Pacific Graphics 2022
N. Umetani, E. Vouga, and C. Wojtan
(Guest Editors)

Volume 41 (2022), Number 7

Multirate Shading with Piecewise Interpolatory Approximation

Yiwei Hu[1,2], Yazhen Yuan[1,4], Rui Wang[†,1], Zhuo Yang[3] and Hujun Bao[1]

[1]State Key Lab of CAD&CG, Zhejiang University [2]Yale University
[3]Guangdong University of Technology
[4]Tencent, CROS, Game Engine Department
[†]Corresponding author: ruiwang@zju.edu.cn

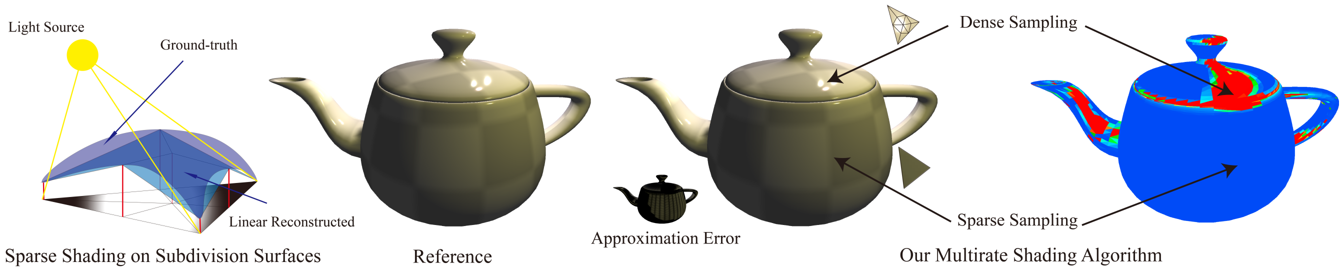

Figure 1: *Ground-truth shading signals can be linearly reconstructed by shading values on the vertices of subdivision surfaces (left). The piecewise interpolation errors are analytically estimated to decide the optimal subdivision given an error threshold. We propose a multirate shading algorithm leveraging sparse piecewise reconstruction (right). High-frequency shading signals require dense sampling while low-frequency signals can be efficiently approximated by sparse sample points. Different shading rates (i.e., granularities of subdivision) are computed at runtime using our derived estimator.*

Abstract

Evaluating shading functions on geometry surfaces dominates the rendering computation. A high-quality but time-consuming estimate is usually achieved with a dense sampling rate for pixels or sub-pixels. In this paper, we leverage sparsely sampled points on vertices of dynamically-generated subdivision surfaces to approximate the ground-truth shading signal by piecewise linear reconstruction. To control the introduced interpolation error at runtime, we analytically derive an L_∞ error bound and compute the optimal subdivision surfaces based on a user-specified error threshold. We apply our analysis on multiple shading functions including Lambertian, Blinn-Phong, Microfacet BRDF and also extend it to handle textures, yielding easy-to-compute formulas. To validate our derivation, we design a forward multirate shading algorithm powered by hardware tessellator that moves shading computation at pixels to the vertices of subdivision triangles on the fly. We show our approach significantly reduces the sampling rates on various test cases, reaching a speedup ratio of 134% $\sim$ 283% compared to dense per-pixel shading in current graphics hardware.

CCS Concepts
• ***Computing methodologies*** $\rightarrow$ ***Rendering;***

1. Introduction

High-quality image synthesis requires shading signals to be reconstructed using densely sampled points (pixels) on geometry surfaces. However, when resolution increases, the shading computation time can surge along with the number of sampled pixels. In the real-time domain, the need to reduce the cost of shading becomes critical, motivating the Variable Rate Shading (VRS) feature in the current hardware pipeline. VRS is adopted by recent work [YZK*19, CKY*22] to improve real-time performance. In our paper, we propose a novel view of multirate shading focusing on the approximation of surface shading signals using sparse reconstruction. We observe that shading signals can be efficiently reconstructed by shading values computed on the vertices of subdivided surfaces as shown in Fig. 1. A higher-frequency shading signal requires more subdivisions, i.e., more vertices, as sample points to ensure a high-quality reconstruction, while a low-frequency shading signal can be quickly approximated using very sparse samples.

Specifically, we apply linear interpolation as a reconstruction operator because it is one of the most efficient arithmetic operations in current hardware.

Sparse reconstruction by piecewise linear interpolation is bound to introduce errors. A proper subdivision is of necessity to control the error on different geometry surfaces. We analytically derive an L_∞ interpolation error bound which is in turn used to compute the required subdivision given a user-specified threshold. Performing subdivision based on analyzed error requires additional computation. To alleviate the extra burden, we derive a practical way to compute a simplified error estimator for runtime evaluation. We apply our derivation on multiple shading functions including Lambertian, Blinn-Phong and Microfacet BRDF. We also extend our assumption of continuous functions to handle discrete variables e.g., textures and provide a conservative error estimate. To the best of our knowledge, this is the first work to analytically build linear interpolation error bounds for various BRDFs.

To validate our derivations, we design a multirate shading algorithm leveraging the hardware tessellator. Specifically, we rely on tessellation shaders in the rendering pipeline, moving per-pixel shading to per-vertex shading on subdivision surfaces. Potential shading errors on a triangle are dynamically estimated and shading functions are piecewise linearly approximated from shading values computed on the vertices of subdivided triangles generated by tessellator. Shading values for pixels not directly sampled are reconstructed by linear interpolation during rasterization. Intuitively, our multirate method is a trade-off between the legacy Gourand shading [Gou71] and the current de-facto standard—per-pixel shading but with quality-guaranteed error control. Although dynamic error analysis and the subdivision process introduce overhead, the overall rendering cost is effectively reduced due to sparse sampling.

Compared to densely-evaluated per-pixel shading, our multirate method performs only $11\% \sim 23\%$ the number of shading function evaluations, achieving a speedup ratio of $134\% \sim 283\%$ on current graphics hardware. Our contributions are summarized as:

- We introduce an approximation of shading signals by sparse linear reconstruction on subdivided geometry surfaces, deriving analytical L_∞ error bounds for suitable subdivision and applying it on various BRDFs and textures.
- We design a multirate shading algorithm that adaptively approximates shading signals with dynamic subdivision supported by the hardware tessellator.

2. Related Works

2.1. Multirate Shading

Multirate shading is a long-standing rendering technique in computer graphics. Many multirate approaches share the observation that low frequency shading can be executed at a low rate without compromising considerable loss in quality. Mixed-resolution shading [YSL08] renders shading components at different resolutions and then reconstructs using bilinear upsampling. Decoupled shading [RKLC*11,CTM13,LD12,CMFL15] separates pixel shading from geometry or visibility calculations by lazy evaluation and reuse of samples when possible. Similar ideas are extended to shading signals on geometry surfaces [BFM10, CTH*14].

Variable Rate Shading (VRS) supported by current graphics hardware runs shaders at different screen-space resolutions. VRS is utilized by many recent approaches to control the real-time shading rate [YZK*19, CKY*22]. Coarse pixel shading [HGF14, VST*14, XLV18, YWB18] is implemented as a set of extensions of graphics hardware that can execute the low frequency part of the shader at lower resolutions. Different from these approaches, we propose multirate shading based on sparse linear reconstruction on subdivision surfaces with controllable analytical errors.

Note that this idea is similar to the classic Reyes Rendering Architecture [CCC87], which achieves high-quality interpolated shading by per-vertex shading on vertices of subdivided triangles. However, Reyes continues subdivision until sub-triangles are at sub-pixel level which ensures quality but sacrifices efficiency. Reyes generates dense sample points as opposed to the variable densities in a multirate method.

2.2. Shading Signal Processing

Processing shading signals on geometry surfaces appears in many areas of computer graphics. An important approach studied in the literature is the meshing process [Hop96]. For shading signals, early work dates back to radiosity [Nak84, GTGB84] which represents shading using polygon-wise colors. Since then, more sophisticated and accurate solutions have been proposed to better capture the frequency of shading signals [HSA91, Zat93].

Precomputed Radiance Transfer (PRT) [SKS02] stores spherical harmonics coefficients on vertices and interpolates to pixels for real-time rendering. Vertex baking [KBS11] treats ambient occlusion or ambient obscurance [STCK13] as low-frequency signals and stores them as vertex attributes to be interpolated at runtime. High-order basis functions e.g., Bezier basis [WYY*14], are also proposed to approximate shading signals on triangles. However, all these methods control errors by precomputation under pre-defined, fixed environments and can only process low-frequency signals. By contrast, our approach estimates interpolatory shading errors and decides sample rates at runtime with an analytical error analysis.

Subdivision or tessellation is a common approach to capture high-frequency signals on geometry surfaces that either can refine highlight shading [CNW96] or enhance geometry details [BA08, YWH*16]. Our method benefits from subdivision as a convenient way to generate sparse samples and perform theoretical analysis.

2.3. Error Analysis on Linear Interpolation

Discussions on error bound analysis for linear interpolation can be found in the field of Finite Elements Methods (FEM). Guessab and Schmeisser [GS05] suggest a sharp error bound for linear interpolation on convex polytopes. More specifically, Subbotin [Sub89] derives an L_∞ error bound of linear interpolation on triangular domain, which is sharp if and only if the triangle is equilateral. Waldron [Wal98] proposes an improved error estimation in multivariate quasi-interpolation on vertices of a simplex and derives a sharp L_∞ inequality when the center of its circumcircle is inside the triangle. However, these derivations pay attention to theoretical analysis. We leverage these conclusions and derive practical solutions to evaluate interpolation errors for shading functions at runtime.

3. Theoretical Analysis

Our key idea is to approximate shading signals with sparsely sampled shading points. Although various patterns exist for point sampling, we apply an efficient way to generate sample points using the vertices of subdivision surfaces. The shading functions are then reconstructed by piecewise linear interpolation using shading values computed on the vertices on subdivided surfaces.

