Constant-Cost Spatio-Angular Prefiltering of Glinty Appearance Using Tensor Decomposition

HONG DENG, School of Computer Science and Engineering, Nanjing University of Science and Technology, China
YANG LIU, School of Computer Science and Engineering, Nanjing University of Science and Technology, China
BEIBEI WANG*, School of Computer Science and Engineering, Nanjing University of Science and Technology, China
JIAN YANG, School of Computer Science and Engineering, Nanjing University of Science and Technology, China
LEI MA, National Engineering Laboratory for Video Technology, Peking University, China
NICOLAS HOLZSCHUCH, University Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK, France
LINGQI YAN, University of California, Santa Barbara, USA

Ours, 5.16 min	Yan et al. [2016], 8.7 h

Fig. 1. Comparison between our method and Yan et al. [2016] on the Car scene, using a seamlessly tileable normal map of metallic flakes with resolution $2K \times 2K$ under geometric optics. The rendered glints are almost identical, but our method costs less than 1% of time compared to Yan et al. [2016], and does not scale with the view distance.

The detailed glinty appearance from complex surface microstructures enhances the level of realism, but is both space- and time-consuming to render, especially when viewed from far away (large spatial coverage) and/or illuminated by area lights (large angular coverage). In this paper, we formulate the glinty appearance rendering process as a spatio-angular range query problem of the Normal Distribution Functions (NDFs), and introduce an efficient spatio-angular prefiltering solution to it. We start by exhaustively precomputing all possible NDFs with differently sized positional coverages. Then we compress the precomputed data using tensor rank decomposition, which enables accurate and fast angular range queries. With our spatio-angular prefiltering scheme, we are able to solve both the storage and performance issues at the same time, leading to efficient rendering of glinty appearance with both constant storage and constant performance, regardless of the range of spatio-angular queries. Finally, we demonstrate that our method easily applies to practical rendering applications that were traditionally considered difficult. For example, efficient bidirectional reflection distribution function (BRDF) evaluation accurate NDF importance sampling, fast global illumination between glinty objects, high-frequency preserving rendering with environment lighting, and tile-based synthesis of glinty appearance.

*Corresponding author.
Authors' addresses: Hong Deng, School of Computer Science and Engineering, Nanjing University of Science and Technology, 200 Xiaolingwei Rd, Nanjing, 210094, China; Yang Liu, School of Computer Science and Engineering, Nanjing University of Science and Technology, 200 Xiaolingwei Rd, Nanjing, 210094, China; Beibei Wang*, School of Computer Science and Engineering, Nanjing University of Science and Technology, 200 Xiaolingwei Rd, Nanjing, 210094, China, beibei.wang@njust.edu.cn; Jian Yang, School of Computer Science and Engineering, Nanjing University of Science and Technology, 200 Xiaolingwei Rd, Nanjing, 210094, China, csjyang@njust.edu.cn; Lei Ma, National Engineering Laboratory for Video Technology, Peking University, Beijing, China, lei.ma@pku.edu.cn; Nicolas Holzschuch, University Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK, 655, avenue de l'Europe, Grenoble, France, Nicolas.Holzschuch@inria.fr; Lingqi Yan, University of California, Santa Barbara, 2119 Harold Frank Hall, Santa Barbara, CA, 93106, USA, lingqi@cs.ucsb.edu.

1 INTRODUCTION

Many materials in the real life have microstructures, such as the tiny metallic flakes in car paints, and scratches and brushes on heavily used cutleries. These microstructures have their own way of interaction with the light, and when aggregated, change the overall visual appearance of the materials. Microstructure rendering [Yan et al. 2014] studies the modeling and rendering of such details, and has increased the realism to rendering individual specular highlights (i.e. glints), a core effect in computer graphics. But in order to represent, store and render the complex appearance from details, the storage is usually prohibitively costly and the performance is often too slow, even for offline productions such as animations.

The fundamental difficulty is that in order to bring out the complex light transport from microstructures, one has to define all the details. For general appearance, it is common practice to use high-resolution normal maps or heightfields to explicitly define every microfacet. For example, Yan et al. [2018] define heightfields with resolutions up to one micron per texel. The size of such high-resolution details is large – $10K \times 10K$ texels just to cover one square centimeter, and it grows with the area of surfaces.

However, an even bigger issue is the computation. During rendering, it is the NDF of the microstructures that determines specific appearance. However, the more microstructures are being considered, the longer it takes to resolve the interactions between the light and the microstructures. This imposes strong difficulties on the rendering performance. When the viewpoint is far away from the objects, the surface region covered by one pixel, a.k.a. pixel footprint, will include more microstructures inside. Also, when large area lights exist, multiple NDF queries will be needed for different samples on the lights. Noise will emerge, and will consume a lot of samples to diminish, because the microstructures contribute to different parts of the light quite differently. Recent works (Gamboa et al. [2018]) have solved this issue with Summed Area Table (SAT) or Integral Histogram (IH), at cost of expensive storage or limited microfacet models (Beckmann only).

From these difficulties, we can see that the performance of rendering glinty appearance heavily depends on the spatial and angular coverages of NDF queries. Therefore, at the core of rendering glinty appearance from microstructures is an efficient method that performs *spatio-angular prefiltering* of NDFs. That is, with some proper treatments before rendering, we should be able to quickly find the NDF value from microfacets within an arbitrary pixel footprint towards an arbitrary range of solid angles. Preferably, the time it takes to answer such questions should be independent of the query sizes, both spatially and angularly, known as constant performance. Even better, if the prefiltering can be combined with dynamic synthesis methods of the microstructure, we will be able to generate microstructures in an infinite positional range, while keeping the prefiltering only on the given finite sized microstructures. This gives us constant storage.

In this paper, we present a precomputation-based prefiltering approach that answers spatio-angular range queries of NDFs, achieving both constant performance and constant storage, a.k.a. constant cost. We start from an exhaustive precomputation of all NDFs at all possible centers and sizes of pixel footprints, to cover all possible NDF queries during rendering. We organize the precomputed data into a 3D "NDF tensor", and use tensor rank decomposition to compress it. Then we propose a novel decompression scheme that answers arbitrarily-sized spatio-angular queries on the original NDF tensor at constant performance.

Our prefiltering scheme leads to efficient rendering of glinty appearance. We demonstrate that our method elegantly applies to practical rendering operations/applications that were traditionally considered difficult: accurate NDF importance sampling, fast global illumination between glinty objects, high-frequency preserving rendering with environment lighting. And we complete our constant-cost goal by combining our prefiltering with tile-based synthesis

methods that achieves constant storage of rendering a large range of microstructures.

Compared with previous glinty appearance rendering methods, our method is the first that does not have to tradeoff between quality and cost. In practice, the performance of our method is also constant w.r.t. view distances and light sizes, and is 1-2 orders of magnitude faster than traditional approaches. And the storage cost of our method is constant to the granularity of the scene, taking up only several megabytes (e.g. for $1K \times 1K$ patches) to hundreds of megabytes (e.g. for $8K \times 8K$ patches) after compression.

To summarize, our contributions include:

(1) a NDF precomputation-based approach with compact compression and decompression using tensor decomposition, which enables fast evaluation of BRDF with large footprint,
(2) a large range of applications to several difficult rendering problems: accurate NDF importance sampling, multiple-bounce glints rendering and prefiltering, and
(3) a combination with tile-based texture synthesis methods, resulting in constant storage for glints rendering.

2 RELATED WORK

In this section, we first briefly review previous work on microstructure rendering and synthesis, and then introduce related work on precomputation and prefiltering.

Microstructure rendering. Yan et al. [2014] proposed to simulate the spatially and directionally varying appearance using patch-local normal distribution functions ($\mathcal{P}$-NDFs), which is later accelerated by Yan et al. [2016] via a position-normal distribution method. Recently, Yan et al. [2018] managed to wave optics effects. As analyzed in Sec. 1, all these methods share a common problem with performance that grows with the size of the pixel footprint.

More generally, Zeltner et al. [2020] exploit the specular manifold to efficiently find specular paths in rendering complex appearance. Wang et al. [2020b] introduce the idea of path cuts to find all specular paths of arbitrary lengths. These methods aim at efficient specific-purpose light transport, possibly with microstructure, but do not focus on appearance models. Our method makes no assumption to light transport methods. It automatically fits the Multiple Importance Sampling (MIS) framework, and naturally handles multiple bounces of light between complex surfaces.

Microstructure synthesis. Considerable efforts have been made to relieve the storage issue in microstructure rendering. And they can be subscribed into two categories. The first category is the procedural creation of the microstructure, realizing the actual distributions of microstructures on the fly from a list of predefined rules [Chermain et al. 2021; Jakob et al. 2014; Raymond et al. 2016; Velinov et al. 2018; Wang et al. 2020a, 2018; Werner et al. 2017; Zirr and Kaplanyan 2016]. These methods work well for specific effects, such as discrete glitters and scratches, but do not support general appearances. The other category of microstructure synthesis is by-example [Wang et al. 2020c; Zhu et al. 2019], using texture synthesis [Cohen et al. 2003; Heitz and Neyret 2018] or using a Generative Adversarial Network to generate NDF images [Kuznetsov et al. 2019]. Our method combines with the by-example synthesis idea, bringing out the constant storage property.

Precomputation followed by efficient compression has been widely adopted in computer graphics. Precomputed Radiance Transfer (PRT) precomputes complex light transport and compresses them in order to allow real-time rendering with complex lighting [Ng et al. 2004; Sloan et al. 2003, 2002; Tsai and Shih 2006; Wang et al. 2009; Xu et al. 2013]. Precomputation also takes place pervasively in appearance modeling and rendering. Yan et al. [2017] use precomputed 2D profiles to describe the scattering behavior in different types of animal fur fibers. Donner et al. [2009] exhaustively precompute one BSSRDF for every different combination of scattering parameters, and use confocal ellipses to fit the resulting data, leading to an efficient empirical BSSRDF model. Wang et al. [2020] propose to represent the multiple scattering in participating media with a precomputed table, which is later improved to neural networks [2019] for further compression. Rainer et al. [2020; 2019] uses a neural network to compress measured 6D Bidirectional Texture Functions (BTFs) to relieve the heavy data storage. We analyze the pros and cons of precomputation in Sec. 3, and exploit precomputation to achieve constant performance.

Prefiltering can be treated as a specific type of precomputation on the appearance, in order to improve run-time performance. For example, to enable efficient level of detail rendering of cloth, Wu et al. [2019] prefilter the heightfield and Zhao et al. [2016] prefilter anisotropic participating media. Prefiltering has also been used in real-time rendering with environment maps [Karis 2013] and to account for the curvature of surfaces [Kaplanyan et al. 2016].

Specifically, prefiltering has been applied to glinty appearance rendering. Belcour et al. [2017] prefilter color-, normal-, and displacement-mapped appearance in the context of multi-bounce global illumination with footprints predicted by covariance tracing. Gamboa et al. [2018] precompute all possible NDFs as histograms for arbitrary spatial range queries at constant performance. However, the resolution of the NDFs is too low (9×32) to preserve the glinty appearance (e.g. metallic flakes) angularly. It focuses on fast performance during rendering, but still suffers from prohibitively expensive storage cost in practice. Concurrent work [Atanasov et al. 2021] proposes a normal map filtering approach via inverse binning mapping, throwing microstructures into corresponding directional bins as a preprocess, taking advantage of the fact that directional resolution can be fixed regardless of the sizes of pixel footprints. However, this work is limited to Beckmann function.

