A Compact Representation of Measured BRDFs Using Neural Processes

CHUANKUN ZHENG*, State Key Lab of CAD&CG, Zhejiang University
RUZHANG ZHENG*, State Key Lab of CAD&CG, Zhejiang University
RUI WANG†, State Key Lab of CAD&CG, Zhejiang University
SHUANG ZHAO, University of California, Irvine
HUJUN BAO†, State Key Lab of CAD&CG, Zhejiang University

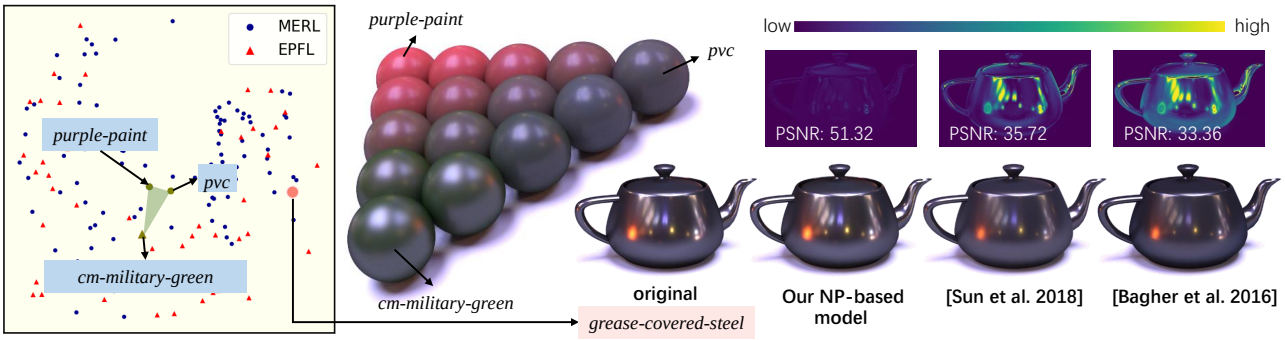


Fig. 1. Our neural process-based approach learns a compact representation from two measured BRDF datasets, MERL and EPFL. BRDFs are encoded with high fidelity. As an example, the reconstructed BRDF *greased-covered-steel* has a much higher accuracy than that of prior methods. Additionally, the learned space preserves traits of BRDFs locally so that we can interpolate existing BRDFs to get new ones.

In this paper, we introduce a compact representation for measured BRDFs by leveraging Neural Processes (NPs). Unlike prior methods that express those BRDFs as discrete high-dimensional matrices or tensors, our technique considers measured BRDFs as *continuous functions* and works in corresponding *function spaces*. Specifically, provided the evaluations of a set of BRDFs, such as ones in MERL and EPFL datasets, our method learns a low-dimensional latent space as well as a few neural networks to encode and decode these measured BRDFs or new BRDFs into and from this space in a nonlinear fashion. Leveraging this latent space and the flexibility offered by the NPs formulation, our encoded BRDFs are highly compact and offer a level of accuracy better than prior methods. We demonstrate the practical usefulness of our approach via two important applications, BRDF compression and editing. Additionally, we design two alternative post-trained decoders to, respectively, achieve better compression ratio for individual BRDFs and enable importance sampling of BRDFs.

*Both authors contributed equally to this research.
†The co-corresponding authors.

Authors' addresses: Chuankun Zheng, ckzheng0320@zju.edu.cn, State Key Lab of CAD&CG, Zhejiang University; Ruzhang Zheng, jaunezheng@gmail.com, State Key Lab of CAD&CG, Zhejiang University; Rui Wang, ruiwang@zju.edu.cn, State Key Lab of CAD&CG, Zhejiang University; Shuang Zhao, shz@ics.uci.edu, University of California, Irvine; Hujun Bao, bao@cad.zju.edu.cn, State Key Lab of CAD&CG, Zhejiang University.

0730-0301/2021/1-ART1 $15.00
https://doi.org/10.1145/3490385

CCS Concepts: • **Computing methodologies** → **Reflectance modeling**.

Additional Key Words and Phrases: Neural Processes, BRDF

ACM Reference Format:
Chuankun Zheng, Ruzhang Zheng, Rui Wang, Shuang Zhao, and Hujun Bao. 2021. A Compact Representation of Measured BRDFs Using Neural Processes. *ACM Trans. Graph.* 1, 1, Article 1 (January 2021), 15 pages. https://doi.org/10.1145/3490385

1 INTRODUCTION

Measured BRDFs faithfully record real-world materials' reflectance profiles and are capable of producing renderings with a remarkable level of fidelity. Consequently, they are used in many real-world applications where visual realism is crucial (e.g., product design and online retail).

As a cost for offering high realism, measured BRDF models are generally data-intensive, making them not only expensive to store and transfer but also difficult to manipulate. Analytical BRDF models, on the contrary, are highly compact and easy to edit but rely on hand-crafted parameterizations that limit their capabilities of accurately reproducing richly diverse appearances of many real-world materials.

To bridge the gap between measured and analytical BRDF models, a few data-driven representations have been introduced for measured BRDFs. These techniques can be broadly put into two categories: *factorization*-based and *fitting*-based.

Factorization-based methods [Bagher et al. 2016; Bilgili et al. 2011; Lawrence et al. 2006, 2004; Matusik et al. 2003a; Nielsen et al. 2015; Tongbuasirilai et al. 2019] treat entire measured BRDF datasets as large matrices (or tensors) and approximate these structures using

low-rank representations obtained using techniques like principal component analysis (PCA). These techniques generally require densely sampled input to work properly. Further, because of the discrete nature of these methods, additional interpolations are typically required to render the resulting BRDFs to avoid visible artifacts.

Fitting-based methods [Bagher et al. 2012; Brady et al. 2014; Holzschuch and Pacanowski 2017; Löw et al. 2012; Pacanowski et al. 2012; Sun et al. 2018], in contrast, use analytic models or manually picked basis functions to fit measured BRDFs. Unfortunately, simple analytic models usually lack the representation power to reproduce detailed reflectance of real-world materials, while sophisticated ones (e.g., those utilizing complex multimodal functions) can make the fitting process unstable or prone to local minima.

In this paper, we introduce a new technique to represent measured BRDFs in a compact and accurate fashion. Unlike prior methods, our representation does not discretize measured BRDFs into matrices or tensors. Instead, we utilize Neural Processes (NPs) to express BRDFs as continuous functions (depicted using neural networks) with learned latent representations. Compared with state-of-the-art methods [Bagher et al. 2016; Hu et al. 2020; Nielsen et al. 2015; Sun et al. 2018], our representation enjoys the advantages of (i) offering higher accuracy, (ii) requiring lower storage, and (iii) retaining a continuous latent space with semantically meaningful structure that facilitates BRDF interpolation.

We demonstrate the practical usefulness of our approach via two important applications: BRDF compression and editing. To this end, we further design two post-trained networks to improve the efficiency of compressing small sets of BRDFs and to enable importance sampling of BRDFs, respectively.

Concretely, our contributions include:

- A new compact representation of measured BRDFs based on the Neural Processes (§3) and an end-to-end pipeline to acquire NP-based representations based on measured BRDF datasets (§4).
- A systematic analysis of the learned latent space of our NP-based representations (§5).
- Post-trained hypernetworks for efficient compression of individual BRDFs (§6) and a new technique for importance sampling according to our NP-based BRDFs (§7).

2 RELATED WORK

BRDF models have been proposed and developed for decades. We refer readers to a comprehensive survey of BRDF representation and acquisition [Guarnera et al. 2016] and only address most relevant works in this section.

2.1 Analytic BRDF Models

Early BRDF models were derived empirically by fitting reflectance data to analytical formulas and were thereby named phenomenological models. Some of the most important phenomenological models are Phong [Phong 1975], Blinn-Phong [Blinn 1977], Ward [Ward et al. 1992], Lafortune [Lafortune et al. 1997], etc. Other types of BRDF models, physically-based models [Cook and Torrance 1982; Jakob et al. 2014; Yan et al. 2014], that are based on physics and optics, were introduced to better capture the roughness at a fine scale. In this work, we represent the BRDF function through neural networks implicitly instead of an analytic form.

2.2 Measured BRDF Datasets

Matusik et al. [2003a] measured 100 real-world isotropic materials, from soft diffuse materials to hard specular materials. These measured BRDFs have been widely used and known as the MERL dataset. Filip and Vávra [2014] constructed the UTIA dataset for anisotropic BRDFs. Lombardi and Nishino [2012] obtained a dataset of objects under natural illumination with calibrated HDR information. Recently, Dupuy and Jakob [2018] employed an adaptive BRDF parameterization and sampling technique to acquire 51 isotropic and 11 anisotropic BRDFs, of which the dataset is named EPFL dataset. In our paper, we focus only on isotropic BRDFs and analyze the distribution of BRDFs across the MERL and EPFL datasets.

2.3 Regression on Measured BRDF Datasets

The development of measured BRDF datasets inspired researchers to improve the representation and model of BRDFs. More precise analytic models [Bagher et al. 2012; Brady et al. 2014; Holzschuch and Pacanowski 2017; Löw et al. 2012; Pacanowski et al. 2012] were proposed to fit on measured datasets. The relation between a specific analytic form, the Lambertian and GGX models, and measured BRDFs was analyzed [Sun et al. 2018]. In our paper, the neural network-based representation better preserves the details in the measured datasets than those analytic models. In §6.3, we compare our work with the latest approach [Sun et al. 2018]. Results show that our method is a level of accuracy better at a low dimension of representation.

Another kind of regressions, BRDF decomposition methods, have been proposed to simplify the structure of measured BRDFs, including PCA decomposition [Matusik et al. 2003a; Nielsen et al. 2015; Serrano et al. 2016], non-negative matrix factorization [Lawrence et al. 2006, 2004], gaussian mixture [Sun et al. 2007], tensor decomposition [Bilgili et al. 2011; Tongbuasirilai et al. 2019] and non-parametric factor model [Bagher et al. 2016]. However, these approaches heavily rely on the discretization of incoming and outgoing directions. Therefore it is intractable for them to handle the EPFL dataset with adaptive and different sampling directions per BRDF. In contrast, our approach does not require certain discretization of directions by treating BRDFs as continuous functions and perform regression in a continuous domain. As a result, our approach not only achieves a compact compression across different measured BRDF datasets but also provides a continuous latent space for BRDF interpolation and exploration.

Soler et al. [2018] proposed a parameterization utilizing Gaussian Process Latent Variable Model (GPLVM) [Lawrence 2005] to embed measured BRDFs into a low dimensional manifold. Their approach shares the basic idea of our work that performs regressions on BRDFs in function space. However, the latent variable model that they used implies an implicitly non-linear mapping from the compact manifold to the BRDFs. Therefore, their approach has to store a full MERL dataset to perform the BRDF interpolation and thereby is unable to compress the data.

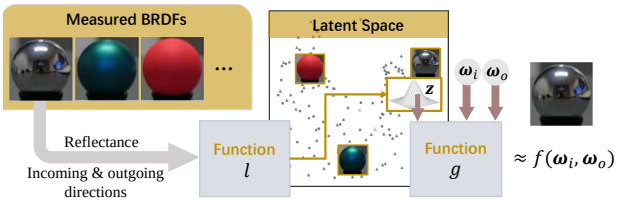


Fig. 2. Overview of our approach. For all BRDFs in measured datasets, we encode observations of one material by function l to the posterior in latent space of BRDF functions, which is learned by NPs. Using a latent vector sampled by posterior, a function g is used to approximate the BRDF reflectance for given incoming directions ω_i and outgoing directions ω_o.