Formally, let $P \in \mathbb{R}^2$ be a convex polygon with vertices v_1, v_2, ..., v_m. Suppose we have a subdivision operator T that uniformly subdivides P into a set of N convex sub-polygons $\mathcal{P} = \{P_i\}_{i=1}^{N}$, each of convex sub-polygons in $\mathcal{P}$ are composed by m vertices noted as $\Theta_i = \{v_{i_k}\}_{k=1}^{m}$, where v_{i_k} is one vertex in the set of all M vertices $\mathcal{V} = \{v_k\}_{k=1}^{M}$ of subdivision surface.

Given these subdivided convex sub-polygons, a continuous function $f \in C(P): \mathbb{R}^2 \to \mathbb{R}$ defined on convex polygon P can be piecewise linearly approximated by a set of values computed at the vertices of subdivided polygons, $\mathcal{F} = \{f(v_k)\}_{k=1}^{M}$. Specifically, for one point $v(x,y) \in P$, the function $f(x,y)$ can be interpolated as:

$$f(x,y) \approx \sum_{i}^{N} \mu_i(x,y) L_{\Theta_i} f(x,y) = \sum_{i}^{N} \mu_i(x,y) \sum_{k=1}^{m} f(v_{i_k}) \lambda_{i_k}(x,y) \quad (1)$$

where $\mu_i(x,y)$ is a discriminant function for P_i where $(x,y) \in P_i$, $\mu_i(x,y) = 1$, otherwise $\mu_i(x,y) = 0$. Meanwhile L_{Θ_i} is a linear interpolation operator that interpolates the values sampled from f at the vertex set $\Theta_i = \{v_{i_k}\}_{k=1}^{m}$ of the convex sub-polygon P_i, and $\lambda_{i_k}(x,y)$ is the linear interpolation coefficient on the convex sub-polygon P_i (e.g., barycentric coordinate) which satisfies the Lagrange condition $\sum_{k=1}^{m} \lambda_{i_k}(x,y) = 1$ and linear precision $v(x,y) = \sum_{k=1}^{m} \lambda_{i_k}(x,y) v_{i_k}$.

In the following sub-sections, we first introduce the piecewise linear interpolation error of a general function defined on arbitrary convex polygonal domain in Sec. 3.1. We also derive a specific error estimation for vector normalization as a widely used operator in shading computation. We then apply these conclusions to analyze interpolation errors on multiple shading functions including Lambertian, Blinn-Phong and Microfacet BRDF (Sec. 3.2∼3.4). Last, we extend our derivation to process discrete variables e.g., textures (Sec. 3.5). In the next section (Sec. 4), we will present a multirate shading algorithm using hardware tessellation, showing the application of the theoretical analysis.

3.1. Interpolation Error

Approximating a non-linear function $f(x,y)$ by piecewise linear interpolation will introduce error. The error can be reduced by a finer subdivision with denser sampling points. To precisely measure the difference, we define the L_∞ norm of interpolation error $e(f)$ on the convex polygon P as follows:

$$\|e(f)\|_{\infty,P} = \sup_{P_i \in \mathcal{P}} \|e(f)\|_{\infty,P_i} = \sup_{P_i \in \mathcal{P}} \|f - L_\Theta f\|_{\infty,P_i} \quad (2)$$

Given that the L_∞ error on the convex polygon P is the maximum L_∞ error among all convex sub-polygons P_i, this error can be regarded as an error function depending on the subdivision operator $T(n)$ where n is a parameter controlling the granularity of the subdivision. To control the error within a threshold ε, we find an optimal granularity of subdivision $T(n)$:

$$\begin{aligned} &\arg\min_{n} \; e(f(x,y), T(n)) \quad \forall (x,y) \in P \\ &\text{s.t. } \|e(f)\|_{\infty,P} \le \varepsilon \end{aligned} \quad (3)$$

The optimal n varies with different $f(x,y)$. As Fig. 2 shows, a fixed subdivision parameter fails to reduce interpolation errors on triangles with shading signals of different frequencies. However, an analytical solution for Eq. (3) is intractable, therefore we compute an appropriate parameter n based on the interpolation error bound.

3.1.1. A General Estimation on $T(n)$ and Error Bound

For an arbitrary convex polygon P with m vertices, the L_∞ error bound of linear interpolation has been proven to be: [GS05]

$$\|e(f)\|_{\infty,P} = \|f - L_{\Theta_i} f\|_{\infty,P} \le \frac{(r^{sc})^2}{2} |f|_{2,\infty,P}, \forall f \in C^2(P) \quad (4)$$

where r^{sc} and v^{sc} specify the smallest circle P^{sc} which contains P:

$$P^{sc} =: \{v \in \mathbb{R}^2 : \|v - v^{sc}\| \le r^{sc}\} \quad \forall v \in P \quad (5)$$

and $|f|_{2,\infty,P}$ is the second order L_∞ semi-norm that is defined as follows:

$$|f|_{2,\infty,P} = \| \, |D^2 f| \, \|_{\infty,P} \quad (6)$$

and

$$|D^2 f|(x,y) = \sup_{\xi \in \mathbb{R}^2, \|\xi\|_2 = 1} |D_\xi^2 f(x,y)| \quad (7)$$

by which $|D^2 f|(x,y)$ is defined as the supremum of the second derivative of f in the arbitrary direction $\xi = [\xi_x, \xi_y]^T$ for all $(x,y) \in P$.

We now define $t = T(n)$ as a uniform subdivision process that

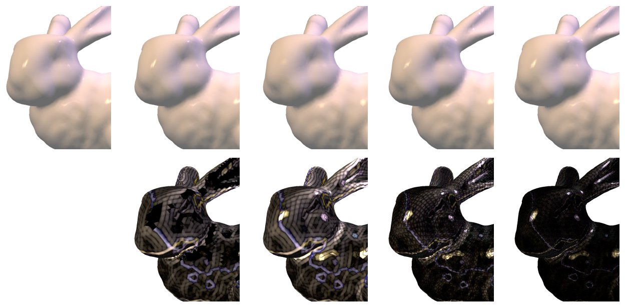

Figure 2: *We show a bunny rendered by per-pixel dense sampling, our multirate sparse sampling, and single-rate sampling (top row). Corresponding error images (30x scaled) are shown below (second row). Single-rate subdivision (n=1, 2 and 4) can reduce overall interpolation error. However, a uniform subdivision on all of the triangles struggles to effectively eliminate errors for triangles with high-frequency shading signals. Interpolation errors (see the shape of highlight) are salient in the specular highlight regions when n = 2 and still noticeable even when n = 4. Instead, our adaptive method only performs dense subdivision on these regions and shades the smooth bunny's body without subdivision.*

lets r_i^{sb}, the radius of circumcircle of subdivided convex polygon P_i (defined as Eq. (5) likewise) be $r_i^{sb} \leq \frac{r^{sb}}{n}$ for all $i = 1 \ldots N$. The L_∞ piecewise interpolation error bound on the subdivided domain can be bounded by:

$$\begin{aligned} \|e(f,t)\|_{\infty,P} &= \sup_{P_i \in \mathcal{P}} \|e(f)\|_{\infty,P_i} \\ &\leq \sup_{P_i \in \mathcal{P}} \left(\frac{(r_i^{ab})^2}{2} |f|_{2,\infty,P_i} \right) \leq \frac{(r^{ab})^2}{2n^2} |f|_{2,\infty,P} \end{aligned} \tag{8}$$

This inequality provides a conservative solution of Eq. (3), that is

$$n \geq r^{ab} \sqrt{\frac{1}{2\varepsilon} |f|_{2,\infty,P}}. \tag{9}$$

Specifically, in the context of computer graphics, we have geometries represented by triangle meshes. The linear interpolation error bound on triangular domain T is studied for a sharper bound [Sub89, Wal98]. Similarly, we define a subdivision process $t = T(n)$ that evenly reduces the diameter h (the length of the longest edge) of the triangle. We can derive

$$\|e(f,t)\|_{\infty,T} \leq \frac{1}{6} \frac{h^2}{n^2} |f|_{2,\infty,T} \quad \forall f \in C^2(T) \tag{10}$$

when the diameters of sub-triangles are all less than $\frac{h}{n}$. Likewise, we conservatively estimate the parameter n under an error threshold ε as

$$n \geq h \sqrt{\frac{1}{6\varepsilon} |f|_{2,\infty,T}}. \tag{11}$$

3.1.2. A Specific Estimation on $T(n)$ and Error Bound

Vector normalization is a fundamental, widely-used operator in shading computations. Normalization is simple but highly nonlinear. Performing interpolation to approximate vector normalization may produce considerable error. On the other hand, due to the complexity of evaluating the second order semi-norms in Eq. (4), direct error analysis using Eq. (10) on vector normalization is impractical at runtime.