3 PRELIMINARIES AND MOTIVATION

In this section, we first briefly provide the fundamentals of rendering complex appearance. Then we analyze the related storage and performance issues of existing methods to motivate our approach. For clarity, we list the symbols used throughout our paper in Table 1.

3.1 Preliminaries

Traditionally, people use the microfacet model [Cook and Torrance 1982; Walter et al. 2007] to describe the BRDF at a point $\mathbf{x}$ on a surface:

$$f_r(\mathbf{i}, \mathbf{o}) = \frac{F(\mathbf{i} \cdot \mathbf{h})\, G(\mathbf{i} \cdot \mathbf{h})\, D(\mathbf{i}, \mathbf{o})}{4\, (\mathbf{i} \cdot \mathbf{n})\, (\mathbf{o} \cdot \mathbf{n})}, \tag{1}$$

where $\mathbf{i}$, $\mathbf{o}$ are the incident and outgoing directions and $\mathbf{n}$ is the normal of the macro surface. F is the Fresnel term giving the total

Table 1. Symbols used in the paper.

symbol	definition
$\mathbf{i}, \mathbf{o}$	incident and outgoing directions
$\mathbf{h}$	half vector
$f_r(\mathbf{i}, \mathbf{o})$	BRDF
$F(\mathbf{i} \cdot \mathbf{h})$	Fresnel term
$G(\mathbf{i} \cdot \mathbf{h})$	Shadowing-Masking function
$D(\mathbf{i}, \mathbf{o})$	normal distribution function
$D_{\mathcal{P}}(\mathbf{x}, \mathbf{h})$	$\mathcal{P}$-NDF
$D^*(\mathbf{x}, \overline{\psi})$	wave optics GNDF
$G(\mathbf{x}; \mu, \sigma)$	Gaussian function
G_c	coherence area
s	sample stride on normal map
t	image block size of a NDF image
L	count of non-empty NDF image blocks in a cluster
R	rank of compression
X_r, Y_r	tensor vectors with size $R \times t$
Z_r	tensor vectors with size $R \times L$
C	tensor vectors with size R

amount of reflection, G is the shadowing-masking term considering self-occlusions from microfacets, and most importantly, D is the NDF term, statistically defining the distribution of microfacets' normals at/around $\mathbf{x}$. Yan et al. [2014] and subsequent work replace the statistical distribution D with an actual distribution of normals $D_{\mathcal{P}}$ (the $\mathcal{P}$-NDF) in an area around $\mathbf{x}$, namely the pixel footprint $\mathcal{P}$ seen through each pixel. This leads to a $\mathcal{P}$-NDF defined as

$$D_{\mathcal{P}}(\mathbf{x}, \mathbf{h}) = \int_{\mathbb{R}^2} G_p(\mathbf{u}; \mathbf{x}, \sigma_p)\, G_r(\mathbf{n}(\mathbf{u}); \mathbf{h}, \sigma_r)\, d\mathbf{u}, \tag{2}$$

where $G(\mathbf{x}; \mu, \sigma)$ is a 2D Gaussian function centered at μ with a standard deviation of σ. $G_p(\cdot; \cdot, \sigma_p)$ and $G_r(\cdot; \cdot, \sigma_r)$ determine the size of the pixel footprint and the "intrinsic roughness" of each microfacet, respectively. $\mathbf{h} = \frac{\mathbf{i}+\mathbf{o}}{|\mathbf{i}+\mathbf{o}|}$ is known as the half vector.

In this way, the actual distributions bring out the variation of appearance at different pixels, resulting in interesting glinting effects, especially when the camera or the light moves.

The wave optics variation of the NDF [Kuznetsov et al. 2019] takes diffraction into account, producing colored appearance. Strictly speaking, the concept of NDFs do not exist in wave optics. Commonly adopted wave optics theories (Harvey-Shack or Kirchhoff) directly predicts the BRDF using Fourier transform of the spatially-varying phase shift of light induced by a heightfield [Yan et al. 2018], in a coherence area *of fixed size* (typically $5 - 10$ microns as the standard deviation of a 2D Gaussian) where the light interferes in a non-linear fashion.

For convenience, Kuznetsov et al. [2019] define a "wave optics GNDF (generalized NDF)", formally written as

$$D^*(\mathbf{x}, \overline{\psi}) = \frac{4}{A_c \lambda^2} \left| \int_{\mathbb{R}^2} G_c(\mathbf{s}; \mathbf{u}, \sigma_c) e^{-i\frac{4\pi}{\lambda}\left[h(\mathbf{s})+\overline{\psi}\cdot\mathbf{s}\right]} d\mathbf{s} \right|^2, \tag{3}$$

where a heightfield h is required instead of normals, G_c is the coherence area, $\overline{\psi}$ is the first two components of the sum $\frac{\mathbf{i}+\mathbf{o}}{2}$, and A_c is a
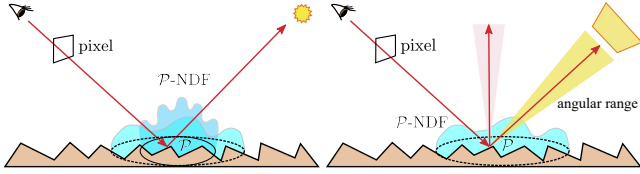
Fig. 2. Motivation of spatio-angular filtering of glinty appearance. Left: larger spatial footprint results in smoother NDF, but takes longer to compute. Right: larger angular range results in smoother NDF, but previously solved with heavy sampling rather than simple range query.

normalization factor. We refer readers to Kuznetsov et al. [2019] for more details.

When the pixel footprint is greater than the coherence area, which is almost true at all times even when viewed from closeup, we are able to quickly integrate over the pixel footprint to get the (linearly) aggregated GNDFs as

$$D_{\mathcal{P}}^*(\mathbf{x}, \overline{\psi}) = \frac{\int_{\mathbb{R}^2} G_p(\mathbf{x}; \mathbf{u}, \sigma_p) \, D^*(\mathbf{u}, \overline{\psi}) \, d\mathbf{u}}{\int_{\mathbb{R}^2} G_p(\mathbf{x}; \mathbf{u}, \sigma_p) \, d\mathbf{u}}. \tag{4}$$

As we can see, the NDFs for both geometric optics and wave optics are able to be defined consistently as 2D images with given sizes of pixel footprints. Also note that, as shown in Figure 3, even when the pixel's footprint becomes rather large, the NDF still contains a lot of high-frequency information. This indicates that the NDF images still needs accurate computation.

3.2 Motivation

From the way $\mathcal{P}$-NDFs and GNDFs are computed (Eqns. 2 and 4), we immediately find that the complexity is corresponding to the spatial size of the pixel footprint. This is undesired and often against intuition. For a large pixel footprint (Fig. 2 (left)), it takes longer to compute or query the NDF, even though there are fewer high frequency details in the NDF. Also, when an area light or environment light is involved (Fig. 2 (right)), sampling the solid angle subtended by the light results in different incident directions, indicating that the NDF needs to be queried multiple times at different half vectors. However, this is equivalent to rendering with a fixed incident direction using an angularly blurred NDF.

The above analysis inspires us to design a spatio-angular prefiltering scheme — given any spatial range on the surface and any angular range on half vectors, we want to find the corresponding NDF image (spatial filtering), then find the average of NDF values within a block on this image (angular filtering). Even better, we should be able to skip producing the entire NDF image and only focus on the angular block, and we should avoid looping over all places inside this block but directly acquiring the average value. This is crucial to achieve constant performance.

To carry out an actual spatio-angular prefiltering method, we start by considering the following question: is it possible to exhaustively precompute all the NDF images for all possible pixel footprints, until the footprint is large enough so that the NDFs become smooth? In this way, the heavy computation of NDFs will become a simple query of the precomputed data, and no matter how large a pixel

footprint is, the query could be performed in constant time. The answer is yes, and we demonstrate the possibility in Sec. 4.

However, precomputation-based approaches often have three fundamental problems that prevent their practical use:

(1) heavy data storage, especially for high-resolution normal maps/heightfields used to define the microstructures,
(2) difficult partial evaluation, i.e. hard to extract part of the compressed data instead of the entire chunk during rendering, and,
(3) fixed representation, that is, the precomputed data usually cannot be manipulated and used for synthesis.

We challenge all these limitations from precomputation, and prove that even exhaustive precomputation can be made both elegant and practical. We elaborate our method in the next sections, starting by introducing the technical part of our method, explaining how to precompute, compress and decompress data, leading to arbitrary constant-cost spatio-angular prefiltering (Sec. 4). With our prefiltering approach, we introduce applications to meet practical rendering needs, such as importance sampling, global illumination and dynamic appearance synthesis (Sec. 5). Finally, we show high quality results with constant cost and compare with previous work (Sec. 6).

Like most other works [Wang et al. 2020c; Yan et al. 2014, 2016] on glinty appearance, we only focus on the NDF term in microfacet models. We leave the (also important) shadowing-masking terms and local multiple bounces among microfacets to the future work (global multiple bounces between objects will be properly addressed). Also, since angular prefiltering is aimed at high performance, similar to other angular prefiltering work [Gamboa et al. 2018], we do not deal with drastic visibility changes in the angular range. Apart from these, no further assumptions are made w.r.t. specific types of microstructure (though quality may vary, as will be analyzed thoroughly), granularity of the scene, optical models (geometric optics or wave optics), and so on.

4 SPATIAL-ANGULAR PREFILTERING: PRECOMPUTATION, COMPRESSION AND DECOMPRESSION

In this section, we focus on data preparation. We first explain how to perform exhaustive precomputation. Then we introduce our efficient compression scheme of the precomputed data, together with an accurate way of decompression that enables us to quickly answer the average value within a spatial and angular range.

4.1 Precomputation

To avoid expensive on the fly computation of NDFs during rendering, we propose to precompute the NDFs at discrete locations on the normal map with different pixel footprint sizes. We arrange such pixel footprints in a pre-determined multi-level structure, as Fig. 3 illustrates. In each level, we sample the pixel footprints' locations (centers) uniformly in a grid, and adjust their sizes accordingly so they can cover the entire normal map. The higher the levels are, the sparser we sample the pixel footprints, and the larger these pixel footprints will be, in order to guarantee full coverage.
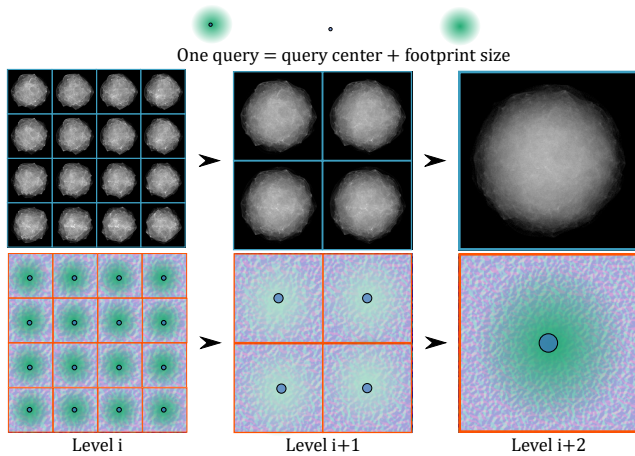
Fig. 3. We precompute NDFs at different levels. At each level, NDFs are computed at uniformly sampled locations (the blue dot) with the same footprint size (green transparent circle). Each higher level has 2×2 sparser sample count and twice larger footprint size than the previous level. The bounding box of the footprint size for each level is 30×30, 60×60 and 120×120 pixels. Normal map: isotropic noise (2K×2K).
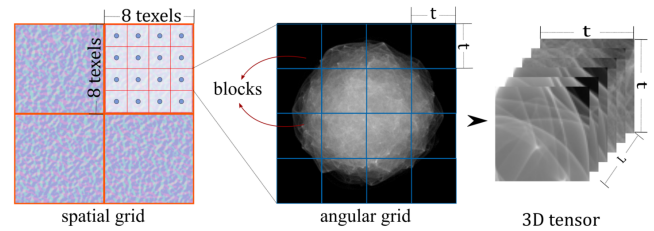

Fig. 4. Before compression, we cluster the NDF image blocks both positionally (on the normal map) and angularly (on the NDF image). All the image blocks which locate in the same angular grid and the same spatial grid form a 3D tensor, and will be compressed together. L is the count of non-empty NDF image blocks in a cluster.
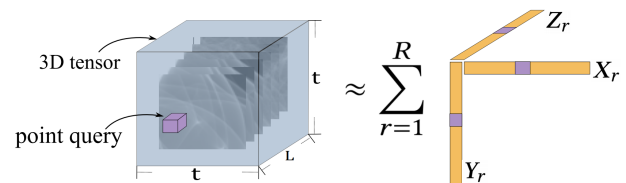

Fig. 5. Illustration of tensor decomposition. Tensor decomposition factorizes a 3D tensor (clustered blocks) into a linear combination of rank one tensors (X_r, Y_r and Z_r), where the number of the rank one tensors is R in this figure. With the factorized tensor vectors, point query (purple box) could be performed.