Several recent works start to encode BRDFs through deep neural networks. Many of them [Deschaintre et al. 2018; Gao et al. 2019] focus on an inverse problem that takes an image as input and outputs estimated BRDF (or SVBRDF) parameters. Moreover, Maximov et al. [2019] introduced deep appearance maps by using a small fully connected network to describe a material. Zsolnai-Fehér et al. [2018] uses a neural network to predict the rendering image and preview a static scene fast. Rainer et al. [2019] design a continuous decoder for an autoencoder to compress BTF datasets. However, all of these neural network-based approaches do not target the goal of this paper that computes and explores a latent space to represent measured BRDFs. Recently, Hu et al. [2020] introduced a CNN-based autoencoder to extract a low-dimensional manifold from the MERL dataset. Although this method shares a similar flavor as our technique, it relies on conventional discretized expressions of BRDFs. Our representation is more compact and provides higher reconstruction accuracy, which we will demonstrate in §6.3.

Beyond numerical metrics defined on measured BRDFs, perception metrics were also introduced to derive low-dimensional perceptual embeddings [Lagunas et al. 2019; Matusik et al. 2003a; Serrano et al. 2016]. We consider the selection of error metrics complementary to this work and use L2 loss for its simplicity.

3 OVERVIEW

We now present an overview of our technique. At the core of our technique is a new compact and *probabilistic* representation of measured BRDFs: We regard a BRDF f as a sample drawn randomly from the set $\mathcal{F}$ of all possible BRDFs. We note that, despite being modeled in a stochastic fashion, the evaluation of any BRDF f will remain mostly deterministic (that is, have negligible variance).

To obtain the corresponding probability distribution, we utilize a data-driven approach by consulting a collection of measured BRDFs $\mathcal{F}_M$ treated as observations of f (drawn from $\mathcal{F}$).

Several previous works on stochastic process tackled such a regression problem, such as Gaussian Processes (GPs) [Rasmussen 2003] and recently proposed Neural Processes (NPs) [Garnelo et al. 2018]. Compared to GPs, NPs show better adaptivity to data (as GPs usually require pre-defined kernels, but NPs do not), and are more efficient to train and evaluate.

We utilize NPs to express the distribution of measured BRDFs which, as we will describe later in this paper, can be used for BRDF reconstruction and interpolation. As outlined in Figure 2, we employ a high-dimensional latent vector z and a (neural) function g such that a BRDF f can be approximately expressed as

$$f(\omega_\mathrm{i}, \omega_\mathrm{o}) \approx g(\omega_\mathrm{i}, \omega_\mathrm{o}; z), \tag{1}$$

where ω_i and ω_o are, respectively, the incident and outgoing directions. Further, we represent a collection of measured BRDFs $\mathcal{F}_M := \{f_k : k \in M\}$ with the same function g and a set of latent vectors $\{z_k : k \in M\}$ such that $f_k(\omega_\mathrm{i}, \omega_\mathrm{o}) \approx g(\omega_\mathrm{i}, \omega_\mathrm{o}; z_k)$ for all k.

By treating the latent vectors z_k as observations of some random variable, Eq. (1) allows a set of measured BRDFs $\mathcal{F}_M$ to be modeled in a probabilistic fashion as follows. We assume z_k to be normally distributed with some mean μ_k and covariance Σ_k for each $k \in M$. In this way, modeling $\mathcal{F}_M$ boils down to (i) determining the function g in Eq. (1), and (ii) finding the distribution parameters μ_k and Σ_k for all k.

In practice, a measured BRDF f is typically expressed as a sequence observations $X := (x_1, x_2, \ldots)$ and $Y := (y_1, y_2, \ldots)$ such that x_j encodes the lighting and viewing directions of the j-th observation and $y_j := f(x_j)$ denotes the corresponding BRDF value. Provided a collection of measured BRDFs $\mathcal{F}_M := \{f_k : k \in M\}$, let $X_k := (x_{k,j} : j = 1, 2, \ldots)$ and $Y_k = (y_{k,j} : j = 1, 2, \ldots)$ denote the observations of f_k for all k. Further, let $X_M := (X_k : k \in M)$ and $Y_M := (Y_k : k \in M)$ indicate observations of all the BRDFs. Then, the target likelihood $p(Y_M \mid X_M)$ can be expressed as:

$$p(Y_M \mid X_M) = \prod_{k \in M} \int p(Y_k \mid X_k, z_k)\, p(z_k)\, \mathrm{d}z_k, \tag{2}$$

where $p(Y_k \mid X_k, z_k)$ models the reconstruction error and measurement noise:

$$p(Y_k \mid X_k, z_k) = \mathcal{N}\left(Y_k;\ g(X_k, z_k),\ \Sigma_\mathrm{err}\right), \tag{3}$$

where $\mathcal{N}(\cdot;\ g(X_k, z_k),\ \Sigma_\mathrm{err})$ denotes the pdf of the (high-dimensional) normal distribution with mean $g(X_k, z_k)$ and variance Σ_err. In Eq. (2), the prior of all possible BRDFs $p(z_k)$ is generally difficult to obtain. Thus, we approximate this term using the conditional posterior:

$$p(z_k) \approx q(z_k \mid X_k, Y_k) = \mathcal{N}(z_k;\ \mu_k, \Sigma_k), \tag{4}$$

which follows the aforementioned assumption of z_k being normally distributed. Let l be a function that predicts the distributional parameters μ_k and Σ_k of z_k given the observations X_k, Y_k:

$$l(X_k, Y_k) = (\mu_k,\ \Sigma_k). \tag{5}$$

Then, Eq. (4) becomes

$$q(z_k \mid X_k, Y_k) = \mathcal{N}(z_k;\ l(X_k, Y_k)). \tag{6}$$

To realize the framework outlined in Eqs. (1–6), we express functions g and l using deep neural networks that are trained by *maximizing* the target likelihood of Eq. (2) using a set of training BRDFs $\mathcal{F}_M$.

With properly trained g and l, all measured BRDFs (similar to those in $\mathcal{F}_M$) can be represented in a highly compact fashion. Specifically, let X and Y be the observations of a measured BRDF f and l_μ denote the μ component returned by l (representing the expected value of the corresponding latent vector z). Then, we have

$$f(\omega_\mathrm{i}, \omega_\mathrm{o}; z) \approx g(\omega_\mathrm{i}, \omega_\mathrm{o};\ l_\mu(X, Y)). \tag{7}$$

Our technique enjoys the following properties:

- It provides a low-dimensional latent space in which new BRDFs could be efficiently defined or interpolated.
- The resulting BRDFs are continuous in incident and outgoing direction with good consistency.
- This technique allows the training observations $\boldsymbol{X}_k, \boldsymbol{Y}_k$ to have different sampling patterns across BRDFs, making it easy to combine multiple measured datasets.

In what follows, we provide more details on applying NPs to measured BRDF in our framework.

4 LEARNING THE LATENT SPACE OF BRDFS

We now detail our realization of the Neural Processes (NPs) framework outlined in §3. Figure 3 summarizes our network architectures and the training process.

4.1 Training Data Preparation

Our training process takes as input a set of measured BRDFs where the k-th entry f_k is depicted as a sequence of lighting and viewing directions $\boldsymbol{X}_k := (\boldsymbol{x}_{k,1}, \boldsymbol{x}_{k,2}, \ldots)$ and the corresponding BRDF values $\boldsymbol{Y}_k := (y_{k,1}, y_{k,2}, \ldots)$ such that $y_{k,i} = f_k(\boldsymbol{x}_{k,i})$ for all i. In what follows, we detail how individual $\boldsymbol{x}_{k,i}$ and $y_{k,i}$ are described.

Parameterization of directions. We parameterize the BRDF using the standard half-difference-angle formulation $(\theta_h, \theta_d, \phi_d)$ introduced by Rusinkiewicz [1998] and used by the MERL dataset [Matusik et al. 2003b]. We note that this coordinate enforces the reciprocity constraint because $f(\theta_h, \theta_d, \phi_d) = f(\theta_h, \theta_d, \phi_d + \pi)$. To ensure continuity at $\phi_d = 0$ and $\phi_d = \pi$, we map ϕ_d onto a circle by setting $t_1 = \sin(2\phi_d)$, $t_2 = \cos(2\phi_d)$ and use $\boldsymbol{x} := (\theta_h, \theta_d, t_1, t_2)$ as the input to our networks.

For BRDFs depicted using other parameterizations, such as the one used by the EPFL dataset [Dupuy and Jakob 2018], we convert them into our parameterization using their original sampling patterns. Benefit from the flexibility of the neural process, our method is capable of taking arbitrary discretization of directions as input and, therefore, does not require resampling the input BRDFs.

Multi-log remapping of BRDF values. BRDFs can have very high dynamic ranges, making the learning process overemphasize high-value regions resulting from specular reflections. To address this problem, Nielsen et al. [2015] introduced a log-relative mapping to compress the dynamic range of BRDF data. In practice, a log transformation $\log1p(x) := \log(1 + x)$ between nonnegative real numbers is commonly used when handling high-dynamic-range data [Eilertsen et al. 2017; Zhang and Lalonde 2017]. In our case, we apply the log1p transformation four times to the BRDF values, which provides slightly better results according to our experiments. Please refer to the supplemental materials for the effect of varying numbers of log1p transformations.

4.2 Network Architecture

At the core of our NP-based BRDF model depicted in Eqs. (1–6) are the functions g and l expressed as neural networks. To adapt to BRDFs with varying sampling patterns, it is desired for l to take observations with arbitrary patterns, orders, and numbers of entries.

Algorithm 1 Training Algorithm

function TrainingProcedure (g, a, h)
  Initialize
  while Stopping criterion not reached **do**
    for each material k **do**
      // Select sequences C and T.
      Randomly select N pairs $C = (\boldsymbol{x}_{k,i}, y_{k,i})_{i=1,\ldots,N}$;
      Randomly select M pairs $T = (\boldsymbol{x}_{k,i}, y_{k,i})_{i=1,\ldots,M}$;
      // Update distributions in latent space.
      $(\boldsymbol{\mu}^{\mathrm{c}}, \Sigma^{\mathrm{c}}) = a(h(\boldsymbol{X}^{\mathrm{c}}, \boldsymbol{Y}^{\mathrm{c}}))$;
      $(\boldsymbol{\mu}^{\mathrm{t}}, \Sigma^{\mathrm{t}}) = a(h(\boldsymbol{X}^{\mathrm{t}}, \boldsymbol{Y}^{\mathrm{t}}))$;
      // Compute KL loss.
      $e_{KL} = KL(\mathcal{N}(\boldsymbol{\mu}^{\mathrm{t}}, \Sigma^{\mathrm{t}}) || \mathcal{N}(\boldsymbol{\mu}^{\mathrm{c}}, \Sigma^{\mathrm{c}}))$;
      // Compute predictions in the set T.
      Sample $(\boldsymbol{z}_k) \sim \mathcal{N}(\boldsymbol{\mu}^{\mathrm{t}}, \Sigma^{\mathrm{t}})$;
      $(y'_i)_{i=1,\ldots,M} = g(\boldsymbol{z}_k, \boldsymbol{x}_{k,i})_{i=1,\ldots,M}$;
      $e_{L2} = \frac{1}{2}(\Delta \boldsymbol{Y}_k^{\top} \Sigma_{\mathrm{err}}^{-1} \Delta \boldsymbol{Y}_k)$;
      (g, a, h)=BackPropagate(e_{KL},e_{L2});
    end for
  end while
end function

To this end, we further decompose l into a nonlinear function h that encodes individual observations and a function a that aggregates a set of encoded input observations. That is,

$$l(\boldsymbol{X}, \boldsymbol{Y}) = a(\{h(\boldsymbol{x}_j, y_j) \,:\, j = 1, 2, \ldots\}), \tag{8}$$

where $\boldsymbol{X} := (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots)$ and $\boldsymbol{Y} := (y_1, y_2, \ldots)$ is a sequence of observations of some BRDF f (i.e. $y_j = f(\boldsymbol{x}_j)$). Based on the roles played by h, a, and g, we call h the *encoder*, a the *aggregator*, and g the *decoder*.

