Technical Report

UCAM-CL-TR-1005
ISSN 1476-2986

Number 1005

UNIVERSITY OF CAMBRIDGE

Computer Laboratory

Neural representation of Bidirectional Reflectance Distribution Function

Zheyuan Hu

March 2026

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

https://www.cl.cam.ac.uk/

This technical report is based on a dissertation submitted May 2025 by the author for the degree of Bachelor of Arts (Computer Science Tripos) to the University of Cambridge, Magdalene College.

The author was supervised by Dr Chenliang Zhou and Dr Alejandro Sztrajman.

Technical reports published by the University of Cambridge Computer Laboratory are freely available via the Internet:

https://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986
DOI *https://doi.org/10.48456/tr-1005*

Abstract

Despite the advent of neural rendering, specifically supervised models trained on Bidirectional Reflectance Distribution Function (BRDF) for appearance modelling, there is limited understanding of their effectiveness, efficiency, and utility for downstream research.
In this work, I designed implicit neural representations of BRDFs and evaluated their real-world performance relative to classical models, with extensions to sparse-sample reconstruction and multi-modal material synthesis. I further investigated importance sampling strategies in the rendering pipeline, alongside both supervised and generative methods.
Building upon these contributions, I implemented a novel multi-modal generative pipeline and proposed new quantitative metrics for material synthesis, addressing a long-standing gap in the evaluation of neural materials.

Keywords: BRDF, material appearance modelling, implicit neural representations, generative models, reconstruction, rendering, importance sampling.

Acknowledgements

I'm deeply indebted to my supervisors, Dr Chenliang Zhou and Dr Alejandro Sztrajman, for introducing me into the exciting field of material appearance modelling and neural rendering. Their insightful guidance and consistent feedback have been instrumental in shaping my research journey.

My sincere thanks go to University Teaching Officer (UTO) Professor Cengiz Öztireli, anonymous thesis markers, examiners Professor Sean Holden and Professor Andrew Moore for their thorough marking followed by a viva review and for honoring this work as the Highest Scoring Undergraduate Dissertation.

Appreciation extends to Dr Markus Kuhn, the technical report editor, for his constructive, detailed guidance that refined this thesis during the submission process. Without valuable feedback from the anonymous reviewers during earlier versions of this work submitted to workshops and conferences, this thesis would not have reached its current form. I am grateful for their time and effort.

Deep gratitude is owed to my Director of Studies, Dr John Fawcett, for his invaluable suggestions and insights throughout my undergraduate studies. Finally, special thanks to my fellow students, friends and familiy members for their unwavering support and encouragement throughout this journey. Their belief in me has been a continuous source of motivation.

Contents

List of Figures

List of Tables

1 Introduction

There are things known and unknown, and in between are the doors of perception.

— Aldous Huxley

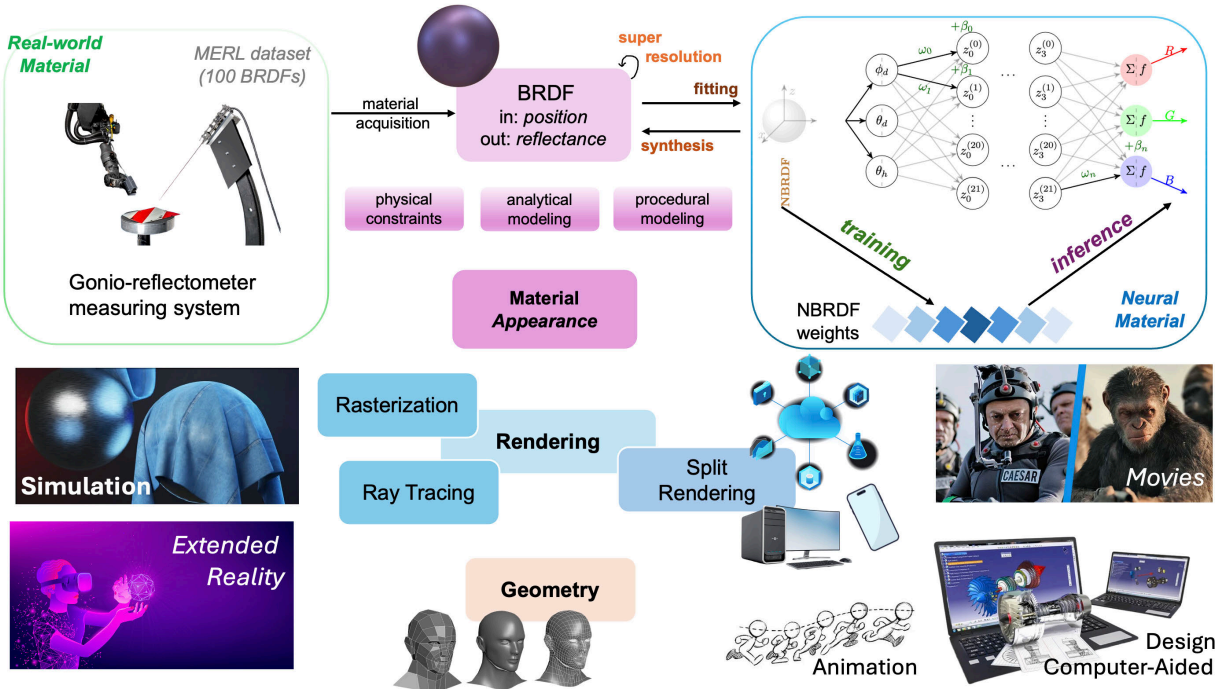


Figure 1.1: Material apperance and its relation to Computer Graphics and applications.

§ 1.1 Motivation

With the deployment of techniques like Ray Tracing (RT) and Global Illumination (GI), modern GPUs are capable of rendering astonishingly photorealistic content. The basic rendering pipeline involves geometry modelling, lighting and texture. However, the gap between the real-time computer-generated and photographic images remains. One remedy is the improved Bidirectional Reflectance Distribution Function (BRDF), which describes the interaction of light with a point on a geometric surface and is widely used in Physically-Based Rendering (PBR).

For material fitting, analytical [Pho98, Bli77, LFTG97] and measured [MPBM03] are two main categories of BRDFs. Each model trade-offs material fidelity, performance (i.e. time and memory) costs and ease of understanding and deployment, which are elaborated upon in the next section. Recent implicit neural BRDF [SRRW21] may open up new possibilities. I propose a MLP-based neural representation to fit the BRDF. Based on experiment results, it maintains an effective balance between quality and memory efficiency.

For generation tasks, diffusion models have shown tremendous potential in various domains, which is likely to excel in material generation as well. However, one bottleneck lies in the datasets, where the acquisition complexity poses challenges for gathering more data and leads

to significant memory consumption. Another challenge is effective evaluation of the generated BRDFs, which is rarely discussed in the literature.

In this work, I propose suitable approaches to address these challenges. Instead of learning on the tabulation data or reconstructing via encoder-decoder architecture, I develop hyperdiffusion on the neural implicit BRDF, a new framework for unconditional and conditional BRDF generation. Hyperdiffusion applies diffusion process over an implicit, concise data-driven BRDF representations, which demonstrates superior performance compared to the alternatives.

§ 1.2 Existing work

There are mainly three categories of BRDF models, namely analytical, data-driven and neural models (Figure 1.2). However, Graphics has not yet arrived at an agreed ideal model.

Developed from an early stage, the **analytical** approach incorporates phenomenological [Lam60, Pho98, Bli77, LFTG97] or physical rules [CT82]. With few tunable parameters, these methods are time and memory efficient. However, due to simplification or approximation, reflectance behaviours are limited using most analytical solutions.

In contrast, the **data-driven** approach [MPBM03], mapping from spherical coordinates to reflected light radiance, yields more realistic rendering results than the analytical approach. Nevertheless, the large data volume[1] needs significant compression (e.g. via PCA) for most real-world use cases.

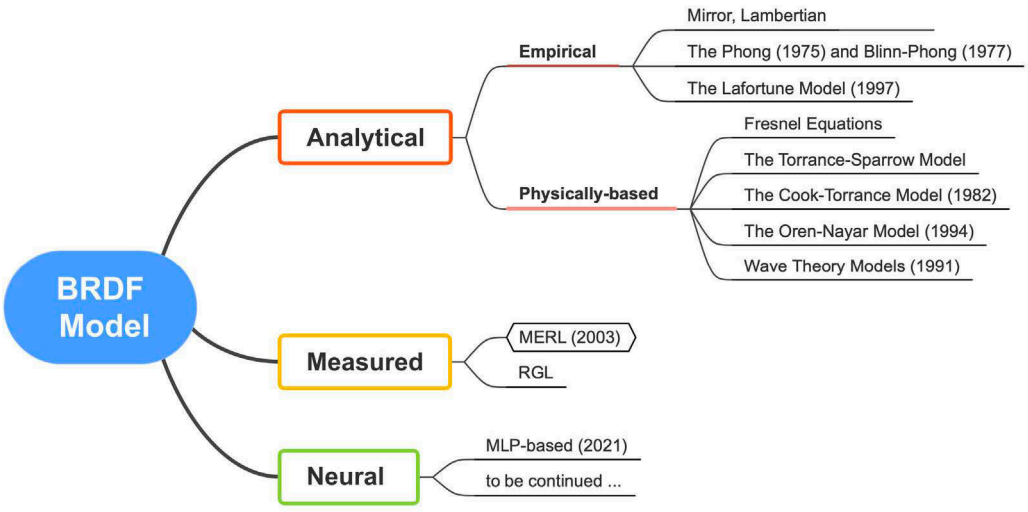


Figure 1.2: An overview of BRDF models.

Recent advances in machine learning bring a novel perspective on material fitting. Neural networks show the potential to compress data-driven measured BRDF effectively, forming the Neural-BRDF (NBRDF) model [SRRW21]. It demonstrates both high fidelity and a low memory footprint. The core aspect of this project is the implementation of the rendering pipeline and NBRDF representation, with various applications and evaluations.

The effectiveness, correctness and applicative utility of the NBRDF representation are still open questions in the graphics field. In addition, despite diffusion models generative ability in image [HJA20, SFH+25], audio [KgLG+24, KPH+21], and text [NDR+22, ZPX+24] domains, it is yet

[1] For example, each BRDF in the MERL dataset is stored using $90 \times 90 \times 180 \times 3$ floating point numbers.

to be explored and its potential in the data-driven material domain remains untapped. Therefore, I further explore application tasks (sparse BRDF super-resolution, multi-modal material synthesis) in this project.

§ 1.3 Aims

This project aims to explore the NBRDF representation and evaluate its correctness, effectiveness and utility.

Core aims

1. To complete the *groundwork* rendering pipeline and a neural BRDF model, i.e.
   (a) to obtain a suitable measured BRDF dataset and apply proper data processing pipeline.
   (b) to integrate the renderer BRDF plugins with the selected BRDF `binary` format, understand and implement different *importance sampling strategies*, such as the uniform hemisphere, Blinn-Phong and microfacet-based.
   (c) to design and implement neural BRDF, mapping from spherical positional inputs to RGB reflectance outputs, and implement the training workflow from scratch, while maintaining the capability of the *Pytorch Lightning* module.
2. To evaluate **correctness** by comparing neural BRDF model correctness (e.g. PSNR and SSIM) against the data-driven ground truth quantitatively and qualitatively.
3. To evaluate **effectiveness** in terms of time and/or memory consumption of different material models.

Extensions

Building upon the core, I explore underlying theories and evaluate its **utility for research** on various applications, including material reconstruction using low-density positional samplings (**super-resolution**), unconditional and conditional **material synthesis**.

1. **Augment** the scarce measured dataset (100 materials in MERL) for BRDF synthesis.
2. **Domain-specific** evaluation metrics in material synthesis, filling in the gap of the research community.
3. Extend to analytical BRDF, beyond data-driven representation, and investigate more advanced microfacet-based importance sampling theories.
4. Further explore the theories behind, e.g. generative methods and hyper-networks, the neural networks that generate weights for another target neural network.

With the aforementioned explored, new possibilities emerge for effective real-time rendering with indistinguishable photorealistic graphics.

2 Preparation

Fortune favors the prepared mind. — Louis Pasteur

This chapter begins with the background theory, including radiometry, material and Bidirectional Reflectance Distribution Function (BRDF), Monte Carlo sampling and rendering. On top of that, many data science approaches are introduced and subsequently applied in different tasks. The chapter proceeds to the requirement analysis and community contribution.

§ 2.1 Background theory

This section attempts to give a systemic overview of scattering, in particular reflective behaviors. Firstly, the scope of discussion and common terminology of materials would be defined. Then, I introduce BRDF and its properties, acquisition and representations. Finally, several data science approaches are introduced for subsequent material fitting, analysis, clustering and generation implementations. K-means clustering is for material statistical analysis and clustering (§ 3.6). PCA is adopted as a material fitting baseline (§ 3.1.1). For material synthesis (§ 3.5.3), PCA, VAE and diffusion models are deployed and compared.

§ 2.1.1 Geometry and material

The **material** focuses on the object characteristics regardless of location $\vec{x}$ on the *geometry* [TAM08]. Examples would be "rubber", "glass", etc. The idea of material captures the key point of the scattering problem. To be more general, the material could also be defined as a function of the position, termed spatially-varying material, while this dissertation focuses on the spatially-invariant material.

§ 2.1.1.1 Types of scattering

Scattering describes all the local interactions of departure light (with direction $\vec{\omega}_o$) with objects, after incident light (with direction $\vec{\omega}_i$) arriving at the surface of the geometry [TAM08]. It's captured by the Bidirectional Scattering Distribution Function (BSDF) $f_s(\vec{\omega}_i, \vec{\omega}_o)$. There are two types of scattering, namely reflective and transmissive.

Reflective measures the light scattered in the upper hemisphere, i.e. $\vec{\omega}_o \cdot \hat{n} \geq 0$. It's captured by the Bidirectional Reflectance Distribution Function (BRDF) $f_r(\vec{\omega}_i, \vec{\omega}_o)$. **Transmissive** measures the light scattered in the lower hemisphere, i.e. $\vec{\omega}_o \cdot \hat{n} \leq 0$. It's captured by the Bidirectional Transmittance Distribution Function (BTDF) $f_t(\vec{\omega}_i, \vec{\omega}_o)$.

The relationship is then defined as, **Scattering = Reflectance + Transmittance**, based on the above definition. Mathematically saying, $f_s(\vec{\omega}_i, \vec{\omega}_o) = f_r(\vec{\omega}_i, \vec{\omega}_o) + f_t(\vec{\omega}_i, \vec{\omega}_o)$. It means that the scattering components could be factored based on the relative position (above or below) with respect to the surface,

§ 2.1.1.2 Radiometry basic quantities

Visible light is a kind of electromagnetic radiation, which is studied by Radiometry. It serves as the basis of the BRDF and rendering equation [PJH16]. I introduce four basic quantities and are summarized in Table 2.1.

The energy Q_e of electromagnetic radiation over a certain time period is called **radiant energy**, measured in Joules (J). The light source emits photons, which each has its wavelength λ and energy is as follows,

$$Q_e = \frac{h \cdot c}{\lambda}, \tag{2.1}$$

where c is the light speed and h is the Planck constant. We are also interested in the rate of energy flow through a space, which is called **radiant flux / power** Φ_e and measured in Joules per second (J/s) or Watts (W).

$$\Phi_e = \lim_{\Delta t \to 0} \frac{\Delta Q_e}{\Delta t} = \frac{dQ_e}{dt}. \tag{2.2}$$

Given a finite surface area A, the average density of power arriving or leaving over this area is called **irradiance** E_e and measured in $\mathrm{W} \cdot \mathrm{m}^{-2}$.

$$E_e = \frac{\Phi_e}{A}. \tag{2.3}$$

To distinguish the directional distribution of irradiance, I introduce **radiance** $L_e(\vec{\omega})$, which is the irradiance per unit solid angle. It's measured in $\mathrm{W} \cdot \mathrm{m}^{-2} \cdot \mathrm{sr}^{-1}$ and sr is the solid angle unit called steradian,

$$L_e(\vec{\omega}) = \lim_{\Delta \vec{\omega} \to 0} \frac{\Delta E_e}{\Delta \vec{\omega}} = \frac{dE_e}{d\vec{\omega}}. \tag{2.4}$$

Table 2.1: Radiometry basic quantities summary.

