Eurographics Symposium on Rendering 2024
E Garces and E. Haines
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 43 (2024), Number 4

Neural Histogram-Based Glint Rendering of Surfaces With Spatially Varying Roughness

I. Shah[1] L. E. Gamboa[2†] A. Gruson[3] P. J. Narayanan[1]

[1]CVIT, International Institute of Information Technology, Hyderabad (IIIT-H), India
[2]Universidad Michoacana de San Nicolás de Hidalgo, Mexico
[3]École de Technologie Supérieure, Canada

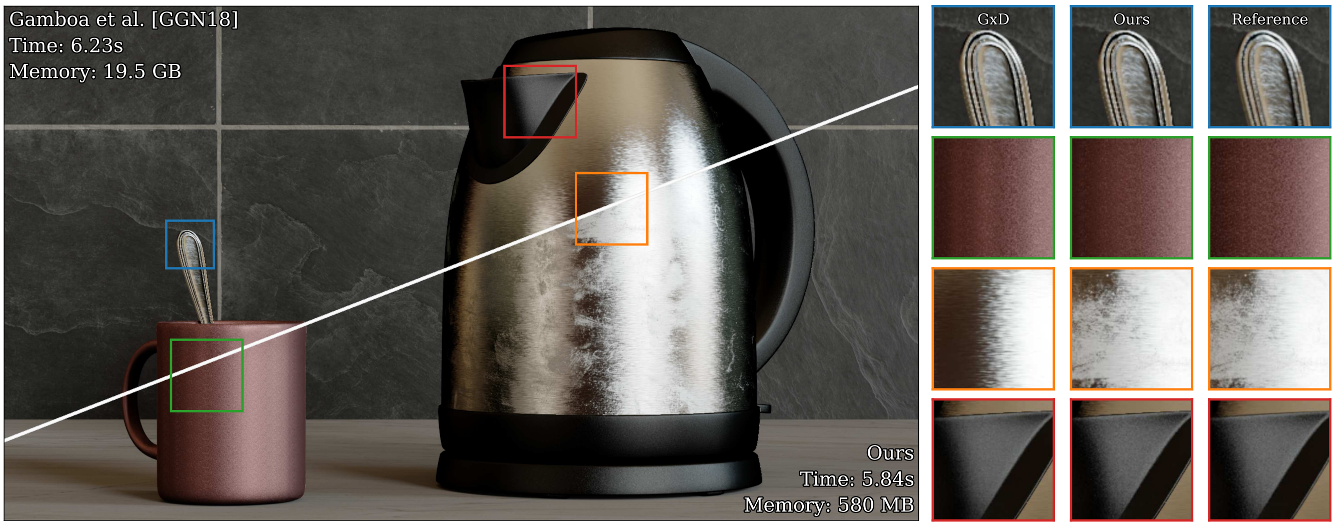


Figure 1: *Our method improves realism by supporting high-frequency normal-mapped surfaces under complex lighting configurations. Compared to previous methods, our approach has moderate memory requirements and scales well to multiple normal maps (we show four). Additionally, it has minimal error compared to the ground truth, and requires only one sample per pixel evaluation, excluding shadows. Our method supports freely varying editable spatial roughness (orange) while maintaining analytical integration of the material and lighting responses at no additional cost.*

Abstract

The complex, glinty appearance of detailed normal-mapped surfaces at different scales requires expensive per-pixel Normal Distribution Function computations. Moreover, large light sources further compound this integration and increase the noise in the Monte Carlo renderer. Specialized rendering techniques that explicitly express the underlying normal distribution have been developed to improve performance for glinty surfaces controlled by a fixed material roughness. We present a new method that supports spatially varying roughness based on a neural histogram that computes per-pixel NDFs with arbitrary positions and sizes. Our representation is both memory and compute efficient. Additionally, we fully integrate direct illumination for all light directions in constant time. Our approach decouples roughness and normal distribution, allowing the live editing of the spatially varying roughness of complex normal-mapped objects. We demonstrate that our approach improves on previous work by achieving smaller footprints while offering GPU-friendly computation and compact representation.

1. Introduction

Many real-world objects exhibit a rich, glinty appearance due to mesoscale features like scratches, bumps, or flakes. Efficiently modeling these appearances at different scales is important for achieving realism in the rendering process. This phenomenon can arise even under large light sources, requiring additional integration.

Material appearance is often modeled using high-resolution normal maps. Unfortunately, normal map filtering is not a linear process and requires special filtering techniques to keep the appearance

† Corresponding author: luis.gamboa@umich.mx

of the surface across scales. Dedicated data structures [AWKK21, GGN18] or hierarchical representations [YHMR16, DLW*22] are used to model the underlying normal distribution (NDF) inside the pixel footprint. These techniques can be expensive in memory or not support analytical integration over large light sources.

The NDF is typically extended by an intrinsic roughness, introduced by Yan et al. [YHJ*14] to avoid singularities. This parameter can also be used to improve the expressiveness of the material model. Intrinsic roughness can be understood by considering a two-level microfacet model similar to Holzschuch et al. [HP17] where the normal map models the distribution of *mesofacets* while the intrinsic roughness models the distribution of *microfacets* on each *mesofacet*. Adding the ability to modulate the intrinsic roughness spatially allows for additional artistic control. For example, one can model impurities such as dirt by using a higher intrinsic roughness (fig. 1). Such details significantly improve realism and cannot be reproduced with only a normal map. For succinctness, we will refer to *intrinsic roughness* as *roughness* in the rest of the article.

We propose a new method to accurately and efficiently compute per-pixel NDF ($\mathcal{P}$-NDF) from high-resolution normal maps and varying roughness under complex lighting. We achieve this by combining a roughness independent binned NDF [GGN18], with a novel neural-histogram and an improved adaptive discretization. Our method needs 60$\times$ less memory for representing the underlying distribution compared to Gamboa et al. [GGN18]. Our approach introduces support for spatially varying roughness at negligible additional cost both in memory and performance. Our main contributions are:

- An adaptive binning strategy to reduce discretization error of the continuous $\mathcal{P}$-NDF.
- A neural network-based method for obtaining the spherical histogram for a patch, reducing the memory requirements and improving the performance of the algorithm.
- Support for double filtering both the material appearance and lighting using Spherical Harmonics (SH).
- A compact and tabulated Zonal Harmonics (ZH) representation and efficient on-the-fly rotation to render glinty surfaces with spatially varying (SV) roughness.

2. Related work

In this section, we relate to existing works grouped by how they represent and query the NDF.

Microfacet theory Similarly to prior works, our method is based on microfacet theory and targets to support the filtering operation of the NDF at different scales, however, unlike the analytical model [BS63, WMLT07], our BSDF model parameterize the NDF by using a normal map. Support for SV roughness in the analytical model is possible by relying on linear filtering of a roughness texture or top level integration, however, this approach cannot be used for normal-mapped NDFs.

Implicit representations Some prior works use an implicit representation for the NDF, meaning the normal distribution is generated on-the-fly. For instance, Jakob et al. [JHY*14] stochastically produce discrete oriented facets and determine how many are well aligned based on the spatial and directional support. Similar ideas have been adapted to real-time rendering based on bi-scale NDF [ZK16], and physically-based multi-scale precomputed NDF [CSDD20]. Our approach differs as we explicitly express the NDF from the normal map using neural networks and target offline rendering. Raymond et al. [RGB16] have developed a specialized SV-BRDF for scratches that support multi-scale filtering and model inter-reflection inside the scratches. Their approach is compact but does not generalize to arbitrary distributions.

Explicit NDF Filtering the underlying normal map accurately is a challenging task. LEAN [OB10] and LEADR [DHI*13] express multiscale auxiliary statistics such as mean and variance that are compatible with MIP-mapping [Wil83]. These methods are cheap and effective but cannot model complex materials' glint distributions across different scales accurately. Yan et al. [YHJ*14] developed a hierarchical technique to prune normals in the NDF that do not contribute to the final shading. While this technique is highly accurate and can support SV roughness, it is expensive for large footprint sizes and light sources. Later approaches have improved performance [YHMR16] or introduced wave-effects [YHW*18], however, there's no support for SV roughness or analytic integration of large light sources. Wang et al. [WDH20] accelerate a 4D Gaussian query by blending and synthesizing different distributions. Additionally, their technique supports environment maps using a prefiltering approach, however, they cannot synthesize macro-scale glints. Deng et al. [DLW*22] developed a prefiltering method that provides constant-cost for rendering glints. They achieve this by expressing the non-overlapping NDF images for a 32×32 footprint and then compressing this representation using tensor decomposition [KB09]. Their approach uses MIP-mapping for larger footprints, however, we found that trilinearly interpolated NDFs result in a blurry distribution. In contrast, we use a Binned NDF representation with a neural network to properly interpolate NDF at different scales without introducing blurriness.