For each sampled pixel footprint, we compute an NDF image using the method by Yan et al. [2016]. As a result, our multi-level structure is similar to a texture mipmap. However, note the significant difference: each "texel" contains a color value from any level in a texture mipmap, but it is *an entire NDF image* in our case. With our multi-level structure, for any query with an arbitrary pixel footprint (location and size), we are able to find the NDF image immediately via trilinear interpolation between different samples in the same level and between different levels. In this way, we have successfully enabled constant performance spatial filtering which is irrelevant to the footprint size.

Practical choices. We generate the pixel footprint samples in a bottom-up fashion, as shown in Fig. 3. In the lowest level, we sample one pixel footprint with a fixed stride $s = 32$, i.e. drawing one sample every $s \times s$ texels. And the size of each pixel footprint is represented with a 2D Gaussian of standard deviation $\sigma_p = 1.5s/\sqrt{12}$, as suggested by Yan et al. [2016]. Starting from the second finest level, the sampled pixel footprints are 2×2 sparser and their sizes are 2×2 larger than those in the next lower level. We continue this sampling process until the NDF images for the neighboring levels are similar (with small MSE) or the stride reaches the normal map size.

As introduced in the background, one advantage of our precomputation is that it does not depend on specific optical models. For each sampled pixel footprint, we compute its NDF image using Yan et al. [2016] under geometric optics, and Yan et al. [2018] under wave optics. The only difference is that the NDF images under wave optics are colored. Similar to [Kuznetsov et al. 2019], we compute these NDF images using 8 spectrum samples then convert them to RGB images.

4.2 Compression

The precomputed NDF images can be used for BRDF evaluation directly, however, the precomputed data can be storage-consuming, which makes it less practical. We propose an efficient compression scheme of the precomputed NDF images in this section.

An important observation is that the NDF images from higher levels must contain the features from lower levels, therefore we propose an immediate solution that stacks all the NDF images together into a 3D tensor and compresses it using tensor decomposition.

Moreover, we find that there are similar parts (e.g. the high frequency curves) and large unoccupied black regions (especially when the overall appearance is glossy) inside each NDF image, which can be better utilized. Hence, we subdivide each NDF image into blocks in an angular grid (Fig. 4 (middle)).

However, stacking all these image blocks together results in a very "thin" tensor, which is not desired for tensor compression. Therefore, we further cluster these image blocks into different groups. Each group contain image blocks generated from pixel footprints with similar locations and levels. And then we compress each group individually using 3D tensor decomposition.

The tensor decomposition generalizes the Singular Value Decomposition (SVD) in 2D into higher dimensions. Specifically, in our case, it factorizes each 3D tensor in a group j into a weighted sum of outer products from 3 vectors, as shown in Figure 5. Each outer product results in a rank one tensor, and we keep the most significant R

ranks with the largest weights. Therefore, we have

$$D \approx \hat{D} = \sum_{r=1}^{R} C_r \otimes X_r \otimes Y_r \otimes Z_r, \quad (5)$$

where D is the original 3D tensor in the group, approximated as $\hat{D}$ after compression, X, Y and Z are 1D vectors of length $R \times t$, $R \times t$ and $R \times L$, respectively. And C is the coefficient for different ranks. For simplicity, we ignore the group index j here.

Practical choices. Angularly, we set the size of each image block as $t \times t$, and we choose $t = 16$ in practice. Spatially, we do not use optimization to guide how to form clusters[Sloan et al. 2003]. Instead, we refer to a simple deterministic method to perform clustering. That is, we divide the entire normal map into a spatial grid, separating the normal map into 8×8 texel regions. We cluster all the NDF images (blocks) into one group as long as their corresponding pixel footprints are centered inside the same region, as shown in Figure 4.

In each group, we stack the image blocks along the third dimension, resulting in a 3D tensor with size $t \times t \times L$, where L is the number of image blocks. Then we conduct tensor decomposition for each group. Specifically, we perform Canonical Polyadic Decomposition [Hitchcock 1927] using alternating least squares with rank set as R, with maximum error set as 10^{-4} and maximum iteration count set as 500.

We set the maximum rank R as 16 or 32, based on specific types of normal maps, as will be elaborated in Sec. 6. Compared to the original 3D tensor, we have reached a compression ratio between $0.72\% \sim 3.76\%$. This ratio is not constant for different normal maps, because we further optimize our compression scheme by throwing away complete blank image blocks, which is especially efficient for glossy appearances. With our choice of R, we have a good balance practically between the compression ratio and the compression quality, which is demonstrated in Fig. 18. Regarding the performance of the compression scheme, it is about $10 \sim 60$ minutes for normal maps of resolution $2K \times 2K$.

4.3 Decompression for angular point query

And as analyzed earlier, spatial prefiltering has been properly addressed using precomputation. With our compression scheme, the storage overhead of the precomputation is also greatly reduced. But it introduces a new issue: during rendering, we usually just need to find the value along individual angles or in an angular range in an NDF image. Thus, we do not need the entire 3D tensor to be decompressed, but only need to query the locations.

In this subsection, we first deal with a specific case corresponding to rendering under a point or directional light or light sampling. Given a pixel footprint, an incident direction and an outgoing direction, we would like to perform *point query* instead of full extraction of the compressed data.

With our compression based on tensor decomposition, the point query can be elegantly achieved. This is because of the property of the outer product operation – any element in the resulting rank-1 tensor is the product of corresponding elements from the resulting 1D vectors, as Fig. 6 illustrates. Therefore, given a query with an arbitrary pixel footprint and the half vector $\mathbf{h} = (h_x, h_y)$ between the incident and outgoing directions, it is trivial to locate the query
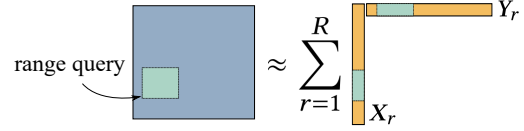

Fig. 6. Angular range query (green box) is performed on the compressed tensors. The average value in a rectangle can be decomposed as the product of the average values along its axis green segment on the right image).

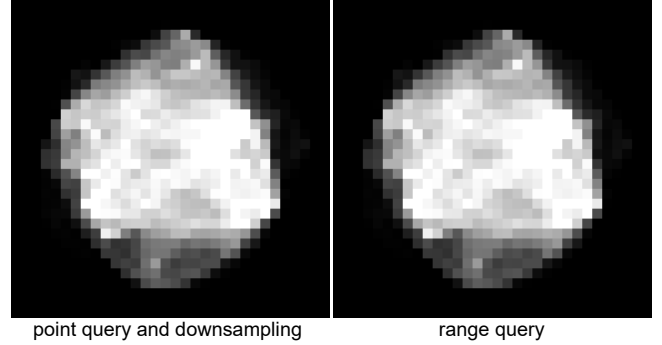point query and downsampling   range query

Fig. 7. The NDFs images are computed by (left) performing point query and downsampling and (right) performing range query directly, and they are identical.

index (x, y, z) in a clustered group j using $\mathbf{x}$ and $\mathbf{h}$. Then we can simply extract the specific NDF value as:

$$\hat{D}(x, y, z) = \sum_{r=1}^{R} C_r \cdot X_r(x) \cdot Y_r(y) \cdot Z_r(z). \quad (6)$$

Recall that when the pixel footprint is not an exact match of the precomputed ones, we perform trilinear interpolation of the NDF values from nearby precomputed pixel footprints. The is equivalent to trilinearly blending the specific NDF values towards $\mathbf{h}$. Therefore, Eqn. 6 will simply be called multiple times. Also note that our decompression for point query is as accurate as acquiring a full NDF image followed by querying. It does not introduce any further approximation error in addition to the compression itself.

4.4 Decompression for angular range query

As motivated in Sec. 3, when large area lights or environment lighting are involved, angular prefiltering can greatly increase the efficiency of the rendering process. And angular prefiltering is essentially asking for the average value within a range of pixels in an NDF image. Therefore, besides angular point query, *range query* is also a common operation. Moreover, as will be introduced in the next section, importance sampling also relies heavily on the range query. Hence, the efficiency of range query is crucial.

Suppose the angular range of half vectors maps to a rectangular (not necessarily square) region $[x_1, x_2] \times [y_1, y_2]$ on an NDF image block with "depth" z in a specific clustered group j, a simple solution to this range query is to perform multiple point queries at different pixels using Eqn. 6 and average them. However, this is costly and its performance scales with the size of the angular range. It seems

difficult to make the performance cost of range query constant to the query size, however, with our tensor decomposition, the range query can be solved cleanly and efficiently as:

$$\hat{D}([x_1, x_2], [y_1, y_2], z) = \sum_{r=1}^{R} C_r \cdot \bar{X}_r([x_1, x_2]) \cdot \bar{Y}_r([y_1, y_2]) \cdot Z_r(z).$$
(7)

where $\bar{X}_r([x_1, x_2])$ means the average value in a segment $[x_1, x_2]$ on the 1D vector X, and similarly for Y.

An illustration of Eqn. 7 can be found in Fig. 6. The key observation is that the average value in a rectangle can be decomposed as the product of the average values along its axis. Note specifically that this observation is *not generally true*, but is *strictly accurate in our case*. We provide proof with detailed derivation in the Appendix. We also verify its accuracy in practice in Fig. 7, showing identical results generated using our range query once and our point query repeatedly.

Now the only remaining task is to quickly attain the average value in any given segment on a 1D vector. This problem is well studied with the help of the Summed Area Table (SAT) [Crow 1984] data structure. In our case, this is even easier because we only need a 1D SAT for each 1D vector. The SAT performs in linear time and exactly doubles the storage, but makes the time cost of angular range query $O(1)$ (strictly, 2 memory look-ups) and gives exact results.

One might think of a 1D mipmap structure to perform the same task, but we would like to point out that (1) mipmaps only provide approximated range queries, (2) mipmaps also double the storage in 1D instead of introducing only 33% additional storage as in 2D cases, and (3) mipmaps require 4 memory look-ups for each query due to trilinear interpolation in 1D. Therefore, SAT is always superior.

Practical choices. When the angular range happen to overlap multiple image blocks, we subdivide it into smaller ones according to the boundary of the image blocks. In practice, this situation does not happen frequently. Also, throughout our paper, we report the full storage cost including the SATs.