Term	Symbol	Definition	Unit
Radiant energy	Q_e	$Q_e = \frac{h \cdot c}{\lambda}$	J
Radiant flux	Φ_e	$\Phi_e = \frac{dQ_e}{dt}$	$\mathrm{W} = \mathrm{J/s}$
Irradiance	E_e	$E_e = \frac{\Phi_e}{A}$	$\mathrm{W/m^2}$
Radiance	$L_e(\vec{\omega})$	$L_e(\vec{\omega}) = \frac{dE_e}{d\vec{\omega}}$	$\mathrm{W/(m^2 \cdot sr)}$

§ 2.1.1.3 Definition of BRDF

The Bidirectional Reflectance Distribution Function (BRDF) is a radiometric function. It measures all the outgoing energy in the upper hemisphere, with respect to the incoming energy of direction $\vec{\omega}_i$ [GGGm16]. The outgoing direction $\vec{\omega}_o$ satisfies $\vec{\omega}_o \cdot \hat{n} \geq 0$. Defined by Fred Nicodemus in 1965 [Nic65] as follows,

$$f_r(\vec{\omega}_i, \vec{\omega}_o) = \frac{dL_o(\vec{\omega}_o)}{dE_i(\vec{\omega}_i)} = \frac{dL_o(\vec{\omega}_o)}{L_i(\vec{\omega}_i) \cos\theta_i \, d\vec{\omega}_i}, \tag{2.5}$$

where θ_i is the angle between $\vec{\omega}_i$ and the normal $\hat{n}$. BRDF is the fraction of two differentials, the outgoing radiance $L_o(\vec{\omega}_o)$ and the incoming irradiance $E_i(\vec{\omega}_i)$.

§ 2.1.2 Properties of BRDF

The bidirectional property means that $f_s(\vec{\omega}_i, \vec{\omega}_o) = f_s(\vec{\omega}_o, \vec{\omega}_i)$. As for user goals, there are two common types, i.e. physically-accurate and art-directed, which differ in their respective focus and application domains. Other goals might exist but are beyond the scope of our discussion.

§ 2.1.2.1 Physically-accurate

Physically accurate BRDFs are supposed to adhere to the following physical properties and constraints,

- **Positivity**:$f_r(\vec{\omega}_i, \vec{\omega}_o) > 0$.
- **Linearity**: multiple BRDFs be added together for a final reflectance result.
- **Helmholtz reciprocity** *(bidirection)*: $f_r(\vec{\omega}_i, \vec{\omega}_o) = f_r(\vec{\omega}_o, \vec{\omega}_i)$.
- **Energy conservation**: the amount of power leaving the passive materials must be no more than the amount arrived, where the unscattered energy is in the form of heat.

§ 2.1.2.2 Art-directed

Disney Principled BRDF prioritises artistic expressibility and controllability over physical accuracy, despite use of some physical rules. It models the diffuse, metallic specular highlights, tails of specularity, retro-reflection and transmission. For details, please refer to the analytical models in the below subsections.

§ 2.1.3 BRDF acquisition

To measure real-world materials, researchers utilize a gonio-spectro reflectometer [BNC09], which employs three to four goniometric arms to position the light source and detectors. The process involves changing light source positions, allowing data from different incoming angles are collected. Alternatively, MERL [MPBM03], which is adopted in this dissertation, employs HDR image-based methods. The MERL database contains 100 materials, each with reflectance values sampled at each degree of incoming and outgoing angles.
In general, the acquisition consumes a significant amount of time and storage [GGGm16], which remains a challenge for most use cases. Consequently, materials are often approximated through mathematical fitting or down-sampling.

§ 2.1.4 BRDF representations

There are mainly three categories of BRDF models (Figure 1.2), i.e. measured, analytical and neural representations. However, Graphics has not yet arrived at a definitive ideal model [HvDM+13].

§ 2.1.4.1 Measured models

The real-world measured BRDFs serve as the ground truth for all modelling techniques, e.g. MERL (2003) measures one RGB triple sample per degree, with a total of $90 \times 90 \times 180$ RGB triples stored in a binary file. They are captured in data-driven approaches, which are crucial for evaluating and improving the accuracy of BRDF models proposed thereafter. Please refer to 2.1.3 for the acquisition details.

§ 2.1.4.2 Analytical models

Empirical and Phenomenological models

For perfectly diffuse material, where the light is scattered uniformly in all directions, the Lambertian [Lam60] model (1760) is used,

$$f_r(\vec{\omega}_i, \vec{\omega}_o) = \frac{\rho_d}{\pi}, \tag{2.6}$$

where ρ_d is the albedo.
The Phong [Pho98] model (1975) is composed of diffuse and specular components k_d and k_s respectively. For specular term, it's based on the cosine law,

$$f_r(\vec{\omega}_i, \vec{\omega}_o) = k_s(\vec{\omega}_o \cdot \vec{\omega}_r)^n, \tag{2.7}$$

where $\vec{\omega}_r$ is the perfectly reflected direction of $\vec{\omega}_i$ over the surface normal $\hat{n}$. To reduce computation of the reflected vector $\vec{\omega}_r$, the Blinn-Phong [Bli77] model (1977) uses the half-angle vector instead of the reflection vector,

$$f_r = k_s(\vec{\omega}_h \cdot \hat{n})^n. \tag{2.8}$$

When compared visually, the Phong model reflects specular light sharper with highlights, whereas the reflection of the Blinn-Phong is softer. They are both nice and simple approximations, but ignoring energy conservation and Helmholtz reciprocity.

The Lafortune [LFTG97] model (1997) generalizes the Phong model (1975) by replacing dot product with weights and summing them up,

$$f_r = (\sum_{p=1}^{3} K_p \vec{\omega}_{o,p} \cdot \vec{\omega}_{r,p})^n, \tag{2.9}$$

where p loops through axes and K_p is the respective weight. It's reciprocal, energy conserving and flexible for more materials phenomena, e.g. the off-specular reflection, with increasing light lobe reflectance pointing to grazing angles. Some surfaces, e.g. on the moon, also reflect backwardly in the direction of the emitting light source, which is known as the retro-reflection and can be captured by the Lafortune model.

Physically-based Scattering models

A series of physically-based models are proposed to model real-world materials, which incorporates strict physical properties or constraints. Among which, the Cook-Torrance [CT82] (1982) is one of the most famous models.

The light is reflected from each **microfacet**, which is a tiny perfectly reflective subsurface, also known as the mirror. The BRDF specular term is modelled by,

$$f_{CT} = \frac{NDF \times G \times F}{4(\hat{n} \cdot \vec{\omega}_i)(\hat{n} \cdot \vec{\omega}_o)}, \tag{2.10}$$

where the halfway vector is $\vec{h} = \frac{\vec{l}+\vec{v}}{|\vec{l}+\vec{v}|}$ and NDF, G, F defined as follows.

Normal distribution function (NDF), e.g. Trowbridge-Reitz GGX, describes the concentration of microfacet along the $\vec{h}$ direction,

$$NDF_{GGX} = \frac{\alpha^2}{\pi[(\vec{n} \cdot \vec{h})^2(\alpha^2 - 1) + 1]^2}, \tag{2.11}$$

where α is square of the roughness coefficient. When the roughness value is low, it indicates a higher amount of microfacets facing in the halfway $\vec{h}$ direction. Hence, the specular highlight is much sharper, which is typical on the smooth surface. On the contrary, the rough surface with higher α tends to reflect lights in random directions, producing uniformly gray specular term.

Fresnel equation describes the percentage of reflection over refraction. It is approximated by Schlick to calculate the contribution of specular reflection between two distinct media,

$$F_{Schlick} = F_0 + (1 - F_0)(1 - (\vec{h} \cdot \vec{v}))^5, \tag{2.12}$$

where the reflectivity coefficient $F_0 = (\frac{IoR-1}{IoR+1})^2$ is derived from index of refraction (IOR).

Geometry function G describes the self-occlusion effects of both incoming and viewing directions,

denote IBL lighting coefficient $k_{IBL} = \frac{\alpha^2}{2}$,

$$G = G_1(\vec{v}) \cdot G_2(\vec{l}), \quad G_i(\vec{v}) = \frac{\vec{n} \cdot \vec{v}}{(\vec{n} \cdot \vec{v})(1 - k_{IBL}) + k_{IBL}}. \tag{2.13}$$

The sub-geometry terms $G_i \in [0, 1]$ scales the BRDF, where the value 1 means no self-occluding shadows and 0 the full shadowing effects.

Table 2.2: Common analytical BRDF models.

model name	year	formula
Lambertian [Lam60]	1760	$f_r(\vec{\omega}_i, \vec{\omega}_o) = \frac{\rho_d}{\pi}$
Phong [Pho98]	1975	$f_r(\vec{\omega}_i, \vec{\omega}_o) = k_s(\vec{\omega}_o \cdot \vec{\omega}_r)^n$
Blinn-Phong [Bli77]	1977	$f_r = k_s(\vec{\omega}_h \cdot \hat{n})^n$
Cook-Torrance [CT82]	1982	$f_r = \frac{NDF \times G \times F}{4(\hat{n} \cdot \vec{\omega}_i)(\hat{n} \cdot \vec{\omega}_o)}$

§ 2.1.4.3 Neural models

Recent machine learning advances bring a brand-new perspective on data compression. Neural networks show the potential to compress measured **data-driven** BRDF effectively, forming the **Neural-BRDF** (**NBRDF**) representation. It demonstrates both high fidelity and low memory footprints. Hence, researchers leverage the inference capability of trained neural network [SRRW21] (2021) as a new BRDF representation.

§ 2.1.5 Rendering and sampling

Solid angle is denoted as $\vec{\omega}$ and its **spherical coordinates** are (r, ϕ, θ), where $r = 1$ for a unit sphere, azimuthal angle[1] $\phi \in [0, 2\pi]$ and zenith angle[2] $\theta \in [0, \frac{\pi}{2}]$.

Given the incoming direction from camera $\vec{\omega}_i$, material BRDF $f_r(\vec{\omega}_i, \vec{\omega}_o)$ and no emitted light at the hit point, $L_e(\mathbf{x}) = 0$, the main task of the renderer is to *sample the outgoing direction* $\vec{\omega}_o$ at each hit point $\mathbf{x}$ to compute the radiance $L_i(\mathbf{x})$ by the **rendering equation**,

$$L_i(\mathbf{x}) = \int_{H^2} f_r(\vec{\omega}_i, \vec{\omega}_o) L_o(\mathbf{x}) \cos\theta_o \, d\vec{\omega}_o. \tag{2.14}$$

Monte Carlo integration approximates the integral $F = \int_{x=a}^{b} f(x)\, dx$ by numerically computing the average of the integrand $f(x_k)$ at random samples x_k, yielding $F \approx (b-a)\frac{1}{N}\sum_{k=1}^{N} f(x_k)$, where the index k ranges from 1 to the total number of samples N. It's an unbiased estimator of the integral with variance $\sigma^2 \propto \frac{1}{N}$, since the expected value of the estimator equals the true value of the integral,

$$\mathbb{E}[\frac{b-a}{N}\sum_{k=1}^{N} f(x_k)] = \int_{x=a}^{b} f(x)\, dx = F. \tag{2.15}$$

Importance sampling enhances the above process by sampling x with the density function $\text{pdf}(x)$, resulting in $\langle F^N \rangle = \frac{1}{N}\sum_{k=1}^{N} \frac{f(x_k)}{\text{pdf}(x_k)}$. The sampling density function $\text{pdf}(x)$ typically chosen to be proportional to the integrand $f(x)$ for lower variance and thus faster convergence.

[1] The polar angle with the bi-normal on the surface.
[2] The polar angle with the surface normal.

Similarly, when sampling the outgoing direction $\vec{\omega}_o$ with $\text{pdf}(\vec{\omega}_o)$ over the hemisphere H^2,

$$L_i(\mathbf{x}) \approx \langle L_i(\mathbf{x})^N \rangle = \frac{1}{N} \sum_{k=1}^{N} \frac{f(\vec{\omega}_{i,k}, \vec{\omega}_{o,k}) L_o(\mathbf{x}) \cos\theta_{o,k}}{\text{pdf}(\vec{\omega}_{o,k})}. \tag{2.16}$$

To efficiently render a scene in real time, importance sampling (implemented in § 3.1) helps the integrand **converge faster** in the limited Monte Carlo samples per pixel (spp). More advanced importance sampling techniques are in Appendix A.1.2.

§ 2.1.6 K-means clustering

I adopt K-means in the constrained synthesis framework (§ 3.6), where unconditional materials are filtered by statistical information rather than a neural network. It is a simple yet effective algorithm for boundary (e.g. specularity) detection, which partitions n data samples $\mathbf{x} \in \mathbb{R}^{n \times p}$ into k clusters $S = \{S_0, ..., S_{k-1}\}$ by minimizing the variance between each sample and its cluster center $\mu_i = \frac{1}{|S_i|} \sum_{\mathbf{x}_i \in S_i} \mathbf{x}_i$,

$$\arg\min_{S} \sum_{i=0}^{k-1} \sum_{\mathbf{x}_i \in S_i} ||\mathbf{x}_i - \mu_i||^2. \tag{2.17}$$

Note that the center of gravity of the data is the solution to the 1-means clustering. Once the clusters are assigned after iteration, we directly obtain the classification decision boundary.

§ 2.1.7 Principal Component Analysis

Principal Component Analysis (PCA), also known as Karhunen-Loève transform or Hotelling transform, is a data reduction technique. I adopt it in the material fitting and synthesis baselines. For material fitting, I utilize prior work [NJR15] to compress each data-driven material BRDF from $p = 90 \times 90 \times 180$ to $k = 300$ dimensions, with proper log-relative mapping (§ 3.1.1). For material generation baselines (§ 3.5.3), I reduce NBRDF weights from $p = 675$ to $k = 100$ for efficient NBRDF-space sampling.

Given n data $\mathbf{x} \in \mathbb{R}^{n \times p}$ each with a dimension of p, PCA reduces from p to a lower k principal components. To find such principal components, I apply singular value decomposition (SVD) on the zero-centered data $\mathbf{x}' = \mathbf{x} - \bar{\mathbf{x}}$, where $\bar{\mathbf{x}} \in \mathbb{R}^{n \times p}$ is broadcasting matrix of the mean vector $\mu \in \mathbb{R}^{1 \times p}$ defined by $\mu_\mathbf{j} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_{i,j}$.

The eigenvectors of covariance matrix $C_u = \frac{1}{n-1} \mathbf{x}'^T \cdot \mathbf{x}'$ and $C_v = \frac{1}{n-1} \mathbf{x}' \cdot \mathbf{x}'^T$ form the columns of $\mathbf{U} \in \mathbb{R}^{n \times k}$ and $\mathbf{V} \in \mathbb{R}^{p \times k}$, where the latter columns correspond to the eigenvectors or principal components. The diagonal matrix $\mathbf{\Sigma} \in \mathbb{R}^{k \times k}$ defined by $diag(\lambda_1, ..., \lambda_k)$, where λ_i stands for the magnitude of variance in eigenvectors.

$$\mathbf{x}' = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathbf{T}. \tag{2.18}$$

New data is generated using a linear combination with weights $\mathbf{c} \in \mathbb{R}^{k \times 1}$ of the scaled principle components matrix,

$$\mathbf{x}^\star = (\mathbf{V}\mathbf{\Sigma})\mathbf{c} + \bar{\mathbf{x}}. \tag{2.19}$$

§ 2.1.8 Variational autoencoder

I employ the Variational Autoencoder (VAE) [KW22] in material generation baselines (§ 3.5.3). By encoding and decoding the NBRDF weights, the VAE learns the latent distribution of the

material, thus providing the generative baselines.

VAE models the dataset distribution $\mathbf{X} = f_\theta(\mathbf{Z})$ by applying a decoder neural network f_θ onto the latent variables $\mathbf{Z}$. When sampling, the VAE decoder outputs the corresponding synthetic data from the latent distribution.