Binned NDF Gamboa et al. [GGN18] combine normal map filtering and environment map lighting. Their technique uses SH expressed in the half-direction domain to compute the double product integral between a histogram-based NDF and environment lighting. Although this approach is computationally efficient, it requires a significant amount of memory, i.e., 4.8 GB for a normal map with a resolution of $2K \times 2K$. We borrow from their ideas, improving data representation to increase accuracy and achieve a smaller memory footprint and higher performance while adding support for SV roughness. The approach by Atanosov et al. [AWKK21] uses a forest of kd-trees to accelerate histogram lookup. They set the bin size based on the roughness of the surface. While this data representation is not memory intensive ($\sim$10-40 MB depending on histogram resolution), the lookup cost still depends on footprint size. Furthermore, it also loses the ability to filter both surface appearance and environmental lighting. This technique could also support SV roughness by choosing a bin size in accordance to the minimum roughness, but, it's impact on the performance needs to be evaluated.

Neural-network in material models Neural models [MESK22] or differentiable indirection [DMD*23] have proven to be an alternative versatile compression scheme. In particular, neural networks have been used extensively for BSDF appearance compression [HGC*20, RGJW20, FWH*22] or improving filtering for SV-BRDF map [GFL*22]. A recent approach by Zeltner et al. [ZRW*23], demonstrates how careful network design and hierarchical MIP-mapped latent code [KMX*21] can capture detailed surface appearance at different scales. We also use NeuMIP [KMX*21] in our technique to compress and pre-filter our representation. Other neural architectures, such as generative ones, can be used to generate on-the-fly targeted NDF distributions [KHX*19] or even complex material graphs [GHS*22, HGH*23]. These generative approaches lift storage limitations regarding the finite texture resolution. Our technique relies on texture representation and these works are considered orthogonal.

Note that most deep appearance models require MC integrations of the incoming lighting. Specialized approaches have been developed to generate samples proportional to the neural material. Xu et al.'s [XWH*23] histogram approach can be seen as similar to ours; however, our histogram captures the NDF profile and supports prefiltering operations.

3. Background

Our method is memory-efficient, GPU-friendly, and capable of analytical integration. We employ explicit NDF and Spherical Harmonics for surface appearance modelling and light integration. We also support spatially varying roughness and don't need additional precomputations when the roughness map changes. We now explain the relevant concepts briefly.

3.1. Explicit NDF

Microfacet models [CT81] have been traditionally used for describing the BRDF at a point $\mathbf{x}$

$$f_r(\mathbf{x}, \omega_i, \omega_o) = \frac{F(\omega_i \cdot \omega_o) G(\omega_i, \omega_o) D(\mathbf{x}, \omega_h)}{4(\mathbf{n} \cdot \omega_o)(\mathbf{n} \cdot \omega_i)}, \tag{1}$$

where ω_i, ω_o are the incident and outgoing directions respectively, $\mathbf{n}$ is the geometric normal at $\mathbf{x}$ and $\omega_h = \frac{\omega_i + \omega_o}{\|\omega_i + \omega_o\|}$ is the half-vector. F and G are the Fresnel and shadowing-masking terms, respectively, and the *normal distribution function* (NDF), $D(\mathbf{x}, \omega_h)$, describes the distribution of microfacets on the surface. In this paper we focus on explicit NDFs defined by an underlying high-resolution normal map instead of infinitesimally small microfacets [YHJ*14].

The explicit NDF ($\mathcal{P}$-NDF) is defined by a set of normals present in a patch $\mathcal{P}$ on the normal map. The patch is given by projecting the pixel footprint on the normal map. Yan et al. [YHJ*14] and subsequent work perform 2D Gaussian-weighting across the footprint. Contrary to this, we use equal-weighting similar to prior work on discrete $\mathcal{P}$-NDF [JHY*14, GGN18, AWKK21]:

$$D(\mathcal{P}, \omega_h) = \frac{1}{N_{\mathcal{P}}} \sum_{\mathbf{x} \in \mathcal{P}} G_r(\omega_h; \omega_x, \sigma_s), \tag{2}$$

where G_r is an isotropic Gaussian that enables intrinsic roughness σ_s around a mean direction obtained from the normal map, i.e. $\omega_x = \text{normal}(\mathbf{x})$, and $N_{\mathcal{P}}$ is the number of normals in the patch.

We can further discretize normals in the normal map into a set $\widehat{\Omega}$ of directions ω_j, resulting in the binned $\mathcal{P}$-NDF:

$$\widehat{D}(\mathcal{P}, \omega_h) = \frac{1}{N_{\mathcal{P}}} \sum_{\omega_j \in \widehat{\Omega}} c^{\mathcal{P}}_{\omega_j} G_r(\omega_h; \omega_j, \sigma_s), \tag{3}$$

where $c^{\mathcal{P}}_{\omega_j}$ indicates the count for normal ω_j inside footprint $\mathcal{P}$. We discuss the set $\widehat{\Omega}$ and how we choose it in section 4.1. Gamboa et al. [GGN18] store this binned NDF as a spherical histogram, which is memory inefficient. To this end, in section 4.2 we propose a neural-based histogram for efficiently querying arbitrary footprints and achieving a low-memory profile.

3.2. Spherical Harmonics

Spherical Harmonics are orthonormal basis functions defined on the sphere that allows us to express directionally-varying functions as a weighted sum. The real basis functions Y for band $l \in \{0, \ldots, l_{\max}\}$ and index $m \in \{-l_{\max}, \ldots, l_{\max}\}$ are defined as:

$$Y^l_m(\omega) = \begin{cases} \sqrt{2}\, K^m_l\, P^{-m}_l(\cos\theta)\, \sin(-m\phi) & m < 0 \\ K^m_l\, P^m_l(\cos\theta) & m = 0 \\ \sqrt{2}\, K^m_l\, P^m_l(\cos\theta)\, \cos(m\phi) & m > 0 \end{cases} \tag{4}$$

where P are the associated Legendre Polynomials and K are the normalization factors.

Given a function f, computing its SH coefficients requires:

$$f^m_l = \int_{S^2} f(\omega) Y^m_l(\omega) \mathrm{d}\omega, \tag{5}$$

and with sufficient SH order (i.e. $l_{\max}$), function f can be perfectly represented by a vector f_{SH} of all coefficients f^m_l.

One very useful property of SH is the double product integral, which can be computed exactly as a dot product of the SH vectors of each function, i.e., $\int_{S^2} f(\omega) g(\omega) \mathrm{d}\omega = (f_{\text{SH}} \cdot g_{\text{SH}})$, meaning that a Light-BRDF integral can be efficiently computed if we have the corresponding SH coefficients.

3.3. Filtered Appearance Rendering

Typically, the rendering equation [Kaj86] is a part of the double integral that filters point-wise contributions to remove aliasing:

$$L_o(\mathcal{P}, \omega_o) = \int_{\mathcal{P}} \int_{\Omega} L_i(\mathbf{x}, \omega_i) f_r(\mathbf{x}, \omega_i, \omega_o)(\mathbf{n} \cdot \omega_i) \mathrm{d}\omega_i \mathrm{d}\mathbf{x}. \tag{6}$$

Monte Carlo integrators perform this filtering task implicitly at no extra cost per sample, however, using high-resolution normal maps requires extremely high sample counts to resolve all normal variation and lighting effects correctly [YHJ*14]. Under a far-field assumption and a microfacet model, we can consider all normal filtering and variation inside the $\mathcal{P}$-NDF [YHJ*14, JHY*14]:

$$L_o(\mathcal{P}, \omega_o) \approx \int_{\Omega} \frac{L_i(\mathbf{x}, \omega_i) F(\omega_i \cdot \omega_o) G(\omega_i, \omega_o) \widehat{D}(\mathcal{P}, \omega_h)}{4(\mathbf{n} \cdot \omega_o)} \mathrm{d}\omega_i, \tag{7}$$

with $\mathcal{P}$ computed around the observed point $\mathbf{x}$ using e.g. ray differentials [Ige99]. We can approximate eq. (7) by decoupling F and G terms, and then re-parameterize to half-direction space to form a

