DOI: 10.1111/cgf.70318

EUROGRAPHICS 2026 / B. Masia and J. Thies
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 45 (2026), Number 2

Real-Time Neural Materials on Mobile VR

Zilin Xu[1,2], Yang Zhou[2], Yehonathan Litman[2,3], Matt Jen-Yuan Chiang[2], Lingqi Yan[1], Anton Michels[2]

[1]Mohamed bin Zayed University of Artificial Intelligence, UAE
[2]Meta Reality Labs Research, USA
[3]Carnegie Mellon University, USA

Figure 1: *We propose an efficient method for rendering neural materials on low-power mobile VR devices. By combining: 1) a texture-space shading approach with 2) spatiotemporal amortization and 3) a compact coarse-to-fine neural material distilled from a larger model, our method achieves over 90 FPS on a mobile VR device (Meta Quest 3) while maintaining quality comparable to NeuMIP [KMX*21].*

Abstract

*Virtual Reality (VR) applications aim to create an immersive virtual world, which demands a high level of visual realism. The analytical material models commonly used in VR often fall short of reproducing complex real-world appearances. Recently, neural materials have emerged as a promising alternative, offering a compact yet effective representation of real-world materials. Deploying neural materials on low-power mobile VR devices poses significant challenges due to the computational complexity of neural networks and the high display resolution and frame rate requirements of VR devices (commonly 72+ frames per second). We address these challenges by leveraging texture-space shading with spatiotemporal computation amortization, driven by a compact, coarse-to-fine neural material model of extremely low capacity. Thanks to our distillation training scheme, our compact neural materials achieve visual quality comparable to NeuMIP [KMX*21] at a much lower cost. Our method reaches over 90 FPS on a mobile VR device (Meta Quest 3) even under multiple light sources.*

CCS Concepts

• ***Computing methodologies*** → ***Reflectance modeling;***

1. Introduction

Traditional microfacet-based models can produce high-quality appearances, but only for simple materials. For many complex materials found in the real world, the expressiveness of these analytical models becomes insufficient. Measured materials such as Bidirectional Texture Functions [DVGNK99] (BTFs) provide highly accurate replications of real-world appearances, but require massive data storage. Even with compression techniques such as Principal Component Analysis (PCA) or tensor decomposition, they still involve complex decoding and interpolation steps, making them impractical.

In recent years, neural material techniques have emerged as a promising alternative. With powerful non-linear approximation capabilities, they can compress complex materials into a single neural network with higher quality than traditional compression methods. However, neural materials have until now required high-power desktop GPUs with special hardware accelerators such as Tensor Cores and Int8 arithmetic support [XCL*25]. This makes neural materials ill-suited for battery-powered mobile devices, which lack computational power and these specific hardware accelerators. In this paper, we will focus on leveraging unique characteristics of Virtual Reality (VR) applications to efficiently render neural materials on mobile VR devices.

VR applications aim to place the user in highly immersive virtual environments. Their demand for visual realism is in some ways higher than other rendering applications, making neural material techniques a desirable fit thanks to their high-quality appearance and realism. But battery-powered VR devices such as the Meta Quest 3 have similar computational limitations to smartphones and other mobile devices while possessing pixel-dense displays with high render resolutions (4128×2208 for both eyes on Meta Quest 3). The combination of these two makes rendering high-quality materials in real-time a significant challenge.

In addition to the constraints mentioned above, the complexity of neural networks is another major factor that fundamentally challenges the rendering of neural materials on VR devices. Typical neural materials use a Multilayer Perceptron (MLP), e.g., with four layers of size 25×25 [KMX*21] (i.e., 25 hidden layer channels). Based on initial experiments, we need to reduce our MLP to hidden layers of size 8×8, which leads to a significant quality drop. To maintain quality with our low-capacity network, we employ knowledge distillation training and utilize an enhanced input parameterization that incorporates an additional half vector. Our approach achieves a comparable level of quality to a previous work [KMX*21], despite using a significantly smaller network architecture.

However, this smaller network architecture alone is not enough to achieve the framerates necessary for a smooth VR experience due to the high screen resolution and low computational power of VR devices. To address this challenge, we developed a texture-space coarse-to-fine neural material shading design that amortizes computation spatially. To further increase performance, we leverage the fact that the viewport and lighting movement in VR is typically smooth and slow (to prevent motion sickness), making small shading latencies unnoticeable to users. Therefore, we introduce a technique that amortizes computation by reapplying the texture-space shading results across multiple frames. This approach enables the efficient rendering of complex neural materials on mobile VR devices, such as the Meta Quest 3, achieving over 90 FPS (the frame rate cap of the Meta Quest 3) with our network design.

In summary, this paper shows how to render realistic neural materials on a mobile VR device, such as the Meta Quest 3 with three contributions:

- a compact coarse-to-fine neural material with a hidden layer size of 8×8 that can run at over 90 FPS on a mobile VR device (Meta Quest 3),
- a VR-specific texture-space neural material shading technique that overcomes the high screen resolution of VR devices and spatiotemporally amortizes heavy inference cost, and
- a distillation training scheme that keeps our neural material at a comparable level of quality to a previous neural material method [KMX*21].

2. Related Work

2.1. Measured Materials & Neural Materials

Measuring the reflectance of a real-world object is the most effective way to obtain its accurate appearance. Measurement results are typically stored as a 6D Bidirectional Texture Function (BTF) [DVGNK99]. Due to its high dimensionality, data compression is necessary. Principal Component Analysis (PCA) compression [KMBK03, RRK09, GMSK09] and Tensor Decomposition [VT04,RK09] are two popular techniques. They achieve a high compression rate, but suffer from inconvenient interpolation and decoding.

Neural-based methods offer a more promising solution for compact BTF representation with higher quality and smooth interpolation. Rainer et al. [RJGW19] propose the first neural-based method for BTF compression, which they later extended to a unified representation [RGJW20]. NeuMIP [KMX*21] introduces a decoder-only design with a neural texture pyramid to represent materials at different levels of detail (LOD), and also incorporates an offset network to achieve parallax effects. This use of neural textures in combination with a decoder network has become the standard scheme adopted by most follow-up works [KWM*22, RPKLMG23, FWH*23, XZJM24, DZJ*24, ZRW*24, XMWY24, XCL*25]. Apart from compression, some works focus on other aspects of neural materials. For example, Kuznetsov et al. [KWM*22] employ an additional network to handle silhouettes, while NeuSample [XWH*23] adds more efficient importance sampling. Xu et al. [XMWY24] focus on dynamic synthesis, and have recently extended their approach to a comprehensive representation [XCL*25].

Unfortunately, none of these methods are efficient enough to run on mobile VR devices. In this paper, we propose a highly compact coarse-to-fine neural material model that can run in real-time on mobile VR devices.

2.2. Texture-Space Shading

Texture-space shading decouples shading from the forward rendering pass by performing shading directly in texture space and storing the results in a texture, which can then be reused in various ways to accelerate rendering. The core idea can be dated back to the object-based shading in Reyes Rendering Architecture [CCC87] that decouples the shading rate from visibility sampling. Many early works [AHTAM14, RKLC*11, BFM10, CTM13] have studied the use of decoupled sampling to reduce the sampling cost of stochastic effects such as motion blur and depth of field. These works primarily focus on accelerating the rendering of a single frame by reusing shading for multiple visibility samples.

Another branch of work [VMG*25, MVD*18, HSS19] leverages texture-space shading for atlas streaming to reduce the computational cost on client devices. By performing expensive shading computations on a host server and streaming the resulting atlases to clients, the rendering workload on clients can be significantly reduced.

One of the most relevant works for us is Texel Shading [HY16]. It utilizes a compute shader to evaluate shading in texture-space and writes the shaded results to a Result Texture. A final Shade Gathering pass then gathers these results from the Result Texture for rendering. Shading updates can be performed at a lower frequency than the frame rate, trading increased latency artifacts for reduced computational cost. Muller et al. [MNV*21] further explored the idea of adaptively reusing shading across frames for real-time rendering and VR applications. Their analysis shows that shading in dynamic scenes remains largely temporally coherent, even in the presence of complex lighting, materials, and animation, demonstrating the feasibility of reusing shading across frames.