Summary. In this section, we have elaborated our precomputation scheme which enables constant performance spatial prefiltering. Then we propose our compression scheme using tensor decomposition, which not only greatly reduces the storage cost, but also leads to efficient and accurate angular prefiltering, for both point query and range query. So, now we have a complete solution of constant performance spatio-angular prefiltering of glinty appearance. In the next section, we show its practical applications in different rendering tasks that were previously considered difficult. We also show how our method could combine with constant storage appearance synthesis approaches, to complete our method to an overall constant cost approach.

5 CONSTANT-COST GLINTY APPEARANCE RENDERING

In this section, we show several applications with our compressed NDF images. We start from efficient BRDF evaluation (Sec. 5.1), then extend to unique applications enabled by spatio-angular prefiltering, such as accurate NDF importance sampling (Sec. 5.2), global illumination with glinty appearance (Sec. 5.3), environment lighting

step 1: sample a tile   step 2: sample sub-quad   step 3: sample sub-quad   step i: get the direction
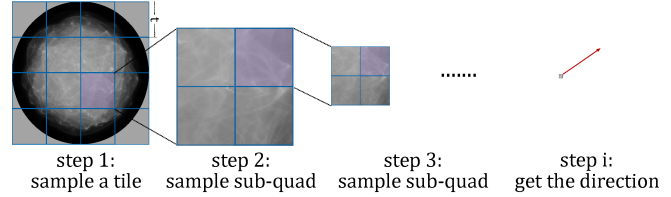
Fig. 8. We sample the outgoing direction hierarchically: start from choosing an image block from NDF, by sampling the averaged NDF values with CDF, and then choose sub-quad with importance sampling until reaching the pixels, and then interpolate the surrounding directions to get the final sampled direction.

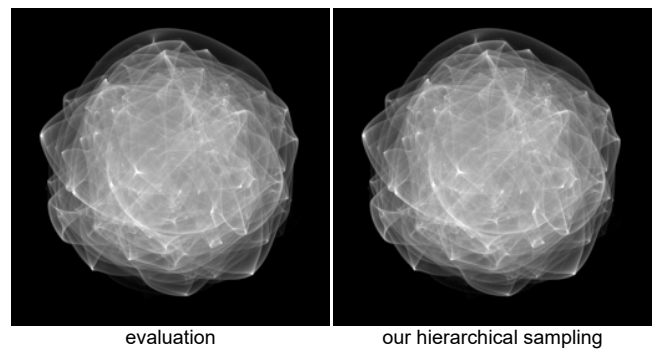evaluation   our hierarchical sampling

Fig. 9. Comparison of NDF images computed with BRDF evaluation and our hierarchical BRDF sampling via the binning method. Binning means sampling a direction and accumulate into the corresponding pixel in the NDF images, like a histogram. They are identical.

prefiltering (Sec. 5.4) and constant storage appearance synthesis (Sec. 5.5).

5.1 BRDF evaluation

With the point angular query, it's straightforward and fast to evaluate the BRDF value for arbitrary pixel footprints, incident and outgoing directions. Our approach avoids the expensive hierarchy traversal and the calculation of contribution from a large number of Gaussian flakes when the footprint is large, yielding much faster constant performance BRDF evaluation.

Since we precompute a range of pixel footprint sizes, in practice, the pixel footprint size can exceed this range. If the pixel footprint size is smaller than the precomputed size at the lowest level of the tensor mipmap, we switch to the method by Yan et al. [2016], which is now efficient with the small footprint size. If the pixel footprint size is larger than the largest size precomputed at the highest level, we directly clamp the size to the largest precomputed, since we stop the precomputation when we believe that the NDFs at the highest level have converged to a smooth distribution.

5.2 Accurate NDF importance sampling

Besides evaluation, importance sampling is another commonly used operation. Given the pixel footprint and an incident direction, importance sampling gives a sampled outgoing direction, which equivalent to a sampled half vector direction. Ideally, the half vector should be sampled strictly according to the shape of the NDF, which would reduce the variance to the minimum thus is called perfect importance sampling.

One possible way to achieve perfect importance sampling is reconstructing the entire NDF image with brute force angular point queries, computing its 2D Cumulative Distribution Function (CDF), and using the inverse sampling method to draw an outgoing direction with this CDF. However, this is very time-consuming and completely offsets the benefits from our efficient BRDF evaluation.

Thanks to our angular range query, we propose an efficient hierarchical importance sampling approach. We start from the entire NDF image, and subdivide it evenly into four quads. Recall that our angular range query answers the average value in an arbitrary range in constant time, we are able to immediately acquire the average values in these four quads. Then we use the four average values as relative probabilities, and randomly choose one quad to proceed. We subdivide this quad into four sub-quads, sample one, and continue this subdivision and sampling process iteratively until reaching the bottom level, i.e. a pixel. In this way, the probability of drawing a sample is guaranteed to be proportional to the "brightness" of each pixel in an NDF image. Therefore, we have achieved perfect importance sampling.

In practice, since an NDF image has already been subdivided into image blocks, we do not have to start from the topmost level of the NDF image. Instead, we first choose an image block to start, again according to their averaged NDF values as relative probabilities. Also, when we reach a pixel, we uniformly perturb the sample location inside it. The full importance sampling approach is illustrated in Fig. 8.

Note that since the range query is accurate, our importance sampling approach is also accurate. Therefore, the Probability Density Function (PDF) associated with the sampling method is exactly the same as the evaluated NDF value. We verify this in Fig. 9. On the left, we show an NDF image generated using point query evaluation. On the right, we show the converged histogram of 10M sampled half vector directions (a.k.a. using the binning method). As expected, these two NDF images are exactly identical.

Discussion. The way we use angular range queries to do importance sampling essentially gives us an *implicit* hierarchical structure on the NDF image. This results in a logarithmic performance w.r.t. the size of NDF images, which is still a constant 256×256. Therefore, our importance sampling approach is still constant performance and runs efficiently in practice. We also would like to point out that building an explicit hierarchical structure on the fly for an NDF image is impractical, since this step will already take linear time, as heavy as the CDF-based sampling approach.

Unlike Yan et al. [2016] that keeps the original normal map to facilitate importance sampling, our importance sampling no longer needs it, which further saves storage. Also note that, there are no previous method that does perfect importance sampling under wave
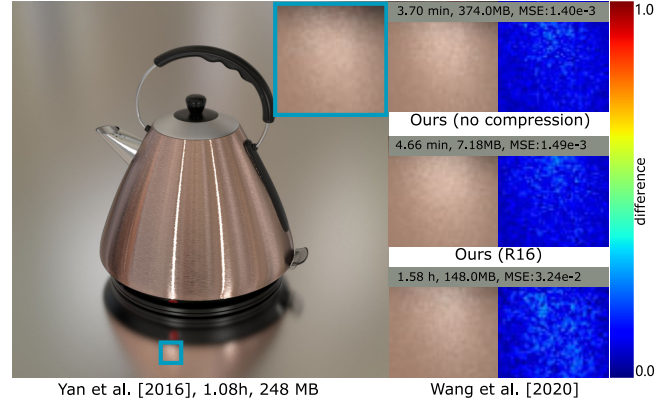

Fig. 10. Global illumination with glinty appearance: Comparison between our method (without compression), our method (with compression with rank 16), Wang et al. [2020c] and Yan et al. [2016] on the Kettle Scene, considering geometric optics rendering and indirect glints. Both of our methods do not consider synthesizing. The results of these methods are identical, however our method is 14× faster. Our method with compression costs much less storage cost than our method (without compression). The picture is with 1024 × 1024 pixels. Normal map: brushed metal.

optics. But in our framework, since we uniformly represent NDFs as precomputed images for both geometric and wave optics, no additional effort will be needed, except that we use the grayscale NDF image to conduct importance sampling.

5.3 Global illumination with glinty appearance

Most of the previous glinty appearance rendering methods limit themselves to the direct illumination only. This is because the pixel footprints are amplified significantly as the light undergoes more bounces during the light transport. To our knowledge, Belcour et al. [2017] deal with explicit multiple-bounce global illumination on glinty appearance. However, it only provides a way to calculate the coverages of pixel footprints at different bounces, but still has to clamp them to a small value for practical rendering, and still yields drastically increasing time cost.

With our spatial prefiltering (plus angular point query), our method automatically solves the global illumination problem, making it practical. We follow the indirect footprint computation by Wang et al. [2018], using accurate footprint for the direct footprint and amply it considering the glossiness during the following bounces. In Fig. 10, we provide examples and discussion.

Note again that the global illumination refers to multiple bounces of light in the level of objects and scenes. It is not related to the multiple scattering of light between microfacets that leads to energy conserving BSDFs.

5.4 Environment lighting prefiltering

Under distant lighting, especially defined using environment maps, rendering of the microstructures is the convolution of the incoming radiance and the BRDF, which in an integral over the directional domain. Traditionally, to compute this integral, a large number of

Ours (with prefiltering)
spp: 64

Ours (without prefiltering)
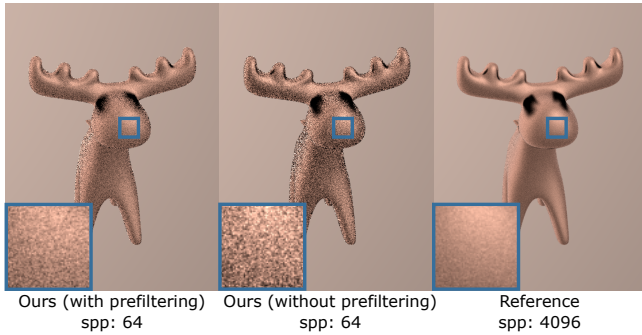spp: 64

Reference
spp: 4096

Fig. 11. Environment lighting prefiltering: Comparison between our method with / without prefiltering and the reference, rendered with Yan et al. [2016]. We do not use multiple importance sampling to verify the prefiltering impact more clear. Our method (with prefiltering) produces much less noise than our method (without prefiltering). The picture is with 920 × 1024 pixels. Normal map: isotropic noise.

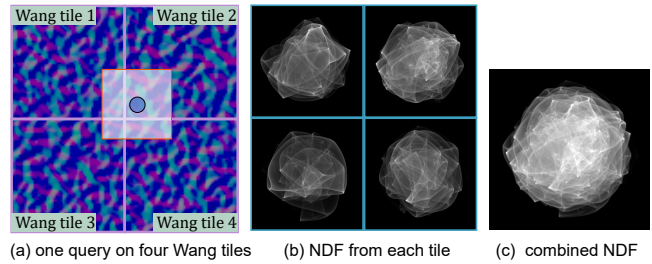(a) one query on four Wang tiles   (b) NDF from each tile   (c) combined NDF

Fig. 12. During precomputation, we precompute the NDF images for each Wang tile, allowing queries whose centers are outside but still intersect the Wang tile. During runtime, the NDF image for a query that crosses the borders between different Wang tiles is accurately computed by combining the precomputed NDF images from all overlapping Wang tiles.
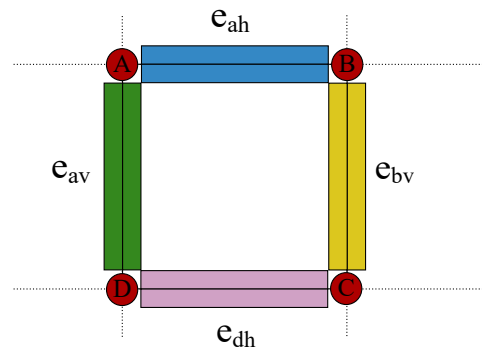

Fig. 13. To generate synthesized normal map on the fly without any index map, we generate two random numbers with the vertex index as seed, and then map these numbers to "color" via a hash table. For example, we generate two random numbers with the index of vertex A, and then map them to two "color" with the hash table. The two "color" are set to edges e_{ah} and e_{av}, which are the right and bottom edges of vertex A. After determining the "color" of all the edges, we get the corresponding Wang tile.

samples are required to obtain noise-free results, even the environment map is low frequency. Given the knowledge of a low-frequency light, prefiltering could be used to reduce required samples, essentially computing blurred versions of the NDF images. Therefore, our angular prefiltering scheme can be conveniently applied in this prefiltering task.