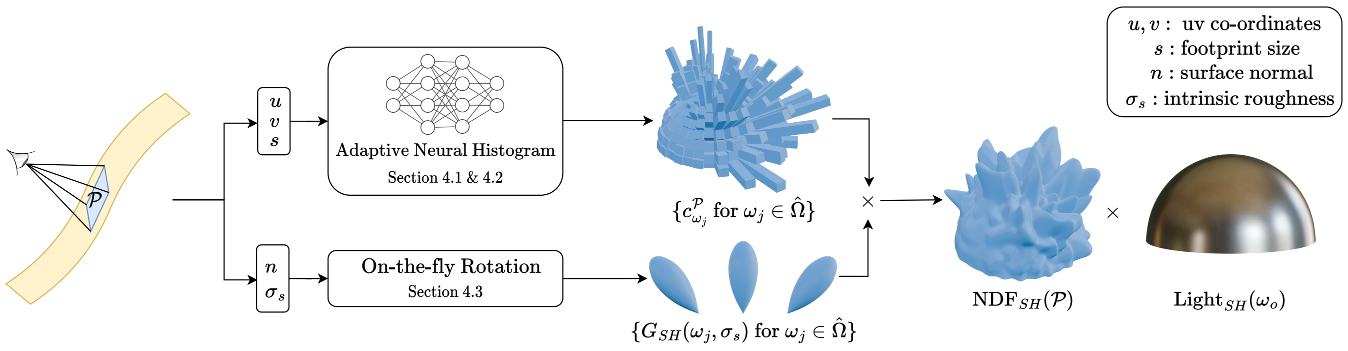


Figure 2: *For a pixel footprint with location (u, v) and size s, we query our Adaptive Neural Histogram (sections 4.1 & 4.2) to get bin counts. SH coefficients for each rotated Gaussian lobe with roughness σ_s are obtained through on-the-fly Zonal Harmonics rotation (section 4.3). Multiplying these coefficients with corresponding bin count yields the SH vector of the $\mathcal{P}$-NDF. Finally, a dot product of the $\mathcal{P}$-NDF and Light SH vectors (section 3.3) results in the final shading.*

double product Light-NDF integral. The effective problem to solve becomes:

$$L_o(\mathcal{P}, \omega_o) \approx \widehat{FG}(\omega_o) \int_{\Omega} L_i(\mathbf{x}, \omega_i) \widehat{D}(\mathcal{P}, \omega_h) \mathrm{d}\omega_i$$
$$= \widehat{FG}(\omega_o) \int_{\Omega_h} \underbrace{\widehat{L}_i(\mathbf{x}, \omega_o, \omega_h) 4(\omega_h \cdot \omega_o)}_{\text{Light}} \underbrace{\widehat{D}(\mathcal{P}, \omega_h)}_{\text{NDF}} \mathrm{d}\omega_h \,. \tag{8}$$

Going between $\widehat{L}_i$ and L_i is trivial as $\omega_i = \text{reflect}(\omega_o, \omega_h)$. Projecting each function to SH allows solving eq. (8) efficiently with:

$$L_o(\mathcal{P}, \omega_o) \approx \widehat{FG}(\omega_o) (\text{Light}_{\text{SH}}(\omega_o) \cdot \text{NDF}_{\text{SH}}(\mathcal{P})) \,. \tag{9}$$

Successfully computing these coefficients is expensive, as projecting a function is an integration process, and under eq. (9) each SH vector is dependent on either view direction or footprint. Light$_{\text{SH}}$ can be precomputed densely for ω_o for environment emitters.

Additionally, Light$_{\text{SH}}$ and NDF$_{\text{SH}}$ are in the global and local frame, respectively, which mandates a change of frame, i.e. an expensive SH rotation, before the dot product. Gamboa et al. [GGN18] precomputed rotated SH lobes tied to a fixed roughness to accelerate this process. Instead, in section 4.3 we show how to obtain an arbitrary roughness NDF$_{\text{SH}}$ relying on on-the-fly Zonal Harmonics rotation while lowering memory requirements and improving performance.

4. Method

Our method pipeline is shown in Figure 2. We introduce an adaptive binning strategy (Sec 4.1), that reduces discretization errors and significantly improves rendering quality for equal histogram resolution. Our neural histogram (Sec 4.2) compresses and predicts this adaptive representation, allowing efficient appearance filtering. Finally, we propose to change how SH coefficients of $\mathcal{P}$-NDF are calculated, reducing the memory requirement and allowing spatial varying or live editing of intrinsic roughness (section 4.3).

4.1. Adaptive Binning

The binned $\mathcal{P}$-NDF introduces discretization errors, especially at lower roughness values or under sharp lighting. This error translates as structural artifacts inside the rendered images (Figure 3, left). A naïve way of reducing this error is to increase the histogram resolution at the cost of runtime and storage requirements. We mitigate this by distributing bins based on the distribution of normals in the normal map (Figure 3, center). Similar to Gamboa et al. [GGN18], we parameterize the histogram bins using spherical coordinates, i.e.

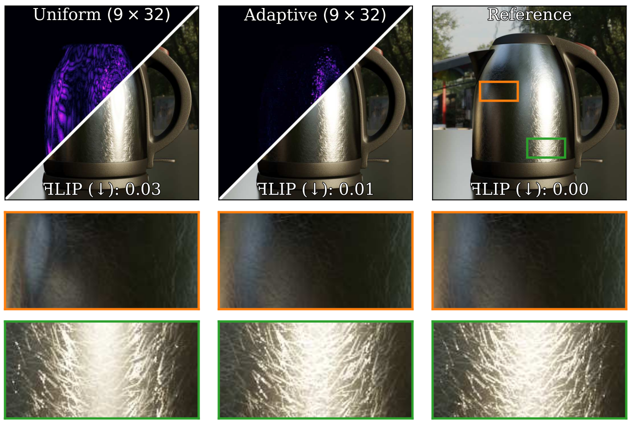


Figure 3: *Comparing uniform and adaptive binning techniques under low-frequency (orange) and high-frequency (green) lighting. We render all images with Monte Carlo to isolate the error caused by using the binned NDF. Adaptive binning clearly improves quality (ꟻLIP difference scaled by 2×).*

$$\theta = \arccos(\omega_h.z)$$
$$\phi = \arctan\left(\omega_h.y, \omega_h.x\right) . \tag{10}$$

Given a normal map, we want to identify the set $\widehat{\Omega}$ of $N_\theta \times N_\phi$ bin centers that best represent the underlying distribution. We have analyzed multiple normal maps and observed that the distribution of ϕ is mostly uniform. Based on this observation, we seek to find only the optimal θ bins while retaining the uniform discretization along ϕ.

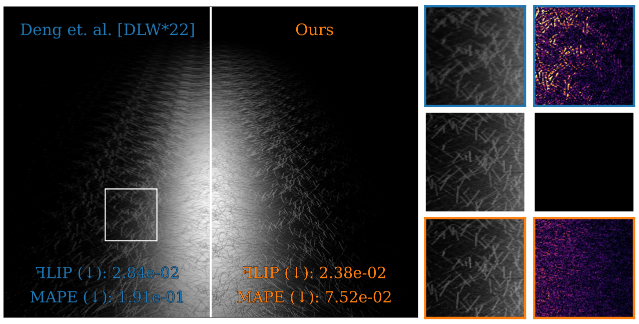


Figure 4: *Comparison between Deng et al. [DLW*22] style linear interpolation and our technique. Our use of neural networks results in fewer interpolation artifacts being observed. As we can see from the reference (middle row insets), our method (bottom row) reproduces the resulting appearance more faithfully. We use Mean Absolute Percentage Error (MAPE) to visualize the difference better and report both MAPE and ꟻLIP scores.*

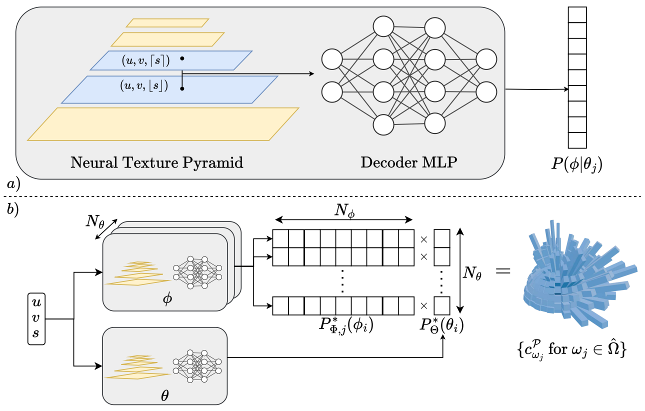


Figure 5: *Our proposed neural architecture. a) Our neural texture pyramid is inspired by [KMX*21] and decoder MLP, which predicts a partial histogram. b) We decompose the histogram into multiple smaller partial histograms and compress them using $N_\theta + 1$ smaller MLPs.*

Adaptive θ discretization Our approach uses K-Means++ [AV07], which improves the seeding procedure for the K-Means algorithm that minimizes an L2 norm. More specifically, we use the greedy initialization variant of this technique. Compared to simpler solutions such as uniformly distributing the bins, K-Means assigns bins to clusters, emphasizing regions with a high concentration of normals and providing a balanced representation important to conserve the material's appearance (Figure 3). The adaptive bins are computed once by considering all of the normals of the normal map.

