. Such a bad approximation is because of the neglect of the integral domains. While the integral domain of $x2$ is considered as $(-\infty,+\infty)$, the actual range of $x1$ is $(-\infty,0]$, given that $x1 = -x*x$. This example shows that the consideration of domain and range of sub-part programs is important for smoothing the shader programs with a complex composition of functions.

To address such a problem, our approach takes the domain and range of composition of functions into consideration. Specifically for the function $y = f(x)$, we assume that the domain of intermediate variable x is $[a^i,b^i]$. Therefore, the mean-variance statistics of the intermediate distribution f_Y in the clamped domain can be computed as:

$$\mu_Y = \int_{a^i}^{b^i} f(u) f_X(u)du, \tag{3}$$

$$\sigma_Y^2 = \int_{a^i}^{b^i} f^2(u) f_X(u)du - \mu_Y^2, \tag{4}$$

where μ_Y and σ_Y^2 are the mean and variance of $f_Y(y)$, f_X is the intermediate distribution of the variable X and f_X meets $\int_{a^i}^{b^i} f_X(u)du = 1$. We recursively apply this range and domain consideration during the decomposition of the shader program.

In Figure 2, we show the comparison of approximations of variable $x2$ in the example shader when $\sigma_X^2 = 0.5$. It can be seen that our method could achieve a much better approximation than that of Yang and Barnes [YB18] by considering the domain and range of composited functions.

† Please refer to the the supplementary document for the detailed derivation.

In the following sections, we first present three new approximations to optimize the mean-variance statistics from Equation 3 and 4, and then describe the algorithm integrating these approximations into an automatic band-limiting shader scheme.

3.2. Renormalized Adaptive Kernel Approximation

The first approximation is an extension of the adaptive kernel approximation [YB18], but we renormalize the probability density function in the clamped domain.

In general, we adopt common kernel functions, such as Gaussian functions or box functions, to approximate the probability density functions of intermediate variables. Let us still use $y = f(x)$ to derive the approximation, where $y \in [a^o, b^o]$, and $[a^o, b^o]$ is the range of $y = f(x)$ with respect to variable x in domain $[a^i, b^i]$.

Under such a new range of y, our key idea is to renormalize the mean-variance statistics to provide a better approximation. Technically, we assume the renormalized distribution $\bar{p}(y)$ in the clamped domain has the same mean and variance of the variable in the real domain, i.e. $\bar{p}(y) \sim f_Y(y)$, $y \in [a^o, b^o]$. Then, we derive that the renormalized adaptive kernel is a second-order accurate approximation when it meets the following equations:

$$\int_{a^o}^{b^o} \bar{p}(u)du = 1, \tag{5}$$

$$\int_{a^o}^{b^o} u\bar{p}(u)du = \mu_Y, \tag{6}$$

$$\int_{a^o}^{b^o} u^2\bar{p}(u)du = \mu_Y^2 + \sigma_Y^2. \tag{7}$$

Please refer to the supplementary document for more details.

Equation 5-7 directly give an explicit solution for compactly supported kernels, such as box functions or tent functions. However, it is difficult to obtain an explicit solution for Gaussian functions in the clamped domain. Hence, we use binary search and cross-validation to optimize the loss function L with p-norm to find the renormalized mean and variance in the clamped domain as:

$$\begin{aligned}\min_{\bar{\mu}_Y, \bar{\sigma}_Y} L = \min_{\bar{\mu}_Y, \bar{\sigma}_Y} \{ &\left\| \int_{a^o}^{b^o} u\bar{G}(u;\bar{\mu}_Y, \bar{\sigma}_Y^2)du - \mu_Y \right\|_p + \\ &\left\| \int_{a^o}^{b^o} u^2\bar{G}(u;\bar{\mu}_Y, \bar{\sigma}_Y^2)du - \mu_Y^2 - \sigma_Y^2 \right\|_p \},\end{aligned} \tag{8}$$

where $\bar{G}(u;\mu,\sigma^2)$ is a renormalized Gaussian function in the clamped domain $[a^o, b^o]$, where $\bar{G}(u;\mu,\sigma^2) = G(u;\mu,\sigma^2) / \int_{a^o}^{b^o} G(u;\mu,\sigma^2)du$, $G(u;\mu,\sigma^2)$ is a standard Gaussian function. More details can be found in Section 4.2.

3.3. Approximation of $f_Y(y)$ with Gaussian Functions

In this approximation, we try to directly use Gaussian functions to approximate $f_Y(y)$ in the clamped domain rather than renormalize the adaptive kernel approximation [YB18] in the unclamped domain.

Theoretically, if probability density function $f_X(x)$ is continuous and f is a monotonically increasing continuous differentiable function with inverse $x = g(y)$, then Y is continuous with probability density function $f_Y(y)$ given by:

$$f_Y(y) = f_X(g(y))g'(y). \tag{9}$$

To approximate $f_Y(y)$ with Gaussian function, we assume that the peak of the approximated Gaussian function is also the maximum value of $f_Y(y)$. Therefore, the mean μ_Y of the Gaussian is the extreme point of $f_Y(y)$, and μ_Y, σ_Y can be solved as:

$$\bigtriangledown_y f_Y(y)\Big|_{y=\mu_Y} = 0, \tag{10}$$

$$\sigma_Y = \frac{1}{\sqrt{2\pi} f_Y(\mu_Y)}. \tag{11}$$

Furthermore, we found that Equation 10 can be directly differentiated given Equation 9 and the assumption that $X \sim N(\mu_X, \sigma_X^2)$:

$$\bigtriangledown_y \left(\frac{1}{\sqrt{2\pi}\sigma_X} \exp\left(-\frac{(g(y)-\mu_X)^2}{2\sigma_x^2}\right) g'(y) \right)\Big|_{y=\mu_Y} = 0. \tag{12}$$

Then, Equation 12 can be simplified as:

$$(\bigtriangledown g(\mu_Y))^2 (g(\mu_Y) - \mu_X) = \sigma_X^2 (\bigtriangledown^2 g(\mu_Y)) \tag{13}$$