Without loss of generality, we represent the environment map with spherical Gaussians (SGs) ([Tsai and Shih 2006]). Other choices are all possible as long as their frequency bandwidths can be acquired. During rendering, we first importance sample the SG-represented environment lighting to find one SG. Based on the SG's bandwidth (size or shape), we compute a corresponding angular range to query the average NDF values.

To obtain this range, we first compute an angle θ in radians, where all directions within θ around the SG's central axis will have a value greater than a threshold ϵ. This property is called SG's compact-ϵ support [Wang et al. 2009]:

$$\theta = \arccos\left(\frac{\ln \epsilon - \ln A}{\lambda} + 1\right), \qquad (8)$$

where A and λ represent the amplitude and bandwidth of an SG, respectively. And in practice, we set ϵ as 0.3.

Now that the incident lighting has a compact-ϵ support of θ, as Fig. 2 (right) indicates, we are able to safely "blur" the NDF using an SG with an approximate compact-ϵ support of $\theta/2$, which maps to a square area on the NDF image with side length

$$Q = 256 \cdot \frac{\theta/2}{\pi} \cdot 2 = 256\theta/\pi \qquad (9)$$

where 256 is the resolution of the NDF image.

The essential effect of our angular prefiltering is that the maximum frequencies of both the light and the BRDF are significantly reduced. This results in much lower noise level compared to naive point sampling (Fig. 11).

5.5 Constant-storage appearance synthesis

With compressed NDF images, our method is able to achieve constant performance, even for large footprint size. However, the storage cost is still expensive for high-resolution normal maps, thus, we would like to implicitly generate infinite large normal maps. Wang tiles [Cohen et al. 2003] is able to synthesize large textures by repeating precomputed seamless tiles. We propose to combine our method with Wang tiles to synthesize a high-resolution normal map from an input sample normal map for constant storage.

Precomputation and compression. We generate the Wang tiles from the input normal map. Then we precompute NDF images for each Wang tile. The precomputation process is similar to the precomputation for a normal map in Section 4.1, except the sampled centers could locate outside the Wang tiles, as shown in Figure 12. In practice, we represent each normal map with 16 Wang tiles and set the Wang tile size as 512 × 512. We then compress the NDF images for each Wang tile as described in Section 4.2 and build the SAT for compressed tensor vectors.

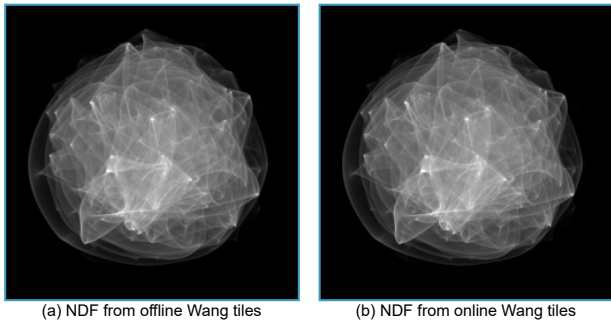(a) NDF from offline Wang tiles   (b) NDF from online Wang tiles

Fig. 14. We visualize the NDF of a normal map generated by Wang tiles offline (generate the synthesized normal map explicitly) and online (using the color hash table to generate the synthesized normal map implicitly).

Dynamic Wang tile generation. Originally, the Wang tiles method still requires a pre-generated index map to determine the locations of individual Wang tiles. Therefore, strictly speaking, it cannot generate an infinitely large normal map at the cost of constant storage. For dynamic Wang tile selection at arbitrary locations, Lagae and Dutré [2005] proposed a method that implicitly build an infinitely large grid, and associate each vertex in this grid with four random numbers. During runtime, the "color" of an edge in this grid is determined using its associated vertices' corresponding random numbers. We slightly improve upon Lagae and Dutré [2005] so each vertex directly controls the "color" of edges to its right and bottom (Figure 13). For each vertex, we generate two random numbers according to a hashed value of its position / index. The two random numbers are used to determine the color of the right and bottom edges to this vertex, respectively. During rendering, given a query center and footprint size, we first find the "colors" of all the edges covered by the query, then we are able to identity the corresponding Wang tiles from a precomputed edge-to-tile map. All the computation is quickly performed on the fly.

Evaluation and sampling. After getting the Wang tiles covered by the footprint, we perform BRDF evaluation for each Wang tile, similar to Section 4.3 and sum all contribution of the Wang tiles, as shown in Figure 12. In Figure 14, we compare the NDF images of normal maps which are generated by the implicit and explicit Wang tile algorithms, and they are identical. For importance sampling, we first choose one Wang tile from all the Wang tiles covered by the pixel footprint with equal probability. Then, we sample a direction from the chosen Wang tile, similar to Sec. 5.2. After getting the sampled direction, we computed the NDF value, which is the same as BRDF evaluation. The PDF is set the same as NDF value. In Figure 15, we compare our method with / without Wang tiles, and other methods, to show our benefits.

6 RESULTS

We have implemented our algorithm inside the Mitsuba renderer [Jakob 2010]. We compare against Yan et al. [2016] for geometric optics glints validation and against Yan et al. [2018] for wave optics glints validation. All timings in this section are measured on a 2.39 GHz Intel i7 (40 cores) with 256 GB of main memory. Unless

Table 2. The precomputation cost of each texture (normal map for geometric optics glints, and heightfield for wave optics glints), including NDF generation time (Pre.), compression time (Com.), total time (Tot.) and the storage cost (Stor.). Regarding the type, G represents geometric optics glints, and W represents wave optics glints. Res. is the resolution of the texture. The NDF of isotropic noise ($8K$) has three channels (RGB) while others have only one channel.

Texture	Type	Res.	Rank	Pre. min.	Com. min.	Tot. min.	Stor. MB
Metallic flakes	G	$2K^2$	32	42.80	60.01	102.81	106.86
Brushed metal	G	$2K^2$	16	37.39	10.01	47.40	15.19
Brushed metal	G	$1K^2$	16	9.56	5.36	14.92	7.18
Isotropic noise	G	512^2	16	59.71	25.32	85.03	6.87
Isotropic noise	W	$8K^2$	16	156.00	347.94	503.94	352.50

Table 3. The rendering time (path tracing (with evaluation only) or path tracing (with sample only) of our method compared to microfacet model. All timings are measured over the Elevator scene with 1024 spp.

Method	Evaluation	Sampling
Ours	2.30 m	6.57 m
Microfacet	2.00 m	3.43 m

otherwise specified, all timings correspond to pictures with 1280×720 pixels.

6.1 Performance Analysis.

Precomputation Cost. In Table 2, we report the precomputation cost for each normal map or heightfield, including the NDF generation time, compression time, and the storage cost.

The compression time depends on the resolution and the NDF behaviors. For example, there are more tiles with zero-value on the brushed metal NDF image, and they are discarded directly, resulting in shorter compression time. Since rank 32 is used for metallic flake normal map, it has longer precomputation time and more storage cost. The $8K \times 8K$ isotropic noise heightfield requires longer time, due to its higher resolution and more channels (RGB).

Shading Cost. In Table 3, we analysis the performance cost (BRDF evaluation and BRDF sampling) of our method during rendering, compared to mircofacet model [Walter et al. 2007]. To better understand the individual cost of evaluation and sampling, we use path tracing with evaluation only or with samplings only. By comparison, the overhead of evaluation in our method is pretty low (about 15%), while the cost of sampling is about 2× of microfact model, due to the multiple tensor reconstruction operations.

Rendering Cost. In Table 4, we report all the scene settings, rendering time and memory costs for our method and the reference methods. The speedup of our method over the reference methods (Yan et al. [2016] or Yan et al. [2018]) varies from 4× to 400×, depending on the footprint size, and using BRDF evaluation or sampling. For the Kettle and Elevator scenes, the high sampling rate leads

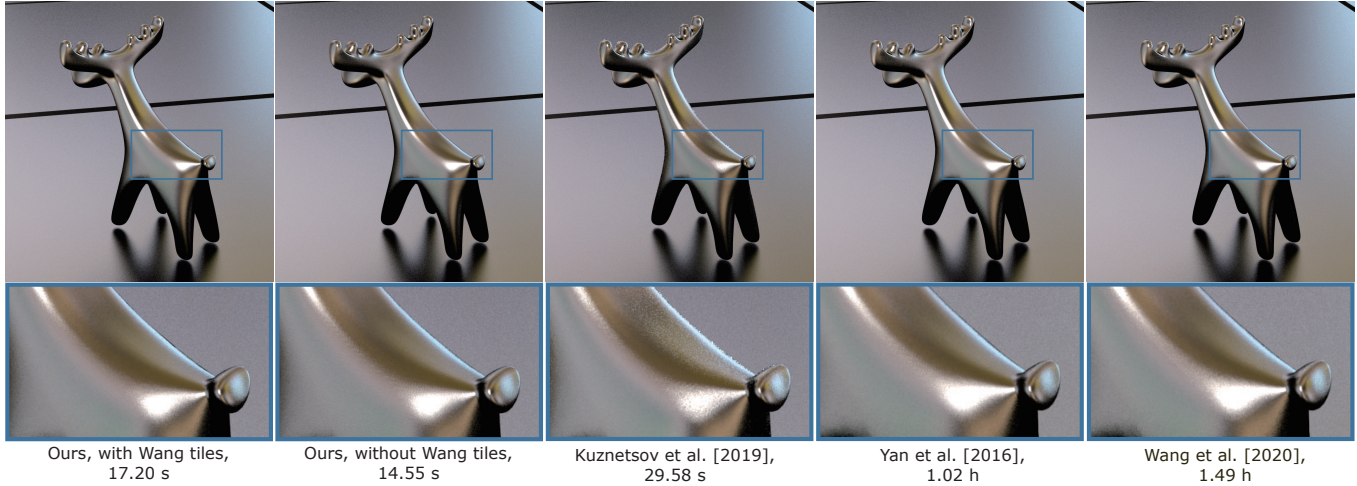| Ours, with Wang tiles, 17.20 s | Ours, without Wang tiles, 14.55 s | Kuznetsov et al. [2019], 29.58 s | Yan et al. [2016], 1.02 h | Wang et al. [2020], 1.49 h |

Fig. 15. Constant-storage appearance synthesis demonstration. Comparison between our method with / without Wang tiles, Kuznetsov et al. [2019], Yan et al. [2016] and Wang et al. [2020c], considering geometric optics rendering. Our method with Wang tiles solves the repetitive pattern issue, compared to our result without Wang tiles and Yan et al. [2016]. Kuznetsov et al. [2019] produces results with lower quality and higher time cost, compared to our method. The picture is with 920 × 1024 pixels. Normal map: isotropic noise.