As l needs to take observations with arbitrary orders, a is required to provide an order-invariant output for all encoded input observations $\{h(\boldsymbol{x}_j, y_j)\}$. In practice, we use the *mean* function to ensure order-invariance. While other operations, such as *max* and *sum*, are also order-invariant, the *mean* function usually works better. Thanks to the *aggregator* a, the input BRDF observations to our model can use arbitrary discretization, making it convenient to train our model using multiple measured BRDF datasets.

In our practice, all these networks are comprised of fully connected (FC) layers. The *aggregator* network a starts with *mean* operation, followed by additional FC layers to introduce more non-linear mapping and compute the mean and covariance of latent vectors. To guarantee the non-negative values of BRDF reflectance, we apply rectified linear units (ReLU) to the last layer of decoder's output. In Figure 3(b), we illustrate the design of each network. Additionally, in §6 and §7, we present two alternative decoder designs for specific applications. We obtain these decoders via a post-training process based on the learned latent space.

4.3 Training the Networks

The goal of our training process is to maximize the variational posterior given by Eq. (2). Unfortunately, this function is difficult to optimize directly. Instead, we leverage the evidence lower-bound (ELBO) [Kingma and Welling 2013] of the variational posterior,

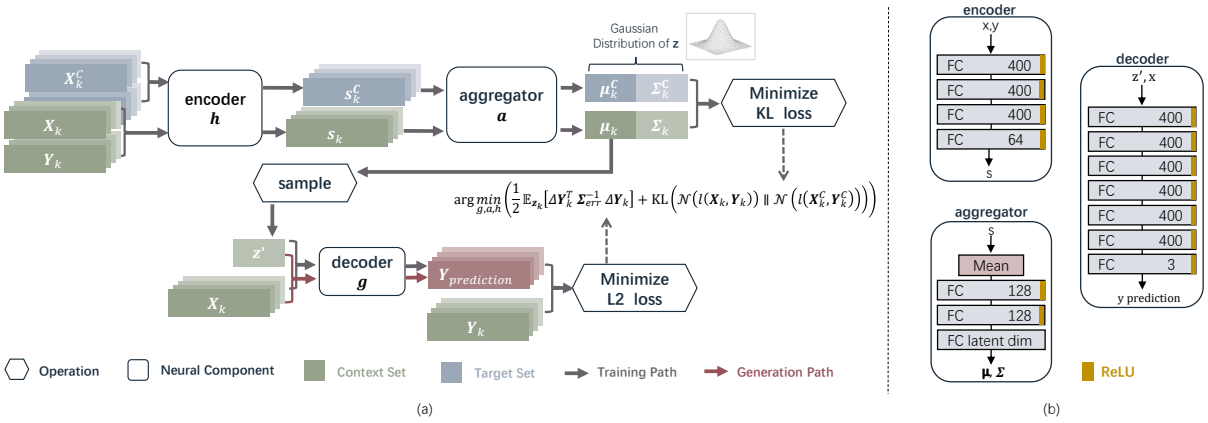


Fig. 3. (a) An architectural view of our model for each BRDF using NPs. While training, the encoder h and aggregator a infer the posteriors over BRDF in function space given sets C and T, and sample a specific latent vector $\boldsymbol{z}'$ from it. We then use $\boldsymbol{z}'$ to predict the y given $\boldsymbol{X}_T$ from the target set T. The model is trained end-to-end to maximize the evidence lower bound (ELBO), which is the sum of $\mathcal{L}_2$ and KL loss. (b) The design of networks used in the model.

which provides a lower bound of $\log p(\boldsymbol{Y}_k \mid \boldsymbol{X}_k)$ for each $k \in M$:

$$\mathbb{E}_{q(\boldsymbol{z}_k \mid \boldsymbol{X}_k, \boldsymbol{Y}_k)}[\log p(\boldsymbol{Y}_k \mid \boldsymbol{X}_k, \boldsymbol{z}_k)] - \mathrm{KL}\left[q(\boldsymbol{z}_k \mid \boldsymbol{X}_k, \boldsymbol{Y}_k) \parallel p(\boldsymbol{z}_k)\right], \quad (9)$$

where $\mathrm{KL}[\cdot \parallel \cdot]$ denotes the Kullback–Leibler divergence between two distributions.

Consider the prior $p(\boldsymbol{z}_k)$ of all possible BRDFs, about which we have little knowledge and therefore cannot access it directly. Moreover, as BRDFs expressing materials with varying albedo and roughnesses can be very different numerically, using a common prior p for all $\boldsymbol{z}_k$ may lead to unsatisfactory results due to unnecessary constraints posed by the common prior. To address this problem, for each $k \in M$, we follow a key idea introduced by neural processes [Garnelo et al. 2018], which is sampling a *context* set of observations $\boldsymbol{X}_k^{\mathrm{c}} := (\boldsymbol{x}_{k,j} : j \in C)$ and $\boldsymbol{Y}_k^{\mathrm{c}} := (y_{k,j} : j \in C)$ for some $C \subset \{1, 2, \ldots\}$.

By forcing the posterior $p(\boldsymbol{z}_k \mid \boldsymbol{X}_k^{\mathrm{c}}, \boldsymbol{Y}_k^{\mathrm{c}})$ of the context set to approximate the prior $p(\boldsymbol{z}_k)$ of the full set, we obtain a new ELBO:

$$\log p(\boldsymbol{Y}_k \mid \boldsymbol{X}_k, \boldsymbol{X}_k^{\mathrm{c}}, \boldsymbol{Y}_k^{\mathrm{c}}) \geq \mathbb{E}_{q(\boldsymbol{z}_k \mid \boldsymbol{X}_k, \boldsymbol{Y}_k)}[\log p(\boldsymbol{Y}_k \mid \boldsymbol{z}_k, \boldsymbol{X}_k)] - \mathrm{KL}\left[q(\boldsymbol{z}_k \mid \boldsymbol{X}_k, \boldsymbol{Y}_k) \parallel p(\boldsymbol{z}_k \mid \boldsymbol{X}_k^{\mathrm{c}}, \boldsymbol{Y}_k^{\mathrm{c}})\right]. \quad (10)$$

In this equation, it is intractable to directly obtain the conditional prior. Thus, the prior is usually approximated with the variational posterior $q(\boldsymbol{z}_k \mid \boldsymbol{X}_k^{\mathrm{c}}, \boldsymbol{Y}_k^{\mathrm{c}})$, causing Eq. (10) to become

$$\log p(\boldsymbol{Y}_k \mid \boldsymbol{X}_k, \boldsymbol{X}_k^{\mathrm{c}}, \boldsymbol{Y}_k^{\mathrm{c}}) \geq \mathbb{E}_{q(\boldsymbol{z}_k \mid \boldsymbol{X}_k, \boldsymbol{Y}_k)}[\log p(\boldsymbol{Y}_k \mid \boldsymbol{z}_k, \boldsymbol{X}_k)] - \mathrm{KL}\left[q(\boldsymbol{z}_k \mid \boldsymbol{X}_k, \boldsymbol{Y}_k) \parallel q(\boldsymbol{z}_k \mid \boldsymbol{X}_k^{\mathrm{c}}, \boldsymbol{Y}_k^{\mathrm{c}})\right]. \quad (11)$$

According to Eqs. (3) and (6), maximizing Eq. (11) further boils down to finding g and l that minimize

$$\frac{1}{2}\mathbb{E}_{\boldsymbol{z}_k}\left[\Delta \boldsymbol{Y}_k^{\top} \Sigma_{\mathrm{err}}^{-1} \Delta \boldsymbol{Y}_k\right] + \mathrm{KL}\left[\mathcal{N}\left(l\left(\boldsymbol{X}_k, \boldsymbol{Y}_k\right)\right) \parallel \mathcal{N}\left(l(\boldsymbol{X}_k^{\mathrm{c}}, \boldsymbol{Y}_k^{\mathrm{c}})\right)\right], \quad (12)$$

where $\Delta \boldsymbol{Y}_k := (y_{k,j} - g(\boldsymbol{x}_{k,j}; \boldsymbol{z}_k) : j = 1, 2, \ldots)$ is a function of $\boldsymbol{z}_k$. At training time, the expected value in Eq. (12) is estimated by randomly drawing $\boldsymbol{z}_k \sim \mathcal{N}\left(l(\boldsymbol{X}_k, \boldsymbol{Y}_k)\right)$ and calculating $\Delta \boldsymbol{Y}_k^{\top} \Sigma_{\mathrm{err}}^{-1} \Delta \boldsymbol{Y}_k$. The KL divergence between two multivariate Gaussians, on the other hand, can be evaluated analytically.

At the high level, the neural process learns to reconstruct $\boldsymbol{Y}_k$ under a regularization that the summary of the *context* set $\boldsymbol{X}_k^{\mathrm{c}}, \boldsymbol{Y}_k^{\mathrm{c}}$ should be similar to that of the full set. Compared to a zero-information prior, a posterior of a subset of the target BRDFs can better reflect the desired behavior. In addition, we train our model with different sizes of contexts, as we will describe later, which encourages our model to be flexible to the size and sampling of the observations at test time.

Training Algorithm. Figure 3 illustrates the training process for our encoder, aggregator, and decoder networks by minimizing Eq. (12) using a collection of input observations. Individual training steps of the training process are further detailed in Algorithm 1.

At each gradient-descent step, we need to calculate the value of Eq. (12) (summed over all input BRDFs) and the corresponding gradients (with respect to the network weights). For each input BRDF with observations $\boldsymbol{X}_k, \boldsymbol{Y}_k$, we evaluate Eq. (12) in a stochastic fashion as follows. First, we randomly select the context set $\boldsymbol{X}_k^{\mathrm{c}} \subset \boldsymbol{X}_k$ and the corresponding $\boldsymbol{Y}_k^{\mathrm{c}} \subset \boldsymbol{Y}_k$. Then, all observation pairs from the context set $\boldsymbol{X}_k^{\mathrm{c}}, \boldsymbol{Y}_k^{\mathrm{c}}$ and the full set $\boldsymbol{X}_k, \boldsymbol{Y}_k$ go through the encoder and the aggregator and yield the distributional parameters $\boldsymbol{\mu}_k^{\mathrm{c}}$, Σ_k^{c} and $\boldsymbol{\mu}_k$, Σ_k, respectively. The KL divergence is then calculated analytically using these parameters.

To estimate the first term of Eq. (12), we randomly draw $\boldsymbol{z}_{k,j} \sim \mathcal{N}(\boldsymbol{\mu}_k, \Sigma_k)$ for each observation pair $\boldsymbol{x}_{k,j}$ from $\boldsymbol{X}_k$ and $y_{k,j}$ from $\boldsymbol{Y}_k$. Then, the L2 difference between $y_{k,j}$ and the network prediction $g(\boldsymbol{x}_{k,j}; \boldsymbol{z}_{k,j})$ is calculated, and the sum of all these L2 differences yields the desired quantity.

As both terms are calculated in a differentiable fashion, the gradients can be easily obtained via automated differentiation.