To simplify computation, we derive a specific error estimation in vector space. First, without loss of generality, we consider vector normalization on a triangle T. A vector $\mathbf{w}$ is interpolated by three normalized vectors $\mathbf{w}_0, \mathbf{w}_1$ and $\mathbf{w}_2$ at three vertices of T as $\mathbf{w} = \sum_{k=1}^{3} \mathbf{w}_k \lambda_k(x,y)$. Its normalized vector is computed as $\frac{\mathbf{w}}{\|\mathbf{w}\|_2}$. Hence, the L_∞ error of the length between the linear interpolated vector $\mathbf{w}$ and its normalized vector can be computed as follows:

$$\| \, \| \frac{\mathbf{w}}{\|\mathbf{w}\|_2} - \mathbf{w} \|_2 \, \|_\infty \leq \max_T \{1 - \|\mathbf{w}\|_2\}. \tag{12}$$

It can be further proved that

$$\max_T \{1 - \|\mathbf{w}\|_2\} \leq 1 - \sqrt{1 - \frac{{h^*}^2}{3}}, \tag{13}$$

where $h^* = \max\{\|\mathbf{w}_0 - \mathbf{w}_1\|_2, \|\mathbf{w}_0 - \mathbf{w}_2\|_2, \|\mathbf{w}_1 - \mathbf{w}_2\|_2\}$. When we subdivide a triangle, h^* on each sub-triangle, noted as h_i^* will subsequently decrease. Under a uniform subdivision, h_i^* varies with n as

$$h_i^* = 2 \sin\left(\arctan\left(\frac{h^*}{2n\sqrt{1-R^2}} \right) \right), \tag{14}$$

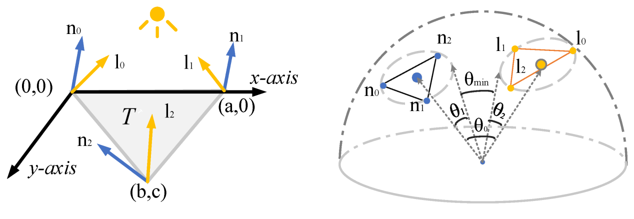

Figure 3: *(a) A triangle in its own local coordinate system whose shape is defined by a, b, c. Shading attributes such as* $\boldsymbol{n}$ *and* $\boldsymbol{l}$ *are defined on each of its vertices. (b) normalized* $\boldsymbol{n}$ *(blue) and normalized* $\boldsymbol{l}$ *(yellow) are distributed on a sphere, and form two cones. The cosine of* $\theta_{\min}$ *is a conservative estimation of possible* $\max\{\cos\langle \boldsymbol{n}, \boldsymbol{l} \rangle\}$.

where R is the radius of circumcircle that includes all three unit vectors, $\mathbf{w}_0$, $\mathbf{w}_1$ and $\mathbf{w}_2$. Given an error threshold ε, we can compute parameter n conservatively as follows:

$$n \geq \frac{\sqrt{1 + 3(1-\varepsilon)^2} h^*}{2\sqrt{3 - 3(1-\varepsilon)^2} \sqrt{1 - R^2}}. \tag{15}$$

Please refer to our supplemental document for a detailed derivation of Eqs. (12 ∼ 15). In the following section, we will describe how to apply the introduced general and specific error estimates for different shading functions. We primarily focus on **triangular-domain error analysis** as defined in Fig. 3a. However, we can extend our derivation to other domains without loss of generality.

3.2. Example: Lambertian Model

The Lambertian model is one of the simplest shading functions that requires normals and light directions as attributes to be interpolated from m vertices of a polygon P to other coordinates. As mentioned before, we consider triangle primitives with $m = 3$. We use $\mathbf{n}$ and $\mathbf{l}$ to denote the linearly interpolated normals and light directions, which are computed as $\mathbf{n} = \sum_{k=1}^{m} \mathbf{n}_k \lambda_k(x,y)$ and $\mathbf{l} = \sum_{k=1}^{m} \mathbf{l}_k \lambda_k(x,y)$, where $\mathbf{n}_k$ and $\mathbf{l}_k$ denote the normals and light directions at each vertex. The interpolated $\mathbf{n}$ and $\mathbf{l}$ are unnormalized. The entire shading function of the Lambertian model is computed as

$$f(x,y) = \hat{f}(\mathbf{n}, \mathbf{l}) = K_d \cdot \frac{\mathbf{n}}{\|\mathbf{n}\|_2} \cdot \frac{\mathbf{l}}{\|\mathbf{l}\|_2}, \tag{16}$$

where K_d is the diffuse coefficient, while $\frac{\mathbf{n}}{\|\mathbf{n}\|_2}$ and $\frac{\mathbf{l}}{\|\mathbf{l}\|_2}$ are the normalized normal and lighting direction at the shading point respectively.

Directly computing the semi-norm $|f|_{2,\infty,P}$ of the Lambertian model is overly complicated due to vector normalization terms, making it impossible to evaluate at runtime. However, we can split the interpolation error of f into two simpler terms, and compute each error bound separately. Theoretically, after subdivision, the error bound on a convex sub-polygon P_i can be computed as following:

$$\begin{aligned}&\|f - L_{\Theta_i} f\|_{\infty,P_i} = \|\hat{f}(n,l) - L_{\Theta_i}\hat{f}(n,l)\|_{\infty,P_i}\\ \le&\|\frac{\mathbf{n}}{\|\mathbf{n}\|_2}\cdot\frac{\mathbf{l}}{\|\mathbf{l}\|_2} - \mathbf{n}\cdot\mathbf{l}\|_{\infty,P_i} + \|\mathbf{n}\cdot\mathbf{l} - L_{\Theta_i}(\mathbf{n}\cdot\mathbf{l})\|_{\infty,P_i}.\end{aligned} \tag{17}$$

3.2.1. Estimation on $\|\frac{\mathbf{n}}{\|\mathbf{n}\|_2}\cdot\frac{\mathbf{l}}{\|\mathbf{l}\|_2} - \mathbf{n}\cdot\mathbf{l}\|_{\infty,P_i}$

The first term of the error can be expanded using Eq. (12):

$$\begin{aligned}&\|\frac{\mathbf{n}}{\|\mathbf{n}\|_2}\cdot\frac{\mathbf{l}}{\|\mathbf{l}\|_2} - \mathbf{n}\cdot\mathbf{l}\|_{\infty,P_i}\\ \le&\Big(\|\ \|\frac{\mathbf{n}}{\|\mathbf{n}\|_2} - \mathbf{n}\|_2\ \|_{\infty,P_i} + \|\ \|\frac{\mathbf{l}}{\|\mathbf{l}\|_2} - \mathbf{l}\|_2\ \|_{\infty,P_i}\Big)\cdot\max\{\cos\langle\mathbf{n},\mathbf{l}\rangle\}\end{aligned} \tag{18}$$

$$\le\big(\max\{1 - \|\mathbf{n}\|_{2,P_i}\} + \max\{1 - \|\mathbf{l}\|_{2,P_i}\}\big)\cdot\max\{\cos\langle\mathbf{n},\mathbf{l}\rangle\} \tag{19}$$

where $\|\ \|\frac{\mathbf{n}}{\|\mathbf{n}\|_2} - \mathbf{n}\|_2\ \|_{\infty,P_i}$ and $\|\ \|\frac{\mathbf{l}}{\|\mathbf{l}\|_2} - \mathbf{l}\|_2\ \|_{\infty,P_i}$ are the errors from vector normalization, and $\max\{\cos\langle\mathbf{n},\mathbf{l}\rangle\}$ is the potential maximum shading value on the sub-triangle.

We compute $\max\{\cos\langle\mathbf{n},\mathbf{l}\rangle\}$ by finding the minimum possible angle between $\mathbf{n}$ and $\mathbf{l}$. For instance, when the convex polygon is a triangle, as shown in Fig. 3b, all $\mathbf{n}$ and $\mathbf{l}$ form two spherical triangles in the hemisphere vector space. For simplicity, we construct two circumcircles to include $\mathbf{n}$ and $\mathbf{l}$ respectively, and denote the angle between these circumcircles as θ_0, and the interior angles of each circumcircle as θ_1 and θ_2. In this way, $\max\{\cos\langle n,l\rangle\}$ can be estimated as

$$\max\{\cos\langle\mathbf{n},\mathbf{l}\rangle\} = \cos(\max\{0, \theta_0 - \theta_1 - \theta_2\}). \tag{20}$$

Providing an error threshold ε, by letting $\varepsilon' = \frac{\varepsilon}{\max\{\cos\langle\mathbf{n},\mathbf{l}\rangle\}}$, we can use Eq. (15) to calculate appropriate subdivision parameters $n_{\mathbf{n}}$ and $n_{\mathbf{l}}$ for $\mathbf{n}$ and $\mathbf{l}$ respectively. For example, $n_{\mathbf{n}}$ can be computed as

$$n_{\mathbf{n}} \ge \frac{\sqrt{1+3(1-\varepsilon')^2}h^*_{\mathbf{n}}}{2\sqrt{3-3(1-\varepsilon')^2}\sqrt{1-R^2_{\mathbf{n}}}}, \tag{21}$$

where $h^*_{\mathbf{n}} = \max\{\|\mathbf{n}_0 - \mathbf{n}_1\|_2, \|\mathbf{n}_0 - \mathbf{n}_2\|_2, \|\mathbf{n}_1 - \mathbf{n}_2\|_2\}$ and $R_{\mathbf{n}}$ is the radius of the circumcircle of $\mathbf{n}_0$, $\mathbf{n}_1$ and $\mathbf{n}_2$.

3.2.2. Estimation on $\|\mathbf{n}\cdot\mathbf{l} - L_{\Theta_i}(\mathbf{n}\cdot\mathbf{l})\|_{\infty,P_i}$

Given ε, the parameter $n_{\mathbf{n}\cdot\mathbf{l}}$ for the second term in Eq. (17) can be computed by the general error estimation formula i.e., Eq. (11). The interpolated function is solely an inner product without normalization and its second order derivative $|D^2 f|$ is a constant:

$$\begin{aligned}&n_{\mathbf{n}\cdot\mathbf{l}} \ge h\sqrt{\frac{1}{6\varepsilon}u},\\ &u = |C_1^T C_2 + D_1^T D_2| + \sqrt{(C_1^T C_2 - D_1^T D_2)^2 + (C_1^T D_2 + C_2^T D_1)^2}\end{aligned} \tag{22}$$

where $C_1 = -\frac{\mathbf{n}_0}{a} + \frac{\mathbf{n}_1}{a}$, $D_1 = \frac{b-a}{ac}\mathbf{n}_0 - \frac{b}{ac}\mathbf{n}_1 + \frac{\mathbf{n}_2}{c}$ and $C_2 = -\frac{\mathbf{l}_0}{a} + \frac{\mathbf{l}_1}{a}$, $D_2 = \frac{b-a}{ac}\mathbf{l}_0 - \frac{b}{ac}\mathbf{l}_1 + \frac{\mathbf{l}_2}{c}$, which are constants computed from attributes of the triangle (see Fig. 3a). The derivation details are provided in the supplemental document.