Table 4. Scene settings, rendering time and memory costs for our test scenes. #Tri. is the count of triangles in the scene. Spp. represents sample per pixel for path tracing, and different spp is used for environment map and point light in the Laptop scene. The details of input normal map or heightfield are shown in Table 2. The reference method is Yan et al. [2016] for geometric optics rendering, and Yan et al. [2018] for wave optics rendering. The tile count means the repeating count of the input textures during rendering. The storage of the Laptop with Yan et al. [2018] is not shown, since it does not require any extra structures (flakes or acceleration structure).

Scene	#Tri.	Spp. (ours)		Spp. (ref.)		Texture	Tile Count		Total time (min.)		Storage cost (MB)	
	K	Env.	Others	Env.	Others		Ours	Ref.	Ours	Ref.	Ours	Ref.
Car	78.1	64	64	64	64	Metallic flakes	1000	1000	5.15	522.00	106.86	176.00
Kettle	175.3	1024	1024	1024	1024	Brushed metal(1K)	16	16	4.66	64.80	7.18	44.00
Elevator	53.5	4096	4096	4096	4096	Brushed metal(2K)	1000	1000	64.80	864.00	15.19	176.00
Laptop	18.4	1024	1	1024	64	Isotropic noise(8K)	79	79	15.97	71.17	352.50	-

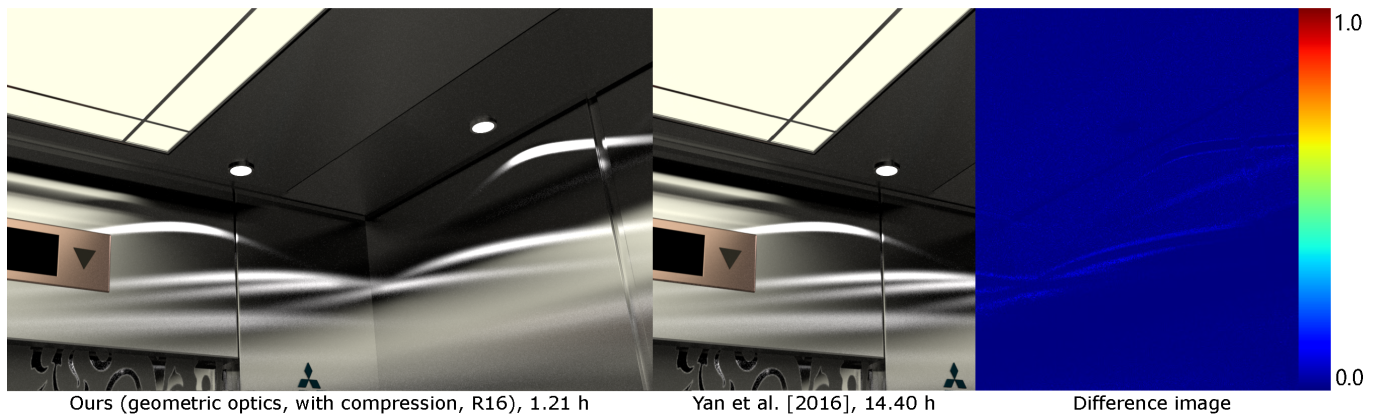| Ours (geometric optics, with compression, R16), 1.21 h | Yan et al. [2016], 14.40 h | Difference image |

Fig. 16. Comparison between our method (without synthesizing) and Yan et al. [2016] on the Elevator Scene, considering geometric optics rendering. The results of the two methods are identical, however our method is about 12× faster. R16 means using rank 16 for compression. Normal map: brushed metal.
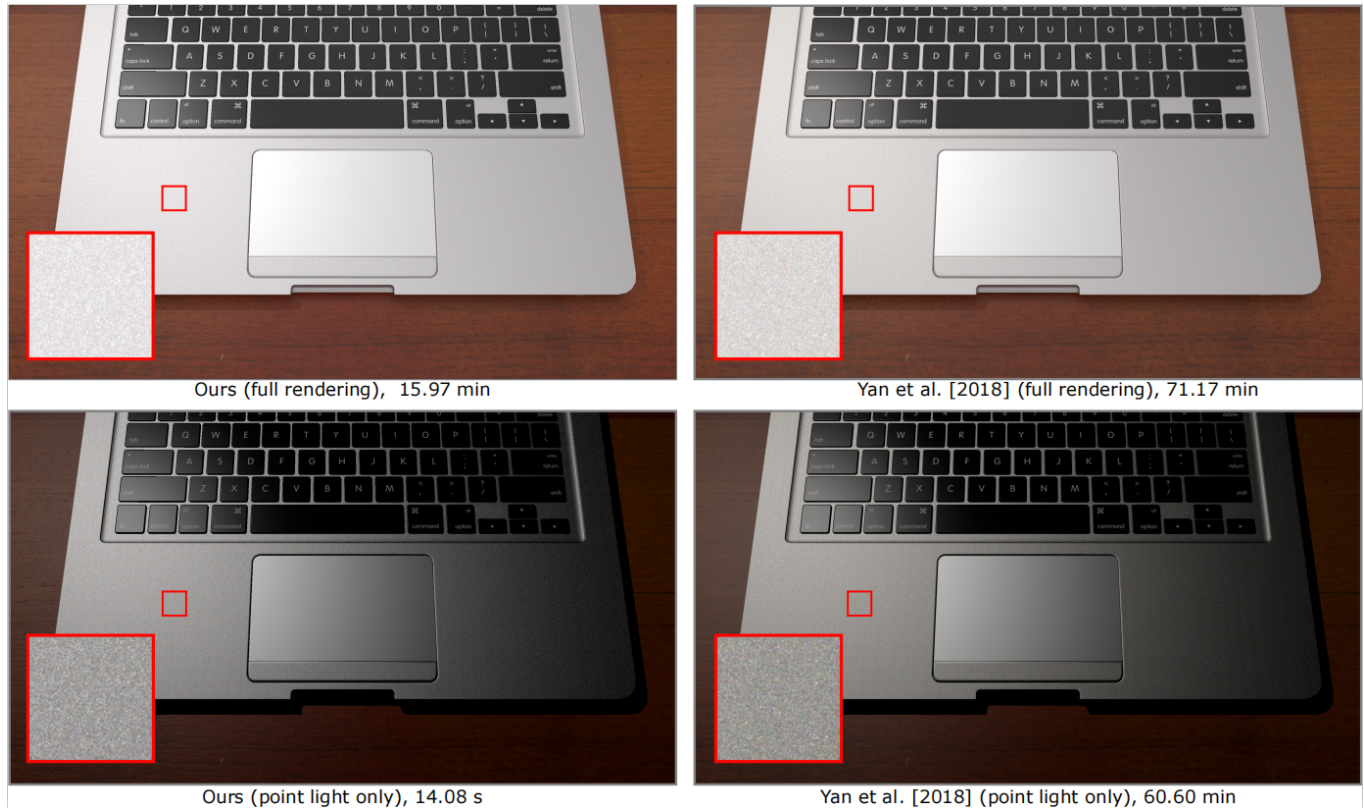
Fig. 17. Comparison between our method (without synthesizing) and Yan et al. [2018] on the Laptop Scene, considering wave optics rendering. We provided rendered results with (top row) and without (bottom row) environment map. With point light only, our method is about 260× faster than the reference. Including the environment map makes our method less impressive, about 4.5× faster, since we sample the BRDF. The picture is with 2550 × 1426 pixels. Heightfield: isotropic noise.

to smaller footprint, which resulting in less speedup compared to the Car scene. However, high sampling rate is required for indirect illumination. For the Laptop scene, since sampling is used for environment map, it decreases the speedup.

6.2 Comparison with Previous Work.

Car scene. The scene in Figure 1 shows a car with coated metallic flakes under environment lighting and a point light, with direct light only. We consider geometric optics rendering without synthesizing. The car is about 200cm wide. The input normal map with resolution $2K \times 2K$ covers 2mm × 2mm. We use compression rank 32 for this metallic flakes normal map, since it's discrete and requires more rank than others. Our method produces similar results to Yan et al. [2016], while our method is about 100× faster.

Elevator scene. In Figure 16, we show an elevator with brushed metallic walls under several area lights, including indirect illumination. We consider geometric optics rendering without synthesizing. The wall of an elevator is about 200cm wide. The input normal map with resolution $2K \times 2K$ covers 2mm × 2mm. Our method is about 13× faster than Yan et al. [2016]. Compared to Figure 1, the performance is less impressive due to the high sampling rate in the Elevator scene. High sampling rate results in small footprint,

since the footprint of each ray is the pixel's footprint divided by the sample count, which decreases the benefit of our method. However, high sampling rate is required to remove the noise, from the indirect lighting.

Kettle scene. Figure 10 illustrates a Kettle with brushed metal on the body under two small area lights and environment lighting, considering geometric optics rendering. This scene is designed to show global illumination with glinty appearance. The kettle is about 30cm high. The input brushed metal normal map with $1K \times 1K$ resolution covers about 9mm × 9mm. Besides the direct glints, we consider the indirect glints (the glints reflected by the glossy surface). The direct glints are computed with Yan et al. [2016] with 2.75 minutes, and the indirect glints are computed with our method with 0.68 minutes, as the footprints of direct glints are not large enough to use our method. The footprints are enlarged by the glossy surface, which yields low performance for Yan et al. [2016]. Our method produces identical results to Yan et al. [2016], with about 14× speedup. We also provide the results of Wang et al. [2020c], which is even slower than Yan et al. [2016].

Laptop scene. The scene in Figure 17 shows a laptop with a roughened aluminum matte finish. It is rendered using a point light and environment light, considering wave optics rendering. The
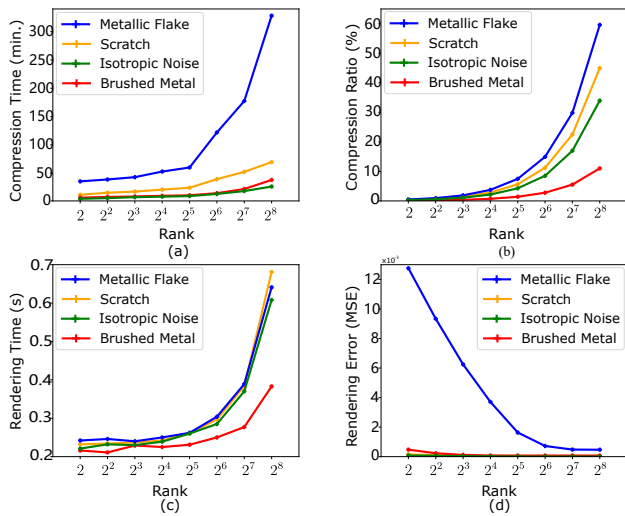
Fig. 18. Impact of compression rank on (a) compression time, (b) compression ratio, (c) rendering time and (d) rendered results error (MSE) on the isotropic noise normal map (2K). The rendering time and error are measured on the BentQuad scene with a point light.
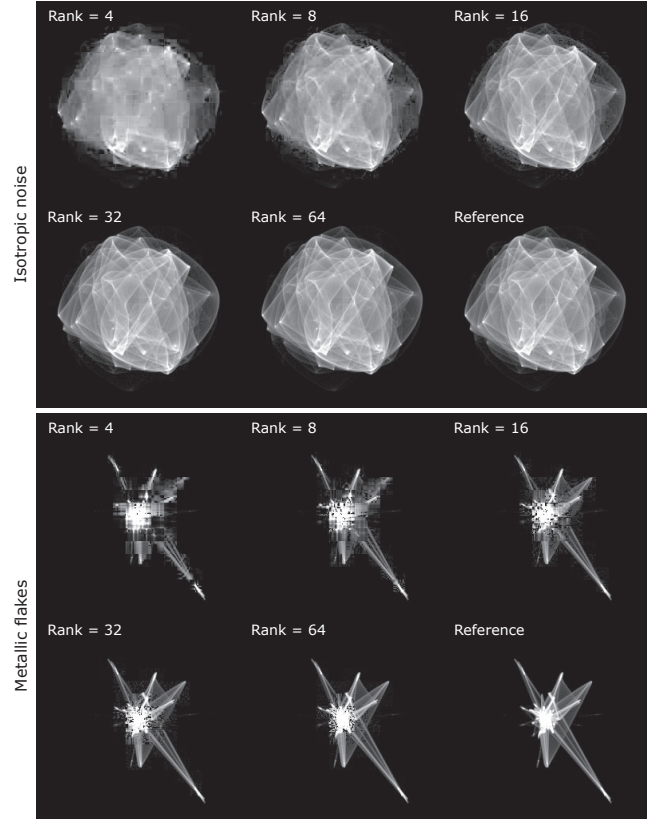

Fig. 19. Reconstructed NDF images from tensor decomposition with varying ranks. We set the rank as 16, balancing both the quality and the storage. Normal map: isotropic noise and metallic flakes.

laptop is about 30cm wide. We compare our method against Yan et al. [2018]. In both methods, we use a tileable input texture with size $8K \times 8K$ with 79 tiles. We render the results for environment map and the point light separately, which is the same as [Yan et al. 2018]. With point light only, our method is about 260× faster than the reference, thanks to our efficient BRDF evaluation. For environment map, we perform BRDF sampling, yielding similar performance to Yan et al. [2018], which slows down the speedup overall.