Taking $y = \exp(x)$ as an example, in this case, $g(y) = \ln(y)$, μ_Y can be computed by solving $\ln(\mu_Y) - \mu_X + \sigma_X^2 = 0$. The mean and variance of the approximated Gaussian function is:

$$\mu_Y = \exp(\mu_X - \sigma_X^2), \tag{14}$$

$$\sigma_Y = \frac{\mu_Y}{\sqrt{2\pi} G(\ln(\mu_Y); \mu_X, \sigma_X^2)} = \frac{\mu_Y \sigma_X}{\exp(-\frac{\sigma_X^2}{2})}, \tag{15}$$

where $G(x;\mu,\sigma)$ is a Gaussian function. In Figure 3, we show

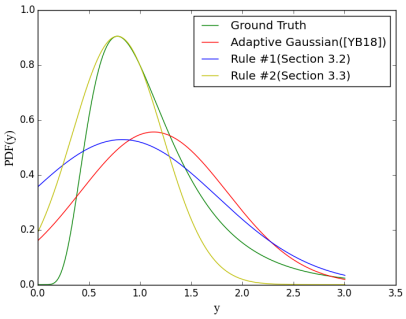


Figure 3: *The comparison of approximations of* $y = \exp(x)$*, where* $X \sim N(0, 0.5^2)$.

the comparison of this approximation and the renormalized adaptive kernels described in Section 3.2. In this example, we consider $y = \exp(x)$ and assume the probability density function of input variable is $X \sim N(0, 0.5^2)$. The green curve in the figure shows the actual probability density function of the variable Y, which is calculated by Equation 9. The red line is the Gaussian distribution that acquired by applying the adaptive Gaussian rule ($G(\mu_Y, \sigma_Y^2)$

through Equation 3 and 4). The blue line shows the Gaussian distribution ($\bar{p}(y)$) which are obtained by renormalization (Rule #1 (Section 3.2)). The yellow line shows the newly fitted Gaussian distribution as presented in this subsection (Rule #2). We use the Mean Squared Error (MSE) with ground truth as the quality metric to compare different approximations. The MSEs from approximations of adaptive Gaussian kernel [YB18], our Rule #1 and #2 are 0.033, 0.038, and 0.021, respectively. Our Rule #2 achieves the best quality in this example.

3.4. Taylor Series Approximation of $f(x)$

Previous two rules improve kernel functions to tighten the approximation in the clamped domain. However, not all functions have a closed-form at computing smoothed output using Equation 2. Therefore, we introduce another rule to simplify the $f(x)$ into polynomials, and then smooth each polynomial function, which has a closed form of smoothed function.

Specifically, second-order Taylor polynomials are used to fit the origin function $f(x)$ around the mean μ_X:

$$f(x) \approx T(x) = f(\mu_X) + \nabla f(\mu_X)(x-\mu_X) + \frac{1}{2}\nabla^2 f(\mu_X)(x-\mu_X)^2 \tag{16}$$

Equation 3 can be reformulated into the integral of Equation 16 with a probability density function $f_X(u)$ as:

$$\begin{aligned}\mu_Y &\approx \int_{a^i}^{b^i} T(u) f_X(u) du \\ &= f(\mu_X)\int_{a^i}^{b^i} f_X(u)du + \nabla f(\mu_X)\int_{a^i}^{b^i}(u-\mu_X)f_X(u)du \\ &\quad + \frac{1}{2}\nabla^2 f(\mu_X)\int_{a^i}^{b^i}(u-\mu_X)^2 f_X(u)du.\end{aligned} \tag{17}$$

Under the assumption that $f_X(u)$ is a Gaussian function, $G(u;\mu_X,\sigma_X^2)$, integrals in Equation 17 could be further computed as:

$$\int_{a^i}^{b^i} f(\mu_X)G(u;\mu_X,\sigma_X^2)du = -\frac{f(\mu_X)}{2}\operatorname{erf}\left(\frac{\mu_X-u}{\sqrt{2}\sigma_X}\right)\Big|_{u=a^i}^{u=b^i}, \tag{18}$$

$$\int_{a^i}^{b^i}(u-\mu_X)G(u;\mu_X,\sigma_X^2)du = -\frac{\sigma_X \exp\left(\frac{-(\mu_X-u)^2}{2\sigma_X^2}\right)}{\sqrt{2\pi}}\Big|_{u=a^i}^{u=b^i}, \tag{19}$$

$$\int_{a^i}^{b^i}(u-\mu_X)^2 G(u;\mu_X,\sigma_X^2)du = \left[\frac{\sigma_X(\mu_X-u)\exp\left(\frac{-(\mu_X-u)^2}{2\sigma_X^2}\right)}{\sqrt{2\pi}} - \frac{1}{2}\sigma_X^2\operatorname{erf}\left(\frac{\mu_X-u}{\sqrt{2}\sigma_X}\right)\right]\Big|_{u=a^i}^{u=b^i}. \tag{20}$$

In Figure 4, we take a function $y = \sin(0.5-0.5\exp(-2x^2))$ as an example to visualize three rules. We assume that the input variable X meets Gaussian distribution and the variance is a constant ${\sigma_X}^2 = 0.13$. We apply three rules to approximate the mean of the output variable Y. The green curve in the figure shows the actual

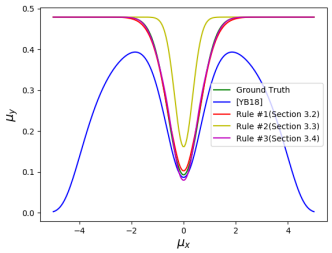


Figure 4: $y = \sin(0.5-0.5\exp(-2x^2))$, *where* $X \sim N(\mu_X, 0.13)$.

mean of the variable Y, which is calculated by the trapezoidal integration. The red line is acquired by the adaptive Gaussian rule through Equation 3 and 4. The blue line shows the band-limited results which are obtained by renormalization (Rule #1 (Section 3.2)). The yellow line shows the band-limited results which are obtained by approximation (Rule #2 (Section 3.3)). The magenta line shows the approximation presented in this subsection (Rule #3). The MSEs from approximations of adaptive Gaussian kernel [YB18], our Rule #1, #2 and #3 are 0.05, 0.000023, 0.0037 and 0.000018, respectively. Meanwhile, the running times of 200 samples are 0.4ms, 13ms, 0.4ms, and 1ms, respectively. Note that this example just exhibits differences of three rules, but is not for comparison. Although Rule #3 shows good quality and small overhead in this example, the other two rules may outperform in different cases.

4. Automatic Band-Limiting Shaders

In this section, we first introduce the overall algorithm of our framework, then present details of using the proposed three approximations to obtain optimal band-limited shaders, finally describe how textures are integrated into our algorithm as discrete functions.

4.1. Algorithm

Our method employs the band-limiting shader framework [YB18] to smooth shader programs. Mainly, it applies approximation rules on shader programs to generate band-limited shader variants, and then uses genetic programming (GP) [SAMWL11, WYY*, YB18] to find the optimal approximation choices (on Pareto frontier) that balance running time and error.