3.2.3. Final Subdivision

We now have three subdivision parameters that are derived from the Lambertian model, namely, $n_{\mathbf{n}}$, $n_{\mathbf{l}}$ and $n_{\mathbf{n}\cdot\mathbf{l}}$. Once given an error threshold ε^*, we evenly divide it into three bounds, $\varepsilon = \frac{\varepsilon^*}{3}$, and individually estimate the corresponding subdivision parameters using Eq. (21) and Eq. (22). We select the maximum value as our final subdivision parameter:

$$n = \max\{n_{\mathbf{n}}, n_{\mathbf{l}}, n_{\mathbf{n}\cdot\mathbf{l}}\}. \tag{23}$$

3.3. Example: Blinn-Phong Model

While similar to the Lambertian model, the Blinn-Phong model requires normals and half-vectors (instead of light directions) as attributes and has an additional power operation. We denote the normal and as $\mathbf{n}$ and the half-vector as $\mathbf{h}$. The entire shading function using Blinn-Phong model is computed as

$$f(x,y) = \hat{f}(\mathbf{n},\mathbf{h}) = K_s\cdot(\frac{\mathbf{n}}{\|\mathbf{n}\|_2}\cdot\frac{\mathbf{h}}{\|\mathbf{h}\|_2})^\alpha, \tag{24}$$

where K_s is the specular coefficient and α is the shininess coefficient. To simplify the derivation, we introduce a new variable t as

$$t = \frac{\mathbf{n}}{\|\mathbf{n}\|_2}\cdot\frac{\mathbf{h}}{\|\mathbf{h}\|_2} \tag{25}$$

By plugging Eq. (25) into Eq. (24), f is simplified as

$$f(x,y) = \hat{f}(t) = K_s\cdot t^\alpha \tag{26}$$

Note that t is not linearly distributed on the surface. However, we can assume that there exists a linear interpolation of t, $L_{\Theta_i}t = \sum_{k=1}^m t_{i_k}\lambda_{i_k}(x)$, where t_{i_k} denotes the values computed at vertices of the convex sub-polygon P_i. We leverage the linear interpolation of t to estimate the error of Blinn-Phong model as

$$\begin{aligned}&\|f(x,y) - L_{\Theta_i}f(x,y)\|_{\infty,P_i} = \|\hat{f}(t) - L_{\Theta_i}\hat{f}(t)\|_{\infty,P_i}\\ \le&\|\hat{f}(t) - \hat{f}(L_{\Theta_i}t)\|_{\infty,P_i} + \|\hat{f}(L_{\Theta_i}t) - L_{\Theta_i}\hat{f}(t)\|_{\infty,P_i}\end{aligned} \tag{27}$$

The first term of the inequality is the error introduced by the assumed linear interpolation of t, while the second term is the error caused by the interpolation of the new function $\hat{f}(t)$ on the convex sub-polygon P_i.

3.3.1. Estimation on $\|\hat{f}(L_{\Theta_i}t) - L_{\Theta_i}\hat{f}(t)\|_{\infty,P_i}$

The second term in Eq. (27) is easy to compute. Note that $L_{\Theta_i}\hat{f}(t) = L_{\Theta_i}\hat{f}(L_{\Theta_i}t)$, which suggests that this error is caused by the interpolation of the power function in Blinn-Phong model. Using the new variable t, we can directly apply the error formula Eq. (10) to derive a close-form solution:

$$\begin{aligned}&\|\hat{f}(L_{\Theta_i}t) - L_{\Theta_i}\hat{f}(t)\|_{\infty,P_i} = \|\hat{f}(L_{\Theta_i}t) - L_{\Theta_i}\hat{f}(L_{\Theta_i}t)\|_{\infty,P_i}\\ \le&\frac{1}{6}\frac{K_s\alpha(\alpha-1)(C_t^2 + D_t^2)h^2}{n^2}(L_\Theta t)^{\alpha-2}_{\max}\end{aligned} \tag{28}$$

where $C_t = -\frac{t_0}{a} + \frac{t_1}{a}$, $D_t = \frac{b-a}{ac}t_0 - \frac{b}{ac}t_1 + \frac{t_2}{c}$ and $(L_\Theta t)_{\max} = \max\{t_0, t_1, t_2\}$. t_0, t_1, t_2 are the values of t calculated on each vertex of P_i.

3.3.2. Estimation on $\|\hat{f}(t) - \hat{f}(L_{\Theta_i} t)\|_{\infty,P_i}$

We can further simplify and expand the first term in Eq. (27) by the Mean Value Theorem:

$$\begin{aligned} &\|\hat{f}(t) - \hat{f}(L_{\Theta_i} t)\|_{\infty,P_i} \\ =&\|\hat{f}_t'(\nu)(t - L_{\Theta_i} t)\|_{\infty,P_i} \qquad \text{where } L_{\Theta_i} t \le \nu \le t \\ \le&|\hat{f}_t'(t_{\max})| \cdot \|t - L_{\Theta_i} t\|_{\infty,P_i} \end{aligned} \tag{29}$$

The above inequality is always satisfied because the first derivative of Eq. (26) is a monotonically increasing function, and $t_{\max}$, which denotes the maximum value of t on the convex sub-polygon P_i, can be calculated by finding the minimum possible angle between $\mathbf{n}$ and $\mathbf{h}$ similar to Eq. (20).

By replacing $\mathbf{l}$ with $\mathbf{h}$, $\|t - L_{\Theta_i} t\|_{\infty,P_i}$ takes a form identical to Lambertian model in Eq. (17). We can apply the same derivation to evaluate this error term.

3.3.3. Final Subdivision

Given an error threshold ε, we apply the same strategy used for the Lambertian model to combine different terms. We evenly divide ε into smaller thresholds. Let each term in Eq. (27) satisfy the split error threshold and take the maximum parameters as the conservative final estimation.

3.4. Example: Microfacet Model

Now we deal with a more sophisticated BRDF model, the Microfacet model, which is widely-used in current graphics applications. We denote normal, light direction, half-vector and view direction as $\mathbf{n}, \mathbf{l}, \mathbf{h}$, and $\mathbf{v}$ respectively. The Microfacet model is computed as

$$f(x,y) = \hat{f}(\mathbf{n},\mathbf{l},\mathbf{h},\mathbf{v}) = \frac{D(\mathbf{n},\mathbf{h})F(\mathbf{v},\mathbf{h})V(\mathbf{l},\mathbf{v})}{4}, \tag{30}$$

where $D(\mathbf{n},\mathbf{h})$ is a GGX (originated by Trowbridge and Reitz) normal distribution function [WMLT07, TR75], $F(\mathbf{v},\mathbf{h})$ is the Fresnel term using the Schlick approximation [Sch94], and $V(\mathbf{l},\mathbf{v})$ is the Smith geometry term [Smi67]. Note that for convenience, the cosine terms in the denominator of the standard microfacet model are included in the term $V(\mathbf{l},\mathbf{v})$.

For such a complex shading function with high-dimension and non-linear properties, we introduce the error propagation formula:

$$\Delta f = |\frac{\partial f}{\partial x_0}|\Delta x_0 + |\frac{\partial f}{\partial x_1}|\Delta x_1 + \cdots + |\frac{\partial f}{\partial x_n}|\Delta x_n, f = f(x_0, \cdots, x_n) \tag{31}$$

On the convex sub-polygon P_i, by letting $\|f - L_{\Theta_i} f\|_{\infty,P_i} = \|\hat{f} - L_{\Theta_i}\hat{f}\|_{\infty,P_i} = \|\Delta\hat{f}\|_{\infty,P_i}$, and applying Eq. (31) to the interpolation error of $\hat{f}$, we obtain:

$$\begin{aligned} \|\Delta\hat{f}\|_{\infty,P_i} =& \| |\frac{\partial\hat{f}}{\partial D}|\Delta D + |\frac{\partial\hat{f}}{\partial F}|\Delta F + |\frac{\partial\hat{f}}{\partial V}|\Delta V \|_{\infty,P_i} \\ =&\|\frac{\hat{f}}{D}(D - L_{\Theta_i} D) + \frac{\hat{f}}{F}(F - L_{\Theta_i} F) + \frac{\hat{f}}{V}(V - L_{\Theta_i} V)\|_{\infty,P_i} \\ \le& \sum_{I \in D,L,V} \|\frac{\hat{f}}{I}\|_{\infty,P_i} \cdot \|I - L_{\Theta_i} I\|_{\infty,P_i}. \end{aligned} \tag{32}$$

which shows the total interpolation error of $\hat{f}$ propagates from the interpolation error from three components $D(\mathbf{n},\mathbf{h})$, $F(\mathbf{v},\mathbf{h})$ and $V(\mathbf{l},\mathbf{v})$. Each term is a function of the dot product of normalized vectors, therefore we can extend Eq. (27) to compute interpolation errors for $\|I - L_{\Theta_i} I\|_{\infty,P_i}, I \in D,L,V$.

Besides, we can conservatively compute $\|\frac{\hat{f}}{I}\|_{\infty,P_i}, I \in D,L,V$. For instance, we have

$$\begin{aligned} &\|\frac{\hat{f}}{D}\|_{\infty,P_i} = \|\frac{F(\mathbf{v},\mathbf{h})V(\mathbf{l},\mathbf{v})}{4}\|_{\infty,P_i} \\ =& \|\frac{F_0 + (1-F_0)(1-\mathbf{v}\cdot\mathbf{h})^5}{4((\mathbf{n}\cdot\mathbf{l})(1-k)+k)((\mathbf{n}\cdot\mathbf{v})(1-k)+k)}\|_{\infty,P_i} \\ \le& \|\frac{F_0 + (1-F_0)(1-\min\{\mathbf{v}\cdot\mathbf{h})^5\}}{4(\min\{\mathbf{n}\cdot\mathbf{l}\}(1-k)+k)(\min\{\mathbf{n}\cdot\mathbf{v}\}(1-k)+k)}\|_{\infty,P_i} \end{aligned} \tag{33}$$

Given the monotonicity of the Fresnel and Geometry terms, the above inequality is always satisfied. The minimum values of $\mathbf{v}\cdot\mathbf{h}$, $\mathbf{n}\cdot\mathbf{l}$ and $\mathbf{n}\cdot\mathbf{v}$ can be efficiently computed in the vector space similar to Eq. (20).