Deer scene. This scene shows a metallic deer statue under an environment lighting with geometric optics rendering. This scene is designed to show constant-storage appearance synthesis. In our method, we use 16 Wang tiles with size 512×512 from a $2K \times 2K$ normal map and the target normal map size is $100K \times 100K$. Our method with Wang tiles produces less repetitive artifacts with over little overhead. We also compare our method with Kuznetsov et al. [2019], which introduced a Generative Adversarial Networks (GAN) model to represent NDFs. Their model only learns the NDFs at certain footprint size(256), while our method supports multi-scale footprints. Compared to Kuznetsov et al. [2019], our method produces better result with about 2× speedup in the Figure 15. Regarding the storage cost, our method costs $6.87MB$, while Kuznetsov et al. [2019] only costs $1.26MB$, as their neural network representation is more compact than ours. We also compare against Yan et al. [2016] and Wang et al [2020c], where Yan et al. [2016] is 427× slower and suffers from repetitive patterns and Wang et al [2020c] is slower than Yan et al. [2016], but is free from the repetitive patterns.

In Figure 11, we compare our results with / without prefiltering and the reference. With prefiltering, our method produces much less noise than without prefiltering.

6.3 Parameter Analysis.

Compression rank R is an importance parameter in our algorithm. We show its impact on the both compression quality and compression storage. The compression quality is measured with difference between the image rendered with compressed NDF and original NDF. Figure 18 shows the impact of compression rank on compression time, compression ratio, rendering time and rendered results error of our algorithm on three normal maps (isotropic noise, brushed metal and metallic flakes). As shown in the curve, larger rank yields longer compression time, more storage cost and longer rendering time but less render error. In our implementation, the rank is set as 16 for most of the test scenes, except for the metallic flake normal map, which uses 32. Different compression ratios are achieved for different normal maps, since they have different blank image blocks. For example, the NDF images of Brushed Metal normal map have the most blank image blocks thus it has the lowest compression ratio. We also show the NDF images with varying ranks in Figure 19. When the rank is smaller than 16, the NDFs suffer from artifacts, while with 32 or larger rank, the NDFs are almost identical to the reference (obtained without compression), thus we think 16 is a
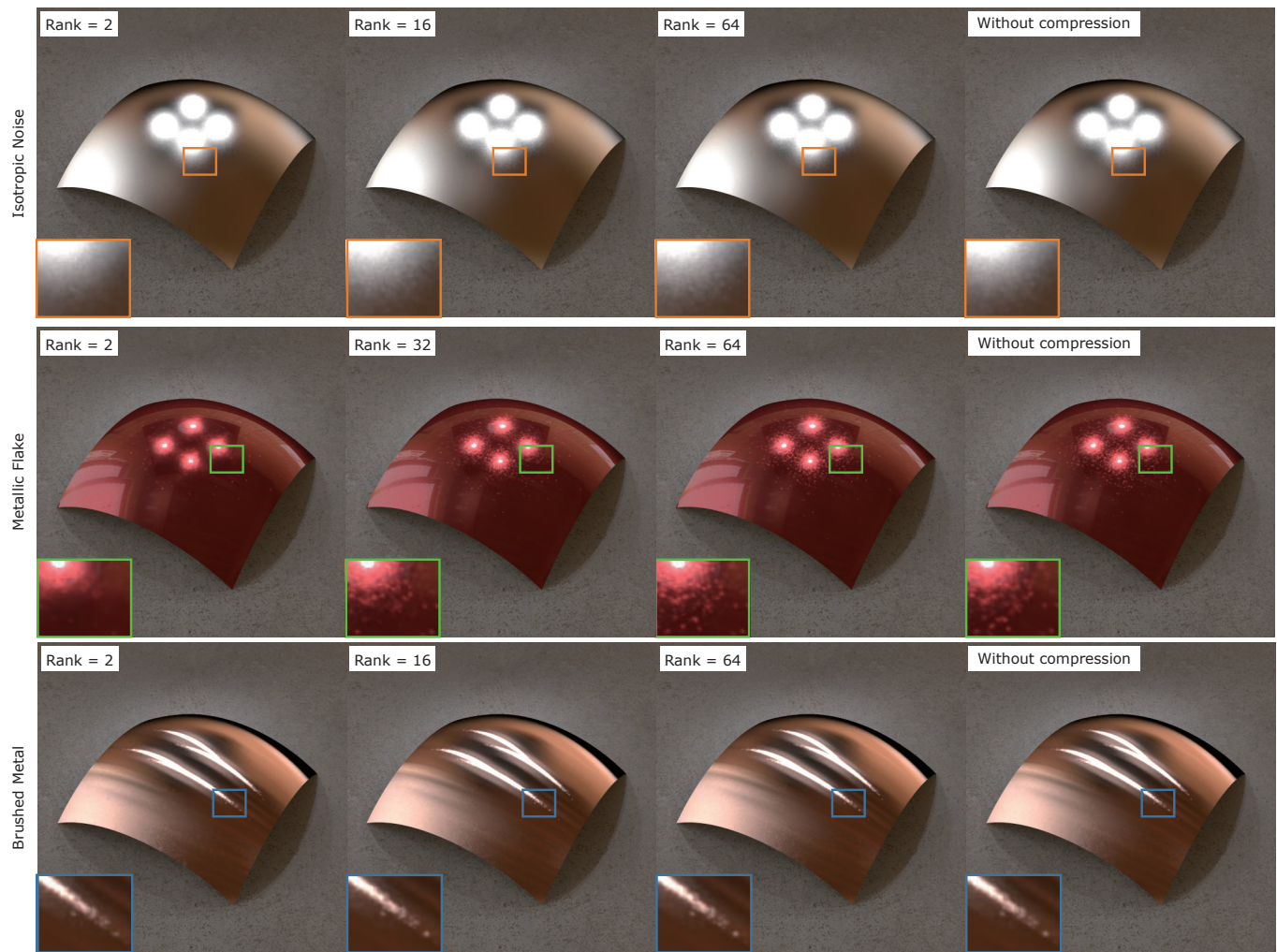
Fig. 20. Rendered images from tensor decomposition with varying ranks on the BentQuad scene with three normal maps. We set the rank as 16 for both isotropic noise and brushed metal and set the rank as 32 for metallic flakes, balancing both the quality and the storage.
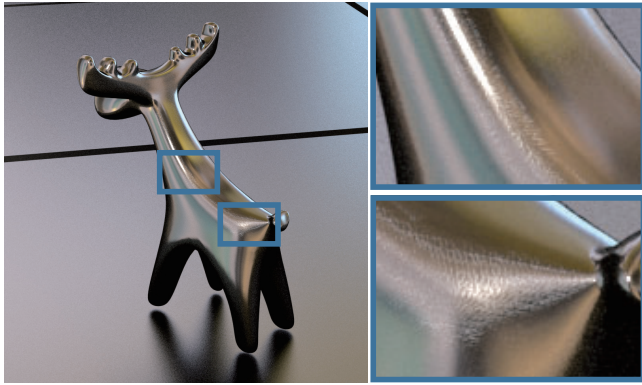
good balance considering both the storage and the quality. In Figure 20, we also show the rendered results of the BentQuad scene over varying rank on the three normal maps.

6.4 Discussion and limitation.

We compare our method with Gamboa et al. [2018], which also proposed to precompute the NDFs to achieve constant performance. They stored the NDFs with summed area table (SAT) like data structure, which has the same resolution as the input normal map, and a 9×32 histogram for NDF at each texel. The NDF of an angular range could be obtained by four queries of table, however, it has very expensive storage cost, about 1.1GB for a $2K \times 2K$ input normal map. Thus, it's almost impossible to handle an $8K \times 8K$ heightfield for wave optics rendering with more than 48 GB. Besides the expensive storage, the resolution of the histogram is too low to preserve the

glinty appearance of some materials (e.g. metallic flakes), as shown in our video. Thanks to our tensor decomposition to compress NDFs, our method is able to support much higher resolution (256×256) with much less storage cost (see Table 2).

Our proposed method has several limitations. First, our method has expensive NDF generation and compression cost, although they could be reused in other scenes. Second, the compression ratio could be improved with more compact representation. Third, our method inherits the issue of Wang tiles method, suffering from visible repetitive patterns for spatial correlated normal maps, as shown in Figure 21. Other synthesizing approach could also be coupled with our method, to solve this issue. Forth, in our prefiltering approach, we only use lights' bandwidth to limit the BRDF's bandwidth, however, the BRDF's bandwidth can also be used to limit lights' bandwidth,

Fig. 21. Our method with Wang tiles suffers from the repetitive patterns for some spatial correlated textures, inheriting the drawback of Wang tiles method. The picture is with 920 × 1024 pixels. Normal map: scratches.

similar to Gamboa et al. [2018]. For example, they used a blurry environment map when the BRDF computation uses a large footprint size.

7 CONCLUSION AND FUTURE WORK

We have presented a method that allows rendering of specular glints with constant performance and constant storage. By introducing compressed NDF images with angular point query and angular range query, our method is able to render glints with large footprint with constant time, and allow elegant importance sampling, prefiltering, and implicit microstructure synthesis. Eventually, our method is able to render both geometric-based glints and wave optics-based glints with constant storage and constant performance.

In the future, it would be interesting to optimize our method for real-time implementation on GPUs. Combining our method with other texture synthesis methods could also be worthwhile directions.

REFERENCES

Asen Atanasov, Alexander Wilkie, Vladimir Koylazov, and Jaroslav Krivánek. 2021. A Multiscale Microfacet Model Based on Inverse Bin Mapping. *Computer Graphics Forum* (2021). https://doi.org/10.1111/cgf.142618

Wang Beibei, Ge Liangsheng, and Holzschuch. Nicolas. 2020. Precomputed Multiple Scattering for Light Simulation in Participating Medium. *IEEE Transactions on Visualization and Computer Graphics* 26, 7 (2020).