Our method builds upon established texture-space shading techniques and adapts them to facilitate highly efficient neural material evaluation, enabling high-performance neural material rendering even on mobile VR devices.

2.3. Lightweight Neural Network Design & Distillation

While smaller network designs lead to greater efficiency, their limited learning capacity leads to lower-quality predictions. Previous methods propose to distill a larger teacher model into a smaller student model by treating predictions from the teacher model as pseudo-groundtruth signals (hints) for guided training. This approach has been used for a variety of tasks such as distilling image diffusion models [YGZ*24] and video diffusion models [YZZ*25, ZLY*24] to few-shot generative models, teacher-student distillation for appearance relighting [ISS*23], implicit neural representation to explicit model distillation [JWT25], as well as compression for 3D reconstruction [FWW*24, WRH*22]. Building on these techniques, we propose a simple teacher-student distillation procedure for neural materials, resulting in high prediction quality while maintaining high efficiency.

3. Overview & Problem Analysis

Our goal is to deliver the most realistic-looking materials on mobile VR devices (Meta Quest 3) while maintaining real-time frame rates ($\geq$ 72 FPS for VR). In this section, we will analyze the challenges of achieving our goal and give an overview of our solution.

3.1. Neural Representation for Measured Materials (BTF)

A BTF is a 6D function that describes a surface's spatially varying reflectance in texture space. Formally,

$$\mathbf{BTF}(\mathbf{u}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) := \frac{\mathrm{d}L_o(\mathbf{u}, \boldsymbol{\omega}_o)}{L_i(\mathbf{u}, \boldsymbol{\omega}_i)\,\mathrm{d}\boldsymbol{\omega}_i}, \tag{1}$$

where $\mathbf{u}$ is the texture coordinate, $\boldsymbol{\omega}_i$ is the incident direction, $\boldsymbol{\omega}_o$ is the outgoing direction, and $L_i(\mathbf{u}, \boldsymbol{\omega}_i)$ (resp. $L_o(\mathbf{u}, \boldsymbol{\omega}_o)$) is the corresponding incident (resp. outgoing) radiance at the surface point parameterized by $\mathbf{u}$. The foreshortening term is conventionally factored in the BTF function.

Due to its high-dimensionality, the raw data size of a BTF can be massive (e.g., up to 40 GB [WGK14] for a $5cm^2$ patch), which makes it impossible to use directly in rendering applications. Traditional approaches often require extreme compression, such as PCA on the 6D BTF data. However, these methods typically require complex interpolation and produce lossy results, which reduces their practical use in rendering.

Neural methods offer a more efficient way to achieve compact BTF compression and smooth interpolation. The entire 6D table can be trained, in an end-to-end fashion, as a neural model $\mathcal{N}$ with learnable parameters Θ (typically a *multilayer perceptron* (MLP) and a set of neural textures):

$$\mathbf{BTF}(\mathbf{u}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \approx \mathcal{N}(\mathbf{u}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o; \Theta). \tag{2}$$

Although neural materials provide efficient BTF representation, they are too computationally expensive for mobile VR devices. Specifically, the inference for neural materials happens in *shader-level*: each shader thread has to perform an independent inference of the entire network. Hence, a small network design is necessary. However, even reducing each fully connected layer's size to 8×8 is insufficient to meet the minimum frame rate requirements of mobile VR experiences. Low-bit quantization [XCL*25] is unfortunately not an option due to the lack of hardware support on mobile GPUs. Therefore, further performance optimization techniques beyond model simplification are necessary to achieve our goal.

3.2. Overcoming High Resolution and Framerate

The challenge of rendering neural materials on VR devices isn't just the computational limitations of battery-powered hardware, it's the need to render two high-resolution images (one for each eye, 2064×2208 per eye on Quest 3) at a high enough frame rate to prevent motion sickness ($\geq$ 72 FPS). Inferencing a neural material network per pixel for each eye at that resolution and framerate would be challenging even for high-end desktop GPUs. Instead, our technique runs the neural network in texture space, as shown in Fig. 3, inferring two radiance textures, one for each eye in one pass. This radiance texture is then reused across multiple (N) frames to amortize the cost of running the neural network. When we render a frame to the viewport, we simply draw objects with their radiance texture applied, i.e. the UV coordinates are used to lookup and return the value in the radiance texture.

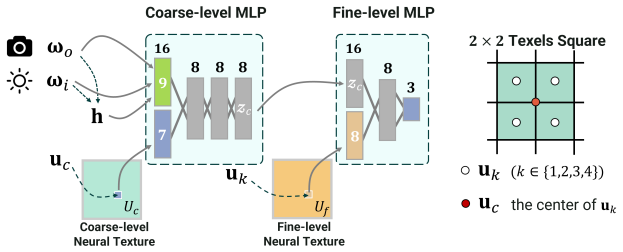


Figure 2: *The structure of our compact coarse-to-fine neural material model. It has two components: a coarse-level MLP and a fine-level MLP. Both of them have a compact design with only an 8-wide hidden layer size. To improve performance, we amortize computation by running the coarse-level MLP once at the center* $\mathbf{u}_c$ *of each* 2×2 *texels square, followed by four times lightweight fine-level MLP inference at individual texels.*

Our texture-space shading strategy relies on several characteristics of VR rendering. First, viewport movements in VR applications are typically small and smooth, as the camera is bound to the user's head. This prevents large angular shifts in view direction $\boldsymbol{\omega}_o$ from frame to frame, which hides the latency between shading updates so that no noticeable perceptual artifacts are introduced to the user. Second, though the shading is updated at a reduced frequency, the scene is still rendered at full framerate ($\geq$ 72 FPS), which is crucial for a comfortable VR experience. This means the shading frequency can be reduced without introducing discomfort or motion sickness, as the user and objects still move smoothly through the world. The reduced shading frequency also does not introduce any ghosting or blurring.

4. Method & Implementation

4.1. Coarse-to-Fine Neural Material

To overcome the constraints of high resolution and limited compute on mobile VR devices, we introduce a highly compact coarse-to-fine neural material model. It utilizes two lightweight MLPs to progressively evaluate reflectance in texture space, from a coarse multi-texel square down to a fine per-texel level. The coarse-to-fine design is motivated by performance considerations. We intentionally kept the size of the fine-level MLP small. By running the coarse-level MLP once followed by multiple lightweight fine-level inferences, we amortize the cost across texels. This results in a significantly lower average inference cost than a full end-to-end MLP per texel. The network architecture is illustrated in Fig. 2.

In the first pass, we train a coarse-level MLP $\mathcal{M}_c$, paired with a neural texture U_c, to predict intermediate latent vectors z_c. Conceptually, each z_c captures the coarse appearance features of a small 2×2 texel patch. This 2×2 square is parameterized by the mean texture coordinate of its texels $\mathbf{u}_k (k \in 1,2,3,4)$: $\mathbf{u}_c = \frac{1}{4}\sum_{k=1}^{4} \mathbf{u}_k$. Given $\mathbf{u}_c$ as well as the angular inputs $\boldsymbol{\omega}_o, \boldsymbol{\omega}_i \in R^3$, we evaluate our coarse-level MLP $\mathcal{M}_c$ as

$$z_c = \mathcal{M}_c(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o, \mathbf{h}, U_c(\mathbf{u}_c)), \tag{3}$$

where we additionally introduce the usage of the half vector $\mathbf{h} \in R^3$ to facilitate network learning.

In the second pass, we predict per-texel appearance with another MLP and neural texture pair, $\mathcal{M}_f$ and U_f. The fine MLP $\mathcal{M}_f$ is *conditioned on* the coarse intermediate latent vector z_c. The final reflectance of each texel $\mathbf{u}_k$ is

$$\mathbf{BTF}(\mathbf{u}_k, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = \mathcal{M}_f(z_c, U_f(\mathbf{u}_k)). \tag{4}$$

Note that the same z_c is shared across all four $\mathbf{u}_k$, thereby amortizing the inference cost of $\mathcal{M}_c$. In this way, $\mathcal{M}_f$ is designed to add details unique to each texel.