4.2. Neural Histogram

We present a compact, flexible representation that supports filtering for different pixel footprints and can be easily parallelized on the GPU. This replaces the memory-intensive summed area table in prior work [GGN18]. Our main idea is to compress the underlying distribution by storing a sparse histogram for a given footprint and perform interpolation at run-time to reconstruct the histogram for the footprint. In practice, we set the minimum footprint size $s = 8 \times 8$ to balance the memory budget and the accuracy of our baseline model. We can explicitly iterate over all the corresponding normals for smaller footprints.

Our approach is similar to Deng et al. [DLW*22], which converts a high-frequency normal map into a set of sparsely distributed NDF images organized as a grid and compresses each image NDF using tensor decomposition [KB09]. MIP-mapping interpolation is then used to reconstruct the NDF image associated with a given query, decompressing their decomposition for a single direction. In contrast, we directly obtain the requested histogram from our network. Additionally, their representation is tied to the input NDF image set computed from fixed intrinsic roughness. This process fundamentally differs from ours since a histogram is naturally roughness-independent.

In Figure 4, we show that trilinear interpolation-based methods suffer from additional blurriness due to linearly blending different NDFs. In contrast, our method uses a deep learning-based approach to produce faithful surface appearances. For this comparison, we use a scratches normal map with a fixed roughness of 0.05 and a point light. Similar results were observed for different normal maps. The reference is computed with MC using square footprints to isolate the differences between the NDF representation and interpolation approaches. In the next subsection, we will empirically describe and justify our network design.

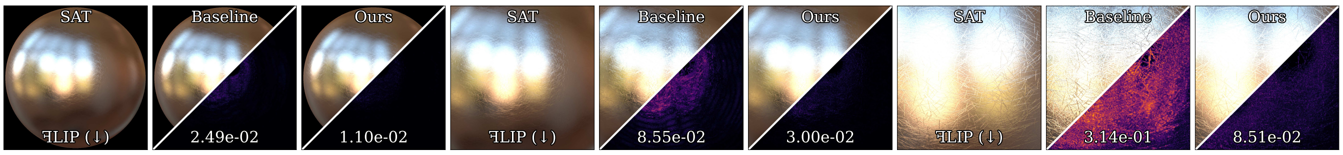
4.2.1. Baseline neural model

We adapt NeuMIP's design [KMX*21], which uses a feature pyramid as a natural solution to our problem. The feature pyramid is arranged as textures with decreasing spatial resolution, similar to MIP-mapping. We obtain the feature vector by trilinearly interpolating inside the feature pyramid for a given footprint size and location. Then, a Multi-Layer Perceptron (MLP) network decodes them to obtain the final histogram for the given footprint.

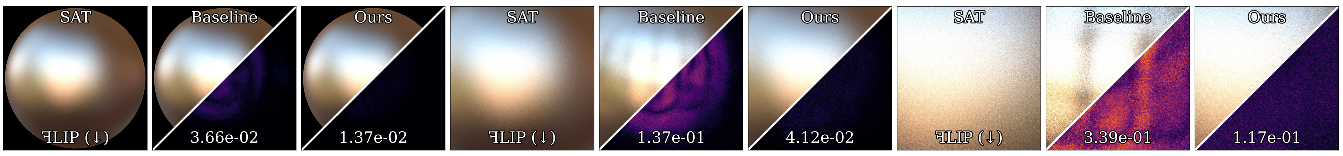
We experimented by replacing trilinear interpolation to choose one level stochastically [ZRW*23], but found the additional noise added to final images detrimental as our integration is noise-free. Usually, our targeted histogram resolution is 9×32 bins. For this resolution, the baseline model uses MLP with 4 layers of width 256 each. Each layer is followed by a **ReLU** non-linearity. We apply the **TriangleWave** non-linearity with amplitude of 1 and period of 2 on the final layer since the normalized range of histogram count will always lie in the range $0 - 1$. We normalize this output by the total sum of predicted logits to ensure they sum to 1.

4.2.2. Improved network design

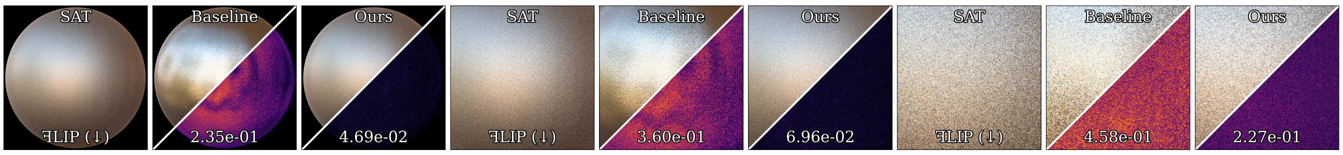
While the baseline method works well, it has two issues. First, the size of the MLP required for high-quality results is too expensive, as it directly affects the performance of histogram queries. Second, scaling to higher histogram resolutions requires an even bigger MLP and potentially larger feature vectors. Keeping multiple MLPs rather than a large one is beneficial as it allows important



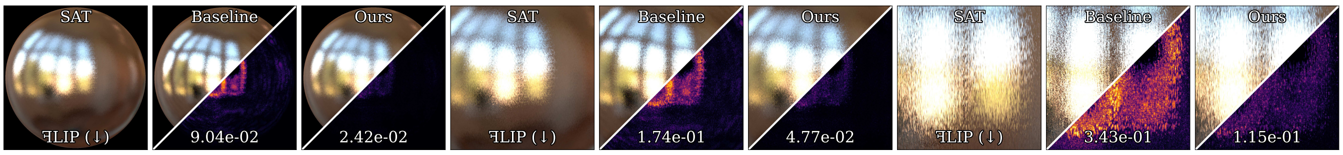
(a) *Scratches*



(b) *Isotropic*



(c) *Flakes*



(d) *Brushed Metal*

Figure 6: *Comparison between the baseline neural model and our multiple small network design for different normal maps and $\mathcal{P}$-NDF at three different scales. The summed area table (SAT) acts as a reference pixel response, but uses $87\times$ more memory than our neural approach. Our network design outperforms the baseline model while maintaining quality across different scales.*

practical optimization, e.g., operation fusing or exploiting the current GPU caching architecture (Nvidia's Ada Lovelace).

Based on these issues, we propose partitioning the histogram into partial histograms and training multiple MLPs to predict them (fig. 5). The normalized bin count can be expressed as a probability mass function $P(\phi_i, \theta_j)$ for a bin oriented in (ϕ_i, θ_j). We can then decompose this density as

$$P(\phi_i, \theta_j) = P(\phi_i|\theta_j)P(\theta_j) \approx P^*_{\Phi,j}(\phi_i)P^*_{\Theta}(\theta_j)\,, \tag{11}$$

where $P^*_{\Phi,j}$ is the learned conditional partial-histogram with θ_j oriented bins, and P^*_{Θ} is the learned densities of ϕs. Figure 5 show our proposed architecture. We need $N_\theta + 1$ neural network inference to estimate the full histogram.

4.2.3. Baseline vs improved architecture

Our improved architecture (section 4.2.2) relies on partitioning the histogram to make learning the signal easier. Instead of using the baseline model (section 4.2.1), which consists of a 256×4 MLP, we can use a much smaller MLP of size 64×2 for better results. Furthermore, since individual evaluations of the smaller MLP are much faster than the larger MLP, evaluating multiple such MLP is still faster. Our improved version is 1.2×-1.3× faster than the baseline model. This difference can be explained as the improved architecture also respects hardware constraints on current GPU, allowing faster computation and fused networks.

Figure 6 shows that both methods resolve different normal distributions at different scales, however, our improved architecture always produces a more accurate NDF distribution. This results in fewer errors in the generated renders. For reference, we use the NDF SAT representation used in Gamboa et al. [GGN18], which accurately reproduces the NDF distribution at the cost of using 87× more memory than our neural representation. More comparisons between our model and the technique of Gamboa et al. are available in the results section.