To find the appropriate latent variables $\tilde{\mathbf{Z}}$, another encoder neural network g_ϕ is trained for $\tilde{\mathbf{Z}} = g_\phi(\mathbf{x})$, given input $\mathbf{x}$. The training objective is derived by evidence lower bound (ELBO) and reparametrization,

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{\mathbf{X}\sim f_\theta(\mathbf{Z}), \tilde{\mathbf{Z}}\sim g_\phi(\mathbf{x})}[\log Pr_{\mathbf{X}}(\mathbf{X}|\tilde{\mathbf{Z}}) - D_{KL}(Pr_{\tilde{\mathbf{Z}}}||Pr_{\mathbf{Z}})]. \tag{2.20}$$

The likelihood term measures the reconstruction quality, while the reverse KL divergence, given a predictive distribution Q and true prior P (2.21), penalizes ill latent distribution $\tilde{\mathbf{Z}}$,

$$D_{KL}(Q||P) = \sum_{x\sim Q} Q(x) \log \frac{Q(x)}{P(x)}. \tag{2.21}$$

§ 2.1.9 Diffusion model

The diffusion model [HJA20] provides an alternative generative architecture for material synthesis. Based on dataset distribution $\mathbf{x}_0 \sim q(\mathbf{x}_0)$, the diffusion model generates synthetic $\mathbf{x}_0^\star \sim p_\theta(\mathbf{x}_0)$ from prior $p(\mathbf{x}_T)$ with bidirectional diffusion processes (Figure 2.1). Both processes are Markov chains with timestamp T. The state transition relations, i.e. per-step learned Gaussian distributions, are defined as forward $q_\theta(\mathbf{x}_t|\mathbf{x}_{t-1})$ and reverse $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, where $t \in [0, T)$.

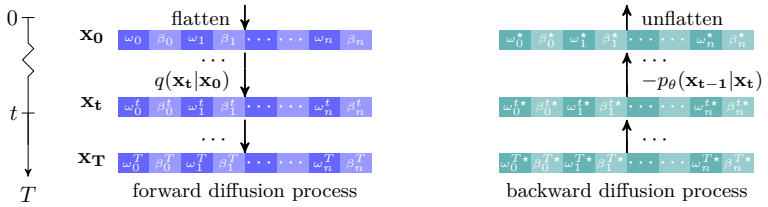


Figure 2.1: Diffusion architecture: add noise and denoise.

Forward process forms a Markov chain with joint distribution $q(\mathbf{x}_1, ..., \mathbf{x}_T|\mathbf{x}_0) = \prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1})$. Gaussian noise $q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$, with variance parameters β_i, is added in each step. *Hyperparameter* β_t is constant instead of learned by reparametrization. Moreover, a closed form can be derived for $\mathbf{x}_t$ distribution from $\mathbf{x}_0$, i.e. $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$, with $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{t=1}^T \alpha_t$,

Reverse process is similar with joint distribution $p_\theta(\mathbf{x}_0, ..., \mathbf{x}_T) = p(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_t|\mathbf{x}_{t-1})$ and per-step unknown Gaussian noise $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t), \sum_\theta(\mathbf{x}_t))$. The synthetic dataset is sampled from $p_\theta(\mathbf{x}_0) = \int p_\theta(\mathbf{x}_0, ..., \mathbf{x}_T)\, d\mathbf{x}_1 \ldots d\mathbf{x}_T$, by the marginal distribution formula.

Training aims to identify θ that best captures the reverse process distribution $p_\theta(\mathbf{x}_0)$. That is to minimize the Negative Log-Likelihood (NLL): $\mathcal{L}_{NLL} = \mathbb{E}_{\mathbf{x}_0\sim q(\mathbf{x}_0)}[-\log p_\theta(\mathbf{x}_0)]$. I derive the

NLL variational bound,

$$
\begin{aligned}
-\log p_\theta(\mathbf{x}_0) &= -\log \int p_\theta(\mathbf{x}_0, ..., \mathbf{x}_T)\, d\mathbf{x}_1 \ldots d\mathbf{x}_T = -\log \int p_\theta(\mathbf{x}_{0:T})\, d\mathbf{x}_{1:T} \\
&= -\log \int \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|x_0)} q(\mathbf{x}_{1:T}|x_0)\, d\mathbf{x}_{1:T} = -\log \mathbb{E}_{q(\mathbf{x}_{1:T}|x_0)}[\frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|x_0)}], \\
&\quad \text{by importance sampling on the sample distribution.} \\
&\leq -\mathbb{E}_{q(\mathbf{x}_{1:T}|x_0)}[\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|x_0)}] = \mathcal{L}_{UB} \quad \text{, by Jensen's inequality.}
\end{aligned}
\tag{2.22}
$$

In practice, the Evidence Upper Bound (EUBO) derived $\mathcal{L}_{UB}$ is the loss function to be minimized,

$$
\mathcal{L}_{UB} = \mathbb{E}_q[-\log \frac{p_\theta(\mathbf{x}_0, ...\mathbf{x}_T)}{q(\mathbf{x}_1, ..., \mathbf{x}_T|\mathbf{x}_0)}] \geq \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0)}[-\log p_\theta(\mathbf{x}_0)] = \mathcal{L}_{NLL}. \tag{2.23}
$$

Plugging in the probability distribution and refining it[3], $\mathcal{L}_{UB}$ is derived as follows,

$$
\mathbb{E}_q[D_{KL}(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T))] + \sum_{t>1} D_{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1). \tag{2.24}
$$

The KL divergence (2.21) terms above measure how prior reverse distribution $p(\mathbf{x}_T)$ matches forward $q(\mathbf{x}_T|\mathbf{x}_0)$ and how denoising posterior forward $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ fits reverse $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$. Plugging in the posterior formula and applying Bayes' rule with reparametrization trick, the optimized loss function is the L2 norm of the random noise ϵ and the predicted noise ϵ_θ,

$$
\mathcal{L} = ||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)||^2. \tag{2.25}
$$

The noise prediction network ϵ_θ is trained together with the diffusion model, which is a transformer model[4] (§ 2.1.11).

Inference samples new data $\mathbf{x}_0^\star \sim p_\theta(\mathbf{x}_0)$ from prior $p(\mathbf{x}_T) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ with the trained reverse process $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ defined above.

§ 2.1.10 Attention module

The attention modules underpin the transformer backbone (§ 2.1.11) in diffusion model (§ 2.1.9).

Scaled dot-product attention. [VSP+17] Given queries $[\mathbf{q}_1, ..., \mathbf{q}_{n_q}]$ and key-value pairs $[(\mathbf{k}_1, \mathbf{v}_1), ..., (\mathbf{k}_{n_x}, \mathbf{v}_{n_x})]$ in the matrix form, i.e. $\mathbf{Q} \in \mathbb{R}^{n_q \times d}$, $\mathbf{K} \in \mathbb{R}^{n_x \times d}$, $\mathbf{V} \in \mathbb{R}^{n_x \times d}$, dot-product term $\mathbf{q}_j \mathbf{k}_i^T$ measures the attention weights between $\mathbf{Q}$ and $\mathbf{K}$. The dot-product is first *scaled* by $\frac{1}{\sqrt{d}}$, i.e. the Euclidean length of $\mathbf{k} \in \mathbb{R}^d$, before applying to $\text{softmax}(\mathbf{x}_i) = \frac{exp(\mathbf{x}_i)}{\sum_{j=1}^n exp(\mathbf{x}_j)}$. Otherwise, extreme values will poison the gradient descent process. Given this correlation as the scaled dot-product, the j-th query output is the weighted sum of values: $\mathbf{o}_j = \sum_{i=1}^n \text{softmax}(\frac{\mathbf{q}_j \mathbf{k}_i^T}{\sqrt{d}})\mathbf{v}_i$,

$$
\begin{aligned}
\text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \in \mathbb{R}^{n_q \times d} &= [\mathbf{o}_{j=1}; ...; \mathbf{o}_{j=n_q}] \\
&= \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V} \quad \text{, via matrix form.}
\end{aligned}
\tag{2.26}
$$

Multi-head cross attention. Given two sets $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_{n_x}] \in \mathbb{R}^{n_x \times d_x}$ and $\mathbf{Y} = [\mathbf{y}_1, ..., \mathbf{y}_{n_q}] \in \mathbb{R}^{n_q \times d_q}$, per-head learnable weight matrices $\mathbf{W}_q^i \in \mathbb{R}^{d_q \times \frac{d}{h}}$, $\mathbf{W}_k^i, \mathbf{W}_v^i \in \mathbb{R}^{d_x \times \frac{d}{h}}$ are applied to the input,

$$
\text{head}^i(\mathbf{X}, \mathbf{Y}) \in \mathbb{R}^{n_q \times \frac{d}{h}} = \text{Attn}(\mathbf{Y} \cdot \mathbf{W}_q^i, \mathbf{X} \cdot \mathbf{W}_k^i, \mathbf{X} \cdot \mathbf{W}_v^i). \tag{2.27}
$$

[3] Details are in Appendix A [HJA20].

[4] Other noise prediction networks are available, e.g. CNN, UNet and many others.

The outputs from h heads are concatenated $[\text{head}^1; ...; \text{head}^h] \in \mathbb{R}^{n_q \times d}$ and assigned different weights with $\mathbf{W}^O \in \mathbb{R}^{d \times d}$, allowing *greater expressibility* than single-head,

$$\text{CrossAttn}(\mathbf{X}, \mathbf{Y}) \in \mathbb{R}^{n_q \times d} = [\text{head}^1; ...; \text{head}^h] \cdot \mathbf{W}^O. \tag{2.28}$$

Self attention takes both inputs from the same source, i.e. $\mathbf{X} = \mathbf{Y}$,

$$\text{SelfAttn}(\mathbf{X}) \in \mathbb{R}^{n_q \times d} = \text{CrossAttn}(\mathbf{X}, \mathbf{X}). \tag{2.29}$$

For each $head^i$, the query $\mathbf{Q}^i$, key $\mathbf{K}^i$ and values $\mathbf{V}^i$ are all projected from the same feature $\mathbf{X}$: $\mathbf{Q}^i = \mathbf{W}_q^i \mathbf{X}, \mathbf{K}^i = \mathbf{W}_k^i \mathbf{X}, \mathbf{V}^i = \mathbf{W}_v^i \mathbf{X}$, with weight matrices $\mathbf{W}_{q,k,v}^i \in \mathbb{R}^{d_x \times \frac{d}{h}}$.

§ 2.1.11 Transformer backbone

For diffusion models (§ 2.1.9), a neural network backbone, e.g. transformer or UNet, predicts the noise as sequential dependencies, with optional condition c on the input data $\mathbf{x}_0$, current timestamp t and random noise ϵ,

$$\epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, t, c). \tag{2.30}$$

Transformer architecture [VSP+17] is a sequence-to-sequence model (Figure 3.6). The input and output tokens are quantized as vectors by **embeddings** and softmax modules. **Positional encoding** incorporates the token relative position in the sequence additionally, via sine and cosine basis oscillating at different frequencies,

$$\forall i \in [0, \frac{d_{out}}{2}). \ \mathbf{P}_{pos,j} = \begin{cases} \sin(\frac{pos}{c^{\frac{j}{d_{out}}}}), & \text{if } j \text{ is even} \\ \cos(\frac{pos}{c^{\frac{j-1}{d_{out}}}}), & \text{if } j \text{ is odd} \end{cases}, \tag{2.31}$$

where $pos \in [0, L)$ is the token position in the sequence, j is the column index, c is the scaling constant and d_{out} is the output embedding dimension. The positional encoding is added with the input embedding, before submitting to the encoder.

The encoder contains multiple identical blocks, each with a Feed-Forward Network (FFN) sublayer after multi-head attention (§ 2.1.10). The FFN is a single-layer MLP with `ReLU` activation,

$$\text{FFN}(\mathbf{x}) = W_2 \max(0, W_1 \mathbf{x} + b_1) + b_2. \tag{2.32}$$

For each sublayer $\in \{\text{FFN}, \text{multi-head attn.}\}$, the residual connection is deployed and followed by layer normalization,

$$\mathbf{y} = \text{LayerNorm}(\mathbf{x} + \text{sublayer}(\mathbf{x})). \tag{2.33}$$

The decoder is similar, but with one multi-head attention taking in the encoder output at the end. The final output goes through a linear layer with softmax activation.

§ 2.2 Requirements analysis

`Mitsuba` [Jak10] has been widely used in Graphics literature. However, there have not been public bug-free Python implementations of major BRDF plugins in the latest `Mitsuba 3`[5] [JSRV22a], where the code structures are greatly changed. As such, the project includes the implementation of BRDF plugins for the MERL `binary` format, with integration from `Pytorch` models. It includes coordinate transformation and proper importance sampling strategies. Besides, I implement classical analytical BRDFs as baselines. The BRDF plugins are then tested

[5]'Mitsuba 3', research-oriented retargetable rendering system in C++ 17, 2022.

for correctness.

For machine learning training, I write the whole script from scratch, which is to be used by both neural BRDF and generative models. As said, two downstream tasks are implemented as in an extension, including sparse samples reconstruction and multi-modal synthesis. Throughout the project, proper software engineering techniques are considered and implemented. The detailed requirements are illustrated in Figure 2.2. The pink, orange and yellow boxes stand for the core implementation, evaluation and extension. The milestones are marked by white boxes.

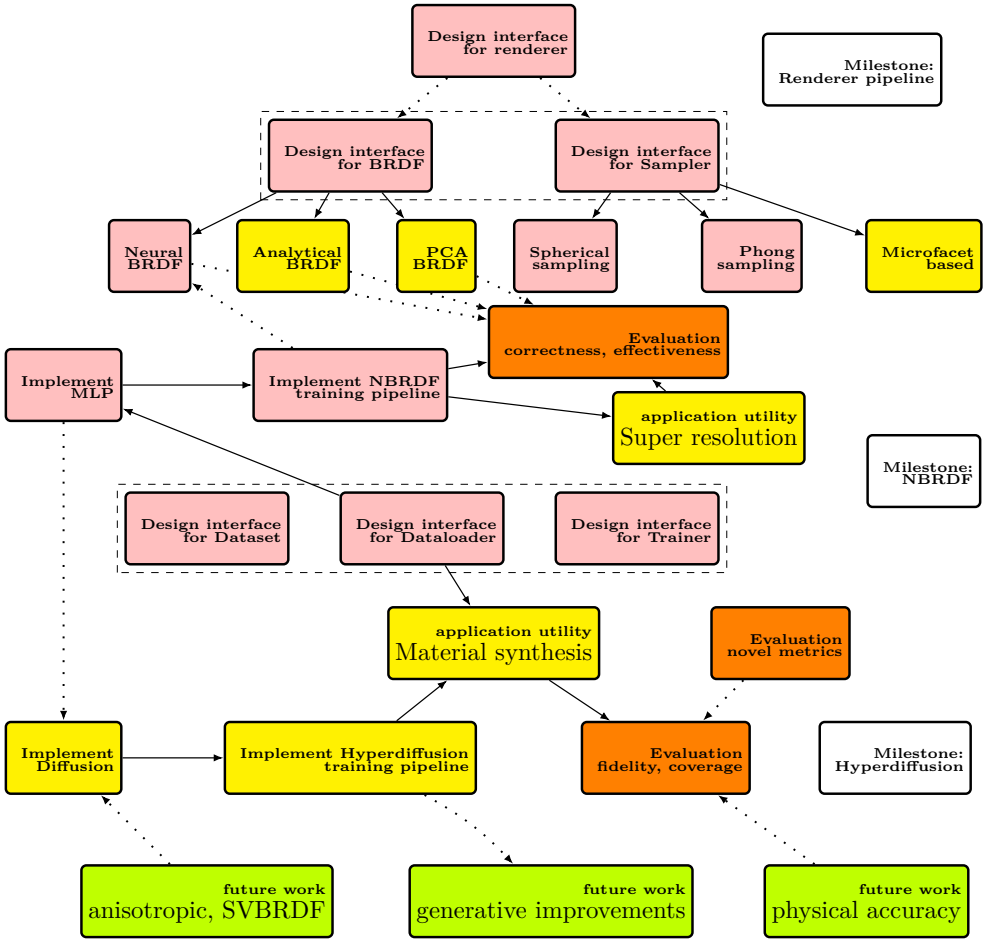


Figure 2.2: Dependency analysis of project requirements.