It is important to note that the incident and outgoing directions $\boldsymbol{\omega}_i, \boldsymbol{\omega}_o$ are obtained with respect to the mean texture coordinate $\mathbf{u}_c$ rather than individual $\mathbf{u}_k$. The 2×2 texels square shares the same angular condition, which introduces minor angular aliasing across the texels square. Nevertheless, as discussed in Sec. 3.2, the difference produced from the accurate coarse lighting with slightly inaccurate finer details is often negligible for human perception, particularly in VR applications.

Discussion of Neural Offset Module. Neural offset module [KMX*21] can produce parallax effects by estimating the texture space UV offset under different viewing directions without requiring an explicit height field. However, we do not incorporate the neural offset module in our design, as our objective is to model only the 6D measured BTF rather than higher-dimensional effects such as parallax. On the other hand, an accurate height field can be easily obtained along with the measurement process and provided to users as part of the measured dataset (as in UBO2014 [WGK14]). We believe that well-established techniques in modern graphics pipelines can effectively produce high-quality parallax effects when the height field is available. Although this choice prevents us from handling materials with unknown height fields, our primary goal is high-performance BTF rendering on mobile VR devices. The computational overhead of a neural offset module is prohibitively expensive for such platforms. Therefore, we prioritize performance over offset estimation and do not include a neural offset module.

4.2. Texture-space Shading

Our texture-space shading method is based on several observations:

Shading latency is acceptable. At high framerates and with limited camera movement (as is the case with minimal 72 FPS VR experiences), the human eye is insensitive to shading differences between consecutive frames.

Quad overdraw is the bottleneck. Forward shading is the most commonly used rendering technique for mobile and VR devices, but it suffers from quad overdraw, especially for detailed high-poly meshes. By dispatching a compute shader across the object's texture space, we eliminate quad overdraw when running expensive shading calculations.

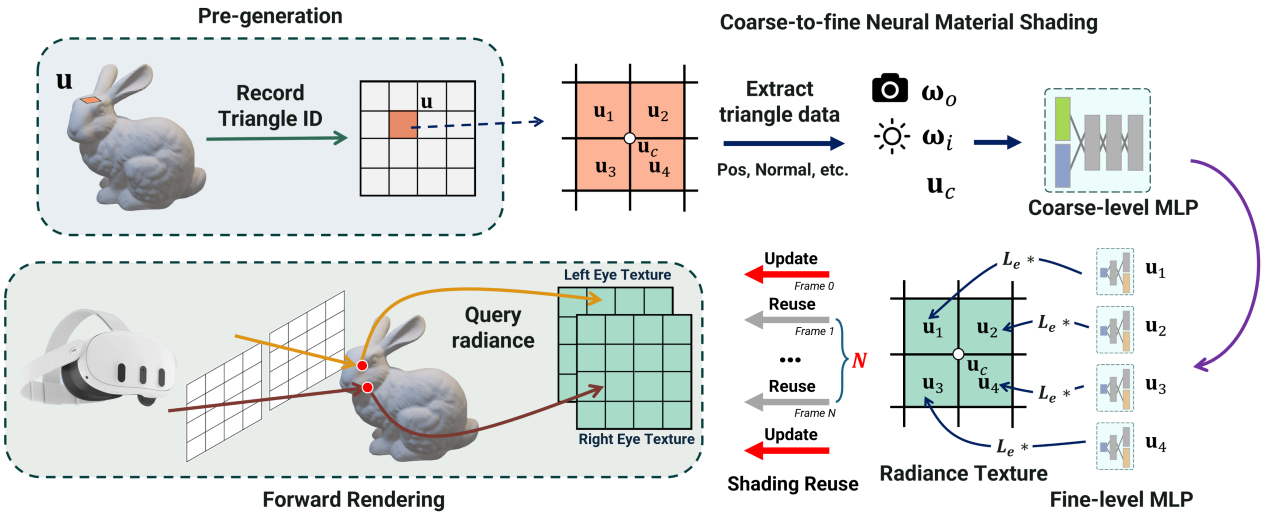


Figure 3: *Method Overview. Our method includes a texture-space shading technique that amortizes computation across frames and a compact coarse-to-fine neural material model. We first pre-generate a texture that stores the Triangle IDs for each texel. At runtime, we extract triangle data using the triangle IDs, then obtain the angular inputs in local space. Each compute thread performs coarse-to-fine neural material shading for a* 2×2 *texel square and records the shading output into a radiance texture. The radiance texture can be reused across frames. Once updated, the next N frames (by default* $N = 4$*) will reuse the same radiance textures for better performance. During forward rendering, the final shading results can be directly queried from the radiance texture.*

Lighting varies smoothly. Lighting across an object typically does not change hugely from pixel to pixel (or texel to texel in our case). For human perception, accurate detailed lighting can be approximated by accurate coarse lighting and synthesized inaccurate details. Because shading is calculated in compute shaders, our technique is not bound to a 1:1 ratio of threads per texel. It can run a single thread for 2×2 texel regions and share computation across texels.

With the aforementioned observations, we design a texture space neural material shading technique as shown in Fig. 3. Our texture-space shading technique first pre-generates a texture specifying the triangle ID at each texel. A compute shader is dispatched over the whole texture (eliminating quad overdraw) and earlies-out on back-face regions of the mesh. It samples the triangle ID using its Dispatch ID as texture coordinates and manually fetches and interpolates the triangle data.

For neural material shading, we employ a coarse-to-fine scheme as detailed in Sec. 4.1. The coarse-to-fine neural material shading produces an object-space radiance texture that is reused for the next N frames (i.e., every $N+1$ frames share the same radiance texture), amortizing the shading computation and significantly increasing performance. During the forward rendering, the final shading results can be directly queried from the radiance texture.

We recommend setting $N = 4$ for balanced performance and latency, but this should also be determined by the actual situation, as high-specular materials will be more sensitive to shading latency. Nevertheless, our internal testing revealed that users were unable to notice latency artifacts on high-end PBR materials, even though we reused the radiance texture for six frames ($N = 6$). In this case, the shading was only updated at a rate of around 10 FPS.

Although our texture-space shading approach builds upon existing methods, our key contribution lies in leveraging the characteristics of VR applications together with the advantages of texture-space shading to improve the performance of neural material rendering.

4.3. Quality Enhancement via Distillation

Previous neural material models typically have hidden layer sizes of 32×32 [XMWY24] or 25×25 [KMX*21]. However, to meet the performance requirements of VR applications, our network is designed with a much smaller hidden layer size of 8×8. Under such constraints, directly learning the BTF results in significant quality loss. To improve quality, we designed a knowledge distillation training scheme to better utilize the limited network capacity. The new training scheme proved to be very effective, allowing our small network to achieve quality on par with that of NeuMIP [KMX*21], a much larger network.

Our distillation scheme aims to transfer the learning of a pre-trained teacher MLP of much larger capacity to a student MLP that is used for final shading. During the training of the student MLP, it learns not only from ground truth data but also from the teacher MLP. The goal is to encourage the student MLP to match the output statistics of the teacher MLP layer by layer. This is straightforward for the final layer, as both MLPs outputs RGB reflectance. However, the latent vectors from hidden layers have different dimensions and are not directly comparable. Given two latent vectors z_s

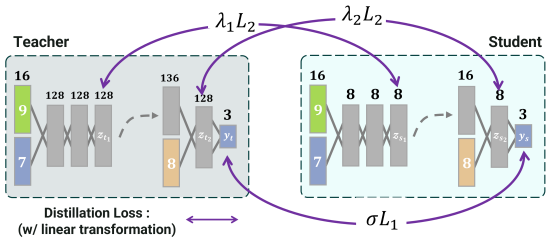


Figure 4: *Our distillation scheme. We use two intermediate feature vectors and the output from the teacher model as hints for the student network. To match the size of the feature vectors from the teacher model and the student model, we apply a learnable linear transformation to the features.*

and z_t with different dimensions D_s and D_t, we generalize the comparison by introducing a linear transform $\mathcal{T}$ of dimension $D_s \times D_t$. The latent vectors are considered similar if

$$z_s \approx \mathcal{T}(z_t).$$

The intuition is that the two vectors capture equivalent underlying latent (presumably nonlinear) features if there exists a linear transform that converts them into each other. $\mathcal{T}$ is jointly optimized along with the student MLP during the distillation.