Implementation Details. In practice, the overhead of the training process can be significant when an input BRDF involves a great number of observation pairs. In this case, we further sample a *target* set $\boldsymbol{X}_k^{\mathrm{t}}, \boldsymbol{Y}_k^{\mathrm{t}}$ which satisfies $\boldsymbol{X}_k^{\mathrm{t}} \subset \boldsymbol{X}_k$ and $\boldsymbol{Y}_k^{\mathrm{t}} \subset \boldsymbol{Y}_k$. This allows the

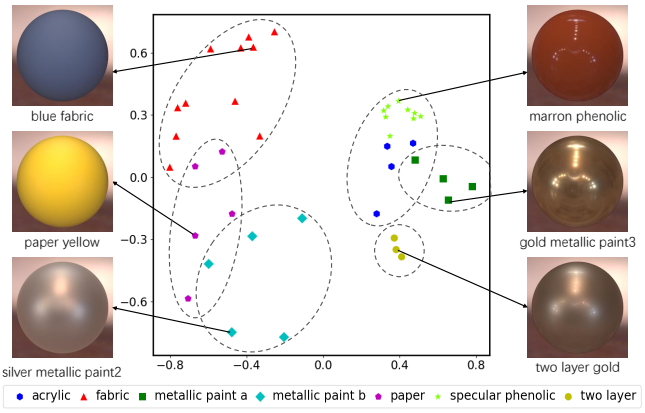


Fig. 4. Projections of several categories of measured BRDF to the two dimension space spanned by first two principal components from a seven dimensions latent space. Different categories of materials can be well distinguished.

calculation of the distribution parameters $\boldsymbol{\mu}_k$, Σ_k as well as the expected value in Eq. (12) to be approximated by using the target set $\boldsymbol{X}_k^{\mathrm{t}}$, $\boldsymbol{Y}_k^{\mathrm{t}}$ instead of the full set.

We train our NPs networks on two measured isotropic BRDF datasets. The MERL dataset [Matusik et al. 2003a] contains 100 materials from soft diffuse materials to hard specular materials. Additionally, we use 51 isotropic BRDFs from the EPFL dataset [Dupuy and Jakob 2018]. The NPs networks were optimized using the Adam method [Kingma and Ba 2014] implemented in TensorFlow [Abadi et al. 2016] with a learning rate of 10^{-4} and a batch size of 16. The entire training takes approximately 40000 iterations in 40 hours on an Nvidia RTX 2080 Ti GPU.

During each training iteration, we choose the size of the *context* set from 1 and 16200 at random and fix the size of the *target* set to 16200. Then, two groups of elements are sampled correspondingly. Lastly, we set $\Sigma_{\mathrm{err}} = 0.2\mathbf{I}$ in Eq. (12), with $\mathbf{I}$ being the identity matrix.

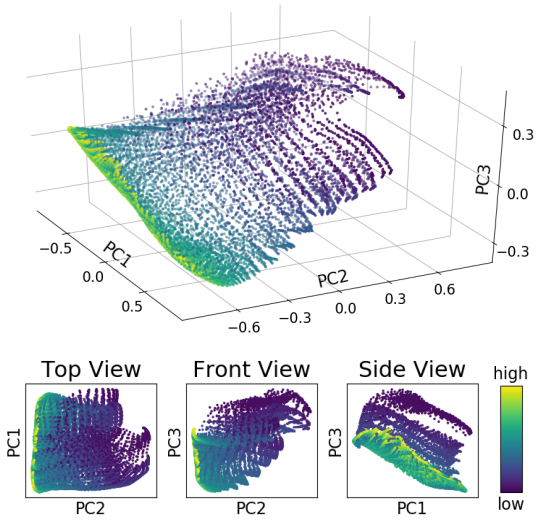
5 ANALYSIS OF LEARNED LATENT SPACE

We apply our neural-process-based technique to measured BRDFs from the MERL [Matusik et al. 2003a] and the EPFL [Dupuy and Jakob 2018] datasets and obtain a latent space for measured BRDFs. Our trained *encoder networks* allow any BRDF (expressed with observation pairs $\boldsymbol{X}$ and $\boldsymbol{Y}$) to be projected into this space.

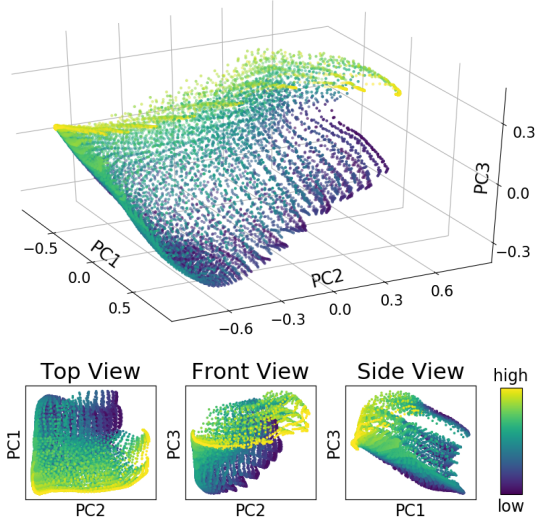
In the following, we analyze the properties of this learned latent space via several experiments. We also develop a web-based tool, which is attached as supplemental material, to visualize this space.

5.1 Dimensionality of the Latent Space

A key parameter to our approach is the dimensionality of the latent space: higher-dimensional spaces generally allow more accurate reconstructions of measured BRDFs. We evaluate the practical implication of this parameter by training multiple networks separately that transform BRDF observations to latent vectors with dimensionalities ranging from two to seven, respectively. Figure 10 shows the average reconstruction accuracy for each of the latent spaces. The results demonstrate that our method can provide high-quality



(a) GGX BRDF projections color-coded using strength of diffuse reflection.



(b) GGX BRDF projections color-coded using surface roughness.

Fig. 5. Color-coded projections of GGX BRDFs in our latent space. The bottom row of each subplot shows orthographic views of the projections. Each axis marked as PC represents one principal component. The smooth color variation in both (a) and (b) indicates that our latent space is well behaved.

reconstructions using even a 2D latent space. However, too low dimensionality makes it difficult for the BRDF encoding to generalize to novel data. This is acceptable for BRDF compression but can be problematic for other applications such as BRDF interpolation. To make our models useful across multiple applications, we use a 7D latent space (which introduce negligible storage overhead compared to the 2D variant) for the rest of the experiments.

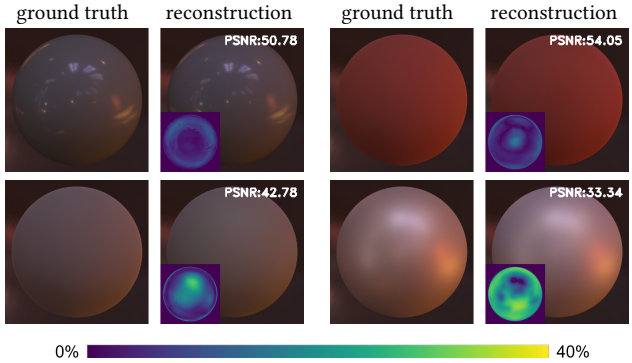


Fig. 6. Reconstruction of novel BRDFs using our NP model. Our model recovers the shape of highlights accurately, however, the color bias may degrade the reconstruction quality (the second row).

5.2 Semantics of the Latent Space

Although our model is trained in a fully data-driven fashion with no semantic input, we do observe the resulting latent spaces to possess semantically meaningful structures. Figure 4 shows a 2D embedding of 7D projections of measured BRDFs from our latent space. In this embedding, we observe that materials with similar appearance tend to form clusters. For instance,

- The "*fabric*" BRDFs—those labeled as "fabrics" from the measured datasets—are located in the upper left;
- The "*paper*" BRDFs are distributed in the lower left;
- The "*metallic-paint-b*" BRDFs, which show glare highlights, lay in the lower middle;
- The "*two-layer*" BRDFs are placed in the lower right;
- The "*metallic-paint-a*" BRDFs, which have sharp highlights without little diffuse reflection, are distributed in the right;
- The "*phenolic*" and "*acrylic*" BRDFs exhibiting sharp highlights with strong diffuse reflection are located in the upper right.

To better understand the overall structure of our latent spaces, we create 16,384 parametric BRDFs each containing a diffuse and a GGX-based specular component [Walter et al. 2007] (that are densely sampled from the parameter space) and project the tabulated version of these BRDFs into our 7D latent space. Figure 5 shows a 3D embedding of the projections. Similar to the 2D embedding example in Figure 4, a material's diffuse reflectivity changes mostly monotonically along the axis marked as PC2 while the roughness varies along PC1, demonstrating that our latent space is smooth and well behaved.

5.3 Accuracy of Stochastic Reconstruction

As depicted in §3, our NP-based method assumes the latent vector z_k to be normally distributed with mean μ_k and covariance Σ_k that are in turn determined (by the aggregator and encoder networks) given a set of observation pairs X_k, Y_k. To reconstruct a BRDF, as expressed in Eq. (7), we simply set z_k to the mean μ_k, making the reconstruction deterministic. The covariance Σ_k, on the other hand, is known to be a good indicator for reconstruction accuracy. In other words, it is desired for Σ_k to be small relative to μ_k (so that the deterministic reconstruction does not deviate greatly from the stochastic model with which the networks are trained).

To evaluate the implication of approximating z_k with its mean μ_k, we perform the *extract* reconstruction by randomly sampling $z_k \sim \mathcal{N}(\mu_k, \Sigma_k)$ and computing the BRDF using Eq. (7) with μ_k replaced by z_k. We repeat this process multiple times for each BRDF from the MERL and EPRL datasets and record the PSNR of renderings obtained using the reconstructed BRDFs. We observe that the variances in PSNR due to the randomness of z_k are negligible (below 0.001) for all materials.

5.4 Generalizability

We project and reconstruct the 306 synthesized BRDFs [Serrano et al. 2016] using our NP model to investigate the generalizability of our representation. As shown in Figure 6, our representation can recover the complex shape of specular highlights accurately. Unfortunately, our representation may struggle with color recovery due to the lack of doucoupling of colors. Please refer to the supplementary material for more results.

Base on the performance of our model on the novel BRDFs, we believe that our model can be directly applied to a small amount of novel data, while fine-tuning can yield better quality. As for the case of a large amount of novel data, adding novel data into the training set and retraining is recommended. Retraining on novel data merely is possible, but MERL and EPFL datasets help to maintain the semantics of the latent space.

5.5 Visualization Tool

We develop a proof-of-concept visualization tool that allows the user to efficiently explore a set of materials characterized by their latent vectors. After importing these vectors, the user is shown a 2D embedding of the corresponding materials (Figure 7). Since materials with similar appearance are generally close to each other in our latent space, our visualization tool can be used for BRDF recommendation. Specifically, when the user clicks a BRDF in the latent space, the tool automatically provides a list of neighboring BRDFs based on the distance in the latent space. In Figure 7, for instance, given a user-selected material, we show the closest, the fourth closest, and a far away BRDF provided by our tool. We can see that the visual similarity indeed decreases with distance.

6 APPLICATION 1: COMPRESSION

We now demonstrate one important application of our Neural-Process-based representation: compressing measured BRDF datasets. This can be done by expressing a BRDF using its low-dimensional (e.g., up to 7D in our case) latent vector (§6.1). Further, as our decoder network g can take more space compared to one BRDF latent vector, this storage overhead can become significant when storing only a small number of BRDFs. To reduce the amortized storage overhead, we post-train light-weight "*hypernetworks*" to decode small sets of BRDFs (§6.2).

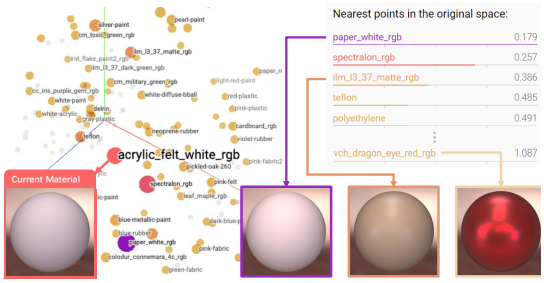


Fig. 7. A visualization of our learned latent space using the developed web-based visualization tool. The left panel displays a 2D embedding of measured BRDFs; and the right interface shows renderings of a set of BRDFs ranked by their latent-space distance from a user-selected BRDF.