In our approach, it takes four steps to create one band-limited shader variant. Firstly, the shader code is parsed and converted into an Abstract Syntax Tree (AST). Secondly, the AST is traversed from bottom to up to calculate the range of each tree node. Thirdly, approximation rules are applied to intermediate variables on each tree node to create a band-limited shader variant. The detailed algorithm steps are shown as pseudocode in Algorithm 1. Specifically, with the AST and approximation rules, we generate the expressions of intermediate variables with their means and variances from bottom to up. The function "RULE" in Line 8 generates different code snippets according to specific rules. Please refer to the supplementary document for more details of the entire algorithm. The three

proposed approximations (Section 3.2-3.4) are regarded as Rule #1, Rule #2 and Rule #3,respectively, in the algorithm.

Algorithm 1 Automatically genarate a band-limted shader variant

Input:
The set of nodes for shaders, N_n;
The chosen rule for each node, R_n;
Output:
A band-limted shader variant, *newshader*;

```
procedure GENERATESHADERCODE()
    newshader = template header codes
    for i = 0; i < n; i++ do
        str = RULE(R_i, N_i)
        newshader += str
    end for
    newshader += template return codes
    return newshader
end procedure
```

Finally, the performance and quality of each generated shader variant are evaluated. Ray-casting technique is applied to acquire the mean and variance of input variable of tree nodes, our framework will record corresponding execution time and the quality of output image by comparing with the ground truth image, which is generated by super-sampling thousands of samples per pixel.

4.2. Approximation Rules

Three approximations introduced in Section 3 are converted into three approximation rules.

For Rule #1, we first calculate the mean and variance using the adaptive kernel approximation [YB18] and then renormalize them in the clamped domain using Equation 5-7. For compactly supported kernels, such as box functions or tent functions, an explicit function between the original and renormalized terms can be derived, which are given in the supplementary document. However, there is no analytical solution for Gaussian kernels, so we numerically compute the mean and variance of the renormalized adaptive Gaussian kernels by binary search and cross validation. Specifically, the computation follows Equation 8 with $p = 1$ at the beginning. We initialize $\bar{\mu}_Y = \mu_Y$ and $\bar{\sigma}_Y = \sigma_Y$. Then, we iteratively fix $\bar{\sigma}_Y$ to optimize $\bar{\mu}_Y$ by minimizing the loss function L in Equation 8, and fix $\bar{\mu}_Y$ to optimize $\bar{\sigma}_Y$. Usually observed in the experiments, after five iterations we obtain the renormalized Gaussian functions.

For Rule #2, we adopt the *atomic* function table in Yang and Barnes' work [YB18] and derive the analytical solution of μ_Y and σ_Y from Equation 13 for each *atomic* function. These solutions are stored as a table in preprocess and used to generate shader codes at runtime. For example, if $y = \exp(x)$, the solutions of μ_Y and σ_Y are given in Equation 14 and 15. When generating shader variants, we lookup the solutions of μ_Y and σ_Y of different *atomic* function from the table to replace the original *atomic* function in the shader program.

For Rule #3, we also create a precomputed table for second-order differential operators of *atomic* functions. While generating the band-limited code snippets, chain rules are applied to derive the differential forms of the composition of multiple *atomic* functions. We successively calculate the zero-order, first-order and second-order differential functions. Let us take $y = \exp(x^2)$ as an example. We automatically compute $t_0 = \exp(\mu_x^2)$, $t_1 = \exp(\mu_x^2) \cdot 2 \cdot \mu_x$, $t_2 = \exp(\mu_x^2) \cdot 2 \cdot \mu_x \cdot 2 \cdot \mu_x + \exp(\mu_x^2) \cdot 2$ through chain rules and substitute these three variables into Equation 18 - 20.

In the supplementary document, we provide more pseudocodes of the algorithm and several example codes of generated band-limited shader variants.

4.3. Texture

In general, texture data can be regarded as a function with discrete outputs. In this paper, we assume the data could be fitted by Gaussian or other basis functions, thereby integrated into our framework. We generally use MIP-mapping technique [Wil83] to smooth color textures. In preprocessing, we store the mean and variance into two mipmapped textures. At runtime, the mean and variance could be directly fetched from two mipmapped textures according to the size of the pixel footprint as:

$$\mu_Y = \frac{1}{2^{2n}} \sum_{i=0}^{2^n} \sum_{j=0}^{2^n} p_{ij}, \tag{21}$$

$$\sigma_Y^2 = \frac{1}{2^{2n}} \sum_{i=0}^{2^n} \sum_{j=0}^{2^n} (p_{ij} - \mu_Y)^2, \tag{22}$$

where n is the corresponding mipmap level of pixel footprint and p_{ij} is the value of color texture.

However, using textures, e.g., normal maps, in shading usually require non-linear filtering, so the traditional mipmapping technique does not work very well. Therefore, we design a specific filtering technique to handle it.

4.3.1. Normal Map in Shading

We use normal map as a proof-of-concept to discuss our filtering technique since normal map plays an important role in shading. We employ Toksvig's method [Tok05] to process normal map. It assumes the angular deviation of normals meet a Gaussian distribution whose variation can be calculated through averaged normal lengths as:

$$\sigma^2 = \frac{1 - \|N\|}{\|N\|}, \tag{23}$$

where N is the averaged normal.

Then, they used a Gaussian function to approximate the Blinn-Phong model. With the averaged normals, they provided a smoothed solution for mipmapping normal maps.

In our work, we follow a similar assumption that the angle between two vectors is a Gaussian distribution. However, we found that using the length of the averaged normal as the variance yields a better approximation of the normal distributions. Therefore, the mean and variance of normal maps are computed as:

$$\mu_\theta = \text{acos}(dot(\frac{N}{\|N\|}, v)), \tag{24}$$

$$\sigma_\theta = \text{acos}(\|N\|)/2.0. \quad (25)$$

Moreover, using NDF based on slope domain [OB10, DHI*13] gives nice band-limited results. However, it requires 2D Gaussian functions to approximate specific BRDF models, such as Blinn-Phong or Beckmann model. Therefore it cannot handle some common operations on normal maps, such as reflect vector, $R = 2 \cdot dot(N,L) \cdot N - L$, where L is the light direction. Results shown in Section 5.4 explains that our solution can not only deal with some BRDF models but also generally handle other basic operations on normal, such as dot product or multiply operations.

Some severe changes may happen in some parts of the normal map and cause large errors under our approximation. In these cases, we use multi-Gaussian lobes to improve the approximation . We partition the angular domain into $m \times n$ parts, where m and n are tile sizes of longitude and latitude coordinates, respectively. We separately compute the normal distribution in each part of the angular domain. In other words, we manually divide normals into different clusters and use one Gaussian lobe to approximate the normal distribution in each cluster, which results in a multi-lobe Gaussians approximation.