Specifically, we opt to match the output and two hidden latent vectors between the teacher and the student MLPs. A schematic diagram is provided in Fig. 4. The overall training loss is

$$\begin{aligned} L = {} & L_1(y_s, y_{\text{gt}}) + \sigma L_1(y_s, y_t) + \\ & \lambda_1 L_2(z_{s_1}, \mathcal{T}_1(z_{t_1})) + \lambda_2 L_2(z_{s_2}, \mathcal{T}_2(z_{t_2})), \end{aligned} \tag{5}$$

where y_s, y_t, and y_{gt} is the predicted reflectance by the student MLP, the teacher MLP, and the ground truth reflectance, respectively. z_{s_*} and z_{t_*} are the latent vectors from the corresponding layers of both MLPs, and $\mathcal{T}_*$ is the linear transform that brings them into the same space. In practice, the teacher MLP is 128-wide and the student MLP is 8-wide, giving $\mathcal{T}_*$ a size of 128×8. σ, λ_1, λ_2 are the weights for each loss term, which are all set to 0.1 in our experiments.

4.4. Training Details

We use PyTorch [PGM*19] for training. We apply the Adam [KB14] optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1 \times 10^{-8}$. We decay the learning rate from 5×10^{-4} to 1×10^{-7} using the `ReduceLROnPlateau` scheduler. It will halve (decay factor of 0.5) the learning rate when the decreased testing L_1 error is smaller than a threshold (1×10^{-4}) in each epoch. For each training step, we take a 2D slice (400×400) from the BTF dataset under the same viewing and lighting angles as one batch. We apply stratified sampling to the neural textures to avoid the discrete artifact.

We first trained the teacher model around 60 epochs, then used it to train the student model from scratch. The distillation training lasts for at most 150 epochs. On a desktop with an RTX 4080 GPU, the whole training including the teacher training takes about 19 hours.

5. Results

We select seven measured BTF examples from UBO2014 [WGK14]. Each BTF exhibits a representative appearance that highlights different aspects of complex material behavior. We first validate our texture-space shading method, then we compare against previous methods by using a modified NeuMIP [KMX*21] as baseline. We use image space ꟻLIP error [ANA*20], peak signal-to-noise ratio (PSNR), and structural similarity index measure (SSIM) as error metrics. By default, all the results are rendered under *three point lights* with four frames between updates ($N = 4$) and 1200^2 texture-space resolution per eye. Please note that some results are displayed with a skybox background for clarity purposes, which is *not* used as a light source. As a common optimization for low-power neural network inference, we use half precision for our inference. We use the Open 3D Engine (O3DE, `https://o3de.org/`) as a framework in which to run our experiments, both while iterating on Desktop and while benchmarking on Meta Quest 3. All performance data is measured on a Meta Quest 3.

Obtaining Reference Images. The measured BTF data is a 6D discrete table requiring 6D interpolation to render reference images. Since the full dataset is huge (about 40 GB), and our rendering runs inside a game engine (O3DE), obtaining aligned reference images is difficult. Therefore, we adopted an uncommon way to generate reference images, based on the fact that previous methods [KMX*21,XMWY24,XCL*25,XZJM24] have extensively shown that neural materials closely approximate references with minimal error. We trained a modified NeuMIP [KMX*21] with 128 hidden layer size (NeuMIP Max) as a compromised reference to replace the direct 6D BTF table interpolation, enabling the generation of accurate and aligned reference images within O3DE. To demonstrate that the compromised NeuMIP Max reference can faithfully reproduce the accurate appearances, and to compare each method with the real reference, we generate rendered images for each material using an offline rendering process in Fig. 17.

5.1. Validation of Texture-Space Shading

We first validate the performance gain from texture-space shading and spatiotemporal amortization. As shown in Fig. 5, ours can only run at 9 FPS in screen space but achieves 80 FPS with a much lower resolution in texture space with spatiotemporal amortization. Note that texture-space shading is a prerequisite for enabling spatiotemporal amortization, which is not feasible in screen space.

Moreover, in Fig. 6, we compared the rendering results at various texture resolutions. At the lowest resolution (800^2 per eye), some noticeable discrete artifacts appear. As the resolution increased, these discrete artifacts became negligible, and the results closely matched the screen-space shading.

Discussion of validating spatiotemporal amortization. Spatial and temporal amortization in our method are tightly coupled with

Figure 5: *Validation of the performance gains of our texture space shading and spatiotemporal amortization. Rendering neural materials directly in high-resolution screen space is prohibitively expensive for mobile VR headsets. With texture-space shading and spatiotemporal amortization, we achieved 80 FPS, which is an* 8.9× *speedup compared to the screen space performance.*

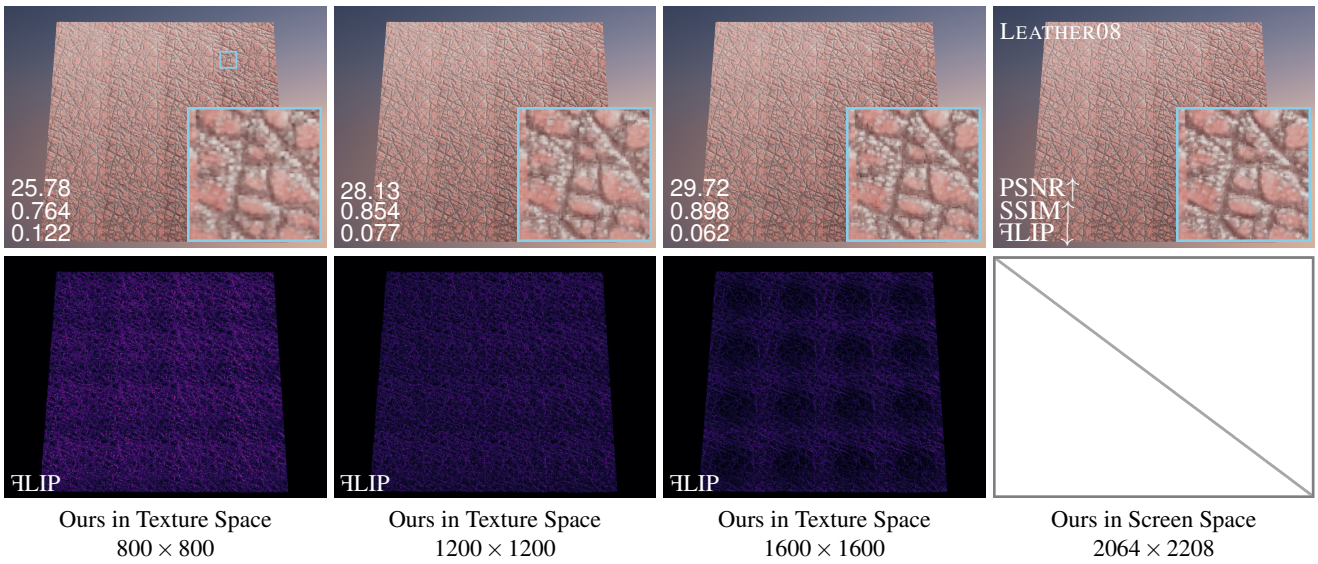


Figure 6: *We compare the results of three different texture resolutions against screen-space shading. The second row shows the ꟻLIP error map against the screen-space result (Ours in Screen Space). For a low resolution of* 800 × 800 *pre-eye, noticeable discrete artifacts can be observed. As the resolution increased, these discrete artifacts became negligible, and the results closely matched the screen-space shading.*

texture-space shading. When both are disabled (w/o spatial and w/o temporal), our method reduces to standard screen-space shading. Enabling spatial amortization without temporal amortization (w/ spatial and w/o temporal) only results in a performance drop, with no quality loss. We cannot independently evaluate the setting with temporal amortization but without spatial amortization (w/o spatial and w/ temporal), as screen-space shading does not support temporal amortization.