6.1 Compressing Full Datasets of Measured BRDFs

By default, our NP-based technique encodes measured BRDFs into a latent space with up to seven dimensions. In this way, the total storage used to store a collection of n measured BRDFs is

$$S_{\text{total}}(n, d) = S_{\text{network}} + n\, d\, S_{\text{dim}}, \tag{13}$$

where S_{network} is the size of the neural network including only the decoder network g; d denotes the dimensionality of the latent space; and S_{dim} indicates the size of each latent vector $\boldsymbol{z} \in \mathbb{R}^d$.

In practice, d ranges from two to seven, and our decoder network takes roughly 3.11 MB of storage (when compressed losslessly [Abadi et al. 2016]). The size of our decoder network dominating the total storage $S_{\text{total}}(n, d)$ (that is, $n\, d\, S_{\text{dim}} \ll S_{\text{network}}$). In other words, the dimensionality d hardly affects the total size of our networks: when going from 2D to 7D, only an extra of 2 KB of storage is needed.

6.2 Compressing Small Sets of BRDFs

Although our technique provides high compression ratio when compressing many (e.g., 100) measured BRDFs, the overhead required for storing our decoder network g can lead to less efficient compression when handling only a small number of BRDFs.

To address this problem, we propose to decompress using post-trained *hypernetworks* [Ha et al. 2016] that are specific to each BRDF and significantly smaller than the full decoder g. Hypernetworks (*hyperNet*) have been widely used in the problem of learning to learn (e.g., meta-learning), which generates the weights of a main neural network (*mainNet*) for a given task.

In our case, we approximate the original decoder g—which takes as input both the latent vector $\boldsymbol{z}$ and the (incident and outgoing) directions $\boldsymbol{x}$—using (i) a *mainNet* that takes the directions $\boldsymbol{x}$ as input and predicts BRDF values, and (ii) a *hyperNet* that takes a latent vector $\boldsymbol{z}$ and generates the weights $\hat{\boldsymbol{z}}$ of the *mainNet*. To be precise,

$$g(\boldsymbol{x}, \boldsymbol{z}) \approx \text{mainNet}(\boldsymbol{x}; \hat{\boldsymbol{z}}), \quad \text{where} \quad \hat{\boldsymbol{z}} = \text{hyperNet}(\boldsymbol{z}). \tag{14}$$

After obtaining the *mainNet*, the *hyperNet* can be discarded, and we only need to generate the *mainNet*'s weights $\hat{\boldsymbol{z}}$ once per BRDF (given its latent vector $\boldsymbol{z}$). Figure 8 shows the architectural details of our *hyperNet* and *mainNet*. Both of them are comprised of FC layers, and the weights $\hat{\boldsymbol{z}}$ of the *mainNet*'s FC layers are computed by the *hyperNet*.

We compute hypernetworks by post-training them in the 7D latent space obtained from measured BRDFs through NPs. To this end, we first randomly sample $\boldsymbol{z}$ from the latent space as input, and then train the networks by supervising the BRDF value output from the original decoder g. The entire training process is performed using the Adam method [Kingma and Ba 2014] in TensorFlow [Abadi et al. 2016] with a learning rate of 10^{-4} and a batch size of 16. Similar to the training of our NP model, we randomly sample directions for each material in a batch rather than computing from all directions. This saves the cost of a single batch (increases the batch size) and improves stability. Hypernetworks are trained for approximately 60000 iterations over the course of 20 hours on an Nvidia RTX 2080 Ti GPU.

Leveraging the learned hypernetworks, we can achieve a very high compression ratio for individual BRDF. Note that the architecture of our *mainNet* is fixed and, thus, does not need to be stored per BRDF. Instead, we only need to store the weights $\hat{\boldsymbol{z}}$ of the *mainNet*, which takes 9 KB per BRDF in our experiments.

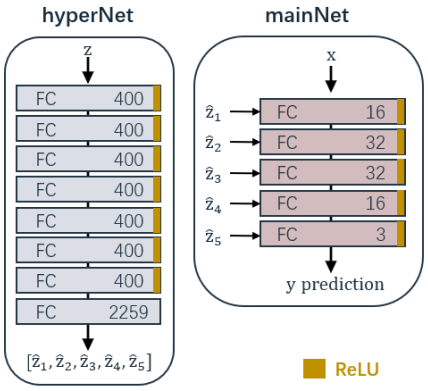


Fig. 8. The design of our *hyperNet* and *mainNet*. The weights of each fully-connected layer of the *mainNet*, which we denote as $\hat{\boldsymbol{z}}$, are generated by the *hyperNet*. In practice, at decompression time, only $\hat{\boldsymbol{z}}$ need to be stored, which greatly reduces overhead.

6.3 Results

To evaluate the visual quality of our compressed BRDFs, we render a sphere using the reconstructed BRDFs under three environmental illuminations (St. Peter's Basilica, Uffizi, and Grace). We use PSNR to quantize the difference between renderings using reconstructed BRDFs and the original (tabulated) ones. The average PSNR is computed from all MERL BRDFs.

We compare our approach with several state-of-the-art BRDF compression approaches [Bagher et al. 2016; Hu et al. 2020; Nielsen et al. 2015; Sun et al. 2018] on the MERL dataset. Figure 9 shows per-BRDF PSNRs, and Figure 10 provides average PSNRs as well as storage comparisons.

Compared to the previous methods, our technique produces higher-quality results for most BRDFs in the MERL dataset—even with a two-dimensional latent space, our method performs on par

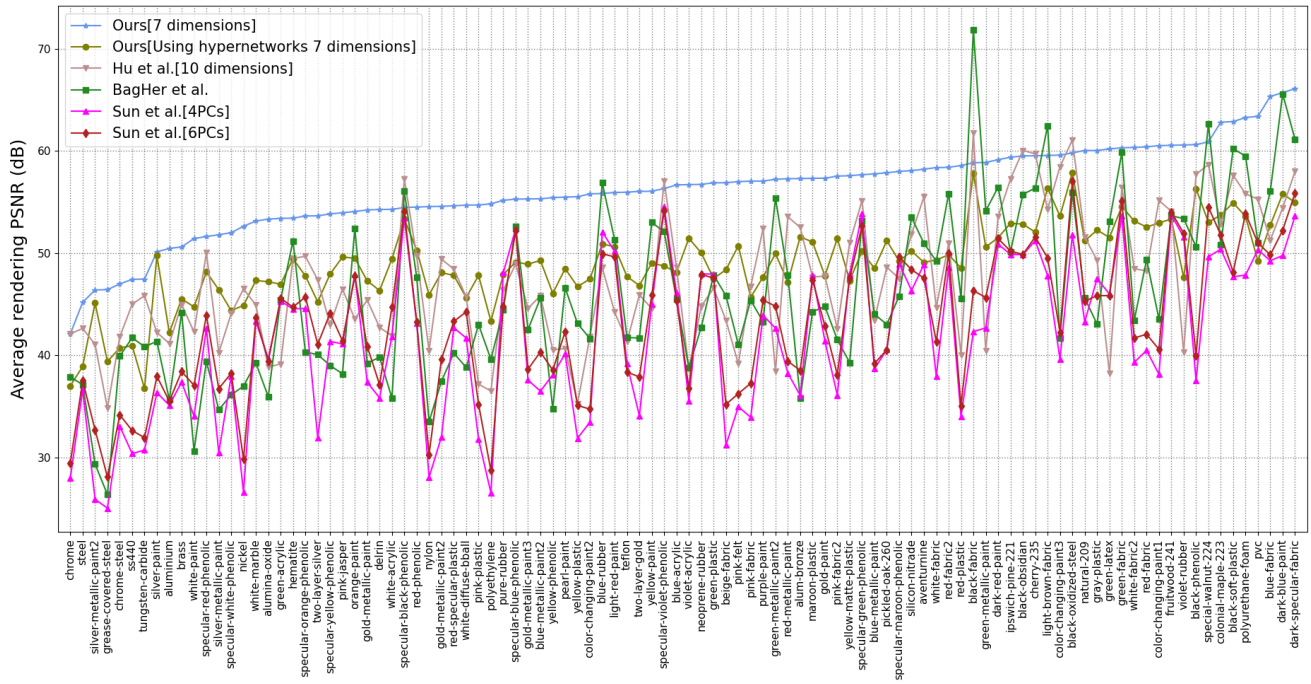


Fig. 9. Reconstruction accuracy measured in PSNR of rendered images. Our model (using a 7D latent space) provides better accuracy than other methods in most cases.

with or better than the state-of-the-art approaches.[1] Further, in these experiments, our model is trained using both MERL and EPFL datasets. We expect training with only the MERL dataset to further improve the performance of our method.

As shown in Figure 10, previous PCA-based compression approaches [Matusik et al. 2003b; Nielsen et al. 2015] require 33 MB per basis. The tensor decomposition approach [Bilgili et al. 2011] requires more than 7 MB for 100 MERL BRDFs. The non-parametric factor model [Bagher et al. 2016], another state-of-the-art method, requires only 3.2 KB per BRDF. However, the quality of reconstructed BRDFs is limited by the functional structure of the microfacet model and cannot be easily improved even with additional storage.

The recent learning-based approach by Hu et al. [2020] requires more than 11 MB to store their decoder. In contrast, the total size of our representation for the MERL dataset is only 3.10 MB. By using hypernetworks to compress individual BRDFs, the storage size can further reduce to 9 KB per BRDF, which is comparable to the work by Bagher et al. [2016]. Although the PSNR value has also reduced to 48.98, the decoder network (i.e., our *mainNet*) is much lighter weighted compared to the original decoder g and, thus, has the advantage of being much less expensive to store and evaluate. In our implementation with TensorRT [Vanholder 2016], it takes 74 ms to decompress a full BRDF (with $180 \times 90 \times 90$ entries) using the original decoder and 3.2 ms to decode the same table using the hypernetworks on an Nvidia RTX 2080 Ti graphics card.

[1]As discussed in §5.1, latent spaces with higher dimensionalities can be better suited for applications beyond BRDF compression as they are generally better-behaved and generalize more easily to BRDFs beyond the training datasets.

In Figure 11, we show renderings of a few reconstructed BRDFs. Compared with previous methods [Bagher et al. 2016; Hu et al. 2020; Sun et al. 2018], our method is more versatile and capable of accurately recovering the complex shapes and colors of specular highlights as well as strong Fresnel reflections. Although there is a small loss of reconstruction quality by using hypernetworks for compression, our reconstructions still well resemble the references. Please refer to the supplementary file for more comparisons of different methods.

We further train our NP model on multiple datasets with different resolutions to investigate the impact of data resolution. Specifically, we uniformly remove a certain percentage of entries from each BRDF (with $180 \times 90 \times 90$ entries). Then we train our NP model on these filtered BRDFs using the same training settings as the original NP model. Table 1 provides average PSNRs of the full-resolution reconstruction at five training resolution settings. Our approach does not introduce a significant quality degradation when 60.3% of entries are filtered out. Even with 90% of entries removed, our approach is still on par with other state-of-art methods.

To evaluate the impact of using our hypernetworks on BRDF interpolation, we randomly select a few pairs of MERL BRDFs and interpolate them using both the original decoder and the hypernetworks. The results demonstrate that our hypernetworks predict interpolated materials similar to our original NP model, as shown in Figure 12.