4.3. On-the-fly Rotation

Given the spherical histogram of a footprint, the SH coefficients of the $\mathcal{P}$-NDF, i.e. substitute eq. (3) in eq. (5), are computed as follows:

$$\mathrm{NDF}_{\mathrm{SH}}(\mathcal{P}) = \frac{1}{N_{\mathcal{P}}} \sum_{\omega_j \in \widehat{\Omega}} c^{\mathcal{P}}_{\omega_j} G_{\mathrm{SH}}(\overline{\omega_j}, \sigma_s)\,, \tag{12}$$

where $c^{\mathcal{P}}_{\omega_j}$ is the count of ω_j in $\mathcal{P}$ and $G_{\mathrm{SH}}(\overline{\omega_j}, \sigma_s)$ are the SH coefficients of a Gaussian lobe oriented in direction $\overline{\omega_j}$ and roughness σ_s. Notice that if $\omega_j = \overline{\omega_j}$ then the resulting SH coefficients are in the local frame, however, when solving eq. (12) we can go straight to the global frame by making $\overline{\omega_j} = \mathrm{toGlobal}(\omega_j)$. Gamboa et al. [GGN18] accelerate this computation by storing G_{SH} coefficients with fixed directions at 4× higher resolution than the his-

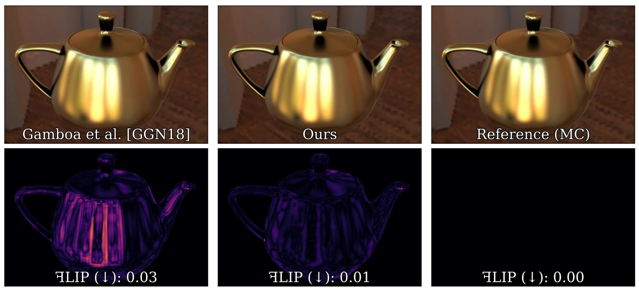


Figure 7: *Comparison between precomputed [GGN18] and our on-the-fly computation of NDF_{SH} coefficients. The $(2\times)$ ꟻLIP difference (bottom row) shows that our method eliminates block-like artifacts stemming from the discretization of stored SH coefficients.*

togram resolution (i.e. 65×128). Then, at runtime, they bi-linearly interpolate to compute the SH coefficients for an arbitrary direction. This approach has two issues:

- Interpolation and discretization can generate visual artifacts with low roughness surfaces (fig. 7, left).
- The rotation array has fixed σ_s and storage cost significantly increases if there are objects with spatially varying roughness or if there are objects with different roughnesses.

4.3.1. Fast Rotation of Zonal Harmonics

Zonal Harmonics (ZH) are a special case of SH that can represent circularly symmetric functions aligned about z using only the $m=0$ coefficients. Of interest, is the fast rotation property of ZH [SLS05] that allows computing the resulting SH coefficients g_l^m after rotating ZH coefficients f_l^0 to an arbitrary direction $\overline{\omega_j}$ by evaluating and scaling the basis functions Y_l^m in that direction:

$$g_l^m = f_l^0 Y_l^m(\overline{\omega_j}) \sqrt{\frac{4\pi}{(2l+1)}}, \tag{13}$$

We propose to use this property along with precomputation to perform fast on-the-fly rotation.

4.3.2. On-the-fly Rotation

Given ZH coefficients for a canonically oriented Gaussian lobe we rotate them on-the-fly to remove discretization error (fig. 7, center). Furthermore, the storage requirement is reduced drastically from 114 MB to 240 B for order-60 SH, using 32-bit floats.

To perform the fast rotation, we need to evaluate the SH basis functions at a given $\overline{\omega_j}$ direction (eq. (13)). To accelerate this operation, we precompute the value of Legendre polynomials densely for different θ, and we bake in the normalization factors as well (see eq. (4)). For a resolution of 2048 and order-60, the precomputed data requires a storage space of 30 MB. Notice that this texture is common for the entire scene and remains independent of the roughness or any other scene parameter.

4.3.3. Spatially Varying Roughness

A benefit of our approach is that the reduced memory footprint allows us to precompute and store the ZH coefficients for many roughness values, enabling support for spatially varying roughness. In practice, we precompute the ZH coefficients for roughness values from $0-1$ at 1000 uniformly distributed points. At runtime, we linearly interpolate the ZH coefficients from the two nearest σ_s and then perform fast rotation to obtain the desired NDF_{SH}. Storing the ZH values for 1000 σ_s values for order-60 takes roughly 240 KB.

5. Implementation

5.1. Training

We implement neural networks using TinyCudaNN [M2̈1] and PyTorch [PGC*17] for training. All experiments are conducted on an NVIDIA RTX 4090 GPU. Training samples are generated by sampling the footprint center $X \sim U(0,1)$ and a random normalized footprint size $\sigma \sim \lambda e^{-\lambda x}$. Experimentally, we found that setting $\lambda = 16$ works well. We clamp the resulting footprint size to a minimum of 8×8 and train each feature pyramid/decoder independently. We use the Adam [KB14] optimizer with a learning rate of 10^{-3} for both the network weights and learnable feature vectors. We generate 2^{18} samples per iteration and train each network for $20k$ iterations, leaving us with ~ 5 billion training samples generated online. Each feature pyramid/decoder pair takes ~ 10 mins to train, which leaves us with a total training time of ~ 1.5 hours. While the preprocessing time required for our method is higher than Gamboa et al. [GGN18], the training is required only once for each normal map and can be reused for different shapes and under different illumination. Latent code and network weights are stored in half-precision because it does not affect the quality of our generated images.

We train the network to minimize the KL Divergence $D_{KL}(P^*||P)$ between the partial-histogram prediction P^* and ground truth partial-histogram P. The ground truth is obtained efficiently on-the-fly while training uses the SAT proposed by Gamboa et al. [GGN18] with our adaptive binning (section 4.1). Note that this SAT is only needed for training and doesn't contribute to the memory footprint during runtime. One hyper-parameter that needs to be chosen is the width of the feature vector. We found that setting the width equal to the size of the histogram works well in practice. The neural network size has to be chosen according to the feature vector size.

5.2. Rendering Pipeline

Our method is implemented completely on the GPU and uses Mitsuba 3 [JSR*22]. After getting the initial intersection points, we build a G-Buffer consisting of the uv-coordinates and footprint size. We then query our Neural Histogram to predict the histogram at each intersection point. Finally, the histograms are projected to SH while lights SH are calculated using bilinear interpolation in half-vector space. We use a custom CUDA kernel to calculate the double integral product from these coefficients and obtain the final shading. Since we use small MLPs, our method can be implemented in a single mega-kernel with inline MLPs similar to Vaidyanathan et al. [VSW*23].

Like Gamboa et al. [GGN18], we also use Ratio Estimators [HHM18] to add visibility information. This introduces some

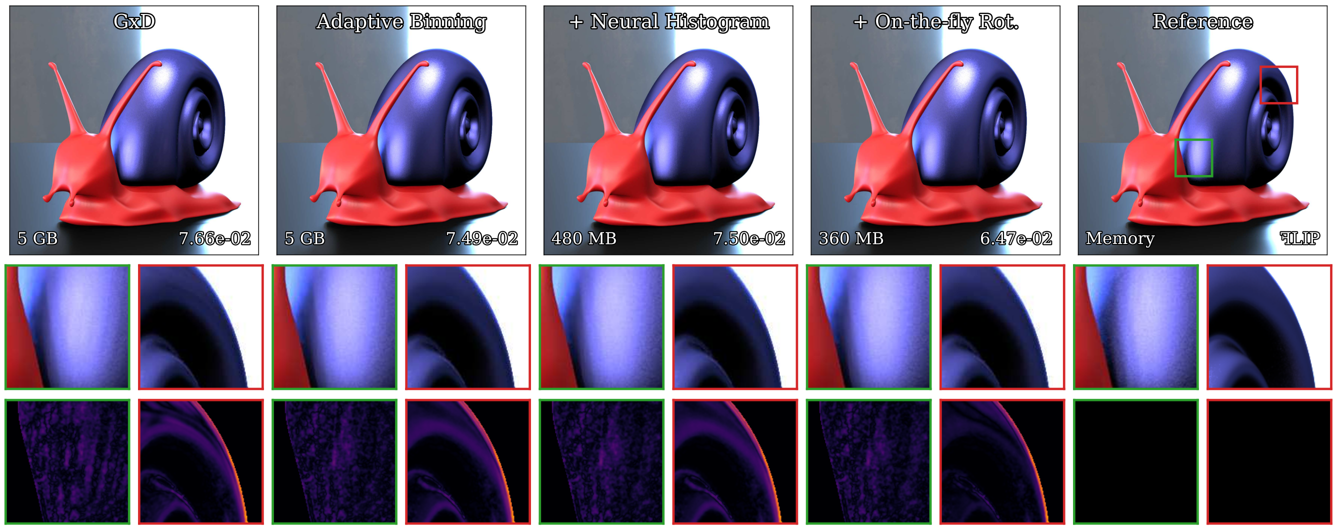


Figure 8: *From left to right: GxD [GGN18], our adaptive binning, neural network representation, on-the-fly rotation, and the $\mathcal{P}$-NDF reference. The reference is calculated using Monte-Carlo with 100k spp. We observe an improvement in both memory usage and rendering quality with our approach. We use ꟻLIP for the difference image.*

variance noise in penumbra regions, to mitigate this, our images are denoised with Optix AI Denoiser. We naively apply temporal denoising for animated sequences by setting the motion vector to zero as we expect small camera movement. More advanced denoising methods are orthogonal to our technique.