5.2. Comparison with Modified NeuMIP

To demonstrate the accuracy of our method, we compare it with a modified NeuMIP [KMX*21] as a baseline (hereafter "NeuMIP" all refers to the modified NeuMIP for simplicity). We made two changes: 1) we removed the feature pyramid structure and always use its finest neural texture; 2) we excluded the offset module.

The reason for 1) is that the feature pyramid is designed to support levels of detail (LoDs). However, in our current setting, we do not consider LoDs (as discussed in Sec. 6). As a result, the feature

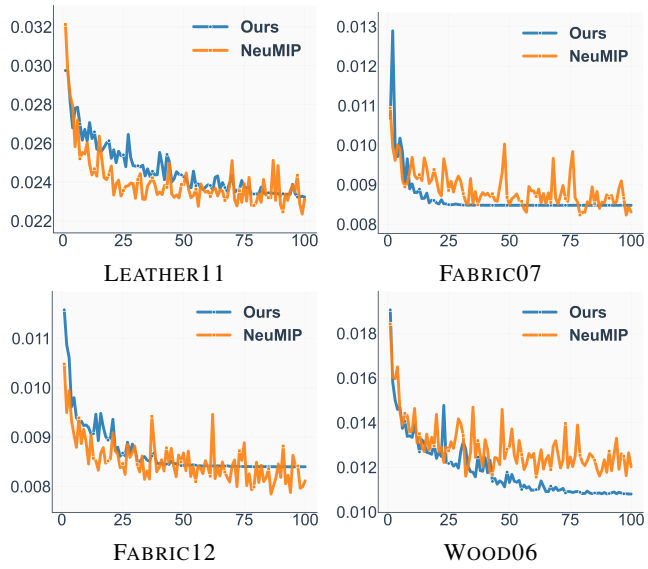


Figure 7: *We compare our method with NeuMIP [KMX*21] in terms of test loss over training epochs. The test loss is measured as the L_1 error across the whole BTF dataset. Our loss is on the same level as NeuMIP.*

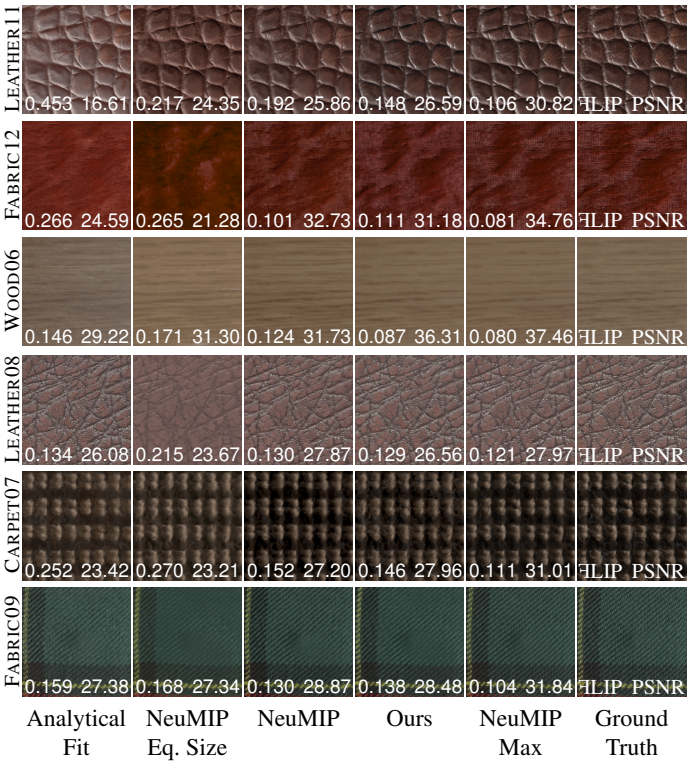


Figure 8: *We compare ours with analytical BTF fitting (with four Lafortune lobes [LFTG97]), NeuMIP [KMX*21], and its variant with the same capacity as our model by directly visualizing the 2D BTF slices. Overall, the analytical fitting approach exhibits larger deviations. Despite NeuMIP using a much larger network than ours, our results exhibit more accurate highlights and self-shadows. When we reduce NeuMIP's network size to match ours, it fails to preserve the appearance of the BTFs.*

pyramid is redundant for comparison, and removing it does not lead to any loss in quality.

As for 2), the offset module in NeuMIP is designed to *estimate* parallax effects for materials with an *unknown* height field. By contrast, when the height field is explicitly available, more accurate parallax effects can be achieved independently using traditional parallax mapping techniques at a significantly lower computational cost. In our case, UBO2014 [WGK14] does provide a height field for each BTF. Therefore, we believe that more accurate parallax effects, or even displacement, can be easily achieved using well-established techniques within a modern graphics pipeline. Since our method models only the 6D BTF, it is more appropriate to compare with NeuMIP without its offset module.

In Fig. 7, we show the L_1 test loss curves on the whole dataset during training for both our method and NeuMIP. Thanks to our distillation training and a more effective training strategy, we achieve a loss on the same level as NeuMIP with an MLP that is 7.4× smaller. On the Wood06 dataset, we even surpass NeuMIP.

Furthermore, we directly visualize the recovered 2D BTF slices in Fig. 8. We also include another NeuMIP variant with the same size as ours (8×8 latent layer size). Ours give more accurate highlights and self-shadows.

In Fig. 16, we show a rendering comparison with NeuMIP on three different materials. It is important to note that our goal is not to surpass previous methods in quality, but to achieve the best possible quality under low computational power constraints while running at 72+ FPS on Meta Quest 3.

5.3. Comparison with Analytical BTF Fitting

BTFs can be effectively approximated as the sum of a diffuse term and multiple anisotropic specular lobes. One of the most classic analytical approaches for BTF fitting represents the specular component using a mixture of k Lafortune lobes [LFTG97] $s(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o; \mathbf{M}, n)$:

$$\mathbf{BTF}(\mathbf{u}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \approx \rho_{d,\mathbf{u}} + \sum_{j=1}^{k} \rho_{s,\mathbf{u},j} \cdot s_{\mathbf{u},j}(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o; \mathbf{M}, n), \quad (6)$$

where ρ_d and ρ_s are diffuse and specular color respectfully. The Lafortune lobe is defined as:

$$s(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o; \mathbf{M}, n) = (\boldsymbol{\omega}_o^t \cdot \mathbf{M} \cdot \boldsymbol{\omega}_i)^n, \quad (7)$$

where $\mathbf{M}$ is a 3×3 matrix that controls the shape, orientation, and anisotropy of the specular lobe, and n is the specular exponent that determines the sharpness of the lobe.