4.4. Selecting Band-limited Shaders

Our method employs genetic programming similar to that in [YB18] to automatically find the optimal shader approximation. At first, we assign our three rules to expression nodes to generate initial shader candidates. Then, in each iteration, we apply standard mutation and cross-over operations to shader candidates. Specifically, the mutation step randomly chooses a candidate and applies a new approximation rule to one expression node. The cross-over step randomly pairs shader candidates and swaps rules in their expression nodes to create offspring. Once one new generation of shader variants are generated, we measure their quality and performance, then update the Pareto frontier of shader variants. The variants on that frontier are selected as candidates for the next iteration, or outputted as the optimal band-limited shaders at the last iteration.

5. Results

We implemented the algorithm using OpenGL on a PC with an Intel Core i7-3770 3.4GHz CPU and an NVIDIA GeForce GTX 1080Ti GPU. We use Yang and Barnes' code [YB18] to reproduce their results for comparison. To evaluate the effectiveness and quality of our technique on more complex shaders, we implemented several procedural and texture-based shaders, including Cook-Torrance microfacet shading model [CT82], punctual lights and image-based lights in a PBR pipeline [PH10]. Since Yang and Barnes' method [YB18] does not support normal map and environment map, and LEAN [OB10] only supports Blinn-Phong and Beckmann distribution functions, we did not include their results in some comparisons. The ground truth image is generated by 1024× super-sampling. Mean squared error (MSE), SSIM [WBSS04] and Peak signal-to-noise ratio (PSNR) are used as error metrics to evaluate the image quality.

Function	MSE		
	Rule #1	**Yang-fract**	**[YB18]**
$\exp(-x^2)$	0.00006	0.00221	0.00099
$\sin(x^2)$	0.00065	0.00216	0.00189
$\exp(-2x^2)$	0.00019	0.00389	0.54072

Table 1: *MSE of approximations shown in Figure 5.*

In our experiments, we model each input pixel coordinate (x,y) as two independent random variables, X and Y. The means of these two variables are $\mu_X = x$ and $\mu_Y = y$, the variances of the inputs are $\sigma_X = \sigma_Y = 0.5$. The mean and variance of input buffers, such as world positions or texture coordinates, are acquired by casting three rays from one pixel to obtain the statistics.

Moreover, we employ a similar approach of Yang and Barnes' method [YB18] to consider the inputs of a binary function as two random variables and the correlation term between these two variables is usually set to zero.

5.1. Analytical Functions

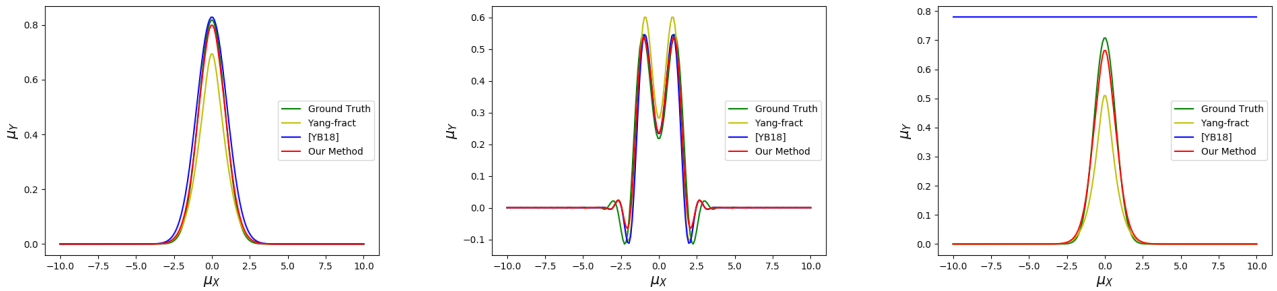
We first compare our method with the adaptive Gaussian approximation [YB18] on analytical functions. Three functions, $y = \exp(-x^2)$, $y = \sin(x^2)$ and $y = \exp(-2x^2)$, are used for comparison, where we assume $X \sim N(\mu_X, \sigma_X^2)$ and $\sigma_X^2 = 0.25$. Results are plotted in Figure 5, where the horizontal axis is μ_X, and the vertical axis is the smoothed output of y. We implement a new method named "Yang-fract" for comparison, which is directly calculating the definite integral in the clamped domain without renormalization or approximation of mean and variance. The blue lines show the results using original adaptive Gaussian approximation rules in the global domain, the yellow lines denote the results using "Yang-fract", the red lines represent the results generated by our renormalized Gaussian kernels in the clamped domain, and the green lines show the original functions as references. MSE is used to measure the error between approximations from three methods and the ground truth. Results shown in Table 1 demonstrate that our method produces more accurate approximations than those using Yang and Barnes' approximation. Note that in some cases, their approximation fails at approximating the smoothed functions, such as those shown in Figure 2 and 5c.

5.2. Simple Procedural Shaders

```
1 float f(float coord)
2 {
3 float x = fract(coord);
4 float x1 = -x * x;
5 float s = exp(10 * x1);
6 return s;
7 }
```

Listing 1: *A simple procedural shader*

We then compose two simple procedural shaders. The first shader is shown in Listing 1. It takes the world position of a pixel in and computes the shading value according to the decimal fraction of



(a) $y = \exp(-x^2)$ (b) $y = \sin(x^2)$ (c) $y = \exp(-2x^2)$

Figure 5: *Approximations of analytical functions from different rules, our renomalizated Gaussian kernel in clamped domain (Section 3.2), the adaptive Gaussian in entire domain [YB18], yang-fract using the adaptive Gaussian kernel in clamped domain, and the ground truth result using the trapezoidal integration to calculate the convolution.*