§ 2.3 Community contributions

During the implementation phase, I actively engage in the Graphics and machine learning community issues discussion, where suggestions are both given and received. I open-source the codebase under `MIT` license[6], due to its openness and broad compatibility. The polished paper is released to arXiv, an open-access preprint repository. Moreover, I share the weights or checkpoints of the trained models, together with the augmented MERL dataset, at AI community Hugging Face under `BSD` license[7], for reproducibility and further research.

[6]Codebase: https://github.com/PeterHUistyping/M3ashy.

[7]NeuMERL dataset: https://huggingface.co/datasets/Peter2023HuggingFace/NeuMERL.

The material synthesis extension of my project [ZHS+24] was submitted to Computer Vision and Pattern Recognition Conference (CVPR) 2025, where the novelty has been well-received by the research community. The reviewers have given constructive feedback, which has been incorporated into the dissertation and the codebase. The revised paper has been accepted by AAAI 2026 main techical track [ZHS+26].

In addition, I compare various state-of-the-art (SOTA) methods[8] against this project method, showcasing my contributions in a more precise BRDF model, multi-modality and quantitative metrics (Table 2.3).

Table 2.3: Comparison of material synthesis methods.

Synthesis	BRDF model	Uncond.	Text	Image	Quant. metrics
Memery et al. [MCS23]	parametric	×	✓	×	×
MATLABER [XLPD23]	analytic	×	✓	×	×
DeepBRDF [HGC+20]	data-driven	×	×	✓	×
FlashImage [HDMR21]	parametric	×	×	✓	✓
This project	**data-driven**	✓	✓	✓	✓

[8]DeepBRDF [HGC+20], [MCS23] via NVIDIA Omniverse parametric material, [HDMR21] via a parametric model from flash images, [XLPD23] via the Cook-Tolerance model.

3 Implementation

What I cannot create, I do not understand. — Richard Feynman

The main pipeline (Figure 3.1) consists of three main stages. Firstly, MERL dataset (§ 3.2.1) are preprocessed (§ 3.2.2) with augmentation. Secondly, the MLP model fits each material BRDF tabulation individually, forming the compact NBRDF representation. Its correctness and effectiveness are evaluated against the tabulated ground truth and compared with other baseline BRDF models in § 4.2. Thirdly, for material generation, the *transformer-based hyper-diffusion* model takes in the flattened NBRDF weights and generates **synthetic BRDF** with optionally multi-modal conditions. These newly generated samples are then evaluated against the reference MERL sets in § 4.3.2.2. Last but not least, I implement two extra NBRDF applications, namely sparse BRDF reconstruction and constrained BRDF synthesis.

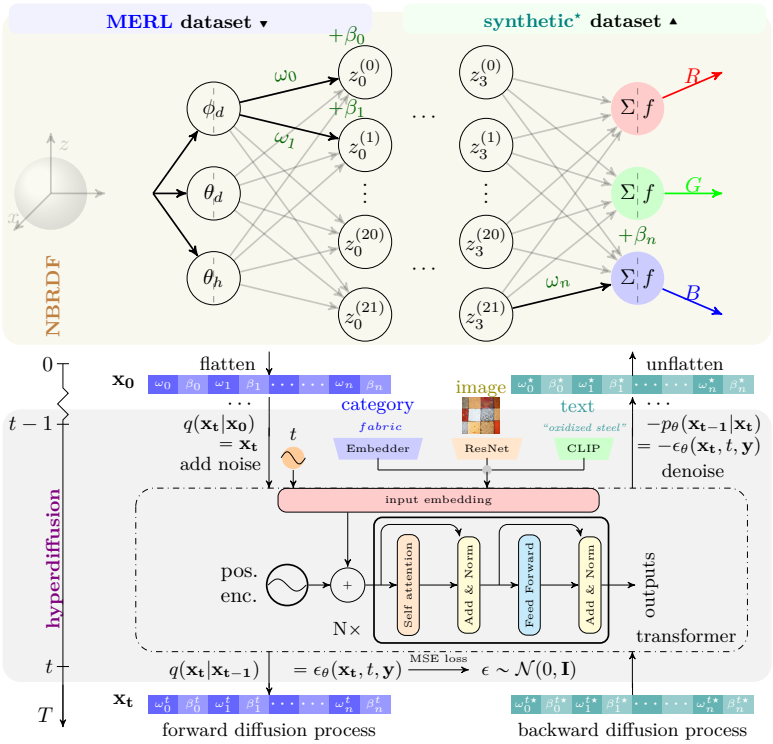


Figure 3.1: An overview of the main pipeline used in the dissertation.

§ 3.1 Renderer and importance sampling

For the renderer module, I implement various BRDF models and importance samplers. I derive the `BSDF` class from `Mitsuba3` to support specific analytical (baseline) and measured BRDF. The latter supports the input format of both binary files and MLP-weights, representing the ground truth and NBRDF respectively. To reuse the rendering functions, the aforementioned different models are encapsulated by a class hierarchy (UML Figure 3.2).

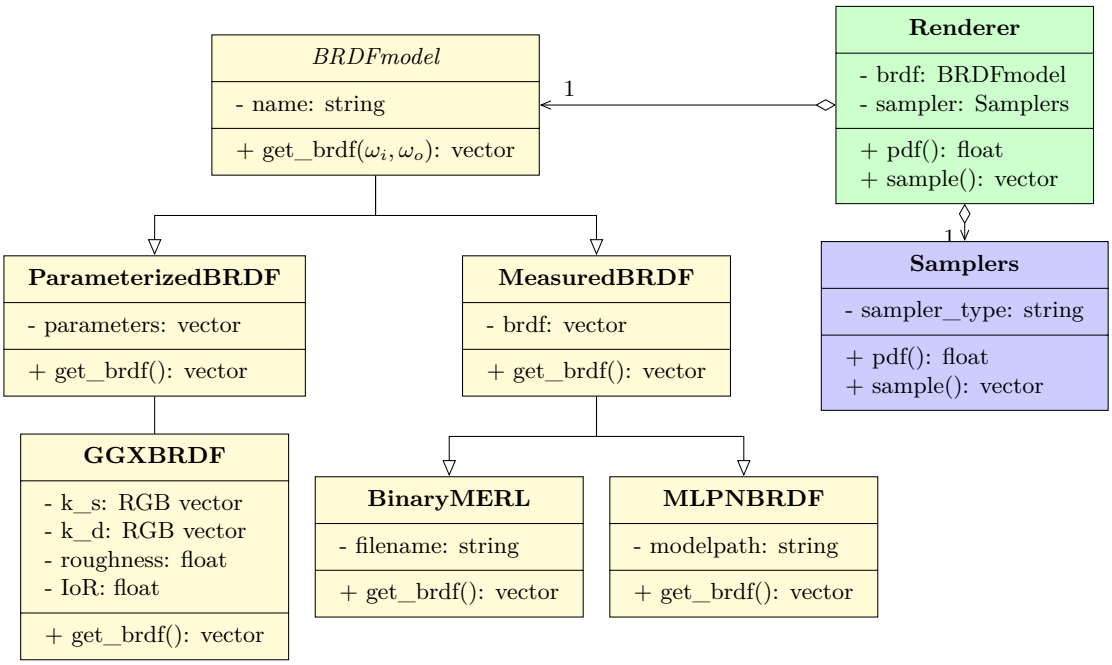


Figure 3.2: Overview of the renderer and BRDF models.

§ 3.1.1 Baseline BRDFs fitting

To evaluate NBRDF performance, I implement a physics-based analytical model and adopt another PCA-based as two baseline methods.

Cook-Torrance [CT82] model takes in diffuse $k_d \in \mathbb{R}^3$, specular coefficient $k_s \in \mathbb{R}^3$, roughness $\in \mathbb{R}^1$ and index of reflection $IoR \in \mathbb{R}^1$. The tabulated database lacks these parameters. To fit them, I utilize existing adaptive reverse rendering techniques based on image quality [BP20], where material coefficients are optimized until minimal image quality loss is achieved.

I implement the Cook-Torrance model *from scratch*, composed of Lambert diffuse and specular reflections: $f_r = k_d f_{Lam} + k_s f_{CT}$. Given incoming and viewing directions $\vec{l}, \vec{v}$, surface normal $\vec{n}$ and halfway vector $\vec{h} = \frac{\vec{l}+\vec{v}}{|\vec{l}+\vec{v}|}$, the **microfacet model** is,

$$f_{Lam} = \frac{\vec{n} \cdot \vec{v}}{\pi}, \quad f_{CT} = \frac{NDF \times G \times F}{4(\vec{n} \cdot \vec{l})(\vec{n} \cdot \vec{v})}. \tag{3.1}$$

The normal distribution function (NDF) Trowbridge-Reitz GGX describes the concentration of microfacet along the $\vec{h}$ direction, denote coefficient $\alpha = roughness^2$,

$$NDF_{GGX} = \frac{\alpha^2}{\pi[(\vec{n} \cdot \vec{h})^2(\alpha^2 - 1) + 1]^2}. \tag{3.2}$$

Fresnel equation describes the light percentage of reflection over refraction, approximated by Schlick F, denote the reflectivity coefficient $F_0 = (\frac{IoR-1}{IoR+1})^2$,

$$F_{Schlick} = F_0 + (1 - F_0)(1 - (\vec{h} \cdot \vec{v}))^5. \tag{3.3}$$

Geometry function G describes the self-occlusion effects of both incoming and viewing directions, denote IBL lighting coefficient $k_{IBL} = \frac{\alpha^2}{2}$

$$G = G_1(\vec{v}) \cdot G_2(\vec{l}), \quad G_i(\vec{v}) = \frac{\vec{n} \cdot \vec{v}}{(\vec{n} \cdot \vec{v})(1 - k_{IBL}) + k_{IBL}}. \tag{3.4}$$

Substitute (3.2) - (3.4) into (3.1), the Cook-Torrance model is complete, where I evaluate its

capturing against ground truth in § 4.2 and interpret its meaning from Computer Graphics perspective in 2.1.4.

Another baseline method PCA (see § 2.1.7) projects high dimension BRDF data to $m = 300$ by singular value decomposition (SVD). When reconstructing from sparse samples, five samples with the most optimal positions are selected. Due to computational complexity, I automate the whole process based on the existing PCA model [NJR15].

§ 3.1.2 Importance sampling

I implement six types of hemisphere importance sampling (§ 2.1.5), including both basic and microfacet strategies. During runtime, a specific sampler function is called each time (an example of the *strategy design pattern* [GHJV94]).

With two random numbers $\xi_1, \xi_2 \in [0, 1]$, the outgoing directions are sampled by $\vec{\omega}_o = (\theta, \phi) = (sample(\xi_1), 2\pi\xi_2)$, where probability density function (pdf) and function `sample()` are defined in Table 3.1.

Table 3.1: Importance sampling strategies summary.

Hemisphere	Solid angle pdf	Spherical sample
Uniform	$\text{pdf}(\vec{\omega}_o) = \frac{1}{2\pi}$	$\theta = \arccos(\xi_1)$
Cosine-weighted	$\text{pdf}(\vec{\omega}_o) = \frac{\cos\theta}{\pi}$	$\theta = \arccos(\sqrt{\xi_1})$
Power cosine	$\text{pdf}(\vec{\omega}_o) = \frac{(\alpha+1)\cos^\alpha\theta}{2\pi}$	$\theta = \arccos(\xi_1^{\frac{1}{\alpha+1}})$
Blinn-Phong	$\text{pdf}(\vec{\omega}_o) = \frac{(\alpha+2)\cos^{\alpha+1}\theta}{2\pi}$	$\theta = \arccos(\xi_1^{\frac{1}{\alpha+2}})$
Beckmann	$\text{pdf}(\vec{\omega}_o) = \frac{1}{\alpha^2\cos^3\theta} e^{\frac{-\tan^2\theta}{\alpha^2}}$	$\theta = \arctan(\sqrt{-\alpha^2\ln(1-\frac{\xi_1}{\pi})})$
GGX	$\text{pdf}(\vec{\omega}_o) = \frac{\alpha^2}{\pi\cos^3\theta(\alpha^2+\tan^2\theta)^2}$	$\theta = \arctan(\frac{\alpha\sqrt{\xi_1}}{\sqrt{1-\xi_1}})$

Derivation details are in Appendix § A. These six sampling strategies are verified in § C.1.3. For the final renderer, I adopt the microfacet-based importance sampling, which is significantly more efficient for a small sample size.

§ 3.2 Data handling

§ 3.2.1 Dataset formation

For supervised learning, a set of data samples is needed, where $\mathbf{y}_i$ is the target output for the input $\mathbf{x}_i$,

$$(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_i, \mathbf{y}_i), ..., (\mathbf{x}_n, \mathbf{y}_n). \tag{3.5}$$

MERL dataset [MPBM03] contains $m = 100$ material BRDF tabulations. The spherical position are sampled once every 1° in each coordinate θ_d, θ_h and ϕ_d, with a total of $n = 90 \times 90 \times 180$ samples.

The tensor shape is denoted in its *superscript*. Each BRDF tabulation contains the target RGB $tensor^{(1,3)}$ for each p-dimension spherical position input $tensor^{(1,p)}$. These $(\mathbf{x}_i : tensor^{(1,p)}, \mathbf{y}_i : tensor^{(1,3)})$ pairs serve as the dataset for each material MLP model.

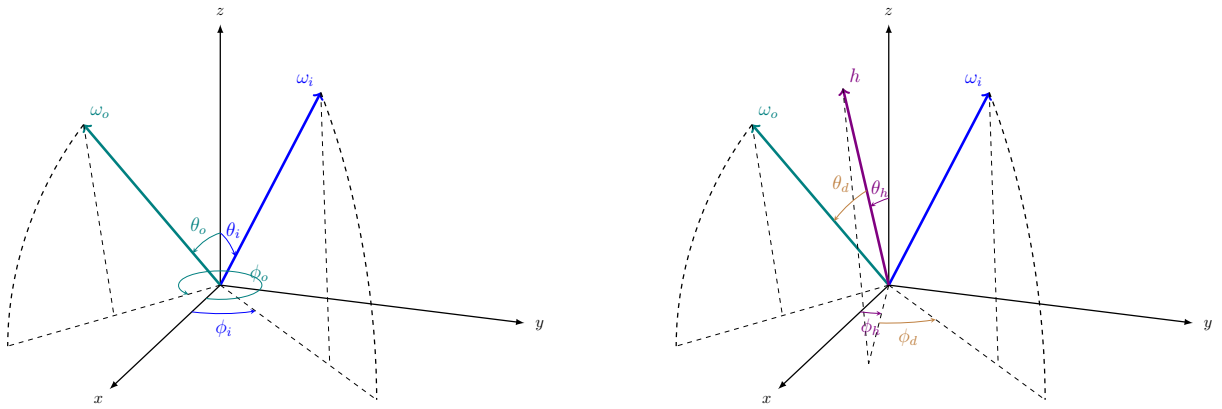
For generative learning, only input $\mathbf{x}_j$ is needed, with optionally[1] embedded condition labels $\mathbf{l}_j$ to synthesize new $x^\star$. The *square bracket* denotes an optional argument,

$$(\mathbf{x}_1, [\mathbf{l}_1]), ..., (\mathbf{x}_j, [\mathbf{l}_j]), ..., (\mathbf{x}_m, [\mathbf{l}_m]). \tag{3.6}$$

Each NBRDF weights are flattened as q-dimension input $\mathbf{x}_j$, the labels are embedded to c dimensions using ResNet18, OpenAI-CLIP or nn.Embedding, as in Figure 3.1. $(\mathbf{x}_j : tensor^{(1,q)}, [\mathbf{l}_j : tensor^{(1,c)}])$ pairs are the generation model dataset.

§ 3.2.2 Data preprocess

Registration. MERL dataset is in Rusinkiewicz's coordinate system (Figure 3.3b), which I convert to the standard coordinate system for rendering. Given the incoming and outgoing directions in spherical coordinate, Rusinkiewicz measures BRDF using halfway vector $\vec{h}$ and difference vector $\vec{d}$ [Rus98]. Hence, MERL BRDF is a function of these two vectors: $f_r(\theta_h, \theta_d, \phi_h, \phi_d)$. To process BRDF in standard coordinate system(Figure 3.3a), I apply reparametrization utilizing the geometric relationship (3.7). After transformation, BRDF is expressed as $f_r(\theta_i, \theta_o, \phi_i, \phi_o)$. In particular, MERL contains only **isotropic BRDF**, meaning f_r is independent of ϕ_o.