We employ a nonlinear Levenberg–Marquardt optimization to fit one diffuse color and $k = 4$ Lafortune lobes (with independent specular colors) to each 4D texel (i.e., apparent BRDF [WHON97,

	Analytical Fit	NeuMIP	Ours
LEATHER11	0.04606	0.02285	0.02311
FABRIC12	0.01914	0.00814	0.00840
WOOD06	0.02930	0.01264	0.01078
LEATHER08	0.04185	0.02811	0.02904
FABRIC07	0.02589	0.00851	0.00847
CARPET07	0.01570	0.01472	0.01668
FABRIC09	0.01082	0.01108	0.00939
Total Storage	35.2 MB	4.48 MB	6.24 MB

Table 1: *Overall L_1 error of each method on each material, together with the corresponding storage. The best result is highlighted. Both NeuMIP [KMX*21] and our method consistently outperform the analytical fitting approach using four Lafortune [LFTG97] lobes. For NeuMIP and our method, the reported total storage includes neural texture parameters as well as network parameters (6.6 KB and 1.6 KB, respectively).*

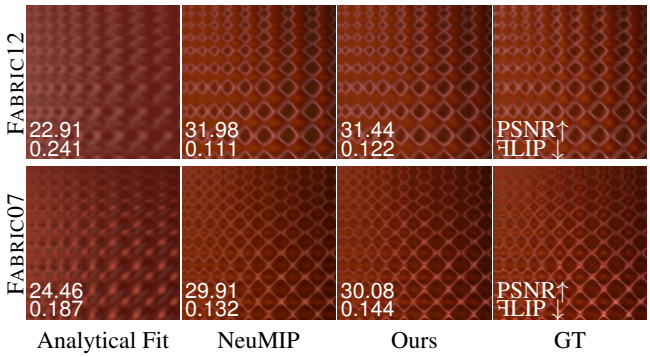


Figure 9: *Visualization of the recovered apparent BRDFs [WHON97, MMS*05] for each method. The 4D apparent BRDF is unfolded into a 2D image for visualization, as commonly done in prior works. With only four Lafortune lobes [LFTG97], the strong sheen effects of fabric materials cannot be captured.*

MMS*05]) in the BTF. The overall L_1 error is summarized in Table 1. To further evaluate the fitting quality in comparison with our method, we visualize the reconstructed apparent BRDFs in Fig. 9 and the 2D BTF slices in Fig. 8. Moreover, we also show the offline rendering results in Fig. 17.

5.4. Performance

We benchmark our method on a Meta Quest 3 device and record the frame rate on a desktop connected via Android Debug Bridge (ADB). Due to hardware constraints of the Meta Quest 3, the frame rate is capped at 90 FPS. We use the same scene as in Fig. 5 for performance testing unless otherwise stated.

Performance under different amortization settings. We summarize our method's rendering performance under different amortization settings in Table 2. We only apply one point light to better benchmark the performance. There are multiple performance tradeoffs: lowering texture resolution will increase the occurrence of discrete artifacts, while increasing intervals between shading updates introduces more latency. We chose 1200^2 texture resolution per eye and four frames between updates as a balanced choice between quality and performance.

Update Interval	Texture Space Resolution 800^2	1200^2	1600^2	2000^2	Screen Space 2064×2208
0 Frames	80 FPS	42 FPS	36 FPS	27 FPS	23 FPS
1 Frame	90 FPS	67 FPS	57 FPS	43 FPS	
2 Frames	90 FPS	82 FPS	71 FPS	58 FPS	
4 Frames	90 FPS	90 FPS	82 FPS	76 FPS	
6 Frames	90 FPS	90 FPS	89 FPS	81 FPS	

Table 2: *Performance of our method when rendering with one point light under different amortization configurations on a Meta Quest 3. The row with "0" frame update interval indicates that temporal amortization is disabled.*

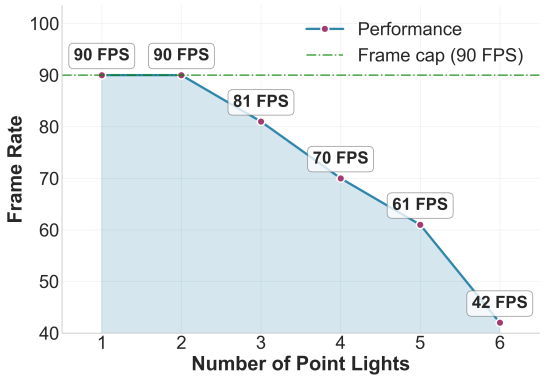


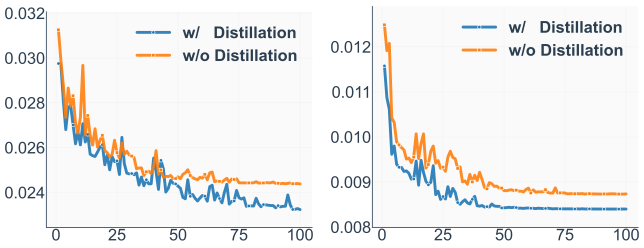
Figure 10: *The performance under different numbers of point lights with four frames between updates and 1200^2 texture resolution. Our performance scales approximately linearly according to the number of point lights.*

Performance under different numbers of point lights. Point lights are the most commonly used light sources in real-time rendering applications. Each point light requires a full inference pass through the neural material model. Therefore, the number of supported point lights directly reflects the practicality of the neural material approach. Under a setting of 1200^2 texture resolution per eye and four frames between updates, in Fig. 10, we conducted a performance test to evaluate the number of point lights our method can support in an actual rendering application.

Performance of rendering multiple materials simultaneously. We evaluate the scalability of our method when rendering multiple materials simultaneously. We place different numbers of square meshes in the scene and assign a distinct material to each square. Table 3 summarizes the rendering performance when simultaneously rendering different numbers of materials. The performance

Num. of Lights	3			
Num. of Materials	1	2	3	4
Frame Rate	81 FPS	62 FPS	51 FPS	39 FPS

Table 3: *Performance of rendering multiple materials simultaneously with three point lights. Each material uses a 1024^2 texture resolution, and the update interval is set to seven. The rendering results will be written to a 32-bit radiance texture, so for each material, the GPU RAM cost for the radiance texture is 8.0 MB.*



LEATHER11 FABRIC12

Figure 11: *We select two representative BTF from UBO2014 [WGK14] to validate the effectiveness of our distillation. For each material, we show the test loss of the whole 6D BTF data during training. With distillation, the convergence of the test loss is significantly better.*

scales approximately linearly with the number of materials (or equivalently, the total shading texture resolution).

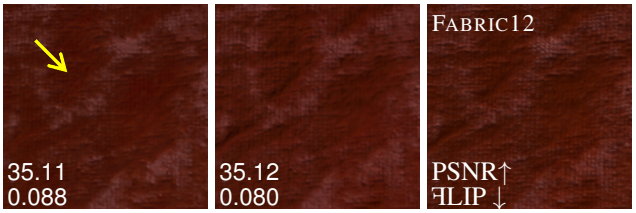
5.5. Informal User Study of the Update Latency

In the supplementary video, we present video recordings of our rendering demo. The latency caused by the shading reuse is hard to notice in the video. To better investigate the potential problems of our method, we conducted an internal user study. We set up three objects with neural materials and texture-space shading in a VR environment and had users put on the headsets and ask questions about potential issues. Some mentioned harsh shadows from the point lights, but nobody mentioned the lag of the lighting on the texture. Even when reusing shading for the next six frames ($N = 6$), the latency was not immediately noticeable. Only when prompted to look for the latency and provided with a low-latency comparison were users able to notice the difference.

5.6. Ablation Studies

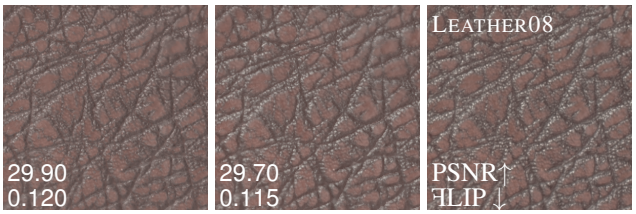
We conduct ablation studies to validate the effectiveness of our two key designs: the distillation training and the coarse-to-fine neural material architecture.

Distillation. We first evaluate the effectiveness of distillation. In Fig 11, we compare the test loss of models trained with and without distillation. The use of distillation significantly improved the convergence. Additionally, we compare the recovered 2D BTF slices in Fig 12. With distillation, the results exhibit finer details and more accurate self-shadows.