The subdivision parameters are determined in the same way that we evenly divide the error threshold and assign to different terms in Eq. (32) to compute n separately. The final subdivision is the maximum value among them.

3.5. Discrete Variables: Textures

We extend our derivation to discrete variables which are often encoded as texture in computer graphics. Textures represent spatially-varying coefficients in shading functions e.g., diffuse or specular albedo, shininess, or roughness values. Formally, the linear interpolation error of shading function f on a convex sub-polygon P_i is:

$$\begin{aligned} &\|f(x,y) - L_\Theta f(x,y)\|_{\infty,P_i} \\ =&\|\hat{f}(\mathbf{A}(x,y),\alpha(x,y)) - L_\Theta f(\mathbf{A}(x,y),\alpha(x,y))\|_{\infty,P_i} \end{aligned} \tag{34}$$

where $\mathbf{A}$ is a set of attributes defined on the shading function, and α is a sampled value from a texture. We consider one texture for simplicity but the following derivation can be applied to multiple textures.

Since texture stores discrete values, we cannot directly obtain an analytical form for $\alpha(x,y)$. However, we observe that most of the shading functions and their second derivatives are usually monotonic functions w.r.t. their coefficients. When evaluating shading functions by sampled α, the maximum interpolation error of $\hat{f}(\mathbf{A},\alpha)$ for all $\alpha^* \in (\alpha_{\min},\alpha_{\max})$ is at $\alpha^* = \alpha_{\min}$ or $\alpha^* = \alpha_{\max}$. We split Eq. (34) into two terms:

$$\begin{aligned} &\|\hat{f}(\mathbf{A},\alpha) - L_{\Theta_i}\hat{f}(\mathbf{A},\alpha)\|_{\infty,P_i} \\ \le& \sup_{\alpha^* \in (\alpha_{\min},\alpha_{\max})} \|\hat{f}(\mathbf{A},\alpha^*) - L_{\Theta_i}\hat{f}(\mathbf{A},\alpha^*)\|_{\infty,P_i} \\ &+ \sup_{\alpha^* \in (\alpha_{\min},\alpha_{\max})} \|L_{\Theta_i}\hat{f}(\mathbf{A},\alpha^*) - L_{\Theta_i}\hat{f}(\mathbf{A},\alpha)\|_{\infty,P_i} \end{aligned} \tag{35}$$

where the first term is the linear interpolation error of $\hat{f}(\mathbf{A},\alpha^*)$ while the second term is the error introduced by replacing α with the fixed value α^*.

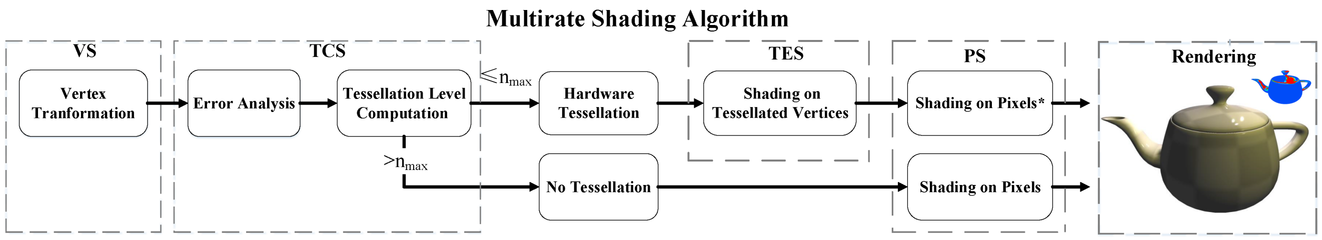

Figure 4: *Overview of our multirate shading algorithm. In the Vertex Shader (VS), we transform vertex coordinates and attributes (e.g., normals) from local space to object space. In the Tessellation Control Shader (TCS), we analyze the interpolation error for each triangle and compute a tessellation level n using a user-specified error threshold* ε*. If n is greater than the maximum-allowed tessellation level* $n_{\max}$*, we revert to conventional dense per-pixel shading (PS: Shading on Pixels). Otherwise, the triangle will be subdivided into sub-triangles by hardware tessellation, and the shading function is only evaluated on subdivided vertices in the Tessellation Evaluation Shader (TES). These shadings values are automatically interpolated to pixels not being sampled. In the Pixel Shader (PS), we perform the rest of the pixel-related operations (Shading on Pixels*). We show our rendering and its tessellation levels at the far right.*

Note that interpolation error of $\hat{f}(\mathbf{A},\alpha^*)$ reaches its maximum at $\alpha^* = \alpha_{\min}$ or $\alpha^* = \alpha_{\max}$. The first term can be computed by:

$$\begin{aligned} &\sup_{\alpha^* \in (\alpha_{\min},\alpha_{\max})} \|\hat{f}(\mathbf{A},\alpha^*) - L_\Theta \hat{f}(\mathbf{A},\alpha^*)\|_{\infty,P_i} \\ &= \max_{\alpha^*=\alpha_{\min},\ \alpha^*=\alpha_{\max}} \{\|\hat{f}(\mathbf{A},\alpha^*) - L_\Theta \hat{f}(\mathbf{A},\alpha^*)\|_{\infty,P_i}\} \end{aligned} \tag{36}$$

For the second term, the supremum of $\|L_\Theta \hat{f}(\mathbf{A},\alpha^*) - L_\Theta \hat{f}(\mathbf{A},\alpha)\|_{\infty,P_i}$ can be constrained as

$$\begin{aligned} &\sup_{\alpha^* \in (\alpha_{\min},\alpha_{\max})} \|L_{\Theta_i} \hat{f}(\mathbf{A},\alpha^*) - L_{\Theta_i} \hat{f}(\mathbf{A},\alpha)\|_{\infty,P_i} \\ &\leq |\hat{f}(\mathbf{A},\alpha_{\max}) - \hat{f}(\mathbf{A},\alpha_{\min})| \end{aligned} \tag{37}$$

In our error estimation, $\alpha_{\min}$ and $\alpha_{\max}$ are the values on the convex sub-polygon P_i. However, we cannot obtain the precise range of $\alpha_{\min}$ and $\alpha_{\max}$ on an individual P_i before subdivision. For conservative estimation, we take $\alpha_{\min}$ and $\alpha_{\max}$ on the original polygonal domain P to obtain $\varepsilon'' = |\hat{f}(\mathbf{A},\alpha_{\max}) - \hat{f}(\mathbf{A},\alpha_{\min})|$. Given an error threshold ε, we compute $\varepsilon' = \varepsilon - \varepsilon''$ and limit the first error term within ε':

$$\max_{\alpha^*=\alpha_{\min},\ \alpha^*=\alpha_{\max}} \{\|\hat{f}(\mathbf{A},\alpha^*) - L_\Theta \hat{f}(\mathbf{A},\alpha^*)\|_{\infty,P_i}\} \leq \varepsilon' \tag{38}$$

With a constant α^*, we estimate its subdivision parameter n using previous derivations.

4. Multirate Shading Algorithm

The goal of our analysis of interpolation error for shading functions is to replace the time-consuming dense sampling process by a dynamic sparse sampling and linear reconstruction with error control. During rendering we first analyze potential interpolation error, determine a proper subdivision, and then compute shading on sparse vertices. To verify such a workflow, we specialize this general sparse sampling idea to a multirate shading algorithm benefiting from hardware automatic subdivision.

4.1. General Framework

We design our multirate shading algorithm in a current forward real-time rendering pipeline, leveraging a hardware tessellator that efficiently subdivides triangles. We show our algorithm in Fig. 4. In a real rendering pipeline, the subdivision parameter n is implemented as the **tessellation level**. In common cases, during forward rendering, shading computations are densely evaluated at pixels in the pixel shaders. We take advantage of tessellation shaders to analyze interpolation errors, subdivide triangles, and perform sparse shading evaluation. In particular, the algorithm accepts a user-specified parameter ε as the L_∞ error threshold. We estimate interpolation error for each triangle in the Tessellation Control Shader (TCS), and determine a tessellation level n based on ε. Given n as input, the hardware tessellator will automatically subdivide the triangle primitive and generate sub-triangles. The subdivided triangles are processed by the Tessellation Evaluation Shader (TES) where we perform a sparse evaluation of shading functions only on the vertices of these triangles. During the rasterization stage, the shading values are automatically interpolated to pixels.

For performance considerations in practice, subdividing numerous sub-triangles at runtime may impose a considerable overload to the rendering pipeline, offsetting all the performance gained from our sparse sampling. To avoid this situation, we introduce another user-specified parameter $n_{\max}$, the maximum-allowed tessellation level, to prevent over-subdivision on triangles. We cancel the subdivision step and revert to conventional per-pixel shading if the computed tessellation level is greater than $n_{\max}$.