(a) Normal coordinate (b) Rusinkiewicz's coordinate

Figure 3.3: Coordinate transformation for MERL BRDF.

Transformation between standard and Rusinkiewicz's coordinate. $rot_{\vec{z},\phi}\vec{\omega}$ represents the rotation of $\vec{\omega}$ around axis $\vec{z}$ with rotation angle ϕ,

$$\vec{h} = \frac{\vec{\omega_i} + \vec{\omega_o}}{|\vec{\omega_i} + \vec{\omega_o}|}, \quad \vec{d} = rot_{\vec{y},-\theta_h} rot_{\vec{z},-\phi_h} \vec{\omega_i}. \tag{3.7}$$

Given $\vec{d} = rot_{\vec{y},-\theta_h} rot_{\vec{z},-\phi_h} \vec{\omega_i}$, apply the reverse rotation operations and rewrite it by matrices,

$$rot_{\vec{y},\theta_h} rot_{\vec{z},\phi_h} \vec{d} = \vec{\omega_i}. \tag{3.8}$$

$$\begin{bmatrix} \cos\phi_h & -\sin\phi_h & 0 \\ \sin\phi_h & \cos\phi_h & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\phi_h & 0 & \sin\phi_h \\ 0 & 1 & 0 \\ -\sin\phi_h & 0 & \cos\phi_h \end{bmatrix} \cdot \vec{d} = \vec{\omega_i}. \tag{3.9}$$

[1] Depending on the existence of condition, the model is termed as either unconditional or conditional generation.

By matrices expansion and $\vec{d}$ spherical coordinate[2],

$$\begin{bmatrix} \dots & \dots & \dots \\ \dots & \dots & \dots \\ -\sin\phi_h & 0 & \cos\phi_h \end{bmatrix} \cdot \begin{bmatrix} \sin\theta_d \cdot \cos\phi_d \\ \sin\theta_d \cdot \sin\phi_d \\ \cos\theta_d \end{bmatrix} = \begin{bmatrix} \dots \\ \dots \\ \cos\theta_i \end{bmatrix}. \tag{3.10}$$

Thus, I derive $\cos\theta_i$, which is used for capturing reflection at grazing angle,

$$\cos\theta_i = \cos\phi_h \cdot \cos\theta_d - \sin\phi_h \cdot \sin\theta_d \cdot \cos\phi_d. \tag{3.11}$$

Data augmentation. MERL size $m_0 = 100$ is too small for most generative models to learn. Therefore, I apply appropriate augmentation, including RGB-channel permutation and interpolation. The results are shown respectively in the first six and last two mosaic images of Figure B.1.

All the material colour channels are permuted, adding total size to $m_1 = 3! \times m_1 = 6 \times m$. Two MERL materials are then randomly selected and linearly interpolated (3.12) as a new mixed material, further increasing the dataset size $m_2 > 600$. The PBR properties of the new BRDF are examined in § 4.3.2.1.

Linear interpolation of two random MERL BRDF data,

$$brdf^{\star} : tensor^{(n,3)} = \alpha \times brdf_1 + (1-\alpha) \times brdf_2, \text{ for } \alpha \in (0,1). \tag{3.12}$$

Invalid values processing choices. For supervised learning, the MERL dataset contains invalid RGB values $(-1,-1,-1)$ due to the original measurement difficulty. These pairs need to be preprocessed. Two options exist, one approach is to fit the original dataset as it is. Another is to apply filtering on the dataset and use MLP inference results.

Mini-batching. To avoid overfitting during training, the dataset is split into 70% for the training set, 10% for the validation set and 20% for the test set.

Shuffling. I random shuffle the training data during each epoch, to avoid overfitting the model. It will help prevent the model from memorizing the data.

§ 3.2.3 Data loading pipeline

UML Figure 3.4 gives an overview of the loading pipeline for neural network models. Notations remain the same (§ 3.2.1). See details in the below subsections.

§ 3.2.3.1 Dataset abstraction

I derive two `Dataset` classes from the `Pytorch` package[3]. The supervised returns the position and RGB vector pair. The generative returns a pair of flattened NBRDF weights and condition labels. The latter is implemented via *a factory design pattern* [GHJV94], where a `LabelFactory` object creates the desired multi-modal condition `Label` object.

§ 3.2.3.2 Efficient dataloader

To avoid overfit, the dataloader will automatically permute and mini-batch dataset in each epoch. A self-implemented `Dataloader` solves the bottleneck of *slow permutation* for large data size in the `Pytorch` package[4]. It permutes the indices instead of the multi-dimensional data by trading a bit more memory. The total training time are measured multiple times and

[2] Note that only the z coordinate of $\vec{\omega}_i$ matters, thus the unrelated terms are omitted.
[3] `torch.util.data.Dataset`.
[4] `torch.util.data.DataLoader`.

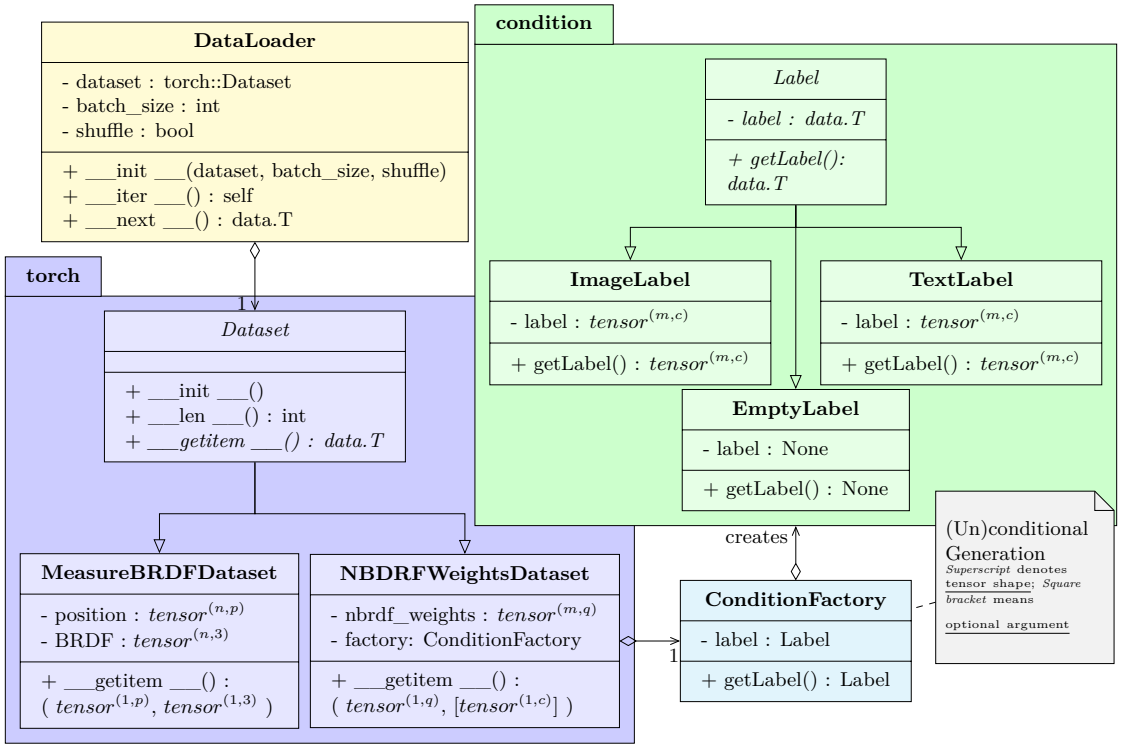


Figure 3.4: Overview of the data loading pipeline.

averaged as $\bar{t}$. Under the same setting[5], my `Dataloader` *speeds up* $\frac{\bar{t}_{old}}{\bar{t}_{new}} = 1.74\times$ than the official one. In addition, mine supports vector *iterable-style* rather than single-data fetch. These make more flexible `Dataset` possible.

As MLP model takes tabulated BRDF separately, I exploit the data parallelism through *multi-threading*. At most six concurrent training threads (*bash scripts*) are scheduled. The `Dataloader` per-epoch IO will be overlapped with other threads execution. It *accelerates* an extra $\frac{\bar{t}_{old}}{\bar{t}_{new}} = 2.25\times$.[6]

§ 3.3 Common training techniques

§ 3.3.1 Gradient descent

Standard gradient descent [Rud16] is performed over the whole dataset, the weight ω is updated based on the average gradient of the loss function $\mathcal{L}$,

$$\omega_{t+1} = \omega_t - \eta \frac{1}{N} \sum_{i=1}^{N} \nabla \mathcal{L}(\omega_t), \tag{3.13}$$

where η is the learning rate, a hyperparameter and i loops over the whole dataset. It converges slowly but is less noisy than other variants, e.g. stochastic gradient descent described below.

Stochastic gradient descent [Rud16] is the iterative optimization process commonly used for neural network training. Given a loss function $\mathcal{L}$ to minimize, the update rule for weight ω is based on a *single* sample,

$$\omega_{t+1} = \omega_t - \eta \nabla \mathcal{L}(\omega_t). \tag{3.14}$$

[5]Both single-threaded taking in tensor shape (2400, 1612) and batch size 25 from hyper-diffusion.
[6]Both models take in data size $n = 921739$ and batch size $\frac{n}{3}$, from valid MERL BRDF tabulation.

As a single data point is needed, it converges faster but is more noisy than standard gradient descent.

Batch gradient descent [LZCS14] is another variant of the standard gradient descent. Instead of the entire dataset, mini-batch gradient descent computes *each batch* of data separately and averages the gradients afterwards,

$$\omega_{t+1} = \omega_t - \eta \frac{1}{|B_i|} \sum_{i \in \mathcal{J}_B} \nabla \mathcal{L}(\omega_t), \tag{3.15}$$

where $|B_i|$ is the batch size and $\mathcal{J}_B$ is the index set of the batch. Despite the noise and computational cost, it is more stable than stochastic gradient descent, which is adopted in this project.

§ 3.3.2 Learning rate decay

In general, the learning rate scheduler [PG17] serves as a tradeoff between exploration and exploitation of state space.

I observe that the generative model loss increased dramatically after every 30 epochs, indicating a higher learning rate than expected. Therefore, a scheduler is adopted for faster loss convergence, switching from exploring further to the exploitation phase.

The `Pytorch` default scheduler is rigid using a fixed patience time. Hence, I implement one myself adopting a **metaheuristic**, where exploration is preferred in the beginning and exploitation at the end. My scheduler takes in the loss from beginning and records the **dynamic** epoch range $t_{threshold}$ for *minimal loss*.

$$\eta_{t+1} = \left\{ \begin{array}{ll} \eta_t & \text{if } t \leq t_{threshold} \\ \eta_t \times 0.1 & \text{otherwise.} \end{array} \right\}, \tag{3.16}$$

where η_t is the learning rate at epoch t. The training is then restarted and continues until the start of range, after which with a ten-time *smaller* learning rate. This process iterates towards convergence efficiently.

§ 3.3.3 Hyperparameter tuning

Firstly, the dataset is split into training, validation and test set, where validation set is used specifically for hyperparameter tuning. During parameter tuning phase, **cross-validation**, or also known as rotation estimation, is applied to avoid over-fitting, where the dataset is split into k folds. The model is trained on any $k-1$ folds and evaluated on the remaining one. The process is iterated k times and the average accuracy and validation loss are reported.

Secondly, a series of parameters are tried using **grid search**. For example, learning rate iterates from 5e-1, 5e-2, 5e-3 to 5e-4. Secondly, the one with the lowest validation loss is picked during *cross validation*. Lastly, the chosen parameter serves as the **random search** interval for finding the precise value.

Lastly, to reduce the inference loss, I apply the **dropout** approach [HSK+12]. During each training epoch, 10% neural nodes of MLP models are randomly disabled and recorded for back-propagation process. This approach brings significantly higher inference accuracy in limited epochs.

§ 3.3.4 Checkpoints

Checkpoints are used for backup and reproducibility. Hyperparameter and training status are saved together with model weights. They provide the foundation for me to compare between models throughout the project. Moreover, I can resume from history rather than training again from scratch, especially useful for large generative networks. Every two weeks, these records are checked and filed properly, which also supports backtracking of my previous training results. At the end of the project, these weights or checkpoints are archived openly at the AI community Hugging Face platform for future reference.

§ 3.4 NBRDF model

Given the MERL dataset $\{Mat_i | i = 1, ..., m\}$, material Mat_i contains spherical position $\mathbf{x}_i = (\theta_h, \theta_d, \phi_d)$ and RGB $\mathbf{y}_i$ pairs (§ 3.2.1). I split the pairs into 70% training set, 10% validation set and 20% test set. The test set is built upon different renderer scenes.

§ 3.4.1 MLP architecture and training

For each material Mat_i, I optimize a fully connected multilayer network (MLP), parametrized by weight ω_i and biases β_i. Concatenate all the coefficients together brings $[\omega_i, \beta_i] \in \mathbb{R}^q$.[7]
In particular, the MLP models effectively compress material Mat_i from high sampling dimension n to compressed dimension q of MLP weights, forming NBRDF $\{f_{Mat_i}(\omega_i, \beta_i) | i = 1, ..., m\}$ dataset following,

$$\{\mathbf{x}_i \in \mathbb{R}^{n \times p}, \mathbf{y}_i \in \mathbb{R}^{n \times 3}, \quad f_{Mat_i}(\mathbf{x}_i; \omega_i, \beta_i) = \mathbf{y}_i\}. \tag{3.17}$$

Network architecture is a Feed-Forward Network (FFN) with two hidden layers and exponential output layer, each with 21 neurons. Denote Leaky ReLU activation function as f_{act} and weight matrix as $W_{i,j}$, $f_{Mat_i}(\mathbf{x}_i) = FFN_{NBRDF}(\mathbf{x}_i) =$

$$f_{act}(exp[W_{i,3} \cdot f_{act}(W_{i,2} \cdot f_{act}(W_{i,1}\mathbf{x}_i + \beta_{i,1}) + \beta_{i,2}) + \beta_{i,3}] - 1). \tag{3.18}$$

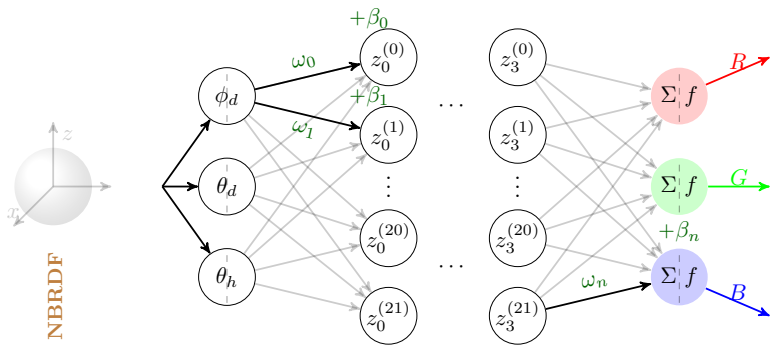


Figure 3.5: MLP architecture.

Hyperparameter. I set the epoch number 100 and learning rate 5e-3, which are evaluated to be the optimal configurations, as introduced in § 3.3.3. Different batch sizes trade-off between

[7] MLP-weight dimensions depend on the architecture. The number of nodes in each layer is $n_0 = 6, n_1 = 21, n_2 = 21, n_3 = 3$. Denote the total number of layers as $l = 4$. Hence, $q = \sum_{i=0}^{l-1} n_i \cdot n_{i+1} + \sum_{i=1}^{l} n_i = 6 \times 21 + 21 \times 21 + 21 \times 3 + 21 + 21 + 3 = 675$.

correctness and training time consumption. Two typical values (512 and 8000) are evaluated in § 4.3.2.2.

MLP weights initialization. As different MLP weights may generate the same BRDF, I use the optimized weights $[\omega_1, \beta_1]$ of the first material Mat_i to initialize the weights of the rest materials $\{Mat_i | i = 2, ..., m\}$.