Our work analytically integrates complex lighting (e.g., an environment map) reflected on a normal-mapped surface. No directional integration is required for delta lights, such as point or directional lights. These delta lights can be easily handled using a pruning approach [GGN18] or directly evaluating our neural histogram. We use the former approach as its overhead is negligible, i.e., $\sim$ 5-20 milliseconds.

6. Results & Discussion

We compare our technique to the histogram representation and SH computation of Gamboa et al. [GGN18]. All techniques have been implemented on GPU using a 9×32 resolution histogram. The source code of our implementation is publicly available at `https://github.com/ishaanshah/neural_glint`. All references are calculated using Monte-Carlo with high sample counts. Full images of all figures are available in the supplemental material for further inspection. It is important to note that while Gamboa et al. [GGN18] supports rectangular queries, we are restricted to a square footprint, however, this limitation is more apparent at grazing angles or other highly anisotropic footprints, where both are inefficient. Finally, we evaluate the results using the LDR-ꟻLIP metric [ANA*20]. We display our results with Troy James Sobotka's AgX tone mapping function.

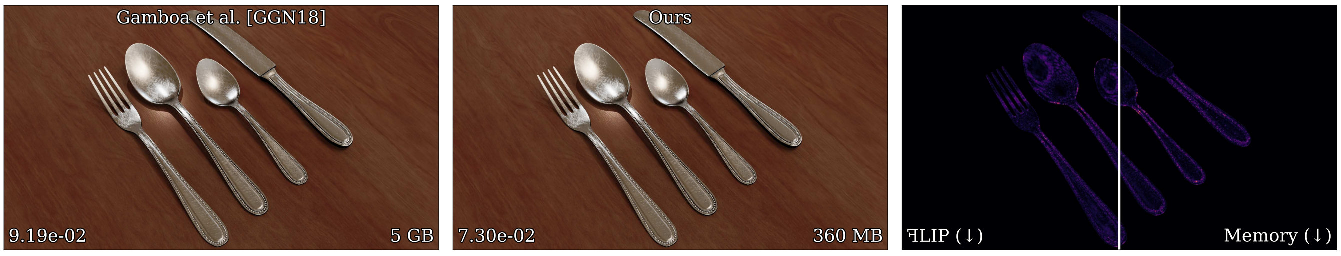
Empirically design study Figure 8 shows an ablation study of our design choices compared to the Gamboa et al. [GGN18] technique. The scene uses the isotropic normal map and a roughness of 0.01. Our adaptive histogram representation and SH computation techniques have improved rendering quality, while our neural network has significantly reduced memory usage. Even though we need to evaluate MLPs to obtain the histogram, this process is still faster than the SAT NDF representation [GGN18]. This can be attributed to GPUs being generally more bandwidth-limited than compute-limited. Table 1 shows the compression ratio achieved compared to prior work except for the light SH coefficients and the runtime comparisons of histogram query and dot product. The light SH coefficients take 342 MB for order 60 SH in both methods. Overall, our technique is faster and one order of magnitude more compact compared to Gamboa et al. [GGN18]'s approach.

Comparison against Gamboa et al. Figure 9 shows additional scenes with different normal maps. All scenes have an environment map and a point light. Animated sequences of each scene can be found in the supplemental video. Each scene contains different normal and environment maps. Our technique consistently produces more accurate results than Gamboa et al. [GGN18] while using less memory. The video demonstrates that our technique does not trade temporal stability for these gains. These results show that our neural network component does not introduce visible artifacts.

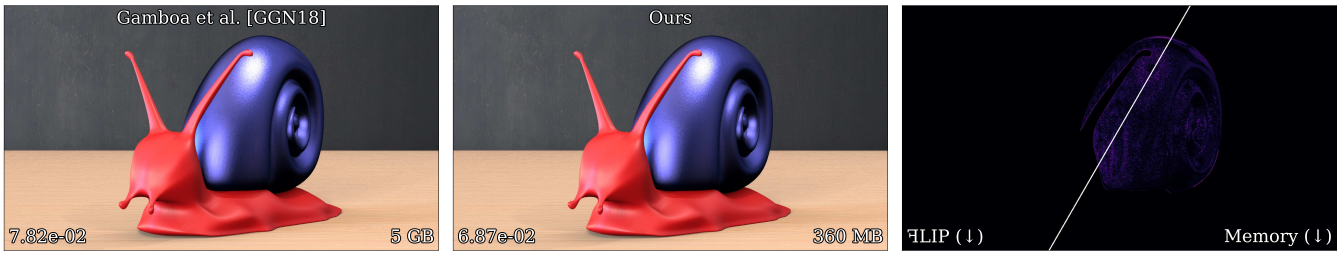
Different histogram resolution Figure 10 shows the impact of a higher-resolution histogram on our technique. Increasing the histogram resolution helps to reduce some discretization errors. Our compact representation enables design choices compared to impractical memory requirements by SAT NDF representations [GGN18], which will require 18.4 GB at higher resolution. Higher histogram resolution impacts the storage of the latent code, which is proportional to the number of histogram bins. We must also increase our decoder network from an MLP of size 64×2 to 256×3 to maximize rendering quality. The runtime increase due to

Table 1: *Comparison of memory usage and runtime of our method with GxD [GGN18], using a* 9×32 *histogram resolution on a* $2K \times 2K$ *normal map with SH-Order of* 60. *Environment map coefficient storage is excluded, as both methods remain the same. The runtimes are calculated assuming the glinty materials fills the entire screen with resolution* 1920×1080.

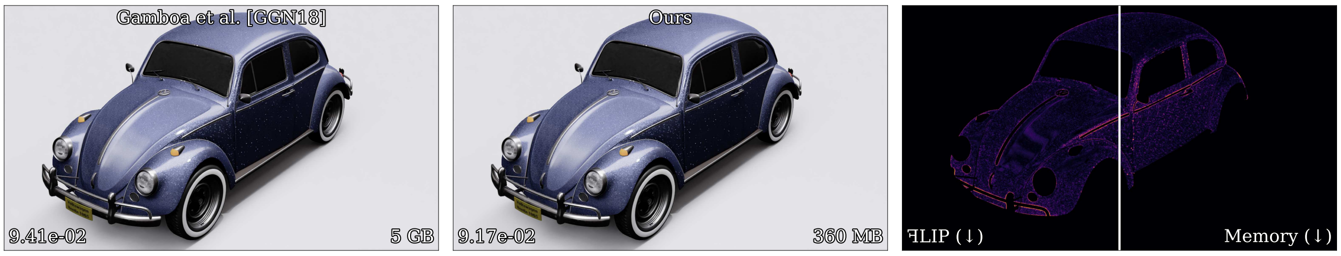
	Memory				Runtime (s)		
	Histogram	BSDF Coeffs	$K_l^m P_l^m(\theta)$	Total	Histogram	SH computation	Total
GxD	4.8 GB	114 MB	–	4.91 GB	0.69	9.15	9.84
Ours	55 MB	240 B	30 MB	85 MB	0.40	7.42	7.82
Ratio / Speedup	$87\times$	$475k\times$	–	$60\times$	$1.7\times$	$1.2\times$	$1.25\times$



(a) *Cutlery (Scratches)*



(b) *Snail (Isotropic)*



(c) *Car (Flakes)*

Figure 9: *Comparison between Gamboa et al. [GGN18] and our method on various complex scenes. The third column shows ꟻLIP error of Gamboa et al. (left) and our method (right) computed against a Monte-Carlo reference with 16k spp. We have a lower ꟻLIP score in all the scenes with a much lower memory footprint. The full reference image is available in supplemental material.*

Table 2: *We isolate rendering cost of our materials (1 spp) shown in fig. 1. Notice how most rendering time is spent on the background objects and the ratio estimator for shadows (1k spp Monte Carlo).*

Object	Normal Map	SH Order	Time (s)	
			GxD	Ours
Spoon	Scratches	60	0.16	0.13
Mug	Flakes	40	0.26	0.29
Handle	Isotropic	40	0.51	0.51
Kettle	Brushed Metal	70	2.04	1.65
Total			2.97	2.58
Ratio Estimator + other objects			3.26	

a bigger network is negligible compared to the 4× increase in SH dot product computation (Table 2).

Spatially varying roughness Figure 1 compares Gamboa et al. [GGN18] (GxD) and our technique with spatially varying roughness. Supporting this is critical for realistic appearances as can be observed by the differences in the kettle object. Our method is more compact and closer to the reference image. Table 2 shows that our technique is slightly more efficient than GxD, where most of our computational budget is spent on dot SH computations and the ratio estimator used to compute the shadow of the environment map. Our supplementary video demonstrates an edition of the surface-varying roughness texture on the kettle object. Editing the roughness texture only requires rendering a new image without any additional computations, thanks to our roughness-independent NDF representation and fast and compact SH projections.