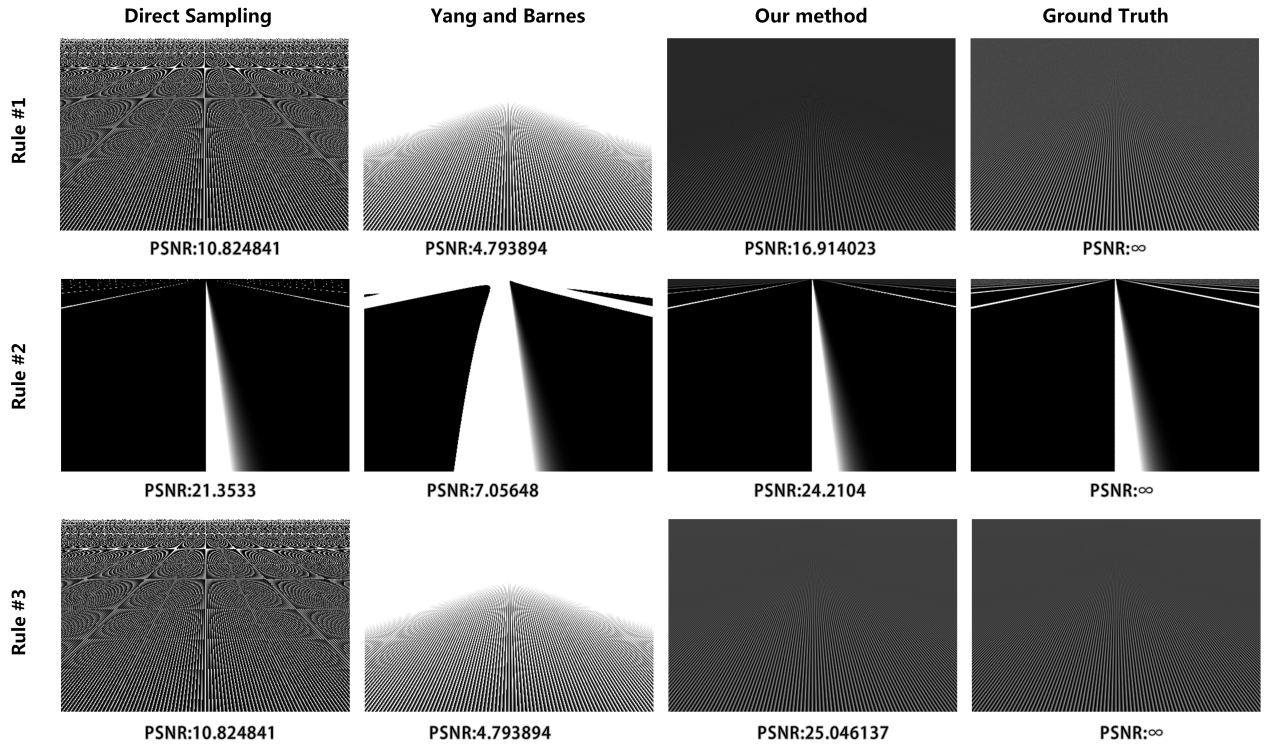


Figure 6: *Comparison of direct sampling, adaptive Gaussian approximation [YB18], our approximations and ground truth result for simple procedural shaders.*

the x coordinate. While we apply it to shade a plane, we could obtain a repetitive strip pattern (Figure 6). Without any smoothness, the direct sampling method computes only one shading sample per pixel. The result shows that it suffers from severe aliasing artifacts. We then apply the renormalized adaptive Gaussian kernel approximation (Rule #1) and the Taylor series approximation (Rule #3) to smooth this shader, and compare them with the original adaptive Gaussian kernel approximation [YB18]. As shown in the first and third row of Figure 6, the results generated by adaptive Gaussian kernels fail at producing correct smoothed results. This is mainly because it incorrectly considers the value range of $x1$. In contrast, our approximations provide much better smoothness of the shader. However, our approximations also bring some errors. The resultant images are a little bit darker than the ground truth image.

The second simple procedural shader is shown in Listing 2. It is a key component of Beckmann function, which has been widely used in real-time rendering as a normal distribution of BRDFs. We apply Rule #2 to smooth this piece of shader code. The result is shown in the second row of Figure 6. Yang and Barnes' approximation lacks the ability to control composition functions with rapidly changed

Shader	Parallax	PSNR			
		Direct	[YB18]	Ours	Gaussian
Check.	Bumps	14.99	22.54	24.21	27.41
	None	15.74	22.16	27.32	27.39
	Ripples	12.53	18.31	22.65	27.41
Circle.	Bumps	15.37	24.47	24.61	27.97
	None	16.76	27.77	29.48	29.83
	Ripples	12.87	20.08	20.30	25.67
Color.	Bumps	21.80	28.29	34.35	35.85
	None	28.58	39.17	40.20	41.41
	Ripples	17.23	19.47	25.31	31.70
Quadr.	Bumps	16.51	24.88	26.50	27.78
	None	19.21	30.87	30.94	35.39
	Ripples	12.99	18.26	22.37	25.44
ZigZag	Bumps	18.04	24.42	29.30	31.12
	None	19.04	33.09	32.83	31.71
	Ripples	15.39	18.10	25.61	28.43
Average		17.14	24.82	27.96	30.30

Table 2: *PSNR Value of Direct Sampling, Yang and Barnes [YB18], Our method, Gaussian Ground Truth with Ground Truth under Checkerboard, Circles, Color Circles, Quadratic Sine and Zigzag shader with 3 choices for parallax mapping: none, bumps, and ripples.*

value ranges. In contrast, our method successfully manages to resemble the ground truth image closely.

Then, we apply three rules onto all intermediate variables in the shader shown in Listing 1 and use the GP to find optimal smoothed shader variants on the Pareto frontier, i.e., balancing time and error. The results are shown in Figure 7 (a). It reveals that our method can generate a much better shader variant than that computed from Yang and Barnes' work [YB18].

```
1 float p_Beckmann(float coord)
2 {
3 float x = fract(coord);
4 float m = 0.02;
5 float s = exp((1.0-1.0/(x*x))/m);
6 return s / (PI * r);
7 }
```

Listing 2: *A shader with Beckmann function*

5.3. Complex Procedural Shaders

We demonstrate our method on 15 complex procedural shaders, which were produced by combining 5 base shaders (*Checkerboard*, *Circles*, *Color Circles*, *Quadratic Sine* and *Zigzag*) with 3 choices for parallax mapping: none, bumps, and ripples. All of them are from Yang and Barnes' code [YB18], but we scale the heights of bumps and ripples about three times.

All shaders are approximated by our method, Yang and Barnes' adaptive Gaussian approximation [YB18] and an ideal Gaussian smoothing. The ideal Gaussian smoothing is the convolution of one shader program with Gaussian function computed by Monte Carlo integration with about 1000 samples. It excludes the errors brought by applying our approximations, thereby can be regarded as the theoretical bound of a smoothed shader with Gaussian function. We illustrate four visual comparisons in Figure 8 and list PSNRs of different approximations for all 15 shaders in Table 2. The Pareto frontiers of optimal smoothed shaders generated by applying two optimizations are illustrated in Figure 7 (b)-(e). The results from method [YB18] are selected with the best approximation quality. Please refer to the supplementary materials for the videos and the complete results.