§ 3.4.2 MLP inference

Let the positional parameter for retrieving the i-th sample of a specific BRDF as $\mathbf{x}_i$. With MAE loss $\mathcal{L}(f_1, f_2) = \frac{1}{N}\sum_{i=1}^{N} |\log[1 + f_1(\mathbf{x}_i)] - \log[1 + f_2(\mathbf{x}_i)]|$, the weights are updated based on back-propagation until convergence. To avoid overfit, The loss on the inference set is also reported. After training, the MLP inference results are evaluated under real-world settings (§ 4.2).

§ 3.5 NBRDF-space diffusion

Experiment on BRDF (un)conditional synthesis. Based on initial MERL dataset $\mathbf{x}_0 \sim q(\mathbf{x}_0)$, the trained model generates $\mathbf{x}_0^\star \sim p_\theta(\mathbf{x}_0)$ from prior $p(\mathbf{x}_T)$.

I split the NBRDF dataset into 95% training and 5% test set. A generative model is then trained on the training set (2280) and evaluated later on the unseen test set (120). After the base model is trained, other conditional models are derived by adding extra embedders for Multi-modal labels. These models are evaluated regarding their diversity and fidelity.

§ 3.5.1 Diffusion architecture and training

As introduced in § 2.1.9, I adopt a diffusion model [HJA20] with forward and reverse processes (Figure 2.1). Both processes are Markov chains with timestamp T, each with a per-step learnt Gaussian distribution, i.e. $q_\theta(\mathbf{x}_t|\mathbf{x}_{t-1})$ and $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ respectively.

Another neural network backbone (Figure 3.6), i.e. transformer model § 2.1.11, outputs the noise $\epsilon_\theta(\mathbf{x}_t, t)$ added at timestamp t.

Algorithm 1 Diffusion training.

Require: timestamp T, conditional context y
  repeat
    $\mathbf{x}_0 \leftarrow \mathbf{X}_0 \sim q(\mathbf{x}_0)$ ▷ sample training data
    $t \sim \text{Uni}(1, T)$
    $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    Gradient descent on $\mathcal{L} = ||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, t)||^2$
  until converged

I introduce the training and architecture details for unconditional and multi-modal synthesis case by case.

Unconditional synthesis. The model optimized on the training set after 700 epochs, with training loss 0.0013. The learning rate is adaptive from 5e-4 to 5e-6.

Category-conditioned synthesis. I train a category-conditioned generative model, with 119 epochs and training loss 0.02.

There are no categories for MERL material available. Hence, I manually extract from material names and filter out unrelated information (e.g. colour), forming 48 distinct categories. The

embedder used is the `Pytorch nn.Embedding` with $\mathbb{R}^{25}$, which projects category id to a high dimensional vector. For interpolated data, two vectors are concatenated. Accordingly, pure material has its vectors duplicated.

Text-conditioned synthesis. I train a generation model based on natural language prompts, with 200 epochs and loss 0.01. The text prompts are adapted from the MERL material tags. The prompt format is either pure *material1* or a mixture of *material1* and *material2* for interpolated BRDF. The embedder used is the CLIP from OpenAI with $\mathbb{R}^{768}$.

Image-conditioned synthesis. Two image-conditioned generation models are trained, with or without RGB condition. Given a scene with the same setup, single-view images are rendered and used as condition labels. I adopt ResNet18, a pre-trained CNN model, to embed center-cropped image into vectors, with $\mathbb{R}^{1000}$. However, purely conditioning on ResNet embedded vectors won't output colour-aware images. This result is also observed in generation-related work. Thus, an extra RGB thumbnail is added for optional colour conditions, adding to $\mathbb{R}^{1768}$. The model without RGB is trained with 200 epochs and loss 0.01. The other converges much faster, achieving a similar loss in only 55 epochs. The latter is trained 147 epochs with loss 0.001.

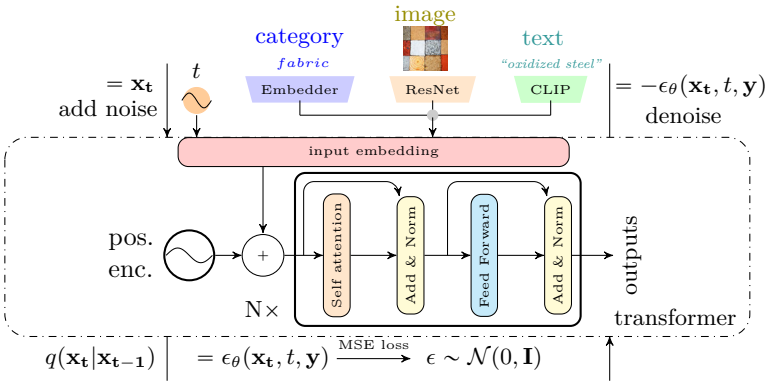


Figure 3.6: Diffusion backbone: transformer architecture.

§ 3.5.2 Diffusion sampling

When predicting noise ϵ_θ from $\mathbf{x}_t$ during the *reverse process*, Classifier-Free Guidance (CFG) approach [HS22] discards an extra classifier $p_\theta(\mathbf{y}|\mathbf{x}_t)$. Instead, it interpolates conditional and unconditional scores with guidance scale ω. Hyperparameter $\bar{\alpha}_t$ is defined above.

To sample synthetic data $\mathbf{x}_0$, I adopt Denoising Diffusion Implicit Models (DDIM) [SME22]. First obtain $\mathbf{x}_T$ from fixed prior $\mathcal{N}(0, \mathbf{I})$. Then iteratively, $\mathbf{x}_{t-1}$ is a linear combination of *denoised observation* $\mathbf{x}_t^\omega$ and CFG score (Algorithm 2).

§ 3.5.3 Baseline synthesis methods

The model I propose for material synthesis is the hyperdiffusion (hyperdiff). In addition, I apply PCA and VAE (§ 2.1.7, 2.1.8) into either brdf space $\mathbb{R}^{90\times90\times180}$ or NBRDF mlp weights space $\mathbb{R}^{675}$, giving four different baseline methods. They are evaluated in § 4.3.2.2.

For PCA (§ 2.1.7), I randomly select the linear combination weights $\mathbf{c} \in \mathbb{R}^{k\times1}$ and apply them to obtained PCA scaled principle components (dimension $k = 300$ for PCA-brdf, $k = 100$ for

[8]Notice that require $\omega > -1$ for conditional generation. Otherwise, the model downgrades to unconditional synthesis when $\omega = -1$.

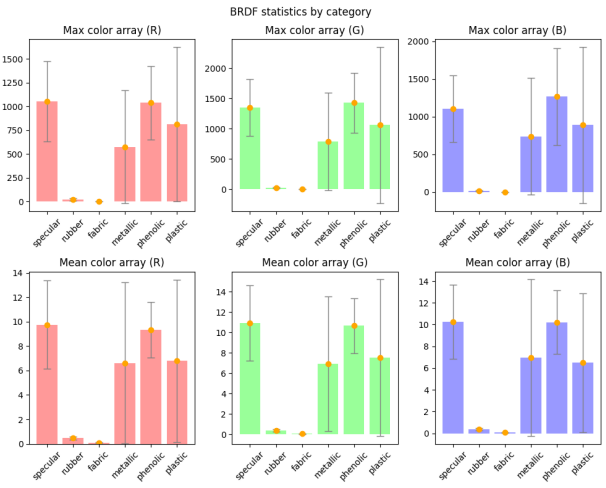
Algorithm 2 Conditional sampling with Classifier-Free Guidance (CFG).

Require: guidance scale $\omega \in \mathbb{R}$ [8], timestamp T, conditional context y
  $\mathbf{x}_T \leftarrow \mathbf{X}_T \sim \mathcal{N}(0, \mathbf{I})$ ▷ sample $\mathbf{x}_T$ from prior
  for $t = T$ to 1 **do**
    $\epsilon_\theta^\omega(\mathbf{x}_t, \mathbf{y}, t) = (1+\omega)\epsilon_\theta(\mathbf{x}_t, \mathbf{y}, t) - \omega\epsilon_\theta(\mathbf{x}_t, t)$ ▷ obtain CFG score
    $\mathbf{x}_t^\omega \leftarrow \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta^\omega(\mathbf{x}_t, \mathbf{y}, t))$ ▷ predicted $\mathbf{x}_0$ from DDIM sampler
    $\mathbf{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_t^\omega + \sqrt{1-\bar{\alpha}_{t-1}}\epsilon_\theta^\omega(\mathbf{x}_t, \mathbf{y}, t)$ ▷ add scaled $\mathbf{x}_t^\omega$ and CFG score
  end for
  return $\mathbf{x}_0$ ▷ return synthetic sample

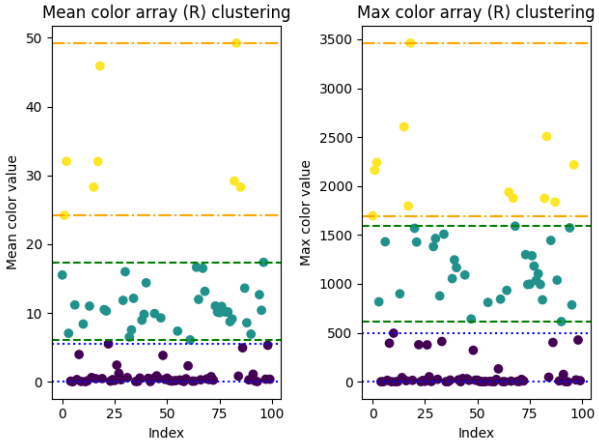
PCA-mlp). Likewise, I train two VAE (§ 2.1.8) models, where design details are left in § B.2.2, on these two materials representations (VAE-brdf, VAE-mlp) as other baseline methods.

§ 3.6 Statistically constrained synthesis

I first perform the statistical analysis on the most data-rich MERL categories obtained from material filename, including the max and mean reflectance values across all colour channels. The per-category summaries are shown in Figure 3.7a, while the individual details are included in Figure B.9 (Appendix).



(a) Category averaged max and mean reflectance.



(b) K means on the maximum reflectance of red channel.

Figure 3.7: MERL BRDF category statistical analysis.

I group the reflectance $f_r(\vec{\omega}_i, \vec{\omega}_o)$ of the unconditional synthetic materials into diffuse $f_r(\vec{\omega}_i, \vec{\omega}_o) \leq f_{diffuse}$ and specular part for those values above that threshold. I deduce the theoretical threshold for the diffuse term: $f_{diffuse} = \frac{1}{\pi}$, assuming that diffuse material BRDF $f_r(\vec{\omega}_i, \vec{\omega}_o) = f_r$ is constant, and by rendering equation at any position $\mathbf{x}$ and energy conservation that every physically-accurate BRDF strictly satisfies,

$$L_i(\mathbf{x}) = \int_{H^2} f_r(\vec{\omega}_i, \vec{\omega}_o) \cos\theta_o \, d\vec{\omega}_o \leq 1 \Rightarrow f_r \leq \frac{1}{\pi}. \tag{3.19}$$

In practice, the threshold depends on the measurement accuracy and unit of BRDF, as it could be pre-multiplied by π. For MERL dataset, the threshold is 1. For synthetic samples, the threshold requires finetuning for optimal classification. However, there is no upper bound on the specular term as long as the integral is satisfied.

Inspired by the above observations and theories, I introduce a novel set of constraints on the unconditional synthetic materials, termed the statistically constrained synthesis. It provides greater interpretability by nature, where SOTA neural network lacks.

- Purely diffuse materials (e.g. fabric): contains only diffuse term in all directions and all colour channels, with very few exceptions $n_{diff-exc.}$,

$$|\{(\vec{\omega}_o, \vec{\omega}_i) | f_r(\vec{\omega}_i, \vec{\omega}_o) \in \mathbb{R}^3 > f_{diffuse}\}| < n_{diff-exc.}. \tag{3.20}$$

- Metal-like materials: these materials are only specular term, with no diffuse component. They are among those classified specular materials mentioned below.
- Specular materials: I split them into low, mid and high specularity, by constraining on the red channel max reflectance over threshold f^i_{spec}, where i is the cluster index,

$$f^i_{spec} \leq \max f_r(\vec{\omega}_i, \vec{\omega}_o, red) \in \mathbb{R} < f^{i+1}_{spec}. \tag{3.21}$$

- Plastic-like materials: contains both diffuse and (mid or high) specular components. The specular part is white (similar intensities for the 3 colour channels). I define mismatch as at least one of the three pairwise RGB distances larger than $\Delta_{plastic}$, which could either be fixed 5% percentage of the maximum values over all channels or a value.
- Mirror-like specular materials: via polar diagrams, I observe that they have a small wideness of the specular lobe. The BRDF $f_r(\theta_h, \theta_d, \phi_d)$ evaluated at $\theta_h = 0$ should be the peak of the lobe (for any values of θ_d and ϕ_d). If incrementing θ_h (without touching the other 2 angles), the BRDF value decreases. The wideness of the lobe can be defined as the distance one needs to increase θ_h from zero, until a BRDF value is half of the value of the peak and the value remains smaller thereafter, while the peak is the BRDF at $\theta_h = 0$. By experiments, I set that value to be 0.349.

To find the classification boundary parameters listed above, I apply the K-means clustering (§ 2.1.6) on them. For example, clustering range $[f^i_{spec}, f^{i+1}_{spec})$ obtained from K-means on max reflectance of a colour channel highly indicates the materials specularity (Figure 3.7b).

§ 3.7 Repository structure

I opt for a microservices-inspired layout, where each module is isolated placed and independently developed. Thus, by decoupling dozens of configs, docs, shell scripts, source code and tests, I ensure cleaner builds, ease of maintenance, extensibility and testability. As opposed to monolithic repositories, my codebase[9] are divided into `configs`, `docs`, `src` and `tests` packages. Within the `src`, I derive the `Mitsuba` renderer's BRDF and sampling strategies (§ 3.1), implement the models in `Pytorch` (§ 3.4, 3.5) and evaluate their performance (§ 4). A high-level repository structure is shown in Figure 3.8. Please note that the list of folders and files shown is not exhaustive and only the code written by me is included (7291 lines of code).

[9]Based on the result of `tree --gitignore --prune` command.

```
Neural-BRDF
├── LICENSE ..... Project MIT License and attribution of others (§ C.1.2)
├── docs/doxygen/ ..... Web-based documentation (§ C.1.4)
├── Doxyfile, README.md ..... Doxygen system configs (§ C.1.4)
├── configs/ ..... Model parameters configs
├── data/ ..... MERL binary BRDFs
│   └── python_merl_scene.xml ..... Scene and geometry setup
├── output/ ..... Folder for rendered results, model checkpoints
├── .github/workflow/ci.yaml ..... Continuous Integration (§ C.1.4) [55 lines]
├── src/ ..... Source code package [Total 6662 lines]
│   ├── renderer/ ..... [Total 1138 lines]
│   │   ├── renderer.py ..... Renderer main control [343 lines]
│   │   ├── brdf_plugin.py ..... Interface for different BRDF [610 lines]
│   │   ├── sampling_strategy.py ..... Six samplers (§ 3.1) [166 lines]
│   │   └── data_type.py
│   ├── pytorch/ ..... [Total 2448 lines]
│   │   ├── dataset/ ..... Format and abstraction (§ 3.2.1, 3.2.3.1) [978 lines]
│   │   ├── dataloader.py ..... Efficient dataloader (§ 3.2.3.2) [50 lines]
│   │   ├── train.py ..... Main training pipeline [696 lines]
│   │   ├── model_factory.py ..... Interface for Pytorch models [295 lines]
│   │   ├── models/ ..... Core and generative models (§ 3.4, 3.5) [323 lines]
│   │   └── utils/ ..... Device setup, loss visualization [106 lines]
│   ├── eval/ ..... [Total 2078 lines]
│   │   ├── metrics.py ..... Performance metrics (§ 4.1) [136 lines]
│   │   ├── eval_gen_metrics.py ..... Generation metrics (§ 4.1.3) [590 lines]
│   │   ├── eval_whole_dataset.py ..... Over whole MERL dataset (§ 4.2) [207 lines]
│   │   └── eval_plot_gen_confusion.py .. Pairwise BRDFs confusion matrix [246 lines]
│   └── tools/merl_workflow/ ..... [Total 998 lines]
│       ├── *.sh ..... Five bash scripts for automation (§ C.1.4) [161 lines]
│       └── util_*.py ..... Commonly-used utility functions
└── tests/ ..... Testing package [Total 574 lines]
    └── chi2/ ..... Renderer samplers verification (§ C.1.3) [196 lines]
```

Figure 3.8: High-level repository structure [7291 lines of code].