Method	Setup	Avg. PSNR
Ours	2D	47.15
	3D	53.78
	4D	53.24
	5D	55.68
	6D	53.60
	7D	**56.20**
	7D + hypernetworks	48.98
[Bagher et al. 2016]	non-parametric	45.99
[Hu et al. 2020]	10D	47.48
[Sun et al. 2018]	1 diff PC	28.70
	1 diff + 3 spec PCs	41.61
	1 diff + 5 spec PCs	43.18
	2 lobes GGX	41.30
[Nielsen et al. 2015]	3 PCs	23.30
	6 PCs	34.00
	9 PCs	40.20

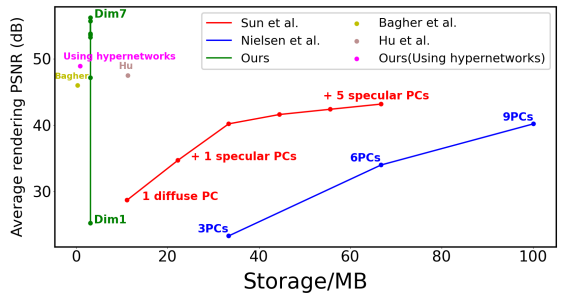


Fig. 10. Comparisons with prior methods [Bagher et al. 2016; Hu et al. 2020; Nielsen et al. 2015; Sun et al. 2018].

Entries	Filter Ratio	Avg. PSNR
(180, 90, 90)	0.0%	56.20
(160, 80, 80)	29.8%	53.78
(133, 66, 66)	60.3%	53.28
(83, 42, 42)	90.0%	48.64
(18, 9, 9)	99.9%	41.83

Table 1. Average PSNRs of the full-resolution BRDFs reconstructed by our NP models trained at five training resolution settings.

7 APPLICATION 2: BRDF EDITING

Besides compressing measured BRDFs (§5), our learned latent space also allows efficient BRDF editing and interpolation. To demonstrate this, we develop a proof-of-concept interactive BRDF editing tool using GPU-based path tracing. In what follows, we first introduce an importance sampling technique specifically developed upon our network-based representation. Then, we show rendered results generated by our tool for BRDF interpolation and exploration.

7.1 BRDF Importance Sampling

Importance sampling is crucial for fast convergence of Monte Carlo rendering. To render our Neural-Process-based BRDFs interactively, our goal is to develop an importance sampling technique that requires minimal preprocessing. In other words, methods relying on

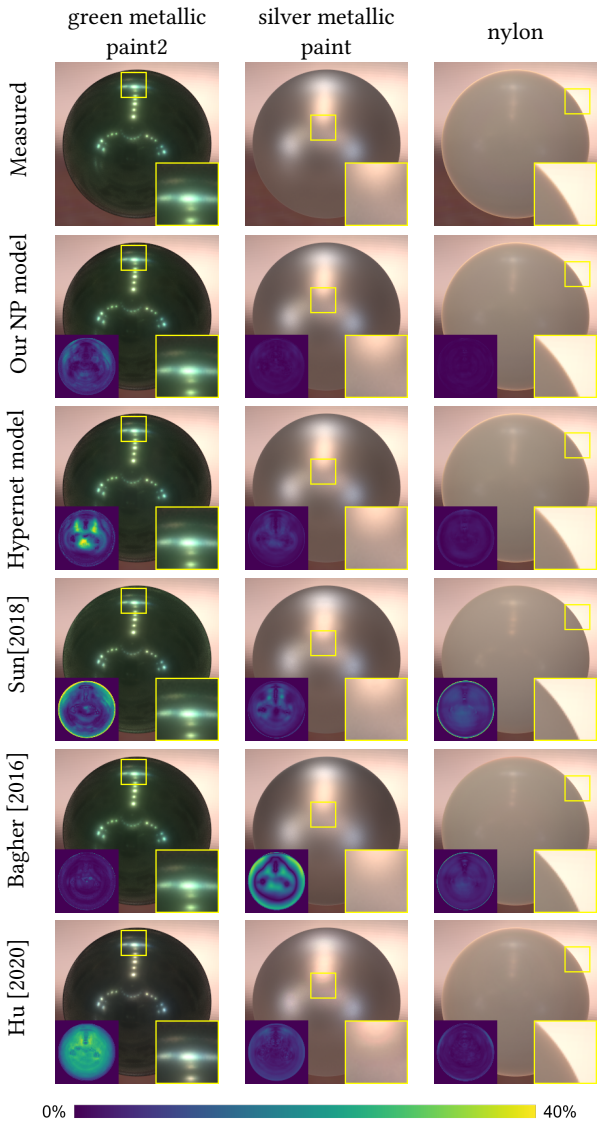


Fig. 11. Comparison with the PCA model [Sun et al. 2018] and factorization approach [Bagher et al. 2016]. Our models (both of the original NP-based model and the hypernetwork model) can accurately recover the color (first column), glossy shape (second column), and the Fresnel effect (third column) of measured BRDFs.

BRDF fitting/decomposition or building summed-area tables should be avoided.

To this end, we leverage Non-linear Independent Components Estimation (NICE) [Dinh et al. 2014, 2016], which allows learning a non-linear deterministic transformation between the target distribution and an easily modeled factorized distribution. In order to maintain the ability to learn complex non-linear transformations, a composition of simple non-linear building blocks is usually employed

$$T = T_{M-1} \circ \ldots \circ T_0, \tag{15}$$

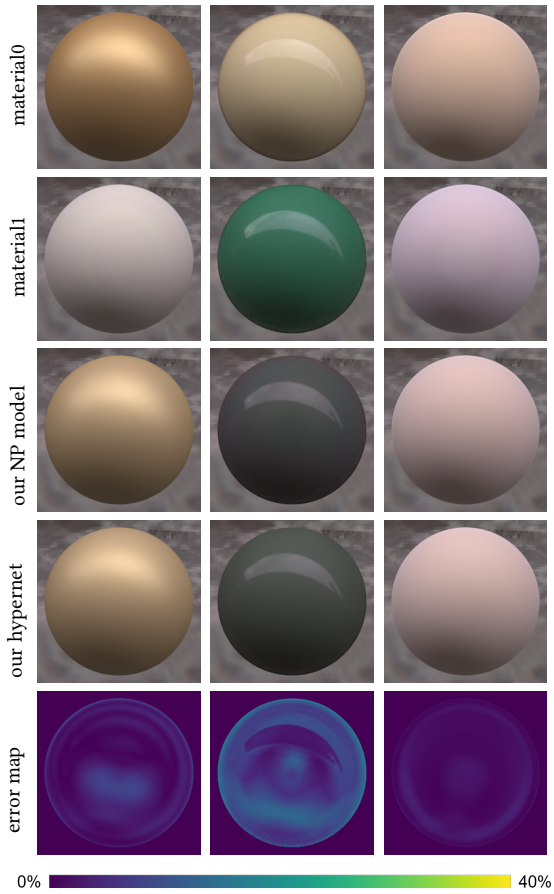


Fig. 12. Difference between interpolations generated with our original model and hypernetworks. The first two rows show the two materials being interpolated, and the rows below show interpolation results.

where, for $0 \le i < M$, T_i, typically referred to as a *coupling layer*, is a specialized network-based bijective transformation, whose Jacobian determinant and inverse transform are easy to compute. Under this framework, to importance sample $\boldsymbol{y}$ with some target PDF, one can first draw $\boldsymbol{x} \sim p_{\boldsymbol{x}}$ from the simple factorized distribution, and then set $\boldsymbol{y} = T^{-1}(\boldsymbol{x})$. In this case, the PDF of $\boldsymbol{y}$ computed via the change-of-variables formula, $p_{\boldsymbol{y}}(\boldsymbol{y}) = p_{\boldsymbol{x}}(T(\boldsymbol{y}))\,|\det(DT)(\boldsymbol{y})|$, should approximately equal the target PDF when T is well learned, where $\det(DT)(\boldsymbol{y})$ denotes the Jacobian determinant of T at $\boldsymbol{y}$.

In our case, we normalize the (cosine-weighted) BRDF[2] to get the target PDF:

$$p(\boldsymbol{\omega}_\mathrm{i}; \boldsymbol{\omega}_\mathrm{o}, \boldsymbol{z}) = \frac{f(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}; \boldsymbol{z}) \cos(\theta_i)}{\int_{\mathbb{H}^2} f(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}; \boldsymbol{z}) \cos(\theta_i)\, \mathrm{d}\boldsymbol{\omega}_\mathrm{i}}. \tag{16}$$

We use NICE to efficiently importance sample the incident direction $\boldsymbol{\omega}_\mathrm{i}$ given any outgoing direction $\boldsymbol{\omega}_\mathrm{o}$ and BRDF latent vector $\boldsymbol{z}$.

Figure 13 shows the architecture of our two-layer NICE. In practice, we importance sample the halfway vector direction instead of the incident direction. Given inputs including the zenith angle θ_o of the outgoing direction $\boldsymbol{\omega}_\mathrm{o}$, the BRDF latent vector $\boldsymbol{z}$, and random

[2]To be precise, we obtain the target PDF by normalizing BRDF luminance.

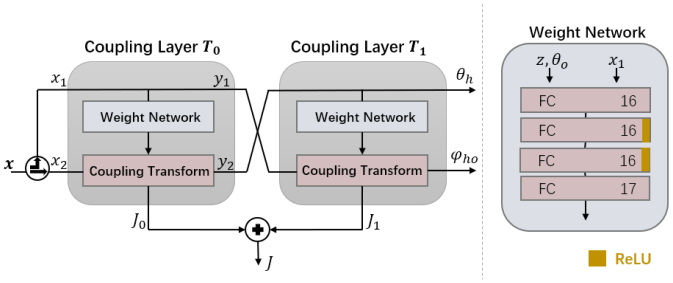


Fig. 13. The two-layer NICE used to importance sample our NP-based BRDFs. We use the same *Coupling Transform* as Müller et al. [2019]. The input $\boldsymbol{x} = (x_1, x_2)$ is a 2D random variable distributed uniformly in $[0, 1)^2$. The output $\boldsymbol{\omega}_\mathrm{h} = (\theta_h, \phi_{ho})$ represents the half-vector sampled from the learned distribution, where ϕ_{ho} is the azimuth angle relative to the outgoing direction. J_i is the logarithmic Jacobian determinant of the coupling layer T_i; θ_o, θ_h, ϕ_{ho} are normalized to between 0 and 1.

samples $\boldsymbol{x}$ drawn uniformly from $[0, 1)^2$, the network generates halfway vector samples from the learned distribution. Benefit from the powerful expressive capacity of the coupling transform module and the continuity of our latent space, the network works well with a small number of parameters. Our NICE network requires roughly 37 KB of storage and takes 8 ms to generate 512×512 samples using TensorRT [Vanholder 2016] on an Nvidia RTX 2080 Ti GPU.

We also employ post-training to obtain T^0 and T^1 from the 7D latent space learned from NPs. Similar to our handling of the hypernetworks discussed in §6, we first randomly sample $\boldsymbol{z}$ from the latent space as input. Next, we sample one θ_o and a set of random directions from the upper hemisphere for each BRDF in a batch, and then train the network by minimizing the Kullback-Leibler (KL) Divergence [Müller et al. 2019]. Thanks to the small scale of the networks involved, the training takes two hours to converge (using a configuration similar to that of the hypernetworks).

We demonstrate the effectiveness of our NICE-based importance sampling by comparing renderings of MERL BRDFs under environmental illuminations with three types of sampling methods: basic cosine-weighted, GGX-based introduced by Sun et al. [2018] (using two lobes), and our technique. As shown in Figure 14, our method produces the best rendering quality under equal samples. In addition, our NICE-based sampling performs equally well on the interpolated BRDFs, which we show in Figure 15. Please refer to the supplemental material for more results.