Results show that our method manages to produce better band-limited results than those from Yang and Barnes' work [YB18]. Firstly and most importantly, we take domain and range into consideration to exclude impossible cases in the composition rules and tighten the approximation. Let us take Blinn-Phong function, $f(x) = \text{pow}(dot(half, view), roughness)$, as an example. The *dot* has a value range of $[0,1]$, while ignoring the range may lead to overexposure or even negative values at approximation. Secondly, our approximation is able to support some functions that cannot be supported by the adaptive Gaussian in Yang and Barnes' work [YB18]. For example, $f(x) = \sqrt{x}$ is only defined on non-negative x, which could not be filtered by Gaussian kernel in Yang and Barnes' work [YB18], but in our method, it could be approximated by Gaussian in the domain $[0,\infty)$. Finally, our method may reduce the accumulative errors caused by composition rules. The mean and variance are computed for each atomic function, while the Rule #3 can approximate multiple atomic functions at one time. It also may help to handle some trouble cases using composition rules, such as $f(x) = 1/x$.

Moreover, we could observe different approximation patterns achieved by different rules in these results. In general, Rule #1 and Rule #2 produce better results for the convolution of atomic functions. They take the mean of the intermediate distribution of root nodes as smoothed output and keep the attributes of intermediate distributions. The major limitation of Rule #1 and Rule #2 is that when they are used to approximate composited functions, they may sometimes bring large variance and result in large errors. Rule #3 simplifies the function into polynomials, which may directly improve rendering efficiency and reduce the accumulative errors caused by composition rules. On the other side, it produces good approximation locally, especially around the expansion points, but may be invalid in a larger range.

5.4. Shaders with Textures

To handle shaders with textures, we demonstrate the quality and effectiveness of our method by testing it in common scenes. In Figure 9, we show the classic Teapot scene with a 1024×1024 normal map. The Blinn-Phong distribution is chosen as the normal distribution function (NDF). Results show that our method can maintain the quality of NDF filtering as LEAN mapping. However, in the Desert Rose scene with GGX distribution as NDF (Figure 7 in the

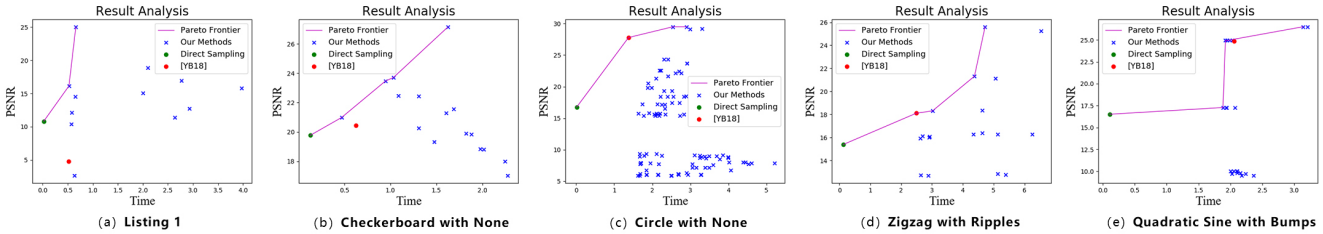


Figure 7: *Pareto frontier of generated program variants.*

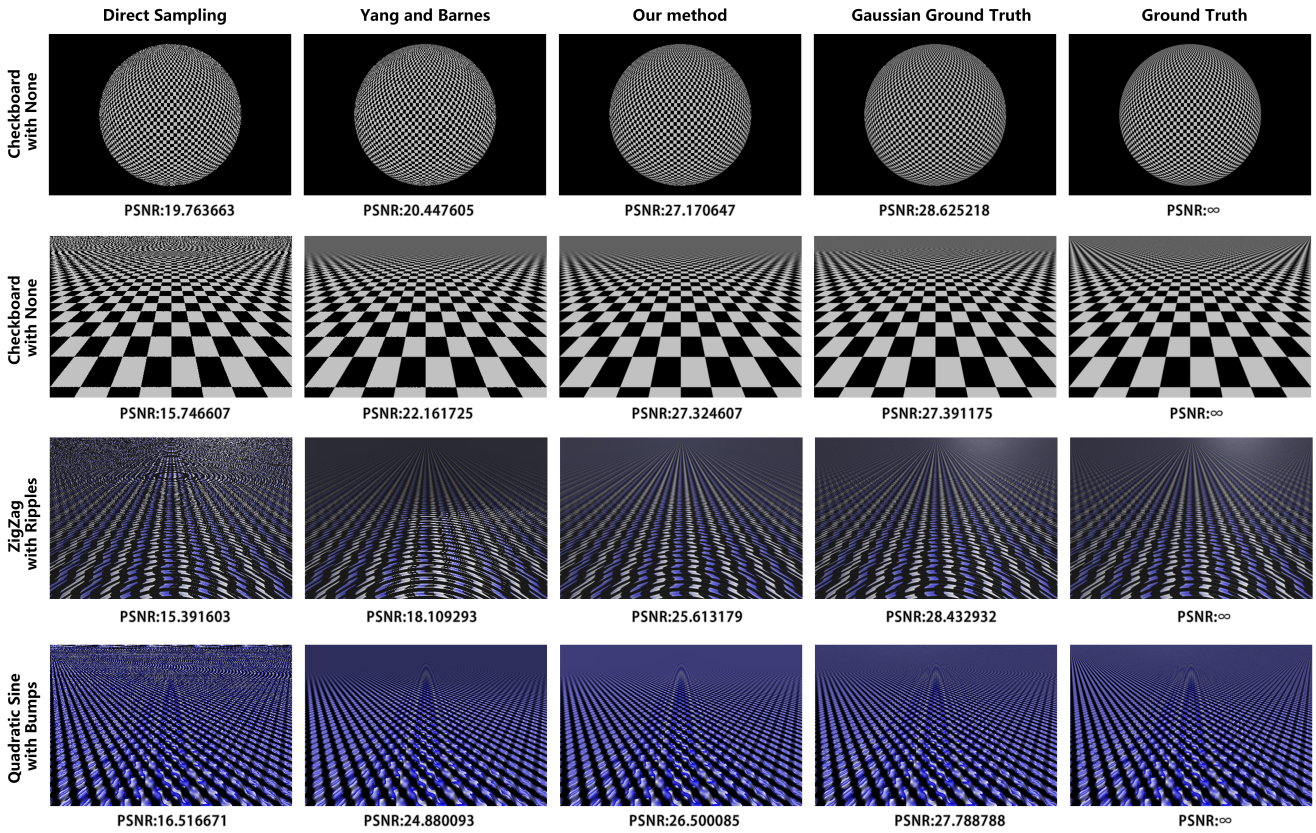
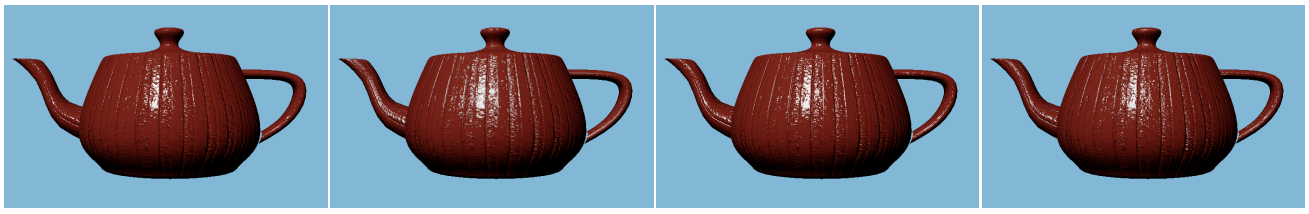
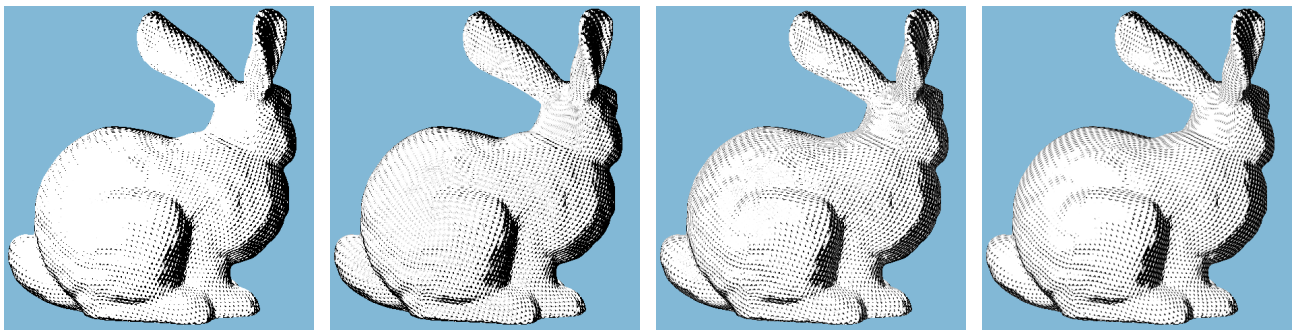