4.2. Implementation Details

In our theoretical analysis (Sec. 3), we require the shading function f to be C^2 continuous. However, even the Lambertian model is not always continuous in practice. For example, the Lambertian BRDF is implemented as $\max(0, n \cdot l)$ on the triangle since light can hit its back surface. To address this problem, we propose two solutions. (1) A conservative solution: we revert to per-pixel shading if a discontinuity is detected. (2) A relaxed solution: we approximate the original function with another C^2 continuous function. For in-

Demo	Teapot	Bunny	Monster	Sofa	Sibenik	Room
Tri	15704	13996	19824	6690	107332	130038
$n_{\max}$	8	5	5	5	5	8
$\varepsilon(L_\infty)$	0.075	0.075	0.090	0.090	0.090	0.090
Nvidia RTX 1060 3GB						
$t_{\text{per-pixel}}$	1.19ms	0.79ms	1.89ms	2.39ms	5.80ms	8.23ms
t_{our}	0.50ms	0.35ms	1.01ms	1.15ms	3.24ms	2.90ms
Speedup	233%	227%	188%	207%	177%	283%
Nvidia RTX 2070 Super						
$t_{\text{per-pixel}}$	0.36ms	0.24ms	0.54ms	0.74ms	1.86ms	2.79ms
t_{our}	0.21ms	0.13ms	0.36ms	0.45ms	1.30ms	1.15ms
Speedup	177%	174%	149%	163%	143%	243%
Nvidia RTX 3090						
$t_{\text{per-pixel}}$	0.14ms	0.09ms	0.21ms	0.29ms	0.72ms	1.05ms
t_{our}	0.08ms	0.06ms	0.13ms	0.17ms	0.54ms	0.41ms
Speedup	163%	161%	165%	170%	134%	258%
Error Control						
L_∞ error	0.013	0.073	0.074	0.046	0.068	0.054
Average L_2	<0.001	<0.001	<0.001	<0.001	0.002	<0.001

Table 1: *Result Summary. We list hyparameters including the number of triangles, the maximum-allowed tessellation levels $n_{\max}$ and L_∞ threshold for different test cases. We show rendering time of dense per-pixel shading, our multirate method as well as speedup ratio, measured at 4K resolution (3840x2160). We report L_∞ and L_2 errors to examine error control quantitatively.*

stance, for $\max(0, n \cdot l)$, we simply ignore the clamping function. We show an example of applying different solutions in Fig. 13. The first strategy will always preserve the shading quality which is suggested when the error threshold is low. By contrast, if a high error threshold is given, applying the second strategy can provide improved performance.

Hardware tessellation is achieved by predefined equal-space partitions parameterized by integers i.e. inner/outer tessellation levels [†]. Sample points are generated either on the edges (outer) or the interior of the primitive (inner) by the tessellator. We set both inner/outer tessellation levels the same. To support textures as input, we need to compute minimum and maximum values of texels on a triangle (Sec. 3.5). We leverage normalized power-weighted filtering [Vli04] to efficiently compute max-min values over an arbitrary region on a texture via Summed Area Tables (SATs).

A uniform error threshold partition (Sec. 3) is improved by a heuristic allocation method. For example, given M error terms, we obtain a set of roughly estimated errors on the triangle as e_1, e_2, ..., e_M using Eq. (10) or Eq. (12). With an error threshold ε, we proportionally assign a sub-error threshold for each term as $\varepsilon_j = \frac{e_j}{\sum_j^M e_j}\varepsilon$.

5. Experimental Results

We implement our multirate shading algorithm based on OpenGL 4.5. As we notice the rendering performance is related to specific hardware, we validate its performance on different graphics cards including Nvidia GTX 1060 3GB, Nvidia RTX 2070 Super, RTX 3090. All images are rendered at 4K resolution (3840x2160). We apply our algorithm on several 3D mesh objects and scenes and compare our method with conventional dense per-pixel shading routines. Multiple light sources are used and their errors are additive. We measure the performance improvement by a **speedup ratio** comparing the time for our method versus dense per-pixel shading with the term $\frac{t_{\text{per-pixel}}}{t_{\text{ours}}} \times 100\%$. All results and user-specified parameters are summarized in Table 1. We report L_∞ errors and average L_2 errors for reference showing our algorithm achieves high-quality reconstruction. Results show our method achieves substantial ($134\% \sim 283\%$) performance improvement while maintaining precise error control. We demonstrate the temporal consistency and robustness of our multirate method in our supplemental video.

For visual results, we show different test cases ranging from mesh objects to large-scale scenes with complex geometries. The models are rendered by either the Microfacet BRDF or Blinn-Phong reflection model with a diffuse component using the Lambertian model. In the following sub-sections, all error images shown are 30x scaled, and all the rendering times reported are measured using Nvidia RTX 2070 Super. We visualize the subdivision parameters (i.e., tessellation levels) by heat maps.

5.1. Mesh Objects

In Fig. 2, we show a bunny rendered with the Blinn-Phong model. We compare our method to a fixed-rate subdivision strategy by which we uniformly subdivide all triangles to reduce shading errors. However, a pre-defined subdivision parameter cannot effectively decrease shading errors on triangles with high-frequency signals and leads to over-subdivision on triangles with low-frequency shading signals. We show more comparisons with manually-tuned fixed subdivision rates in Fig. 9. Our method outperforms this naive approach in both error control and time cost.

In Fig. 7, we show a classical Utah teapot rendered with the Microfacet BRDF model. Dense subdivision and sampling are performed on triangles with high specularity such as those on the lid of teapot, while the body of the teapot is mostly shaded by the lowest sampling rate and the BRDF is reconstructed by simple interpolation. Our multirate method determines sample rates adaptive to input material parameters – e.g., roughness value α in this example. The overall errors are controlled by our algorithm. Also note that the shading rate is not only related to specular reflections. It depends on the complex, arbitrary light/view/normal configurations. For example, the thin edge of the teapot lid is shaded with a higher shading rate because of its normal variations.

We show texture mapping is supported in Fig. 8. The sofa model is rendered with a Microfacet BRDF model with a roughness value represented by a texture. The monster model is shaded by a Blinn-Phong model with a spatially varying shininess texture. Different from diffuse albedo or specular albedo, the roughness and shininess values are hard to decouple from the full shading function, therefore the interpolation error must be considered. Naive Gouraud-shading (per-vertex) causes significant errors when the texture encodes non-continuous variables. Our algorithm, on the other hand, properly increases the sample rate to reduce interpolation errors and achieves considerable speedup ratio (up to 163%).

[†] www.khronos.org/opengl/wiki/Tessellation

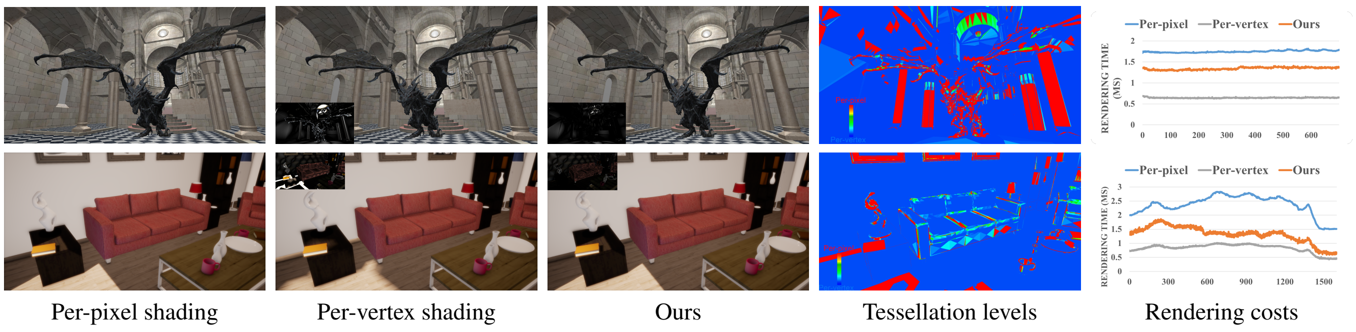

Figure 5: *Rendering results for the Dragon and Room scenes (one frame). We compare dense sampling (Per-pixel), naive interpolation (Per-vertex), and our multirate sparse sampling method (Ours). We show error images (insets), tessellation levels, and time-varying rendering costs. See supplemental video for the full animation.*

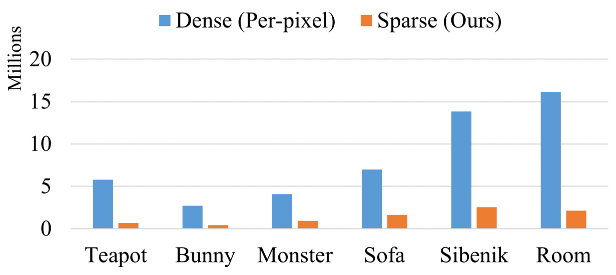

Figure 6: *Our multirate shading method significantly reduces the sample rate for shading functions, only performing* 11% ∼ 23% *shading function evaluations compared to full-rate per-pixel sampling. The X axis shows the times of shading function evaluation (unit: million)*

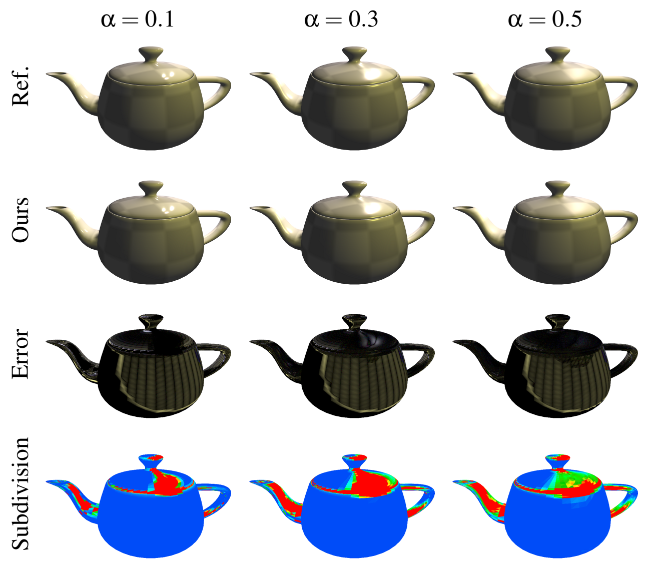

Figure 7: *A Utah teapot rendered by Microfacet BRDF with different roughness values. Our multirate method achieves* 177%, 154% *and* 157% *speedup ratios at different roughness values i.e.,* α*=0.1, 0.3 and 0.5 respectively.*

5.2. Demo Scenes

In Fig. 5, we validate our algorithm on large-scale scenes with complex geometries. The Dragon scene is a dragon model within the Sibenik model, each of which has very distinct characteristics. The dragon model has bumpy surfaces, while the Sibenik model has smooth surfaces. The Room scene is a living room with several furniture and tens of small items such as books, bottles, plants, statues, etc. We demonstrate our multirate method can be incorporated with other rendering techniques. We implement screen-space multirate shadow mapping [HGF14] using shadow boundaries and multiple post-process effects [Unr22] (e.g., blooming, SSAO, eye adaptation and tone mapping) in this living room demo. All post-process effects are performed in a color framebuffer containing direct illumination generated by our method.