7.2 BRDF Interpolation and Editing

Our BRDF editing tool allows the user to interactively edit the BRDF of an object in two ways. First, two BRDFs can be interpolated by linearly mixing their corresponding latent vectors. As demonstrated in §5, materials with similar appearance are typically located close to each other in our latent space. Figure 16 shows two interpolation results, where our interpolation allows the shapes of specular highlights and the strength of diffuse reflections to both vary smoothly.

Second, by leveraging the semantically meaningful structure of our latent space, when editing a BRDF, our tool allows the exploration of its neighborhood by moving along "*trait vectors*". We use

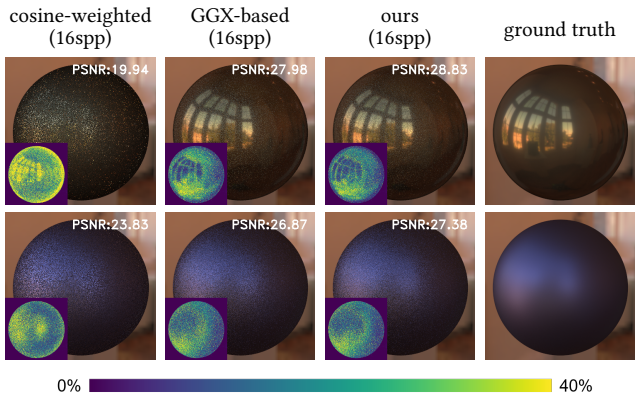


Fig. 14. Renderings of MERL BRDFs using varying importance sampling methods. Compared with cosine-weighted sampling, our approach produces significantly less noise under equal samples. Further, our approach performs slightly better than the GGX-based sampling technique [Heitz 2017] without the need of per-BRDF parameter fitting.

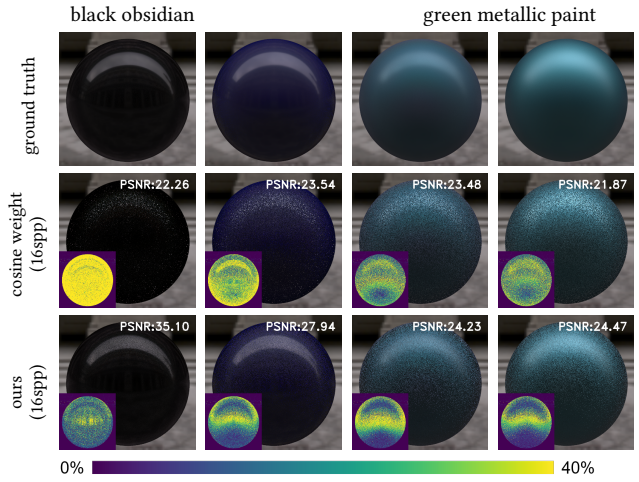


Fig. 15. Renderings of interpolated BRDFs using varying importance sampling methods. Our method significantly outperforms cosine-weighted sampling. We do not include Sun's GGX-based method in this example as it requires per-BRDF fitting and is not well suited for interactive rendering of interpolated BRDFs.

the Support Vector Machine (SVM) in a similar fashion as in prior work [Matusik et al. 2003a] to obtain semantic traits. Specifically, we label each BRDF using 13 perceptual traits proposed by Serrano et al. [2016] including *rubber*, *metallic*, *fabric*, *ceramic*, *soft*, *hard*, *matte*, *glossy*, *bright*, *rough*, *strength of reflections*, *sharpness of reflections*, and *tint of reflections*. For each trait, we compute a hyperplane in our latent space that separates BRDFs with and without the trait. Then, we move along the normal direction of this plane, which we name the *trait vector*, and observe how material appearance changes. We further project trait vectors into a 2D space spanned by the first two principal components of our 7D latent space. As shown in Figure 17, despite being projected from our 7D latent space to a 2D space, the arrangement of these trait vectors

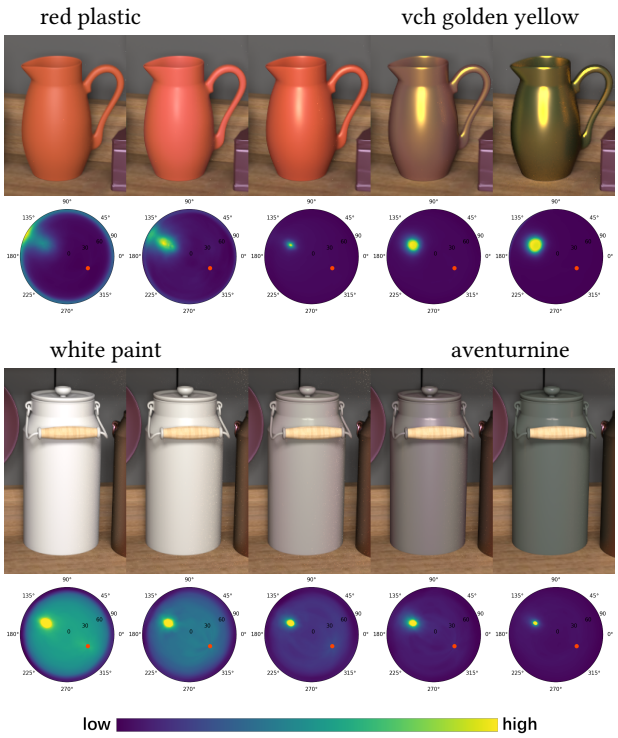


Fig. 16. Interpolation between two measured BRDFs using our representations. For each BRDF, we visualize a slice with a fixed viewing direction (marked in red). As can be seen, such an interpolation allows the shapes of specular highlights and the strength of diffuse reflections to vary smoothly.

remains semantically meaningful. For example, as most *metallic* materials are *glossy*, the corresponding trait vectors are well aligned. As for *glossy* and *fabric*, which generally contradict each other, the trait vectors point to approximately opposite directions.

Figure 18 shows renderings generated using BRDFs obtained by moving along the five trait vectors. The leftmost and rightmost images are rendered using BRDFs at, respectively, the start and end points of trait vector shown in Figure 17. Please see the supplementary video for more BRDF editing results.

8 DISCUSSION AND CONCLUSION

8.1 Limitations and Future Work

Extrapolation in the latent space. The results from §5 have demonstrated that our learned latent space is useful for not only high-quality compression of measured BRDFs but also the interpolation of such BRDFs. However, extrapolated BRDFs that are far away from any training BRDF in the latent space may not preserve any traits of those BRDFs. This is mainly because our key encoding and decoding networks are trained to optimize reconstruction accuracy and may behave poorly for unseen BRDFs that are very different from those used for training. In Figure 19, we show an example of such extrapolated BRDFs.

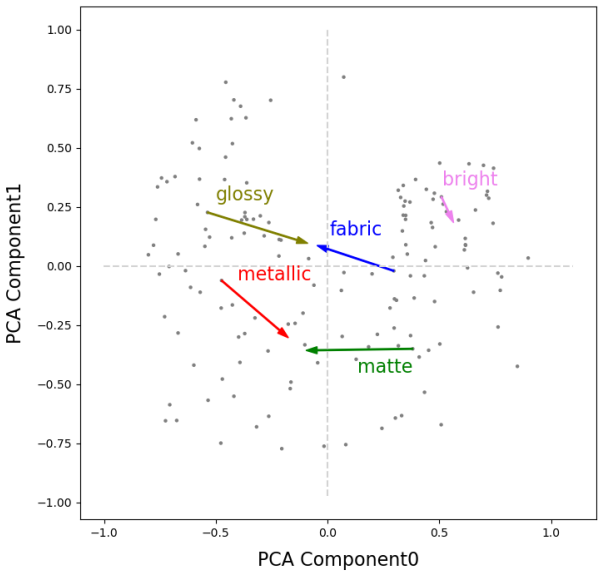


Fig. 17. Projections of trait vectors learned by SVM into the PCA space of our 7D embedding.

Physical plausibility. When developing our technique, we made several design choices to preserve the physical plausibility of the learned BRDFs: we use the half-difference-angle parameterization to enforce reciprocity and employ the rectified linear units (ReLU) at the last layer of the decoder to guarantee non-negativity. On the other hand, similar to most data-driven approaches, it is challenging to theoretically guarantee the conservation of energy through neural networks. In our experiments, we observed that the optimization of Eq. (9) inherently retains high accuracy of reconstructed BRDFs on observed data as well as preserves the traits of BRDFs in interpolation. We did not observe any interpolated BRDFs to violate energy conservation. To further illustrate this, we compute the maximum reflected ratio of the interpolated BRDFs in the latent space:

$$a(z) = \max \left\{ \int_{\Omega} f(\omega_i, \omega_o; z)(\omega_i, \mathbf{n}) d\omega_i \;\middle|\; \omega_o \in \Omega \right\}, \qquad (17)$$

which should be in $[0, 1]$ for the BRDF to conserve energy. This is mostly the case in our experiments, which we show in Figure 20, except for extrapolated BRDFs that are located in the top-right corner of the space and far away from all training samples. We consider a new network architecture that guarantees the energy conservation an interesting future topic.

Colored materials. As we train our Neural-Process-based models with RGB colors baked in, our latent space does not guarantee any separation of diffuse and/or specular albedo, especially at low dimensionalities. The absence of color separation makes our latent space representation color-dependent, which can bring trouble when editing: the color of the interpolated material may not variate as expected, as shown in Figure 21. Also, when applying our model to novel materials, the color bias causes a decrease in reconstruction quality, which limits the generalizability of our representation. We plan to test on more exotic BRDFs in the future , and it will be an interesting topic to explore a new network architecture that decomposes albedos from full reflectance profiles.

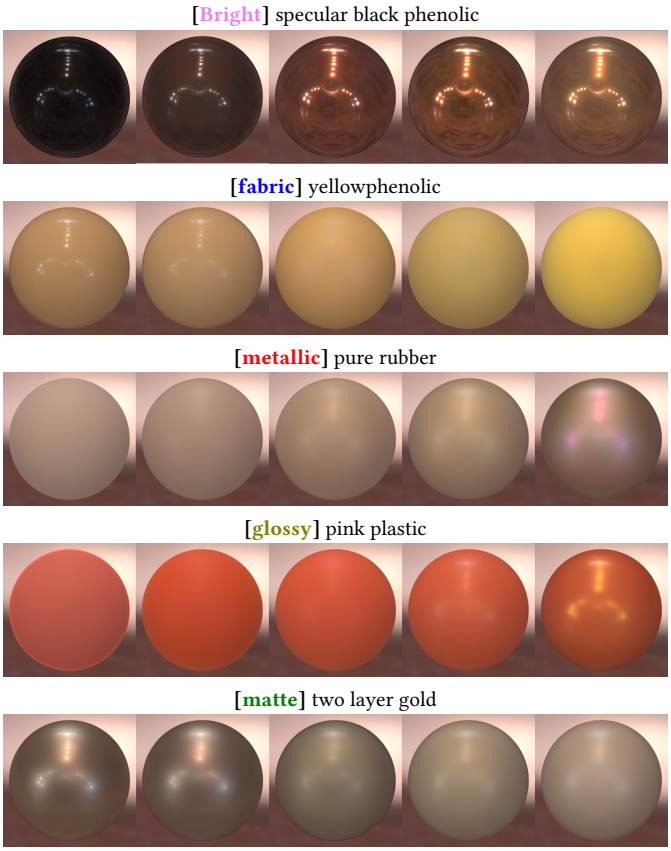


Fig. 18. Visualization of the changes in appearance along each trait vector in Figure 17.