Laurent Belcour, Ling-Qi Yan, Ravi Ramamoorthi, and Derek Nowrouzezahrai. 2017. Antialiasing Complex Global Illumination Effects in Path-Space. *ACM Trans. Graph.* 36, 4, Article 75b (Jan. 2017), 13 pages.

Xavier Chermain, Simon Lucas, Basile Sauvage, Jean-Michel Dischler, and Carsten Dachsbacher. 2021. Real-Time Geometric Glint Anti-Aliasing with Normal Map Filtering. *Proc. ACM Comput. Graph. Interact. Tech.* 4, 1, Article 1 (April 2021), 16 pages.

Michael F. Cohen, Jonathan Shade, Stefan Hiller, and Oliver Deussen. 2003. Wang Tiles for Image and Texture Generation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2013)* 22, 3 (2003), 287–294.

R. L. Cook and K. E. Torrance. 1982. A Reflectance Model for Computer Graphics. *ACM Trans. Graph.* 1, 1 (Jan. 1982), 7–24.

Franklin C. Crow. 1984. Summed-Area Tables for Texture Mapping. *SIGGRAPH Comput. Graph.* 18, 3 (Jan. 1984), 207–212.

Craig Donner, Jason Lawrence, Ravi Ramamoorthi, Toshiya Hachisuka, Henrik Wann Jensen, and Shree Nayar. 2009. An Empirical BSSRDF Model. In *ACM SIGGRAPH 2009 Papers (SIGGRAPH '09)*. Association for Computing Machinery, Article 30, 10 pages.

Luis E. Gamboa, Jean-Philippe Guertin, and Derek Nowrouzezahrai. 2018. Scalable Appearance Filtering for Complex Lighting Effects. *ACM Transactions on Graphics* 37, 6, Article 277 (Dec. 2018), 13 pages.

L. Ge, B. Wang, L. Wang, X. Meng, and N. Holzschuch. 2019. Interactive Simulation of Scattering Effects in Participating Media Using a Neural Network Model. *IEEE Transactions on Visualization and Computer Graphics* (2019).

Eric Heitz and Fabrice Neyret. 2018. High-Performance By-Example Noise Using a Histogram-Preserving Blending Operator. *Proc. ACM Comput. Graph. Interact. Tech.* 1, 2 (2018), 31:1–31:25.

Frank L. Hitchcock. 1927. The Expression of a Tensor or a Polyadic as a Sum of Products. *Journal of Mathematics and Physics* 6, 1-4 (1927), 164–189.

Wenzel Jakob. 2010. Mitsuba Renderer. http://www.mitsuba-renderer.org/.

Wenzel Jakob, Miloš Hašan, Ling-Qi Yan, Jason Lawrence, Ravi Ramamoorthi, and Steve Marschner. 2014. Discrete Stochastic Microfacet Models. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2014)* 33, 4 (2014).

Anton S. Kaplanyan, Stephen Hill, Anjul Patney, and Aaron Lefohn. 2016. Filtering Distributions of Normals for Shading Antialiasing. In *Eurographics/ ACM SIGGRAPH Symposium on High Performance Graphics*, Ulf Assarsson and Warren Hunt (Eds.). The Eurographics Association. https://doi.org/10.2312/hpg.20161201

Brian Karis. 2013. *Real Shading in Unreal Engine 4.* Technical Report. Epic Games. http://blog.selfshadow.com/publications/s2013-shading-course/karis/s2013_pbs_epic_notes_v2.pdf

Alexandr Kuznetsov, Miloš Hašan, Zexiang Xu, Ling-Qi Yan, Bruce Walter, Nima Khademi Kalantari, Steve Marschner, and Ravi Ramamoorthi. 2019. Learning Generative Models for Rendering Specular Microgeometry. *ACM Trans. Graph.* 38, 6 (2019), 14.

Ares Lagae and Philip Dutré. 2005. A Procedural Object Distribution Function. *ACM Transactions on Graphics* 24, 4 (October 2005), 1442–1461.

Ren Ng, Ravi Ramamoorthi, and Pat Hanrahan. 2004. Triple Product Wavelet Integrals for All-Frequency Relighting. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 477–487.

Gilles Rainer, Abhijeet Ghosh, Wenzel Jakob, and Tim Weyrich. 2020. Unified Neural Encoding of BTFs. *Computer Graphics Forum (Proceedings of Eurographics)* 39, 2 (June 2020).

Gilles Rainer, Wenzel Jakob, Abhijeet Ghosh, and Tim Weyrich. 2019. Neural BTF Compression and Interpolation. *Computer Graphics Forum* 38, 2 (2019), 235–244.

Boris Raymond, Gaël Guennebaud, and Pascal Barla. 2016. Multi-scale Rendering of Scratched Materials Using a Structured SV-BRDF Model. *ACM Transactions on Graphics* 35, 4 (2016), 57:1–57:11.

Peter-Pike Sloan, Jesse Hall, John Hart, and John Snyder. 2003. Clustered Principal Components for Precomputed Radiance Transfer. *ACM Trans. Graph.* 22, 3 (July 2003), 382–391.

Peter-Pike Sloan, Jan Kautz, and John Snyder. 2002. Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments. *ACM Trans. Graph.* 21, 3 (July 2002), 527–536.

Yu-Ting Tsai and Zen-Chung Shih. 2006. All-Frequency Precomputed Radiance Transfer Using Spherical Radial Basis Functions and Clustered Tensor Approximation. *ACM Trans. Graph.* 25, 3 (July 2006), 967–976.

Zdravko Velinov, Sebastian Werner, and Matthias B. Hullin. 2018. Real-Time Rendering of Wave-Optical Effects on Scratched Surfaces. *Computer Graphics Forum (Proc. of EUROGRAPHICS 2018)* 37, 2 (2018).

Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. 2007. Microfacet Models for Refraction through Rough Surfaces. In *Rendering Techniques*. The Eurographics Association.

Beibei Wang, Hong Deng, and Nicolas Holzschuch. 2020a. Real-Time Glints Rendering With Pre-Filtered Discrete Stochastic Microfacets. *Computer Graphics Forum* 39, 6 (2020), 144–154.

Beibei Wang, Miloš Hašan, Nicolas Holzschuch, and Ling-Qi Yan. 2020c. Example-Based Microstructure Rendering with Constant Storage. *ACM Transactions on Graphics* 39, 5, Article 162 (2020), 12 pages.

Beibei Wang, Miloš Hašan, and Ling-Qi Yan. 2020b. Path Cuts: Efficient Rendering of Pure Specular Light Transport. *ACM Trans. Graph.* 39, 6, Article 238 (Nov. 2020), 12 pages.

Beibei Wang, Lu Wang, and Nicolas Holzschuch. 2018. Fast Global Illumination with Discrete Stochastic Microfacets Using a Filterable Model. *Computer Graphics Forum (Proceedings of Pacific Graphics 2018)* 37, 7 (2018), 55–64.

Jiaping Wang, Peiran Ren, Minmin Gong, John Snyder, and Baining Guo. 2009. All-Frequency Rendering of Dynamic, Spatially-Varying Reflectance. *ACM Trans. Graph.* 28, 5 (Dec. 2009), 1–10.

Sebastian Werner, Zdravko Velinov, Wenzel Jakob, and Matthias B. Hullin. 2017. Scratch iridescence: Wave-optical rendering of diffractive surface structure. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2017)* 36, 6 (2017), 207:1–207:14.

Lifan Wu, Shuang Zhao, Ling-Qi Yan, and Ravi Ramamoorthi. 2019. Accurate Appearance Preserving Prefiltering for Rendering Displacement-Mapped Surfaces. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2019)* 38, 4 (2019).

Kun Xu, Wei-Lun Sun, Zhao Dong, Dan-Yong Zhao, Run-Dong Wu, and Shi-Min Hu. 2013. Anisotropic Spherical Gaussians. *ACM Trans. Graph.* 32, 6, Article 209 (Nov.

2013), 11 pages.

Ling-Qi Yan, Miloš Hašan, Wenzel Jakob, Jason Lawrence, Steve Marschner, and Ravi Ramamoorthi. 2014. Rendering Glints on High-Resolution Normal-Mapped Specular Surfaces. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2014)* 33, 4 (2014).

Ling-Qi Yan, Miloš Hašan, Steve Marschner, and Ravi Ramamoorthi. 2016. Position-Normal Distributions for Efficient Rendering of Specular Microstructure. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2016)* 35, 4 (2016).

Ling-Qi Yan, Miloš Hašan, Bruce Walter, Steve Marschner, and Ravi Ramamoorthi. 2018. Rendering Specular Microgeometry with Wave Optics. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2018)* 37, 4 (2018).

Ling-Qi Yan, Weilun Sun, Henrik Wann Jensen, and Ravi Ramamoorthi. 2017. A BSSRDF Model for Efficient Rendering of Fur with Global Illumination. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2017)* 36, 6 (2017).

Tizian Zeltner, Iliyan Georgiev, and Wenzel Jakob. 2020. Specular Manifold Sampling for Rendering High-Frequency Caustics and Glints. *ACM Trans. Graph.* 39, 4, Article 149 (July 2020), 15 pages.

Shuang Zhao, Lifan Wu, Frédo Durand, and Ravi Ramamoorthi. 2016. Downsampling Scattering Parameters for Rendering Anisotropic Media. *ACM Trans. Graph.* 35, 6, Article 166 (Nov. 2016), 11 pages.

Junqiu Zhu, Yanning Xu, and Lu Wang. 2019. A Stationary SVBRDF Material Modeling Method Based on Discrete Microsurface. *Computer Graphics Forum (Proceedings of Pacific Graphics 2019)* 38, 7 (2019), 745–754.

Tobias Zirr and Anton S Kaplanyan. 2016. Real-time rendering of procedural multiscale materials. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games.* ACM, 139–148.

APPENDIX

Proposition: Suppose D is a *rank-1* 2D tensor, i.e., $D = X \otimes Y$. Then the average of D in range $[x_1, x_2] \times [y_1, y_2]$ equals to the product of the average of X and Y in ranges $[x_1, x_2]$ and $[y_1, y_2]$, respectively, viz.

$$\bar{D}([x_1, x_2], [y_1, y_2]) = \bar{X}([x_1, x_2]) \cdot \bar{Y}([y_1, y_2]). \quad (10)$$

Proof: According to the definition of outer product, for rank-1 tensor D, we immediately have

$$D(i, j) = X(i) \cdot Y(j). \quad (11)$$

Denoting $M = x_2 - x_1 + 1$ and $N = y_2 - y_1 + 1$,

$$
\begin{aligned}
&\bar{D}([x_1, x_2], [y_1, y_2]) \\
&= \frac{\sum_{i=x_1}^{x_2} \sum_{j=y_1}^{y_2} D(i, j)}{MN} = \frac{1}{M} \sum_{i=x_1}^{x_2} \left(\frac{1}{N} \sum_{j=y_1}^{y_2} (X(i) \cdot Y(j)) \right) \\
&= \frac{1}{M} \sum_{i=x_1}^{x_2} \left(X(i) \cdot \frac{1}{N} \sum_{j=y_1}^{y_2} Y(j) \right) = \frac{1}{M} \sum_{i=x_1}^{x_2} \left(X(i) \cdot \bar{Y}([y_1, y_2]) \right) \\
&= \bar{Y}([y_1, y_2]) \cdot \frac{1}{M} \sum_{i=x_1}^{x_2} X(i) = \bar{X}([x_1, x_2]) \cdot \bar{Y}([y_1, y_2]). \quad (12)
\end{aligned}
$$

Note that the derivation also trivially extends to higher dimensions.