Ours w/o distillation | Ours w/ distillation | Ground Truth

Figure 12: *To better validate the effectiveness of the distillation, we visualize the predicted 2D BTF slices of each model along with the ground truth. Distillation gives more accurate results and self-shadows.*



Ours Single Level | Ours Coarse-to-Fine | Ground Truth

Figure 13: *Validation of our coarse-to-fine neural model design. Our coarse-to-fine design does not introduce quality loss compared with the single-level variant with the same capacity.*

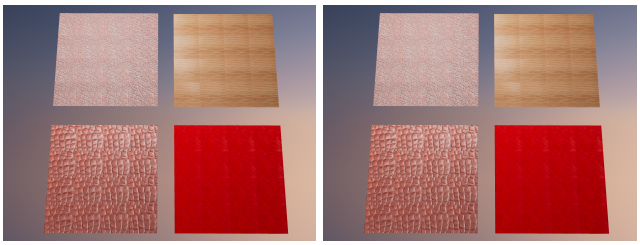
Coarse-to-Fine Network Design. Compared to typical end-to-end network architectures (4 latent layers with 8×8 size), our coarse-to-fine network design does not introduce quality loss as shown in Fig. 13. Instead, it enables the computation to be spatially amortized, improving efficiency without sacrificing quality.

6. Limitation & Discussion

Texture Resolution & Level of Details. Our neural material evaluation currently does not offer a robust Level of Details (LoD) mechanism. In close-up scenarios such as Fig. 14, materials must be rendered at very high texture resolutions to maintain detailed appearance, which hinders performance. However, it should be noted that our amortized texture-space shading can still shade much more texels than screen space shading. As alternatives to LoD techniques or high-resolution textures, tiling or by-example synthesis [XMWY24, XCL*25] could be employed to enhance detail.

Fine-grained Details. Due to the limited capacity of our model, even with distillation training, it may still struggle to capture extremely fine details. A failure case is provided in Fig. 15.

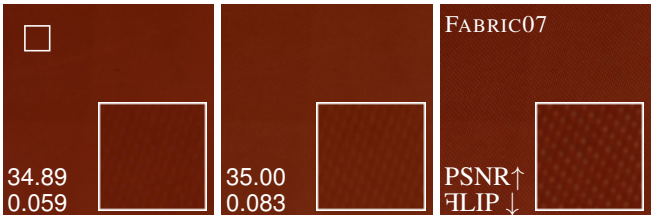
Tiling Issues. While we do not explicitly handle tiling issues, the neural textures with an MLP decoder design of our model suggest that it can be compatible with the previous dynamic neural BTF synthesis approach [XMWY24].



15 FPS
Screen Space
2064×2208
(not covered all pixels)

14 FPS
Texture Space
4×2400^2

Figure 14: *The performance of our technique drops with very large texture sizes. Still, our amortized texture-space shading can output roughly 5× more pixels/texels than screen-space shading under the same camera condition.*



Ours NeuMIP Ground Truth

Figure 15: *A failure case. Our MLP can not handle very fine-grained details due to its extremely low capacity. Nevertheless, even for NeuMIP [KMX*21] that has a much larger network size, such fine details remain difficult to be captured.*

7. Conclusion & Future Work

We have presented an efficient solution for rendering neural materials on low-power mobile VR devices. By combining a texture-space shading approach with spatiotemporal amortization and a compact coarse-to-fine neural model, our method achieves over 90 FPS on Meta Quest 3 while maintaining high visual fidelity. Thanks to distillation training, our lightweight network delivers quality comparable to that of NeuMIP [KMX*21], making neural materials practical for real-time VR rendering for the first time.

For future work, we believe that designing efficient image-based lighting (IBL) solutions for neural materials is an important research direction. Current real-time shading applications heavily rely on the efficient split-sum approximation [Kar13]. However, the split-sum approximation only supports analytical BRDFs. Neural materials still require a large number of samples to achieve IBL, which is prohibitively expensive on mobile devices. While our work tackles the challenge of efficient point-wise evaluation of neural materials, it remains an open problem how to extend the current neural material paradigm to support efficient integration over the angular domain.

Another promising direction for future work is to leverage the stereoscopic nature of VR rendering by sharing a single radiance texture between both eyes. Although the two eyes have slightly different viewing directions, they are highly correlated. Our current method performs inference independently per eye and does not exploit this relationship. A shared one-pass inference for both eyes could further improve performance.

References

[AHTAM14] ANDERSSON M., HASSELGREN J., TOTH R., AKENINE-MÖILER T.: Adaptive texture space shading for stochastic rendering. In *Computer Graphics Forum* (2014), vol. 33, Wiley Online Library, pp. 341–350. 3

[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:10.1145/3406183. 6

[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), pp. 19–28. 3

[CCC87] COOK R. L., CARPENTER L., CATMULL E.: The reyes image rendering architecture. *ACM SIGGRAPH Computer Graphics 21*, 4 (1987), 95–102. 3

[CTM13] CLARBERG P., TOTH R., MUNKBERG J.: A sort-based deferred shading architecture for decoupled sampling. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 1–10. 3

[DVGNK99] DANA K. J., VAN GINNEKEN B., NAYAR S. K., KOENDERINK J. J.: Reflectance and texture of real-world surfaces. *ACM Transactions On Graphics (TOG) 18*, 1 (1999), 1–34. 2

[DZJ*24] DOU Y., ZHENG Z., JIN Q., NI B., CHEN Y., KE J.: Real-time neural brdf with spherically distributed primitives. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024), pp. 4337–4346. 2

[FWH*23] FAN J., WANG B., HAŠAN M., YANG J., YAN L.-Q.: Neural biplane representation for btf rendering and acquisition. In *Proceedings of SIGGRAPH 2023* (2023). 2

[FWW*24] FAN Z., WANG K., WEN K., ZHU Z., XU D., WANG Z.: Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. In *NeurIPS* (2024). 3

[GMSK09] GUTHE M., MÜLLER G., SCHNEIDER M., KLEIN R.: Btf-cielab: A perceptual difference measure for quality assessment and compression of btfs. In *Computer graphics forum* (2009), vol. 28, Wiley Online Library, pp. 101–113. 2

[HSS19] HLADKY J., SEIDEL H.-P., STEINBERGER M.: Tessellated shading streaming. In *Computer Graphics Forum* (2019), vol. 38, Wiley Online Library, pp. 171–182. 3

[HY16] HILLESLAND K. E., YANG J.: Texel shading. In *Eurographics (Short Papers)* (2016), pp. 73–76. 3

[ISS*23] IWASE S., SAITO S., SIMON T., LOMBARDI S., BAGAUTDINOV T., JOSHI R., PRADA F., SHIRATORI T., SHEIKH Y., SARAGIH J.: Relightablehands: Efficient neural relighting of articulated hand models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), pp. 16663–16673. 3

[JWT25] JIANG Z., WANG S., TANG S.: Dnf-avatar: Distilling neural fields for real-time animatable avatar relighting. In *ICCV* (2025). 3

[Kar13] KARIS B.: Real shading in unreal engine 4. *Proc. Physically Based Shading Theory Practice 4*, 3 (2013), 1. 11

[KB14] KINGMA D. P., BA J.: Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980* (2014). 6

[KMBK03] KOUDELKA M. L., MAGDA S., BELHUMEUR P. N., KRIEGMAN D. J.: Acquisition, compression, and synthesis of bidirectional texture functions. In *3rd International Workshop on Texture Analysis and Synthesis (Texture 2003)* (2003), vol. 59, p. 64. 2

Figure 16: *Comparison between NeuMIP [KMX*21], ours, and the compromised NeuMIP Max reference. Due to NeuMIP's large network architecture, it failed to run in real time on Meta Quest 3. Note that our goal is not to surpass previous methods in quality, but to achieve the best possible quality under low computational power constraints. Some small black artifacts appear in the results of* FABRIC07*, which, based on our experiments, originate from the O3DE rendering pipeline rather than our method.*

[KMX*21] KUZNETSOV A., MULLIA K., XU Z., HAŠAN M., RAMAMOORTHI R.: Neumip: Multi-resolution neural materials. *ACM Trans. Graph. 40*, 4 (jul 2021). URL: https://doi.org/10.1145/3450626.3459795, doi:10.1145/3450626.3459795. 1, 2, 4, 5, 6, 7, 8, 9, 11, 12