4 Evaluation

Measure what is measurable, and make measurable what is not so. — Galileo Galilei

To evaluate the proposed NBRDF model, I thoroughly measure its correctness, effectiveness and utility. In terms of correctness, both qualitative and quantitative image quality metrics show that the neural BRDF outperforms other BRDF models, against the ground truth. For effectiveness, the memory usage and inference time are collected and analyzed, with respect to classical BRDF models. Regarding the utility for research, I apply the NBRDF model to different downstream tasks, with separate evaluation. For material generation, I measure both the coverage and fidelity over the metrics proposed. Finally, I validate the effectiveness of the proposed novel metrics.

§ 4.1 Performance metrics

For material fitting evaluation, I proceed by rendering the set-up scenes with both NBDRF and ground truth BRDF, and then comparing the image quality degradation between those two rendered images. Denote the width and height of the rendered image as W and H respectively, and the number of colour channels as C. Given two rendered images $I_1, I_2 \in \mathbb{R}^W \times \mathbb{R}^H \to [0,1]^C$, there are two types of metrics measuring their similarity, namely pixel-wise and perception-based.

§ 4.1.1 Pixel-wise

Mean Squared Error (MSE) (↓) checks if pixel values at the same coordinates match. The lower the score, the better the quality.

$$\mathrm{MSE}(I_1, I_2) = \frac{1}{W \cdot H} \sum_{w=1}^{W} \sum_{h=1}^{H} [I_1(w,h) - I_2(w,h)]^2, \quad \mathrm{RMSE}(I_1, I_2) = \sqrt{\mathrm{MSE}(I_1, I_2)}, \tag{4.1}$$

where **Root Mean Squared Error (RMSE)** is the square root of MSE. However, their absolute values both suffer from the image intensity scaling. Hence, to measure via these metrics, the pixel values are normalized to the same range.

Peak Signal-to-Noise Ratio (PSNR) (↑) [HZ10] is the scaled MSE, solving the intensity scaling issue described above. The higher the score, the better the quality.

$$\mathrm{PSNR} = 10 \log_{10} \frac{P^2}{\mathrm{MSE}} = 10 \log_{10} \frac{P^2}{\frac{1}{W \cdot H} \sum_{w=1}^{W} \sum_{h=1}^{H} [I_1(w,h) - I_2(w,h)]^2} [\mathrm{dB}], \tag{4.2}$$

where peak signal P is the maximum pixel value. For 8-bit images used in my experiments, $P = 2^8 - 1 = 255$[1].

§ 4.1.2 Perception-based

Structural Similarity Index (SSIM) (↑) [WBSS04] is a perception-based metric. It solves the issue of pixel-wise metrics lacking in human perception. SSIM metric approximates local *luminance* μ_I, contrast σ_I via the image I's Y-channel mean, variance and covariance respec-

[1] when generalize to n-bit image, $P = 2^n - 1$.

tively,

$$\mu_I = \bar{I^Y} = \frac{1}{W \cdot H} \sum_{w=1}^{W} \sum_{h=1}^{H} I^Y(w,h), \tag{4.3}$$

$$\sigma_I = Var(I^Y) = \sqrt{\frac{1}{W \cdot H - 1} \sum_{w=1}^{W} \sum_{h=1}^{H} [I^Y(w,h) - \mu_I]^2}. \tag{4.4}$$

The covariance σ_{I_1,I_2} between two images are defined by the covariance formula,

$$\sigma_{I_1,I_2} = Cov(I_1^Y, I_2^Y) = \frac{1}{W \cdot H - 1} \sum_{w=1}^{W} \sum_{h=1}^{H} (I_1^Y(w,h) - \mu_{I_1})(I_2^Y(w,h) - \mu_{I_2}). \tag{4.5}$$

Separate functions for luminance, contrast and structure are defined below

$$l(I_1,I_2) = \frac{2\mu_{I_1}\mu_{I_2} + C_1}{\mu_{I_1}^2 + \mu_{I_2}^2 + C_1}, \quad c(I_1,I_2) = \frac{2\sigma_{I_1}\sigma_{I_2} + C_2}{\sigma_{I_1}^2 + \sigma_{I_2}^2 + C_2}, \quad s(I_1,I_2) = \frac{\sigma_{I_1,I_2} + C_3}{\sigma_{I_1}\sigma_{I_2} + C_3}, \tag{4.6}$$

where C_i are constants to avoid instability.

$$\text{SSIM}(I_1,I_2) = [s(I_1,I_2)]^\alpha [c(I_1,I_2)]^\beta [s(I_1,I_2)]^\gamma, \quad \alpha > 0, \beta > 0, \gamma > 0. \tag{4.7}$$

SSIM is finally defined by a product of the above luminance, contrast and structure functions.

§ 4.1.3 Generation-based

For material generation evaluation, I evaluate the fidelity and diversity of the generated set S_g against the reference set S_r, via FID and self-proposed metrics adapted from the point cloud generation field.

Fréchet Inception Distance (FID) (↓) [HRU+17] is the de-facto image generation evaluation metric, originally designed for Generative Adversarial Networks (GANs) and later extended to wider applications. The lower the score, the higher synthetic set coverage. A pre-trained Inception Network f_{inc} maps the image set S_r and S_g to the latent space, which is parametrized by two Gaussian distributions,

$$f_{\text{inc}}(S_r) \subset \mathbb{R}^n \sim \mathcal{N}(\mu_r, \Sigma_r), \quad f_{\text{inc}}(S_g) \subset \mathbb{R}^n \sim \mathcal{N}(\mu_g, \Sigma_g). \tag{4.8}$$

The FID score is measured by the difference between their fitted Gaussian distributions,

$$\text{FID}((\mu_r,\Sigma_r),(\mu_g,\Sigma_g))^2 = |\mu_r - \mu_g|^2 + tr(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}), \tag{4.9}$$

where $tr(A)$ is the trace of matrix A and $(\mu,\ \Sigma)$ are the first two moments, i.e. mean and covariance of the Gaussian. For material generations, I evaluate the FID score of BRDFs rendered images in a controlled real-world scene.

As an extension, I propose a set of domain-specific metrics to evaluate the fidelity and diversity between reference set S_r and generative set S_g. To begin with, I exploit two types of underlying distance functions d_D, namely BRDF-space d_D^{BRDF} and rendered image space d_D^{image}.

Pairwise distance for the tabulated BRDFs. Inspired by the loss used in BRDFs fitting, I propose to adapt the BRDF per-sample loss functions $\mathcal{L}(f_1,f_2)$ as the distance function d_D in the high-dimensional BRDF-space,

$$\mathcal{L}(f_1,f_2) \rightarrow \begin{cases} \mathcal{L}_{L1} & = |f_1 - f_2| \\ \mathcal{L}_{L1-log} & = |\log(1+f_1) - \log(1+f_2)| \\ \mathcal{L}_{L1-log-\cos} & = |\log(1+f_1\cos\theta_i) - \log(1+f_2\cos\theta_i)| \end{cases}, \tag{4.10}$$

where (f_1, f_2) are the two BRDFs to be compared and θ_i is the angle between the incident light direction and point normal. According to the Lambert's Cosine law [Lam60], which relates the amount of light arriving at a surface I^s to the source intensity I_0,

$$I^s = I_0 \cos(\theta_i) \implies I^s \propto \cos(\theta_i), \tag{4.11}$$

the cosine term in $\mathcal{L}_{L1-log-\cos}$ is used to compensate for the reflectance increase towards grazing directions [FFG12]. The BRDF-space distance function d_D is defined as the Mean Absolute Error (MAE) of the above loss functions over the material parameter space,

$$d_D^{\text{BRDF}} = MAE(\mathcal{L}) = \frac{1}{N}\sum_{s=1}^{N}\mathcal{L}(f_1(p_s), f_2(p_s)), \tag{4.12}$$

where p_s is the parameter to retrieve the s-th sample of a specific BRDF, i.e. $\vec{\omega}_i$ and $\vec{\omega}_o$ and $\mathcal{L}$ is one of the above loss functions. Hence, there are three instances of d_D^{BRDF} with different loss functions.

Pairwise distance for the BRDFs rendered images. To augment the evaluation, I also adapt the image quality scores, i.e. MSE, PSNR and SSIM, as the distance function d_D^{image} in the image space. Note that for MSE and RMSE, the lower the score, the better the quality. It's naturally a distance function d_D^{image}. Whereas for PSNR and SSIM, it is the opposite way around. Hence, I negate[2] the latter two metrics for d_D^{image}. Hence, there are three instances of d_D^{image} with different image quality metrics.

With the above distance functions, i.e. either d_D^{BRDF} or d_D^{image}, I introduce Minimum Matching Distance (MMD) for quality, Coverage (COV) for diversity and 1-Nearest-Neighbour Accuracy (1-NNA) for both fidelity and diversity, They are widely-used metrics in the 3D point cloud generation field [PAG18], where I adapt it to the BRDF synthesis evaluation. To my best knowledge, the proposed metrics tailored for BRDF synthesis evaluation and are novel in the field.

Minimum Matching Distance (MMD) (↓) [PAG18] captures the average distance between the reference BRDFs (from S_r) and their closest neighbours in the generative set S_g, where a lower score is preferred. With $d_D(X, Y)$ introduced as above, i.e. the two types of distance functions between a reference BRDF and a sample BRDF, MMD is defined as,

$$\text{MMD}(S_g, S_r) = \frac{1}{|S_r|}\sum_{Y\in S_r} min_{X\in S_g} d_D(X, Y). \tag{4.13}$$

Coverage (COV) (↑) [PAG18] reports the coverage by measuring the *unique* and closest neighbour of the generated BRDF (from S_g) in the reference set S_r, quantifying the diversity of the generation process. Usually, we prefer a higher score, with $d_D(X, Y)$ introduced as above. Cov is defined as,

$$\text{COV}(S_g, S_r) = \frac{|\arg\min_{Y\in S_r} d_D(X, Y)|X \in S_g|}{|S_r|}. \tag{4.14}$$

1-Nearest-Neighbour Accuracy (1-NNA) (50%) [LPO17] is the leave-one-out accuracy signaling whether reference set S_r and generated set S_g are coming from the same distribution, of which 50% is preferred. The accuracy measures which set the nearest neighbour of each

[2] due to the positivity of PSNR and SSIM, the negation preserves the monotonicity of the distance function.

sample belongs to. With $d_D(X, Y)$ introduced as above, 1-NNA is defined as,

$$\text{1-NNA}(S_g, S_r) = \frac{\sum_{X \in S_g} \mathbb{I}[N_{-X} \in S_g] + \sum_{Y \in S_r} \mathbb{I}[N_{-Y} \in S_r]}{|S_g| + |S_r|}, \tag{4.15}$$

where the indicator function is denoted as $\mathbb{I}[\cdot]$ and the nearest neighbour of X in both sets as $N_{-X} \in S_r \cup S_g - \{X\}$.

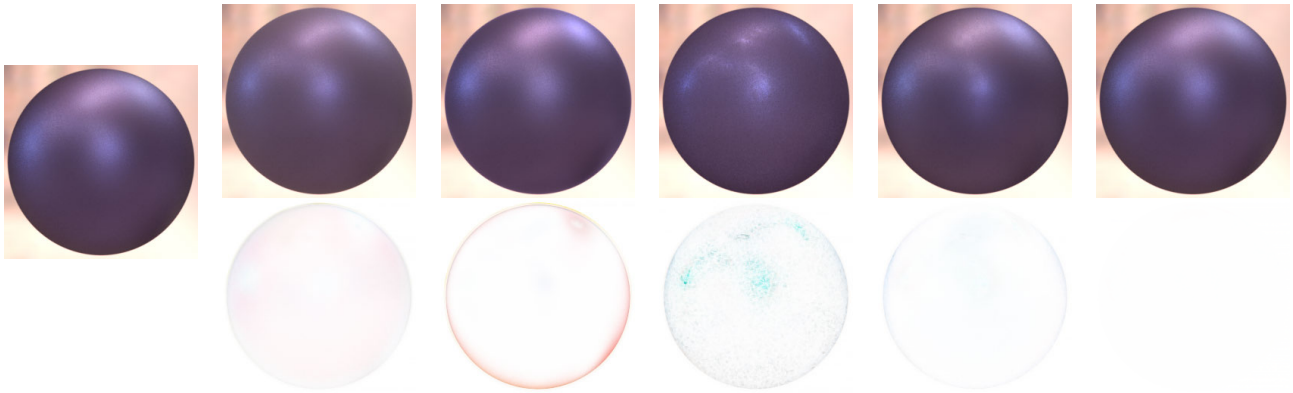
Hence, I adapt the above three metrics into the BRDF synthesis evaluation, which is an approach previously unexplored in the field, to the best of my knowledge. The proposed metrics are validated in § 4.3.3.

§ 4.2 Techniques evaluation

In this section, different NBRDF models are evaluated and the best is chosen. The baseline methods are the PCA and analytical Cook-Torrance model. I first train a model over the unprocessed original MERL (ori-MERL) and another model using preprocessed dataset (pre-MERL), as introduced in § 3.2.2. For *hyperparameters*, I try batch sizes of different magnitudes. Among these, 8000 (NBRDF-L) and 512 (NBRDF-s) are reported.

§ 4.2.1 Correctness evaluation

A scene is set up with the same IBL[3] environment and target sphere but different BRDF models. I compare rendered image quality against tabulated ground truth. To visualize the difference, SSIM index maps are attached beside scores (Figure 4.1). The material is blue-metallic-paint, which is particularly challenging to capture among the MERL dataset.



(a) Tabulated MERL (ground truth) | (b) Baseline PCA (SSIM=0.9749) | (c) Baseline Cook-Torr (SSIM=0.9852) | (d) NBRDF-L on ori-MERL (SSIM=0.9828) | (e) NBRDF-L on pre-MERL (SSIM=0.9938) | (f) NBRDF-s on pre-MERL (SSIM=0.9992)

Figure 4.1: Rendered images and SSIM index maps comparison from different BRDF models against ground truth (material: blue-metallic-paint).

Compared with *Cook-Torrance* and *PCA* baseline methods, all NBRDFs achieve similar or better results than the analytical baseline. Particularly, NBRDF-s trained over the pre-MERL dataset captures the details accurately, leaving near-zero SSIM difference. To further demonstrate it over the whole dataset, I measure MSE, PSNR and SSIM scores with mean ± standard deviation (Table 4.1). **Top** results are **bold**.

[3]Image-based lighting (IBL) measures radiance in an omnidirectional image (ODI). The common ODI representations are the cube map, light probe image (stereographic projection) and (longitude-latitude) equirectangular map (cylindrical projection) [ML06].

Table 4.1: Different BRDF models image quality.

dataset	model	MSE(↓)	PSNR (↑)	SSIM (↑)
ori-MERL	*Cook-Torr*	201.42 ± 32490.4	26.9 ± 18.6	0.9718 ± 0.0003
ori-MERL	*PCA*	154.15 ± 38861.3	29.0 ± 26.4	0.9703 ± 0.0015
ori-MERL	NBRDF-L	155.01 ± 31669.2	28.7 ± 21.9	0.972 ± 0.002
pre-MERL	NBRDF-L	96.57 ± 12551.2	31.3 ± 30.2	0.987 ± 0.0002
pre-MERL	**NBRDF-s**	**1.585 ± 9.598**	**48.7 ± 15.6**	**0.998 ± 7e-6**

Ablation studies. From the third and fourth rows of Table 4.1, the image quality of NBRDF trained over **preprocessed** dataset is higher, confirming the effectiveness of the preprocessing techniques. In addition, the model with a **small** batch size best fits the BRDF, as shown in the last row of Table 4.1.

§ 4.2.2 Efficiency evaluation

The measurements are conducted on a Mac M1. For NBRDF fitting, experiments are conducted on a single-core CPU running at 3.2 GHz. For material generation, the experiments are conducted on a seven-core GPU at 1.28 GHz. To reduce variance, the measured time is averaged **over multiple executions**.