For these two demos, our algorithm produces a 134% ∼ 283% performance improvement on different graphics hardware. Additionally, we create animation sequences for these two scenes. In the last column of Fig. 5, we plot the rendering time (ms) per frame, showing our method achieves a consistent performance improvement over dense per-pixel shading.

6. Discussion

Our algorithm leverages sparsely-sampled shading points to reconstruct the original shading functions. To examine the sparsity of sample points generated by our multirate method, we count the shading function evaluation times for each of our scenes when rendered at 4K resolution. In Fig. 6, we show our sparse sampling only performs around 11% ∼ 23% the number of shading function evaluations compared to standard full-rate sampling. We analyze the impact of screen resolution and primitive size on our method, demonstrating concrete performance improvement.

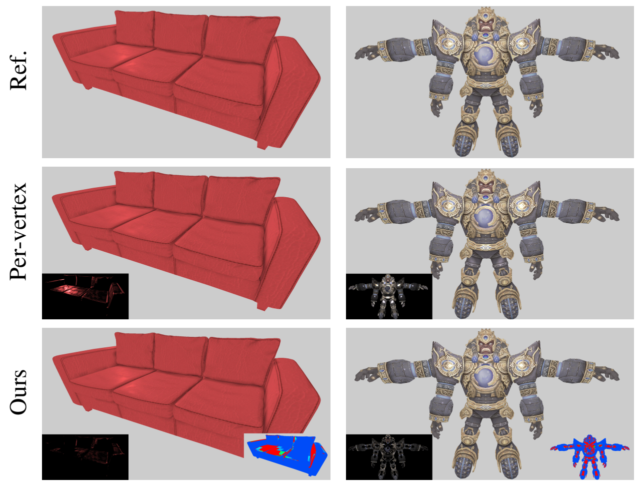

Figure 8: *We dynamically evaluate interpolation error of shading functions with discrete variables as textures. Our multirate approach gains performance improvement up to 163% over per-pixel dense sampling and controls the reconstruction errors caused by spatial varying texture inputs. Error images and subdivision levels are shown as insets.*

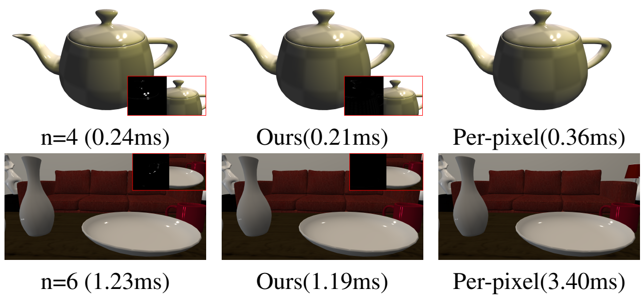

Figure 9: *We show our multirate method outperforms fixed-rate subdivision strategy using uniform tessellation levels n. Our method efficiently suppresses high-frequency shading error and still shows better performance. The Room scene is rendered with direct illumination only to better illustrate the difference. Error maps are scaled by 10. Images are 4K. Please zoom-in to see errors on the high-frequency regions.*

6.1. Independence on Resolutions

Since most of shading computations are executed on the vertices of subdivision surfaces, our method is almost independent of sample resolution, while the time complexity of the reference algorithm — conventional per-pixel shading — scales with the screen resolution. Fig. 10 shows computation cycles occurred in different shader stages collected by the Nvidia Nsight Profiler [Nsi22]. The computation cycles in the vertex shader and tessellation shader at different resolutions are nearly the same in our algorithm because they depend only on geometry complexity. The cycles of TCS are additional cost by error analysis. Since triangles whose subdivision parameter is greater than $n_{\max}$ are reverted to pixel shading, the computation cycles in pixel shader merely increases along with the resolution. Compared to full-rate per-pixel shading, our sparse, multirate shading can produce even better improvement when rendering high-resolution images.

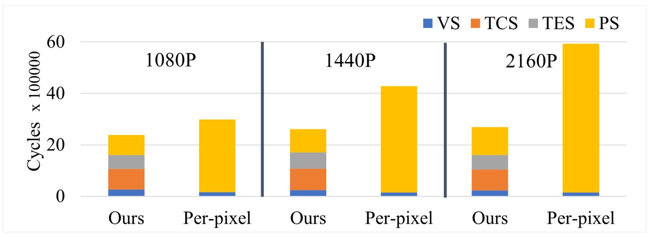

Figure 10: *Breakdown of each shader stage at different resolutions (from left to right, resolutions are 1920*1080, 2560*1440 and 3840*2160 respectively) when shading the bunny model.*

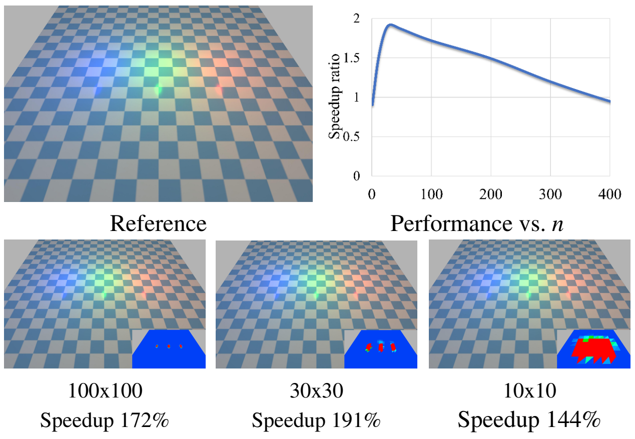

Figure 11: *Performance analysis with different number of triangles. We show performance improvement by our method against the number/size of triangles. The 2D plane is uniformly partitioned into $n \times n$ grids. Each grid is composed by two triangles.*

6.2. Performance with Primitive Size

Our algorithm introduces an additional process to compute subdivision parameters for each visible primitive. Small primitives (e.g., triangles) may require unnecessary error analysis as direct per-pixel shading would be fast enough. On the other hand, large primitives with high-frequency shading signals may require a very fine-grained subdivision to ensure quality.

We conduct an experiment to analyze the influence of primitive size on our algorithm. In Fig. 11, a planar surface is lit by three sharp light sources with different colors. We uniformly partition the 2D plane into triangles with various sizes (i.e., with different numbers of triangles). Additional regions require subdivision to reduce interpolation error when the size of triangle increases (from left to right, bottom row of Fig. 11). On the other hand, the computation cost for error analysis will increase when the number of triangles increases. We plot the speedup ratios against the number of triangles. In this example, our algorithm achieves the maximum speedup ratio at approximately 1800 triangles, and improves the rendering performance for a wide range of numbers of triangles

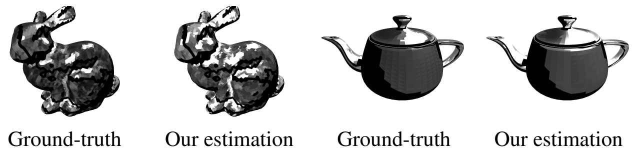

Figure 12: *Our method estimates interpolation errors conservatively and analytically. We show our predicted errors of a Blinn-Phong model on the bunny example and a Microfacet BRDF on the Utah teapot example.*

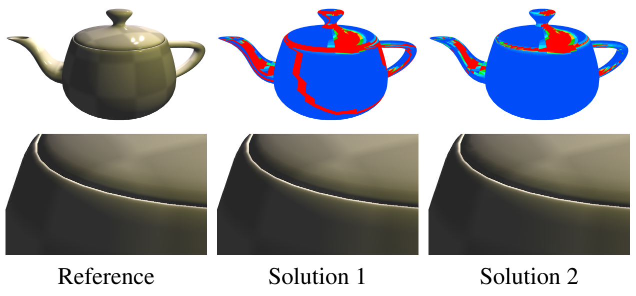

Figure 13: *We propose two solutions to address discontinuities such as in* $\max(0, \boldsymbol{n} \cdot \boldsymbol{l})$*. With the first solution, more triangles (see the long arc region on the body of the teapot) are reverted to per-pixel shading. On the other hand, the second solution ignores the discontinuity. Though it is non-conservative, interpolation errors are barely noticeable.*

even with overhead. The performance slows down when triangles are either too large or too numerous.

6.3. Accuracy of Error Estimation

Our approach predicts interpolation error and computes subdivision parameter n according to a L_∞ threshold. The accuracy of error estimation is crucial for the performance. In Fig. 12, we show a comparison between the ground-truth and our estimated interpolation errors. Our method conservatively evaluates the interpolation error and therefore can effectively reduce the possible reconstruction error by subdivision.

7. Limitations and Future Work

We analyze interpolation errors on multiple shading functions and verify our multirate shading approach is capable of reducing computation costs on a variety of scenarios. However, our approach still relies on certain preliminary assumptions and have limitations.