Figure 8: ***Comparison with*** *Direct Sampling, Yang and Barnes [YB18], Our method, Gaussian Ground Truth and Ground Truth under complex procedural shaders.*



(a) Direct Sampling (**0.93029**) (b) LEAN (**0.94265**) (c) Our method (**0.943403**) (d) Ground Truth (**1.0**)

Figure 9: ***Comparison of shading images*** *of the Teapot model computed by (a) Direct sampling, (b) LEAN, (c) Our method and (d) Ground truth under a physical-based shading shader with Blinn-Phong BRDF model and normal maps. SSIM values are shown in the bracket.*



(a) Direct Sampling (b) Our method with one Gaussian lobe (c) Our method with adaptive Multi-Gaussian lobes (d) Ground Truth

Figure 10: ***Comparison of visibility image*** *of the Bunny model generated by (a) Direct sampling, (b) Our method with one Gaussian lobe, (c) Our method with adaptive multi-Gaussian lobes and (d) Ground truth under a physical-based shading shader with the Smith shadowing functions and normal maps.*

Scene	Shader	Performance (FPS)				
		Direct	Toks.	LEAN	Ours	GT.
Teapot	Blinn	5900	5800	5700	5700	9
Bunny	Beck.	5700	5700	5700	5200	12
Rose	GGX	4500	4400	-	4360	4

Table 3: *Performance comparison*

supplementary document), it shows the capability of our method that can handle different operations and functions in shaders.

Finally, we include a scene with a shader having a Smith shadowing function [WMLT07]. Results are shown in Figure 10. It shows that our approximation rules have the ability to smooth very complex functions, such as the complex shadowing term.

Table 3 shows the performance in frames per second (FPS) for shaders with textures. Our method takes a little extra computation compared with LEAN mapping and direct sampling, but it preserves more details of band-limited shaders. On the other hand, our method is significantly faster than super-sampling while offers similar quality.

More results can be found in the supplementary document.

5.5. Limitations and Future Work

Our method has some limitations deserving further considerations. Firstly, our method bears the same assumption of Yang and Barnes' method that the distributions of intermediate variables are Gaussian functions. However, it may not be true in real applications. In Figure 8, we show the errors caused by such an assumption. The error between the reference and the result from the ideal Gaussian smoothing method reflects the difference between the assumed Gaussian distribution and the actual distribution. In the future, it would be an interesting direction to explore the better approximation of actual distribution. For example, Gaussian Mixture Models (GMMs) might be a good candidate. In Figure 11, we show a simple function $y = \exp(-x^2)$ under an input distribution of multi-models, (Equation 26). The horizontal axis presents different values

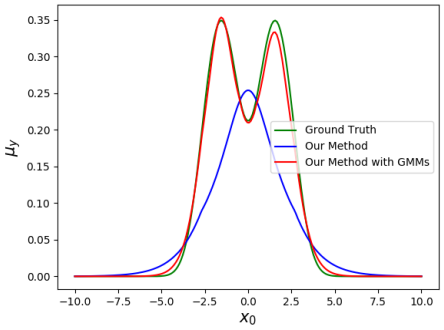


Figure 11: $y = \exp(-x^2)$ *with non-Gaussian input distribution* f_X.

of x_0 in Equation 26, and the longitudinal axis denotes the mean of f_Y. In this case, we can still achieve a good approximation when the input distribution is approximated by GMMs. However, the more complex assumption of intermediate variables, the more computations are required at optimization and runtime to do the smoothing. We would regard it as an important future work.

$$f_X = \begin{cases} \frac{1}{4}|\sin((x-x_0))|, & x_0 - \pi < x < x_0 + \pi \\ 0, & \text{otherwise} \end{cases} \quad (26)$$

Secondly, composition rule may bring large variance and then result in large errors. Other approximation methods for the shader program will be another interesting future work. Thirdly, some approximations such as the filtering of normal maps may not restore all information about how normal vary within the footprint. Toksvig's method and ours both give a compromise about the real normal distribution. However, it is still worthy of finding a new, general, and more accurate expression about filtering normal maps. Finally, in this work, we only considered range and domain at processing composited functions in shaders. However, besides range and domain, there may have other relations among computations in one shader, such as the correlations of different intermediate variables, certain distributions of variables in the scene, etc. Other relations to better approximate smoothed values will be an important future direction.