[KWM*22] KUZNETSOV A., WANG X., MULLIA K., LUAN F., XU Z., HASAN M., RAMAMOORTHI R.: Rendering neural materials on curved surfaces. In *ACM SIGGRAPH 2022 Conference Proceedings* (New York, NY, USA, 2022), SIGGRAPH '22, Association for Computing Machinery. URL: https://doi.org/10.1145/3528233.3530721, doi:10.1145/3528233.3530721. 2

[LFTG97] LAFORTUNE E. P., FOO S.-C., TORRANCE K. E., GREENBERG D. P.: Non-linear approximation of reflectance functions. In *Proc. SIGGRAPH '97* (1997), vol. 31, pp. 117–126. doi:10.1145/258734.258801. 8, 9

[MMS*05] MÜLLER G., MESETH J., SATTLER M., SARLETTE R., KLEIN R.: Acquisition, synthesis, and rendering of bidirectional texture functions. In *Computer Graphics Forum* (2005), vol. 24, Wiley Online Library, pp. 83–109. 8, 9

[MNV*21] MUELLER J. H., NEFF T., VOGLREITER P., STEINBERGER M., SCHMALSTIEG D.: Temporally adaptive shading reuse for real-time rendering and virtual reality. *ACM Trans. Graph. 40*, 2 (2021). URL: https://doi.org/10.1145/3446790, doi:10.1145/3446790. 3

[MVD*18] MUELLER J. H., VOGLREITER P., DOKTER M., NEFF T., MAKAR M., STEINBERGER M., SCHMALSTIEG D.: Shading atlas streaming. *ACM Transactions on Graphics (TOG) 37*, 6 (2018), 1–16. 3

[PGM*19] PASZKE A., GROSS S., MASSA F., LERER A., BRADBURY J., CHANAN G., KILLEEN T., LIN Z., GIMELSHEIN N., ANTIGA L., DESMAISON A., KOPF A., YANG E., DEVITO Z., RAISON M., TEJANI A., CHILAMKURTHY S., STEINER B., FANG L., BAI J., CHINTALA S.: Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf. 6

[RGJW20] RAINER G., GHOSH A., JAKOB W., WEYRICH T.: Unified neural encoding of BTFs. *Computer Graphics Forum (Proc. Eurographics) 39*, 2 (July 2020), 167–178. doi:10.1111/cgf.13921. 2

[RJGW19] RAINER G., JAKOB W., GHOSH A., WEYRICH T.: Neural btf compression and interpolation. *Computer Graphics Forum (Proceedings of Eurographics) 38*, 2 (Mar. 2019). 2

[RK09] RUITERS R., KLEIN R.: Btf compression via sparse tensor decomposition. *Computer Graphics Forum 28*, 4 (2009), 1181–1188. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2009.01495.x, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2009.01495.x, doi:https://doi.org/10.1111/j.1467-8659.2009.01495.x. 2

[RKLC*11] RAGAN-KELLEY J., LEHTINEN J., CHEN J., DOGGETT M., DURAND F.: Decoupled sampling for graphics pipelines. *ACM Transactions on Graphics (TOG) 30*, 3 (2011), 1–17. 3

[RPKLMG23] RODRIGUEZ-PARDO C., KAZATZIS K., LOPEZ-MORENO J., GARCES E.: Neubtf: Neural fields for btf encoding and transfer. *Computers & Graphics 114* (2023), 239–246. 2

[RRK09] RUITERS R., RUMP M., KLEIN R.: Parallelized matrix factorization for fast btf compression. In *EGPGV@ Eurographics* (2009), pp. 25–32. 2

[VMG*25] VINING N., MAJERCIK Z., GU F., TAKIKAWA T., TRUSTY T., LALONDE P., MCGUIRE M., SHEFFER A.: Fastatlas: Real-time compact atlases for texture space shading. In *Computer Graphics Forum* (2025), Wiley Online Library, p. e70010. 3

[VT04] VASILESCU M. A. O., TERZOPOULOS D.: Tensortextures: Multilinear image-based rendering. In *ACM SIGGRAPH 2004 Papers*. 2004, pp. 336–342. 2

[WGK14] WEINMANN M., GALL J., KLEIN R.: Material classification based on training data synthesized using a btf database. In *Computer Vision – ECCV 2014* (Cham, 2014), Fleet D., Pajdla T., Schiele B., Tuytelaars T., (Eds.), Springer International Publishing, pp. 156–171. 3, 4, 6, 8, 10

[WHON97] WONG T.-T., HENG P.-A., OR S.-H., NG W.-Y.: Image-based rendering with controllable illumination. In *Rendering Techniques '97* (Vienna, 1997), Dorsey J., Slusallek P., (Eds.), Springer Vienna, pp. 13–22. 8, 9

[WRH*22] WANG H., REN J., HUANG Z., OLSZEWSKI K., CHAI M., FU Y., TULYAKOV S.: R2l: Distilling neural radiance field to neural light field for efficient novel view synthesis. In *ECCV* (2022). 3

[XCL*25] XU Z., CHEN X., LIU C., WANG B., WANG L., MONTAZERI Z., YAN L.-Q.: Towards comprehensive neural materials: Dynamic structure-preserving synthesis with accurate silhouette at instant inference speed. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers* (New York, NY, USA, 2025), SIGGRAPH Conference Papers '25, Association for Computing Machinery. URL: https://doi.org/10.1145/3721238.3730626, doi:10.1145/3721238.3730626. 2, 3, 6, 10

[XMWY24] XU Z., MONTAZERI Z., WANG B., YAN L.-Q.: A dynamic by-example btf synthesis scheme. In *SIGGRAPH Asia 2024 Conference Papers* (New York, NY, USA, 2024), SA '24, Association for Computing Machinery. URL: https://doi.org/10.1145/3680528.3687578, doi:10.1145/3680528.3687578. 2, 5, 6, 10

[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, 2023), SIGGRAPH '23, Association for Computing Machinery. URL: https://doi.org/10.1145/3588432.3591524, doi:10.1145/3588432.3591524. 2

[XZJM24] XUE B., ZHAO S., JENSEN H. W., MONTAZERI Z.: A hierarchical architecture for neural materials. In *Computer Graphics Forum* (2024), vol. 43, Wiley Online Library, p. e15116. 2, 6

[YGZ*24] YIN T., GHARBI M., ZHANG R., SHECHTMAN E., DURAND F., FREEMAN W. T., PARK T.: One-step diffusion with distribution matching distillation. In *CVPR* (2024). 3

[YZZ*25] YIN T., ZHANG Q., ZHANG R., FREEMAN W. T., DURAND F., SHECHTMAN E., HUANG X.: From slow bidirectional to fast autoregressive video diffusion models. 3

[ZLY*24] ZHAI Y., LIN K., YANG Z., LI L., WANG J., LIN C.-C., DOERMANN D., YUAN J., WANG L.: Motion consistency model: Accelerating video diffusion with disentangled motion-appearance distillation. In *NeurIPS* (2024). 3

[ZRW*24] 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. *ACM Transactions on Graphics 43*, 3 (June 2024), 1–17. URL: https://dl.acm.org/doi/10.1145/3659577, doi:10.1145/3659577. 2

Appendix A: Comparison with the Real 6D BTF Reference

Figure 17: *We compare each method against the original 6D BTF rendering results (Reference). Since accurate 6D BTF interpolation is not affordable for real-time rendering, we generate the reference using an offline Monte Carlo path tracing process. We reuse the same Monte Carlo samples for all methods to eliminate differences caused by random noise. We provide three error metrics (PSNR↑, SSIM↑, and ꟻLIP ↓), and visualize the ꟻLIP error map to better evaluate each method's result. The compromised NeuMIP Max reference produces results that are very close to the real reference. Analytical fitting with four Lafortune lobes struggles to capture the complex appearance of the measured materials. Ours shows a similar level of quality compared to NeuMIP.*