First, the C^2 continuity of shading functions is not always satisfied, though we practically can address the discontinuity points by approximation or conservative shading (Fig. 13). Since our analytical derivations introduce multiple inequalities to constrain the interpolation error and enable fast evaluation, interpolation errors on a few low-frequency triangles could still be over-estimated such as in Fig. 12. Besides, we derive the error formula by an L_∞ metric because it is invariant to projections or transformations. However, an L_∞ metric could be overly strict, leading to unnecessary subdivision. For instance, as reported in Table 1, the average L_2 is almost zero on the whole image for many cases. Moreover, we leverage a heuristic model to assign the error threshold to different terms (Sec. 4.2), but this may not result in optimal separation. Hence, a relaxed but still quality-guaranteed sparse sampling and error analysis remains an open problem.

Second, though we show our derivation is extended to discrete variables encoded as textures, the current theoretical analysis is hard to extend to non-continuous vector attributes requiring normalization such as normals. Practically, normal mapping is a commonly-used technique to simulate high-frequency geometry details. As we aim at a fully analytical derivation, we estimate errors from vector normalization using our special derived formula (Sec. 3.1.2). Deriving an analytical relationship between the subdivision parameter n and the interpolation error of discrete normal directions is non-trivial. But since the normal map is bound to specific geometry, in practice, the problem could be solved by precomputing such a relation on each triangle: enumerate possible subdivisions, numerically compute the interpolation error, and store the precomputed subdivision parameters for reuse at runtime.

Last, our multirate shading algorithm is built upon a forward rendering framework. We take advantage of hardware tessellation and rasterization as a fast and efficient implementation for sparse sampling and linear reconstruction. As a recent addition to rendering pipelines, mesh shading supports programmable geometry processing which could provide a finer control of the shading rates. We also show the extensibility of our method that can be combined with other rendering techniques such as screen-space post-processing effects. However, the core idea of reconstruction by sparse sampling is universal. Extending our method as a VRS technique that generates samples in screen space is interesting, and how to apply the error analysis on different frameworks such as deferred shading in real-time rendering or ray tracing in off-line rendering is a potential research direction.

8. Conclusion

We propose a novel multirate rendering approach that leverages piecewise linear reconstruction with sparse shading on the vertices of subdivision surfaces. We present an analytical error estimation framework to evaluate interpolation errors and provide an efficient way to compute the granularity of subdivision for various shading functions. We design a multirate shading algorithm based on real-time error analysis and subdivided per-vertex shading, demonstrating that our method achieves considerable performance improvement on a variety of scenes. We hope our work introduces new insights in multirate shading and will inspire more ideas on rendering optimization.

Acknowledgment

We sincerely thank Holly Rushmeier for her discussion and suggestions on this work. This work was supported in part by NSFC (No. 61872319), Key R&D Program of Zhejiang Province (No. 2022C01025), and the Fundamental Research Funds for the Central Universities.

References

[BA08] Boubekeur T., Alexa M.: Phong tessellation. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia 2008) 27*, 5 (2008). 2

[BFM10] Burns C. A., Fatahalian K., Mark W. R.: A lazy object-space shading architecture with decoupled sampling. In *Proceedings of the Conference on High Performance Graphics* (2010), HPG '10, pp. 19–28. 2

[CCC87] Cook R. L., Carpenter L., Catmull E.: The Reyes image rendering architecture. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1987), SIGGRAPH '87, ACM, pp. 95–102. URL: http://doi.acm.org/10.1145/37401.37414, doi:10.1145/37401.37414. 2

[CKY*22] Cardoso J. L., Kerbl B., Yang L., Uralsky Y., Wimmer M.: Training and predicting visual error for real-time applications. *Proc. ACM Comput. Graph. Interact. Tech.* (2022). URL: https://doi.org/10.1145/3522625. 1, 2

[CMFL15] Crassin C., McGuire M., Fatahalian K., Lefohn A.: Aggregate G-buffer anti-aliasing. In *Proceedings of the 19th Symposium on Interactive 3D Graphics and Games* (2015), ACM, pp. 109–119. 2

[CNW96] Cho Y., Neumann U., Woo J.: Improved specular highlights with adaptive shading. In *Proceedings of CG International '96* (1996), pp. 38–46. doi:10.1109/CGI.1996.511785. 2

[CTH*14] Clarberg P., Toth R., Hasselgren J., Nilsson J., Akenine-Möller T.: AMFS: adaptive multi-frequency shading for future graphics processors. *ACM Trans. Graph. 33*, 4 (July 2014), 141:1–141:12. 2

[CTM13] Clarberg P., Toth R., Munkberg J.: A sort-based deferred shading architecture for decoupled sampling. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 141. 2

[Gou71] Gouraud H.: Continuous shading of curved surfaces. *IEEE Transactions on Computers C-20*, 6 (1971), 623–629. doi:10.1109/T-C.1971.223313. 2

[GS05] Guessab A., Schmeisser G.: Sharp error estimates for interpolatory approximation on convex polytopes. *SIAM journal on numerical analysis 43*, 3 (2005), 909–923. 2, 3

[GTGB84] Goral C. M., Torrance K. E., Greenberg D. P., Battaile B.: Modeling the interaction of light between diffuse surfaces. *ACM SIGGRAPH computer graphics 18*, 3 (1984), 213–222. 2

[HGF14] He Y., Gu Y., Fatahalian K.: Extending the graphics pipeline with adaptive, multi-rate shading. *ACM Trans. Graph. 33*, 4 (July 2014), 142:1–142:12. 2, 9

[Hop96] Hoppe H.: Progressive meshes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (1996), SIGGRAPH '96, pp. 99–108. 2

[HSA91] Hanrahan P., Salzman D., Aupperle L.: A rapid hierarchical radiosity algorithm. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques* (1991), SIGGRAPH '91, pp. 197–206. 2

[KBS11] Kavan L., Bargteil A. W., Sloan P.-P.: Least squares vertex baking. In *Proceedings of the Twenty-second Eurographics Conference on Rendering* (2011), EGSR '11, pp. 1319–1326. 2

[LD12] Liktor G., Dachsbacher C.: Decoupled deferred shading for hardware rasterization. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2012), ACM, pp. 143–150. 2

[Nak84] Nakamae E.: A screen subdivision method for half-tone representation of 3-D objects using mini-computers. *Journal of Information Processing 7*, 2 (1984). 2

[Nsi22] Nsight N.: Nvidia, 2022. URL: http://www.nvidia.com/object/nsight.html. 10

[RKLC*11] Ragan-Kelley J., Lehtinen J., Chen J., Doggett M., Durand F.: Decoupled sampling for graphics pipelines. *ACM Trans. Graph. 30*, 3 (May 2011), 17:1–17:17. 2

[Sch94] Schlick C.: An inexpensive BRDF model for physically-based rendering. In *Computer graphics forum* (1994), vol. 13, Wiley Online Library, pp. 233–246. 6

[SKS02] Sloan P.-P., Kautz J., Snyder J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. Graph. 21*, 3 (July 2002), 527–536. 2

[Smi67] Smith B.: Geometrical shadowing of a random rough surface. *IEEE transactions on antennas and propagation 15*, 5 (1967), 668–671. 6

[STCK13] Sloan P.-P., Tranchida J., Chen H., Kavan L.: Ambient obscurance baking on the gpu. In *SIGGRAPH Asia 2013 Technical Briefs* (2013), SA '13, pp. 32:1–32:4. 2

[Sub89] Subbotin Y. N.: The dependence of estimates of a multidimensional piecewise-polynomial approximation on the geometric characteristics of a triangulation. *Trudy Matematicheskogo Instituta imeni VA Steklova 189* (1989), 117–137. 2, 4

[TR75] Trowbridge T., Reitz K. P.: Average irregularity representation of a rough surface for ray reflection. *JOSA 65*, 5 (1975), 531–536. 6

[Unr22] Unreal: Unreal Engine 4, 2022. URL: https://www.unrealengine.com/. 9

[Vli04] Vliet L. J. V.: Robust local max-min filters by normalized power-weighted filtering. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.* (2004), vol. 1, pp. 696–699 Vol.1. 8

[VST*14] Vaidyanathan K., Salvi M., Toth R., Foley T., Akenine-Möller T., Nilsson J., Munkberg J., Hasselgren J., Sugihara M., Clarberg P., et al.: Coarse pixel shading. In *High Performance Graphics* (2014). 2

[Wal98] Waldron S.: The error in linear interpolation at the vertices of a simplex. *SIAM Journal on Numerical Analysis 35*, 3 (1998), 1191–1200. 2, 4

[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), EGSR'07, pp. 195–206. 6

[WYY*14] Wang R., Yang X., Yuan Y., Chen W., Bala K., Bao H.: Automatic shader simplification using surface signal approximation. *ACM Trans. Graph. 33*, 6 (Nov. 2014), 226:1–226:11. 2

[XLV18] Xiao K., Liktor G., Vaidyanathan K.: Coarse pixel shading with temporal supersampling. I3D '18, Association for Computing Machinery. URL: https://doi.org/10.1145/3190834.3190850, doi:10.1145/3190834.3190850. 2

[YSL08] Yang L., Sander P. V., Lawrence J.: Geometry-Aware Framebuffer Level of Detail. *Computer Graphics Forum* (2008). doi:10.1111/j.1467-8659.2008.01256.x. 2

[YWB18] Yuan Y., Wang R., Bao H.: Tile pair-based adaptive multi-rate stereo shading. *IEEE Transactions on Visualization and Computer Graphics 26*, 6 (2018), 2303–2314. 2

[YWH*16] Yuan Y., Wang R., Huang J., Jia Y., Bao H.: Simplified and tessellated mesh for realtime high quality rendering. *Computers & Graphics 54* (2016), 135–144. 2

[YZK*19] Yang L., Zhdan D., Kilgariff E., Lum E. B., Zhang Y., Johnson M., Rydgård H.: Visually lossless content and motion adaptive shading in games. URL: https://doi.org/10.1145/3320287, doi:10.1145/3320287. 1, 2

[Zat93] Zatz H. R.: Galerkin radiosity: A higher order solution method for global illumination. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (1993), SIGGRAPH '93, pp. 213–220. 2