6. Conclusion

In this paper, we extend the framework of mean-variance statistics to smooth shader program. We first consider the domain and range of functions to more closely approximate the weighted average of shading values over the pixel footprint. Three new shader smoothing rules are given to approximate specific composition of functions and reduce the approximation errors. The first approximation rule is based on the adaptive kernels, but improved by renormalizing the kernel in the clamped domain. The second rule directly approximates the function outputs using Gaussian distributions. The third approximation rule handles complex functions using Taylor series. Besides these approximations, we further integrate textures, including normal map and color texture, as discrete functions into the framework. Compared with previous work, our algorithm achieves better quality and preserves higher smoothed details. Experiments show that our method is capable of smoothing general shaders such as physical-based shaders and shaders with complex visibility functions.

Acknowledgements

We thank Dr. Yifan Peng for his suggestions. This work was supported in part by National Key R&D Program of China (No. 2017YFB1002605), NSFC (No. 61872319), Key R&D Program of Zhejiang Province (2018C01090), Zhejiang Provincial NSFC (No. LR18F020002) and Zhejiang University Education Foundation Global Partnership Fund.

References

[AGB00] APODACA A. A., GRITZ L., BARZEL R.: *Advanced RenderMan: Creating CGI for motion pictures*. Morgan Kaufmann, 2000. 2

[BSS*13] BELCOUR L., SOLER C., SUBR K., HOLZSCHUCH N., DURAND F.: 5d covariance tracing for efficient defocus and motion blur. *ACM Transactions on Graphics (TOG) 32* (06 2013). doi:10.1145/2487228.2487239. 2

[Cro77] CROW F. C.: The aliasing problem in computer-generated shaded images. *Communications of the ACM 20*, 11 (1977), 799–805. 2

[Cro84] CROW F. C.: Summed-area tables for texture mapping. In *ACM SIGGRAPH computer graphics* (1984), vol. 18, ACM, pp. 207–212. 1, 2

[CT82] COOK R. L., TORRANCE K. E.: A reflectance model for computer graphics. *ACM Transactions on Graphics (TOG) 1*, 1 (1982), 7–24. 7

[DBLW15] DORN J., BARNES C., LAWRENCE J., WEIMER W.: Towards automatic band-limited procedural shaders. In *Computer Graphics Forum* (2015), vol. 34, Wiley Online Library, pp. 77–87. 1, 2

[DHI*13] DUPUY J., HEITZ E., IEHL J.-C., POULIN P., NEYRET F., OSTROMOUKHOV V.: Linear efficient antialiased displacement and reflectance mapping. *ACM Transactions on Graphics (TOG) 32*, 6 (2013), 211. 1, 2, 7

[HNPN13] HEITZ E., NOWROUZEZAHRAI D., POULIN P., NEYRET F.: Filtering color mapped textures and surfaces. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2013), ACM, pp. 129–136. 2

[HSRG07] HAN C., SUN B., RAMAMOORTHI R., GRINSPUN E.: Frequency domain normal map filtering. *ACM Transactions on Graphics (TOG) 26*, 3 (2007), 28. 1, 2

[JHY*14] JAKOB W., HAŠAN M., YAN L.-Q., LAWRENCE J., RAMAMOORTHI R., MARSCHNER S.: Discrete stochastic microfacet models. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 115. 2

[Kaj86] KAJIYA J. T.: The rendering equation. In *ACM Siggraph Computer Graphics* (1986), vol. 20, ACM, pp. 143–150. 1

[KHPL16] KAPLANYAN A. S., HILL S., PATNEY A., LEFOHN A. E.: Filtering distributions of normals for shading antialiasing. In *High Performance Graphics* (2016), pp. 151–162. 1

[NPS13] NEUBELT D., PETTINEO M., STUDIOS R. A. D.: Crafting a next-gen material pipeline for the order: 1886. *part of Physically Based Shading in Theory and Practice, SIGGRAPH* (2013). 1

[OB10] OLANO M., BAKER D.: Lean mapping. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games* (2010), ACM, pp. 181–188. 1, 2, 7

[PH10] PHARR M., HUMPHREYS G.: Physically based rendering, second edition: From theory to implementation. 7

[SAMWL11] SITTHI-AMORN P., MODLY N., WEIMER W., LAWRENCE J.: Genetic programming for shader simplification. In *ACM Transactions on Graphics (TOG)* (2011), vol. 30, ACM, p. 152. 5

[TLQ*08] TAN P., LIN S., QUAN L., GUO B., SHUM H.: Filtering and rendering of resolution-dependent reflectance models. *IEEE Transactions on Visualization and Computer Graphics 14*, 2 (2008), 412–425. 1, 2

[Tok05] TOKSVIG M.: Mipmapping normal maps. *journal of graphics tools 10*, 3 (2005), 65–71. 1, 2, 6

[VAZH*09] VELÁZQUEZ-ARMENDÁRIZ E., ZHAO S., HAŠAN M., WALTER B., BALA K.: Automatic bounding of programmable shaders for efficient global illumination. 2

[WBSS04] WANG Z., BOVIK A. C., SHEIKH H. R., SIMONCELLI E. P.: Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing 13*, 4 (2004), 600–612. 7

[Wil83] WILLIAMS L.: Pyramidal parametrics. In *Acm siggraph computer graphics* (1983), vol. 17, ACM, pp. 1–11. 1, 2, 6

[WMLT07] WALTER B., MARSCHNER S. R., LI H., TORRANCE K. E.: Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics conference on Rendering Techniques* (2007), Eurographics Association, pp. 195–206. 11

[WYY*] WANG R., YANG X., YUAN Y., CHEN W., BALA K., BAO H.: Automatic shader simplification using surface signal approximation. *Acm Transactions on Graphics 33*, 6, 1–11. 5

[WZYR19] WU L., ZHAO S., YAN L.-Q., RAMAMOORTHI R.: Accurate appearance preserving prefiltering for rendering displacement-mapped surfaces. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2019) 38*, 4 (2019). 1, 2

[XWZB17] XU C., WANG R., ZHAO S., BAO H.: Real-time linear brdf mip-mapping. In *Computer Graphics Forum* (2017), vol. 36, Wiley Online Library, pp. 27–34. 1, 2

[YB18] YANG Y., BARNES C.: Approximate program smoothing using mean-variance statistics, with application to procedural shader bandlimiting. In *Computer Graphics Forum* (2018), vol. 37, Wiley Online Library, pp. 443–454. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

[YHJ*14] YAN L.-Q., HAŠAN M., JAKOB W., LAWRENCE J., MARSCHNER S., RAMAMOORTHI R.: Rendering glints on high-resolution normal-mapped specular surfaces. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 116. 2

[YHMR16] YAN L.-Q., HAŠAN M., MARSCHNER S., RAMAMOORTHI R.: Position-normal distributions for efficient rendering of specular microstructure. *ACM Transactions on Graphics (TOG) 35*, 4 (2016), 56. 2