Different light configuration Our method is primarily designed to work with large light sources, such as environmental lighting, to produce noise-free images. Figure 11 shows a comparison between the reference and our technique under various lighting configurations. For each lighting configuration, we produce a single environment map that is then projected onto SH. Using clipped SH expansion [BXH*18] to support area light sources in our technique is left as future work.

7. Limitations & Future Work

Our focus is on rendering of glinty surfaces controlled by a normal map lit under environment lighting. We can analytically project a point light or directional light to spherical harmonics for a holistic approach using closed-form solutions. However, projecting delta lights is impractical due to the necessary order and potential SH banding artifacts. In our current implementation, we choose a more practical solution using the pruning approach proposed in prior work [GGN18]. Delta light poses no integration challenges, as we only need to evaluate the filtered NDFs. However, our approach is based on the assumption of performing analytical light integration using SH. The balance between histogram resolution, roughness and SH order is key to getting the most efficient representation.

Our method focuses on resolving aliasing caused due to shading using a single sample. However, aliasing due to geometric discontinuities is still visible. Such geometric aliasing can be easily mitigated by well known methods such as super sampling (SSAA) [Ake93] the entire image, or supersampling only at the geometric boundaries (MSAA) [SKS11].

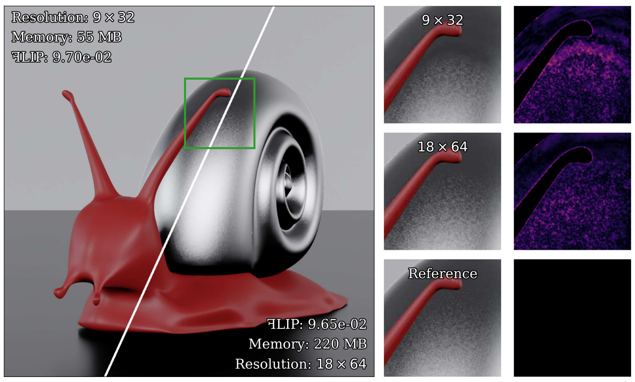


Figure 10: *Comparison of different histogram resolutions. Increasing the resolution by a factor of two has minor local improvements in discretization error at the expense of increased memory usage. Memory figures are only for our neural histogram storage.*

One of our current challenges is that the cost of our neural histogram depends on normal map and $\mathcal{P}$-NDF resolutions. We experimented with differentiable codebook [TMND*23] and differentiable indirection [DMD*23], but found difficulties balancing compression and keeping sharp features of the $\mathcal{P}$-NDF, essential to faithful surface appearance. In section 4.1, we introduced a global parameterization for our adaptive histogram. An interesting idea is to perform this adaptive parametrization on-the-fly based on the pixel footprint $\mathcal{P}$ and local lighting information. Frame prediction methods such as Zeltner et al. [ZRW*23] may inspire, however, it remains unclear how this approach can be scaled up to our binned NDF representation. Another orthogonal solution to reduce our memory consumption is to explore the use of tiling [DLW*22] and generative approach [AWKK21] to produce on-the-fly infinite spatial NDF.

Like all SH-based techniques, we suffer from SH order constraints. We can observe that SH computation is one of the biggest bottlenecks to our technique (table 1). The fast rotation array for the light SH is 342 MB, which is 4 times more than the memory usage for all our other components. As future work, we want to explore alternative bases [XZW*22] to overcome these issues.

Our method uses square footprints relying on a representation of MIP-mapping style. To obtain a better $\mathcal{P}$-NDF, we could use multiple queries, similar to anisotropic filtering, at an added cost. As future work, we want to explore other feature space designs to support anisotropic filtering in a single query, similar to recent work by Deliot et al. [DB23].

8. Conclusion

We presented a novel method to capture complex material appearances arising from spatially varying roughness and high-resolution normal maps. Our solution relies on an adaptive histogram rep-

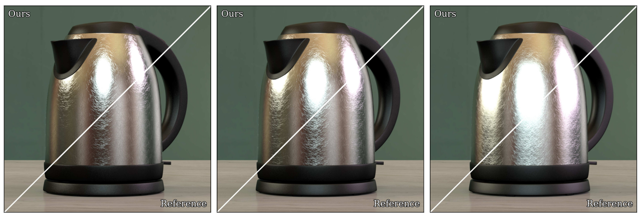


Figure 11: *We show renderings of our method for different-sized light sources. Our method matches closely to the ground truth obtained using MC with 10k spp for various lighting configurations.*

resentation coupled with neural network interpolation to filter $\mathcal{P}$-NDF at different scales. Our representation allows for decoupling of the histogram representation and the spatially varying roughness. Roughness is computed and filtered at run-time, and we perform a fast SH coefficient computation using our new compact ZH representation. Finally, we compute the complex $\mathcal{P}$-NDF - Light integral in a single dot product of SH vectors. We believe our technique pushes the realism of materials and opens a venue for future research.

9. Acknowledgements

We would like to thank Ling-Qi Yan for providing normal maps and the Snail and Cutlery models. All figures use environment maps obtained from PolyHaven and are available under a CC0 license. The Beetle Car was made by Rodrigo Marini and is available at BlenderKit under Royalty Free license. We also thank Aakash K. T. and Rahul Goel for their feedback on the paper. Ishaan Shah was partially funded by KCIS during his Master's.

References

[Ake93] Akeley K.: Reality engine graphics. In *Annual Conference Series (Proceedings of SIGGRAPH)* (New York, NY, USA, 1993), SIGGRAPH '93, Association for Computing Machinery, p. 109–116. doi:10/fqv9pn. 10

[ANA*20] Andersson P., Nilsson J., Akenine-Möller T., Oskarsson M., Åström K., Fairchild M. D.: ꟻLIP: A Difference Evaluator for Alternating Images. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 3*, 2 (2020), 15:1–15:23. doi:10/gnbnp5. 8

[AV07] Arthur D., Vassilvitskii S.: k-means++: the advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (USA, 2007), SODA '07, Society for Industrial and Applied Mathematics, p. 1027–1035. 5

[AWKK21] Atanasov A., Wilkie A., Koylazov V., Křivánek J.: A Multiscale Microfacet Model Based on Inverse Bin Mapping. *Computer Graphics Forum (Proceedings of Eurographics) 40*, 2 (2021), 103–113. doi:10/gsmm6q. 2, 3, 10

[BS63] Beckmann P., Spizzichino A.: *The Scattering of Electromagnetic Waves from Rough Surfaces.* Pergamon Press, NY, 1963. 2

[BXH*18] Belcour L., Xie G., Hery C., Meyer M., Jarosz W., Nowrouzezahrai D.: Integrating clipped spherical harmonics expansions. *ACM Transactions on Graphics 37*, 2 (Mar. 2018), 19:1–19:12. doi:10/gd52pf. 10

[CSDD20] Chermain X., Sauvage B., Dischler J.-M., Dachsbacher C.: Procedural Physically-based BRDF for Real-Time Rendering of Glints. *Comput. Graph. Forum (Proc. Pacific Graphics) 39*, 7 (2020), 243–253. 2

[CT81] Cook R. L., Torrance K. E.: A reflectance model for computer graphics. *Computer Graphics (Proceedings of SIGGRAPH) 15*, 3 (Aug. 1981), 307–316. doi:10/br5ps6. 3

[DB23] Deliot T., Belcour L.: Real-Time Rendering of Glinty Appearances using Distributed Binomial Laws on Anisotropic Grids. *Computer Graphics Forum 42*, 8 (2023), e14866. doi:10/mq8s. 10

[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 (Proceedings of SIGGRAPH Asia) 32*, 6 (2013), 211:1–211:11. doi:10/gbd53f. 2

[DLW*22] Deng H., Liu Y., Wang B., Yang J., Ma L., Holzschuch N., Yan L.-Q.: Constant-Cost Spatio-Angular Prefiltering of Glinty Appearance Using Tensor Decomposition. *ACM Transactions on Graphics 41*, 2 (Jan. 2022), 22:1–22:17. doi:10/mq8t. 2, 5, 10

[DMD*23] Datta S., Marshall C., Dong Z., Li Z., Nowrouzezahrai D.: Efficient Graphics Representation with Differentiable Indirection. In *SIGGRAPH Asia 2023 Conference Papers* (New York, NY, USA, Dec. 2023), Association for Computing Machinery, pp. 1–10. doi:10/mq8z. 2, 10

[FWH*22] Fan J., Wang B., Hasan M., Yang J., Yan L.-Q.: Neural Layered BRDFs. In *ACM SIGGRAPH 2022 Conference Proceedings* (New York, NY, USA, July 2022), Association for Computing Machinery, pp. 1–8. doi:10/gshv3t. 3