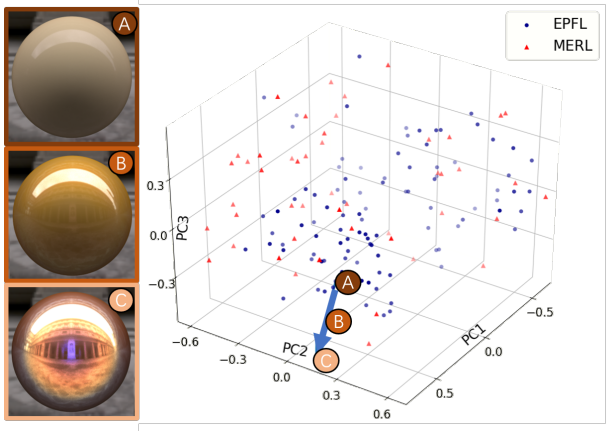


Fig. 19. **Extrapolated BRDFs:** The extrapolation path is shown on the right, and three renderings of extrapolated BRDFs are shown at left. When the BRDFs are far away from all training examples in the latent space, the extrapolated BRDF might not preserve any traits of the training ones.

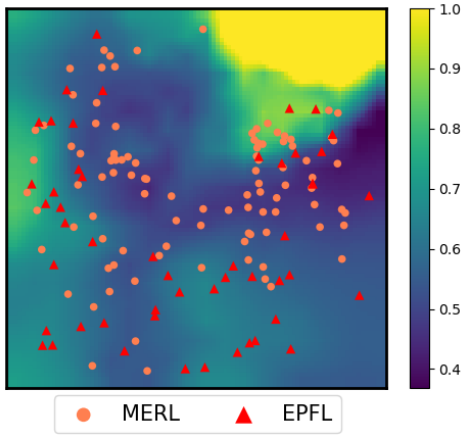


Fig. 20. Visualization of the maximum reflected ratio $a(z)$ of interpolated and extrapolated BRDFs projected into a 2D space spanned by the first two principal components from our 7D latent space. We can see that $a(z)$ is within [0, 1] for the interpolated BRDFs.

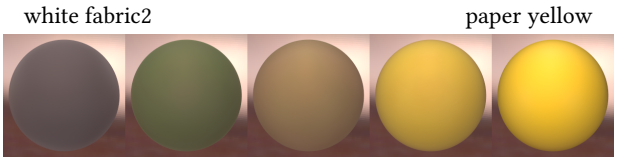


Fig. 21. Interpolation between *white-fabric2* and *paper-yellow*. Since our latent space is color-dependent, the color of the interpolated material may change suddenly and not as expected.

8.2 Conclusion

We presented a new representation for measured BRDFs by leveraging the Neural Processes (NPs). Treating the input BRDFs as continuous functions rather than discrete matrices or tensors, our technique learns a latent space by allowing complex and nonlinear relations, represented with neural networks, between measured BRDFs and their latent representations. This function-level flexibility offers combined compactness and accuracy that has not been achieved before.

We evaluated the effectiveness of our method by providing detailed statistics and comparing to state-of-the-art methods. Further, we demonstrated the practical usefulness of our technique via two applications. Leveraging our latent representation, measured BRDF can be compressed compactly. Using our post-trained hypernetworks, even a small number of BRDFs can be compressed with low storage overhead. We also developed an interactive editing tool integrated with a post-trained importance sampling neural network.

ACKNOWLEDGMENTS

We would like to thank all reviewers for their insightful comments. This research was partially funded by NSFC (No. 61872319), Zhejiang Provincial NSFC (No. LR18F020002), National Key R&D Program of China (No. 2017YFB1002605), and Zhejiang University Education Foundation Global Partnership Fund. Shuang Zhao is grateful to Adobe Research for providing gift fund.

REFERENCES

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 265–283.

Mahdi M Bagher, John Snyder, and Derek Nowrouzezahrai. 2016. A non-parametric factor microfacet model for isotropic brdfs. *ACM Transactions on Graphics (TOG)* 35, 5 (2016), 159.

Mahdi M Bagher, Cyril Soler, and Nicolas Holzschuch. 2012. Accurate fitting of measured reflectances using a Shifted Gamma micro-facet distribution. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 1509–1518.

Ahmet Bilgili, Aydın Öztürk, and Murat Kurt. 2011. A general brdf representation based on tensor decomposition. In *Computer Graphics Forum*, Vol. 30. Wiley Online Library, 2427–2439.

James F Blinn. 1977. Models of light reflection for computer synthesized pictures. In *ACM SIGGRAPH computer graphics*, Vol. 11. ACM, 192–198.

Adam Brady, Jason Lawrence, Pieter Peers, and Westley Weimer. 2014. genBRDF: discovering new analytic BRDFs with genetic programming. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 114.

Robert L Cook and Kenneth E Torrance. 1982. A reflectance model for computer graphics. *ACM Transactions on Graphics (TOG)* 1, 1 (1982), 7–24.

Valentin Deschaintre, Miika Aittala, Fredo Durand, George Drettakis, and Adrien Bousseau. 2018. Single-image svbrdf capture with a rendering-aware deep network. *ACM Transactions on Graphics (ToG)* 37, 4 (2018), 1–15.

Laurent Dinh, David Krueger, and Yoshua Bengio. 2014. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516* (2014).

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2016. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803* (2016).

Jonathan Dupuy and Wenzel Jakob. 2018. An adaptive parameterization for efficient material acquisition and rendering. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 274.

Gabriel Eilertsen, Joel Kronander, Gyorgy Denes, Rafał K Mantiuk, and Jonas Unger. 2017. HDR image reconstruction from a single exposure using deep CNNs. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 178.

Jirí Filip and Radomír Vávra. 2014. Template-based sampling of anisotropic BRDFs. In *Computer Graphics Forum*, Vol. 33. Wiley Online Library, 91–99.

Duan Gao, Xiao Li, Yue Dong, Pieter Peers, Kun Xu, and Xin Tong. 2019. Deep inverse rendering for high-resolution SVBRDF estimation from an arbitrary number of images. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 134.

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. 2018. Neural processes. *ICML 2018 workshop on Theoretical Foundations and Applicatons of Deep Generative Models* (2018).

Darya Guarnera, Giuseppe Claudio Guarnera, Abhijeet Ghosh, Cornelia Denk, and Mashhuda Glencross. 2016. BRDF representation and acquisition. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 625–650.

David Ha, Andrew Dai, and Quoc V Le. 2016. Hypernetworks. *arXiv preprint arXiv:1609.09106* (2016).

Eric Heitz. 2017. A Simpler and Exact Sampling Routine for the GGX Distribution of Visible Normals. (2017).

Nicolas Holzschuch and Romain Pacanowski. 2017. A two-scale microfacet reflectance model combining reflection and diffraction. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–12.

Bingyang Hu, Jie Guo, Yanjun Chen, Mengtian Li, and Yanwen Guo. 2020. DeepBRDF: A Deep Representation for Manipulating Measured BRDF. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 157–166.

Wenzel Jakob, Miloš Hašan, Ling-Qi Yan, Jason Lawrence, Ravi Ramamoorthi, and Steve Marschner. 2014. Discrete stochastic microfacet models. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 115.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

Eric PF Lafortune, Sing-Choong Foo, Kenneth E Torrance, and Donald P Greenberg. 1997. Non-linear approximation of reflectance functions. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 117–126.

Manuel Lagunas, Sandra Malpica, Ana Serrano, Elena Garces, Diego Gutierrez, and Belen Masia. 2019. A similarity measure for material appearance. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–12.

Jason Lawrence, Aner Ben-Artzi, Christopher DeCoro, Wojciech Matusik, Hanspeter Pfister, Ravi Ramamoorthi, and Szymon Rusinkiewicz. 2006. Inverse shade trees for non-parametric material representation and editing. *ACM Transactions on Graphics*

(TOG) 25, 3 (2006), 735–745.

Jason Lawrence, Szymon Rusinkiewicz, and Ravi Ramamoorthi. 2004. Efficient BRDF importance sampling using a factored representation. In *ACM Transactions on Graphics (TOG)*, Vol. 23. ACM, 496–505.

Neil Lawrence. 2005. Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *Journal of machine learning research* 6, Nov (2005), 1783–1816.

Stephen Lombardi and Ko Nishino. 2012. Reflectance and natural illumination from a single image. In *European Conference on Computer Vision*. Springer, 582–595.

Joakim Löw, Joel Kronander, Anders Ynnerman, and Jonas Unger. 2012. BRDF models for accurate and efficient rendering of glossy surfaces. *ACM Transactions on Graphics (TOG)* 31, 1 (2012), 9.

Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. 2003a. A Data-Driven Reflectance Model. *ACM Transactions on Graphics* (2003).

Wojciech Matusik, Hanspeter Pfister, Matthew Brand, and Leonard McMillan. 2003b. Efficient isotropic BRDF measurement. (2003).

Maxim Maximov, Laura Leal-Taixe, Mario Fritz, and Tobias Ritschel. 2019. Deep appearance maps. In *Proceedings of the IEEE International Conference on Computer Vision*. 8729–8738.

Thomas Müller, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. 2019. Neural importance sampling. *ACM Transactions on Graphics (TOG)* 38, 5 (2019), 1–19.

Jannik Boll Nielsen, Henrik Wann Jensen, and Ravi Ramamoorthi. 2015. On optimal, minimal BRDF sampling for reflectance acquisition. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 186.

Romain Pacanowski, Oliver Salazar Celis, Christophe Schlick, Xavier Granier, Pierre Poulin, and Annie Cuyt. 2012. Rational brdf. *IEEE transactions on visualization and computer graphics* 18, 11 (2012), 1824–1835.

Bui Tuong Phong. 1975. Illumination for computer generated pictures. *Commun. ACM* 18, 6 (1975), 311–317.

Gilles Rainer, Wenzel Jakob, Abhijeet Ghosh, and Tim Weyrich. 2019. Neural btf compression and interpolation. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library, 235–244.

Carl Edward Rasmussen. 2003. Gaussian processes in machine learning. In *Summer School on Machine Learning*. Springer, 63–71.

Szymon M Rusinkiewicz. 1998. A new change of variables for efficient BRDF representation. In *Rendering techniques' 98*. Springer, 11–22.

Ana Serrano, Diego Gutierrez, Karol Myszkowski, Hans-Peter Seidel, and Belen Masia. 2016. An intuitive control space for material appearance. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 186.

Cyril Soler, Kartic Subr, and Derek Nowrouzezahrai. 2018. A Versatile Parameterization for Measured Material Manifolds. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 135–144.

Tiancheng Sun, Henrik Wann Jensen, and Ravi Ramamoorthi. 2018. Connecting measured BRDFs to analytic BRDFs by data-driven diffuse-specular separation. In *SIGGRAPH Asia 2018 Technical Papers*. ACM, 273.

Xin Sun, Kun Zhou, Yanyun Chen, Stephen Lin, Jiaoying Shi, and Baining Guo. 2007. Interactive relighting with dynamic BRDFs. *ACM Transactions on Graphics (TOG)* 26, 3 (2007), 27.

Tanaboon Tongbuasirilai, Jonas Unger, Joel Kronander, and Murat Kurt. 2019. Compact and intuitive data-driven BRDF models. *The Visual Computer* (2019), 1–18.

Han Vanholder. 2016. Efficient Inference with TensorRT.

Bruce Walter, Stephen R Marschner, Hongsong Li, and Kenneth E Torrance. 2007. Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*. Eurographics Association, 195–206.

Gregory J Ward et al. 1992. Measuring and modeling anisotropic reflection. *Computer Graphics* 26, 2 (1992), 265–272.

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 (TOG)* 33, 4 (2014), 116.

Jinsong Zhang and Jean-François Lalonde. 2017. Learning high dynamic range from outdoor panoramas. In *Proceedings of the IEEE International Conference on Computer Vision*. 4519–4528.

Károly Zsolnai-Fehér, Peter Wonka, and Michael Wimmer. 2018. Gaussian material synthesis. *ACM Transactions on graphics (TOG)* 37, 4 (2018), 76.