[GFL*22] Gauthier A., Faury R., Levallois J., Thonat T., Thiery J.-M., Boubekeur T.: MIPNet: Neural normal-to-anisotropic-roughness MIP mapping. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 41*, 6 (Nov. 2022), 246:1–246:12. doi:10/grqvfc. 3

[GGN18] Gamboa L. E., Guertin J.-P., Nowrouzezahrai D.: Scalable appearance filtering for complex lighting effects. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 37*, 6 (Dec. 2018), 277:1–277:13. doi:10/ggfg4g. 2, 3, 4, 5, 6, 7, 8, 9, 10

[GHS*22] Guerrero P., Hašan M., Sunkavalli K., Měch R., Boubekeur T., Mitra N. J.: MatFormer: A generative model for procedural materials. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 41*, 4 (July 2022), 46:1–46:12. doi:10/gqjn68. 3

[HGC*20] Hu B., Guo J., Chen Y., Li M., Guo Y.: DeepBRDF: A Deep Representation for Manipulating Measured BRDF. *Computer Graphics Forum (Proceedings of Eurographics) 39*, 2 (2020), 157–166. doi:10/gpwj7j. 3

[HGH*23] Hu Y., Guerrero P., Hasan M., Rushmeier H., Deschaintre V.: Generating procedural materials from text or image prompts. In *ACM SIGGRAPH 2023 Conference Proceedings* (New York, NY, USA, 2023), Association for Computing Machinery. doi:10/mrj7. 3

[HHM18] Heitz E., Hill S., McGuire M.: Combining analytic direct illumination and stochastic shadows. In *Proceedings of the Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2018), ACM Press, pp. 2:1–2:11. doi:10/gfznb7. 7

[HP17] Holzschuch N., Pacanowski R.: A two-scale microfacet reflectance model combining reflection and diffraction. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 36*, 4 (July 2017), 66:1–66:12. doi:10/gbxhc7. 2

[Ige99] Igehy H.: Tracing ray differentials. In *Annual Conference Series (Proceedings of SIGGRAPH)* (Aug. 1999), vol. 33, ACM Press, pp. 179–186. doi:10/c2t9t9. 3

[JHY*14] Jakob W., Hašan M., Yan L.-Q., Lawrence J., Ramamoorthi R., Marschner S.: Discrete stochastic microfacet models. *ACM Transactions on Graphics 33*, 4 (July 2014), 1–10. doi:10/f6cnzx. 2, 3

[JSR*22] JAKOB W., SPEIERER S., ROUSSEL N., NIMIER-DAVID M., VICINI D., ZELTNER T., NICOLET B., CRESPO M., LEROY V., ZHANG Z.: Mitsuba 3 renderer, 2022. URL: https://mitsuba-renderer.org. 7

[Kaj86] KAJIYA J. T.: The rendering equation. *Computer Graphics (Proceedings of SIGGRAPH) 20*, 4 (Aug. 1986), 143–150. doi:10/cvf53j. 3

[KB09] KOLDA T. G., BADER B. W.: Tensor decompositions and applications. *SIAM review 51*, 3 (2009), 455–500. doi:10/dzcrv6. 2, 5

[KB14] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. *arXiv:1412.6980 [cs]* (Dec. 2014). arXiv:1412.6980. 7

[KHX*19] KUZNETSOV A., HAŠAN M., XU Z., YAN L.-Q., WALTER B., KALANTARI N. K., MARSCHNER S., RAMAMOORTHI R.: Learning generative models for rendering specular microgeometry. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 38*, 6 (Nov. 2019), 225:1–225:14. doi:10/ggfg2w. 3

[KMX*21] KUZNETSOV A., MULLIA K., XU Z., HAŠAN M., RAMAMOORTHI R.: NeuMIP: multi-resolution neural materials. *ACM Transactions on Graphics 40*, 4 (July 2021), 175:1–175:13. doi:10/gpwj7q. 3, 5

[M2̈1] MÜLLER T.: tiny-cuda-nn, 4 2021. URL: https://github.com/NVlabs/tiny-cuda-nn. 7

[MESK22] MÜLLER T., EVANS A., SCHIED C., KELLER A.: Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics 41*, 4 (July 2022), 102:1–102:15. doi:10/gqkpt7. 2

[OB10] OLANO M., BAKER D.: LEAN mapping. In *Proceedings of the Symposium on Interactive 3D Graphics and Games* (2010), pp. 181–188. doi:10/fkbvpn. 2

[PGC*17] PASZKE A., GROSS S., CHINTALA S., CHANAN G., YANG E., DEVITO Z., LIN Z., DESMAISON A., ANTIGA L., LERER A.: Automatic differentiation in pytorch. 7

[RGB16] RAYMOND B., GUENNEBAUD G., BARLA P.: Multi-scale rendering of scratched materials using a structured SV-BRDF model. *ACM Transactions on Graphics 35*, 4 (July 2016), 57:1–57:11. doi:10/f89j85. 2

[RGJW20] RAINER G., GHOSH A., JAKOB W., WEYRICH T.: Unified Neural Encoding of BTFs. *Computer Graphics Forum (Proceedings of Eurographics) 39*, 2 (2020), 167–178. doi:10/gg792d. 3

[SKS11] SOUSA T., KASYAN N., SCHULZ N.: Secrets of cryengine 3 graphics technology. In *ACM SIGGRAPH* (2011), vol. 1. 10

[SLS05] SLOAN P.-P., LUNA B., SNYDER J.: Local, deformable precomputed radiance transfer. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 24*, 3 (July 2005), 1216–1224. doi:10/bvb8qw. 7

[TMND*23] TAKIKAWA T., MÜLLER T., NIMIER-DAVID M., EVANS A., FIDLER S., JACOBSON A., KELLER A.: Compact neural graphics primitives with learned hash probing. In *SIGGRAPH Asia 2023 Conference Papers* (2023). 10

[VSW*23] VAIDYANATHAN K., SALVI M., WRONSKI B., AKENINE-MOLLER T., EBELIN P., LEFOHN A.: Random-access neural compression of material textures. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 42*, 4 (July 2023), 88:1–88:25. doi:10/gsk4fz. 7

[WDH20] WANG B., DENG H., HOLZSCHUCH N.: Real-Time Glints Rendering With Pre-Filtered Discrete Stochastic Microfacets. *Computer Graphics Forum 39*, 6 (Sept. 2020), 144–154. doi:10/mq9k. 2

[Wil83] WILLIAMS L.: Pyramidal parametrics. *Computer Graphics (Proceedings of SIGGRAPH) 17*, 3 (July 1983), 1–11. doi:10/cq4xrd. 2

[WMLT07] WALTER B., MARSCHNER S. R., LI H., TORRANCE K. E.: Microfacet models for refraction through rough surfaces. In *Rendering Techniques (Proceedings of the Eurographics Symposium on Rendering)* (June 2007), Eurographics Association, pp. 195–206. doi:10/gfz4kg. 2

[XWH*23] XU B., WU L., HASAN M., LUAN F., GEORGIEV I., XU Z., RAMAMOORTHI R.: NeuSample: Importance Sampling for Neural Materials. In *ACM SIGGRAPH 2023 Conference Proceedings* (New York, NY, USA, July 2023), Association for Computing Machinery, pp. 1–10. doi:10/gsk4js. 3

[XZW*22] XU Z., ZENG Z., WU L., WANG L., YAN L.-Q.: Lightweight Neural Basis Functions for All-Frequency Shading. In *SIGGRAPH Asia 2022 Conference Papers* (New York, NY, USA, Nov. 2022), Association for Computing Machinery, pp. 1–9. doi:10/grqvdr. 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 (Proceedings of SIGGRAPH) 33*, 4 (July 2014), 116:1–116:9. doi:10/f6cprr. 2, 3

[YHMR16] YAN L.-Q., HAŠAN M., MARSCHNER S., RAMAMOORTHI R.: Position-normal distributions for efficient rendering of specular microstructure. *ACM Transactions on Graphics 35*, 4 (July 2016), 56:1–56:9. doi:10/f89kqw. 2

[YHW*18] YAN L.-Q., HAŠAN M., WALTER B., MARSCHNER S., RAMAMOORTHI R.: Rendering specular microgeometry with wave optics. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 37*, 4 (July 2018), 75:1–75:10. doi:10/gd52td. 2

[ZK16] ZIRR T., KAPLANYAN A. S.: Real-time rendering of procedural multiscale materials. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, Feb. 2016), I3D '16, Association for Computing Machinery, pp. 139–148. doi:10/gmdkv6. 2

[ZRW*23] ZELTNER T., ROUSSELLE F., WEIDLICH A., CLARBERG P., NOVÁK J., BITTERLI B., EVANS A., DAVIDOVIČ T., KALLWEIT S., LEFOHN A.: Real-Time Neural Appearance Models. 3, 5, 10