For the proposed NBRDF fitting training time, controlled measurements with different batch sizes are listed in Table 4.2. The smaller the batch size, the longer the training time, but with higher image quality as shown in Table 4.1. However, since the training time is a one-time cost, the additional cost is marginal.

Table 4.2: Training time for NBRDF models in 100 epochs.

batch size	512 (-s)	1024	2048	4096	8000 (-L)
time (s)	763.4 ± 52	342.6 ± 24	205.9 ± 11	112.3 ± 5	**62.7 ± 4**

I compare the computational performance of the NBRDF models with other baseline methods. The rendering speed (million rays per second) is the number of rays rendered per second. The time measured is *solely the rendering function*, excluding the unrelated initialization, IO and many others. For static memory storage, the material is represented by single-precision floating numbers. Table 4.3 shows the averaged numbers of rendered rays per second and memory storage needed, where the **best** results are **bold**.

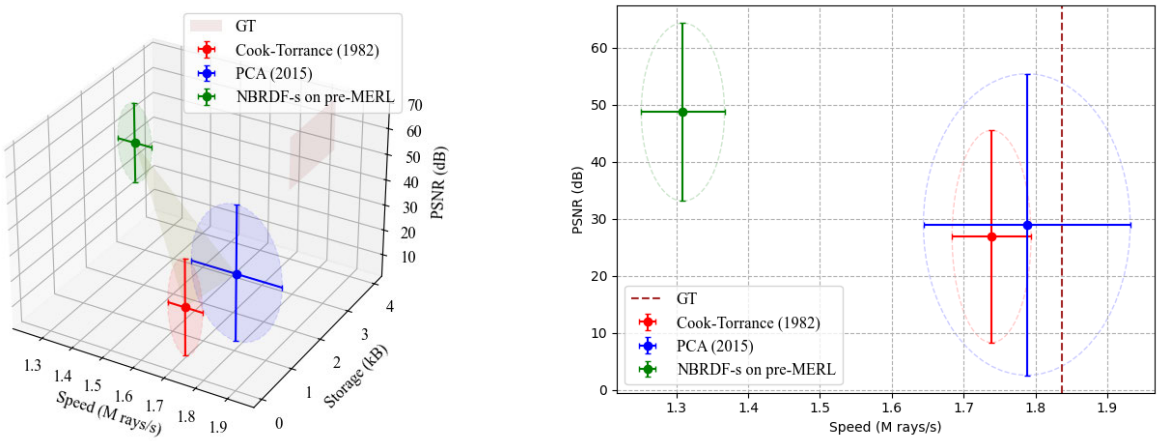
Table 4.3: Computational performance of different BRDF models.

model	speed (M rays/s)	memory (kB)
GT	**1.837 ± 0.037**	35840
PCA	1.788 ± 0.144	1.171875
Cook-Torr	1.739 ± 0.055	**0.03125**
NBRDF	1.309 ± 0.058	2.63672

All the experimented methods compress the BRDF effectively. Note that there is a trade-off between the memory usage and rendering speed, where I assume that no pre-computation, e.g. caching, is allowed. For NBRDF, the compression ratio $CR = \frac{memory_{NBRDF}}{memory_{GT}} = 0.00735\%$. Meanwhile, its rendering speed is yet *competitive* ($71.25\% \pm 3.03\%$ of the ground truth), as the real-time inference time is taken into account.

§ 4.2.3 Discussions

The above evaluations are summarized in Figure 4.2, covering the image quality (↑), rendering speed (↑) and memory usage (↓). The models displayed here are NBRDF-s trained over the pre-MERL dataset with a batch size of 512, PCA (2015) and Cook-Torrance (1982) as the baselines. The ground truth has ∞ PSNR, rendering speed of 1.837 million rays per second and memory usage of 35.84 MB. I pick the PSNR as the image quality metric, which is consistent with the other metrics.



(a) Models performance over related metrics.

(b) Image quality versus rendering speed.

Figure 4.2: Evaluation of models correctness and efficiency.

From the multivariate plot (Figure 4.2), with standard deviational ellipses[4], NBRDF models are able to **outperform** all the baselines in image quality, while maintaining a *competitive* rendering speed with tiny memory usage. The training time is also acceptable with a small batch size, since it's a one-time cost.

§ 4.3 Utility for research

This section evaluates how NBRDF model can be applied in different domains, paving the ways for future research directions. Two main applications are discussed, i.e. super-resolution for low-density BRDF and material synthesis with various modality.

§ 4.3.1 Super-resolution for low-density BRDF

Since the data-driven approach to BRDF modelling consumes a large amount of memory, it would be a great idea to store only the compact neural model and reconstruct the BRDF on-the-fly when needed [GSZ+24]. Hence, I showcase the super-resolution capability of the proposed NBRDF model.

[4]The ellipse semi-minor and semi-major axes represent the standard deviations of the dual metrics, which correspond to the eigenvalues of the two-dimensional data covariance matrix.

As the MLP model captures $\mathbf{y} = \text{FFN}(\theta_h, \theta_d, \phi_d)$, the cardinality of the material reflectance $\mathbf{y}$ is unlimited since the domain of position samples are any arbitrary floating-point numbers. Hence, I experiment with the BRDF reconstruction for low-density input, where the position samples density are decreased x-time in all three dimensions $\theta_h, \theta_d, \phi_d$ from $90 \times 90 \times 180$. Given each dimension x-time lower density and take $x = 1, 2, 4, 8, 16, 32$ as the scaling factor, the number of position samples left are

$$(1 + \lfloor \frac{89}{x} \rfloor) \times (1 + \lfloor \frac{89}{x} \rfloor) \times (1 + \lfloor \frac{179}{x} \rfloor). \tag{4.16}$$

The reconstruction baseline is to fetch BRDF values taking in position indices, which are rounded down to the nearest neighbour. On the other hand, NBRDF model is trained over low-density data. Under the same scene setting, I measure SSIM of rendered images between full-density ground truth and reconstructed BRDF (Table 4.4). **Top** results among the two methods are **bold**.

Table 4.4: Low-density reconstruction comparison (SSIM).

x	# position samples	baseline	NBRDF
1×	$90 \times 90 \times 180$	**1 ± 0**	0.9987 ± 7e-6
2×	$23 \times 23 \times 45$	**0.996 ± 9.3e-5**	**0.996** ± 0.00012
4×	$12 \times 12 \times 23$	0.984 ± 0.001	**0.9934 ± 1.475e-4**
8×	$6 \times 6 \times 12$	0.915 ± 0.024	**0.962 ± 0.0067**
16×	$4 \times 4 \times 8$	0.823 ± 0.0527	**0.896 ± 0.02614**
32×	$3 \times 3 \times 6$	0.745 ± 0.07467	**0.831 ± 0.04942**

The NBRDF model showcases overwhelming well reconstruction capability over the MERL dataset in all ranges of position sample densities. It shows that despite the huge reduction in input density or memory storage in need, the NBRDF model can still reconstruct the BRDF with high fidelity. In addition, this result might be useful in scenarios where high-resolution BRDF collection is not available. Moreover, it also lowers the entry bar and intensity for BRDF measurement.

§ 4.3.2 Material synthesis

§ 4.3.2.1 Data augmentation evaluation

To begin with material synthesis evaluation, I analyze the physical properties of the augmented MERL dataset through colour channel permutation and linear interpolation described in § 3.2.2. The physical properties of the BRDF are listed below,

- **Positivity**:$f_r(\vec{\omega}_i, \vec{\omega}_o) > 0$.
- **Helmholtz reciprocity** *(bidirection)*: $f_r(\vec{\omega}_i, \vec{\omega}_o) = f_r(\vec{\omega}_o, \vec{\omega}_i)$.
- **Energy conservation**: the amount of power leaving the passive materials must be no more than the amount arrived, where the un-scattered energy is in the form of heat.

For RGB channel permutation, only the order of the output array is changed and thus not affecting the above statements. Hence the physical properties are preserved. For interpolation,

the physical properties are also preserved due to the linearity of the augmented BRDFs with the original ones. Hence, it is safe to use the augmented dataset for synthesis tasks.

§ 4.3.2.2 Quantitative evaluation

To evaluate how NBRDF to be used in multi-modal material synthesis (§ 3.5.1, 3.5.3), I evaluate all the unconditional generative models in the hold-out test set. For reference, I also evaluate the performance of the chosen training set, which is the same size as the test set.

The metrics include FID, averaged MMD from the reference set, COV and 1-NNA between the sample and reference set[5] (§ 4.1.3).

I measure the standard FID of rendered images between reference and sample sets, where the Inception Network dimension is 64. It is worth noting that the lower the FID score, the better the generative quality.

For pairwise distances in the MMD, COV and 1-NNA metrics, I propose to use two categories of d_D, namely the BRDF-space d_D^{BRDF} and the rendered image space d_D^{image}. (1) For captured BRDF-space $d_D^{\text{BRDF}} \in \mathbb{R}^{90\times90\times180}$, three different auxiliary loss functions are designed and measured $\mathcal{L}_{L1-log-\cos} = |\log(1+f_1\cos\theta_i) - \log(1+f_2\cos\theta_i)|$, $\mathcal{L}_{L1-log} = |\log(1+f_1) - \log(1+f_2)|$, $\mathcal{L}_{L1} = |f_1 - f_2|$. The MAE of them directly demonstrates the quality without projection into a particular scene through rendering. (2) For single-view image space $d_D^{\text{image}} \in \mathbb{R}^{256\times256\times3}$, RMSE(↓) is used without negation, whereas PSNR and SSIM (↑) scores are negated for shortest distance comparison.

The results are reported in Table 4.5 using my novel metrics[6], where detailed results are reported in Appendix B.2. **In conclusion**, the proposed generative model `hyperdiff` excels in the coverage, fidelity and diversity, where the other baseline methods suffer from non-convergence and mode dropping. The `hyperdiff` model is easier to train and requires less computation and memory, due to the compact input neural BRDF dataset.

Ablation study. I measure the performance of both models, without (`hd-wo-aug`) and with (`hyperdiff`) the data augmentation (§ 3.2.2), on the hold-out test set in Table 4.5. The proposed data augmentation method significantly improves the model performance in terms of coverage, fidelity and diversity.

Efficiency. Each *training* epoch of the hyperdiffusion model takes on average 43.5 seconds on a MacBook M1 laptop. It uses a batch size of 16 and a learning rate of 0.0005 on the dataset consisting of 2,400 NBRDF samples. Under these conditions, the total training time is 2.42 hours for 200 epochs and 8.46 hours for 700 epochs. During the *inference* phase, the total sampling time for 120 NBRDF instances takes 8.22 seconds.

§ 4.3.2.3 Qualitative evaluation

Beside reporting metrics, I demonstrate the rendered images using synthetic materials. The unconditional generation results are shown in Figure 4.3 and 4.9a. To further demonstrate the capability of the proposed framework and results, I render a real-world scene with the selected synthetic materials (Figure 4.3).

Statistically constrained synthesis (§ 3.6) results are shown in Figure 4.4 for each proposed

[5] Following Computer Vision convention, the standard deviation is not reported for generative models.

[6] Note that in the table, model {PCA, VAE}-brdf, {PCA, VAE}-mlp are shorthanded as {PCA, VAE}-b and {PCA, VAE}-m respectively. -PSNR, -SSIM represent negated values are used.

	metrics	training set	PCA-b	PCA-m	VAE-b	VAE-m	hd-wo-aug	hyperdiff
(←) MMD	RMSE×10^2	7.54	33.3	30.2	63.7	15.5	13.4	**9.34**
	-PSNR	−28.7	−13.9	−14.8	−8.30	−20.9	−22.6	**−25.6**
	-SSIM×10	−9.55	−6.74	−6.29	−2.68	−6.86	−8.27	**−9.40**
	L1×10^{-3}	2.51	9.05	9.22	9.09	5.83	4.30	**4.02**
	L1-log	1.05	2.72	1.66	3.70	1.26	**1.16**	**1.21**
	L1-log-cos	0.620	2.35	1.35	3.13	0.857	**0.750**	**0.765**
(→) COV %	RMSE	55.8	18.3	28.3	0.833	16.7	25.0	**50.0**
	-PSNR	56.7	18.3	28.3	0.833	18.3	25.0	**50.0**
	-SSIM	59.2	23.3	16.7	0.833	17.5	22.5	**51.7**
	L1	60.8	2.50	30	0.833	20.8	28.3	**50.8**
	L1-log	63.3	8.33	**30**	1.67	22.5	22.5	27.5
	L1-log-cos	64.2	8.33	**30.8**	0.833	21.7	21.7	27.5
(→)1-NNA %	RMSE	55.4	96.3	93.4	100	93.3	84.6	**60.0**
	-PSNR	55.0	94.2	90.0	100	93.3	84.6	**60.4**
	-SSIM	57.5	96.3	96.7	100	93.8	86.3	**61.7**
	L1	58.8	100	95.4	100	96.7	92.5	**80.0**
	L1-log	98.3	100	**92.1**	100	99.6	100	100
	L1-log-cos	93.3	100	**90.4**	100	99.6	98.8	99.6
	FID (↓)	0.187	10.9	23.8	26.1	10.0	7.56	**0.440**

Table 4.5: Unconditional generative models performance on test set.

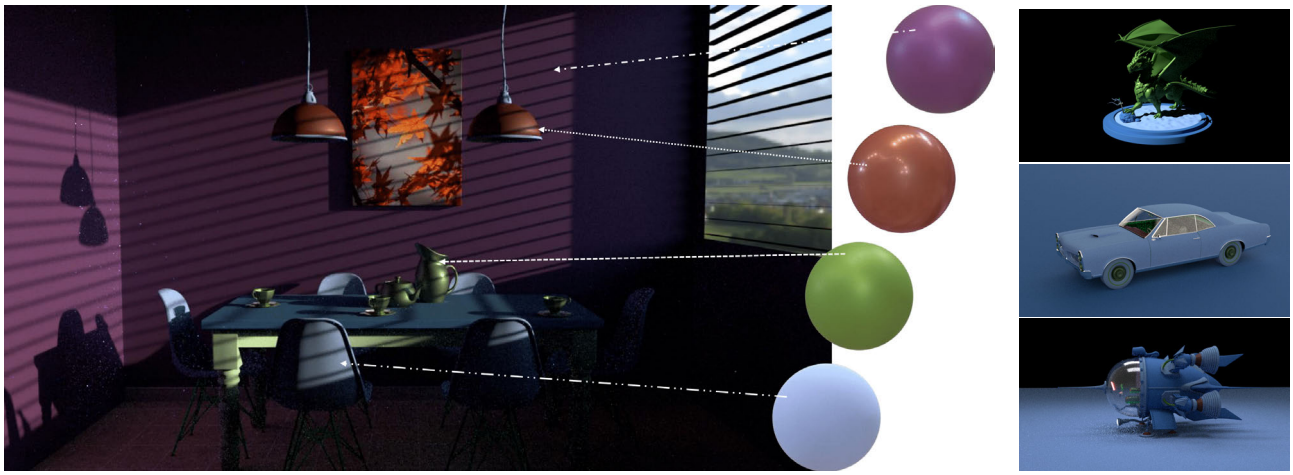


Figure 4.3: Rendered scene via my unconditional synthetic material.

category. Note that these categories do not represent all of the synthetic dataset, but are based on the constraint introduced in § 3.6. The statistically constrained synthesis serves as an alternative approach to control the generation process without re-training the model.

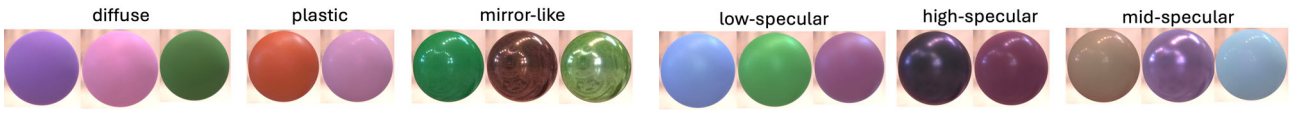


Figure 4.4: Statistically constrained synthesis results.

Multi-modal synthesis (§ 3.5.1) three generated samples with **category** (Figure 4.5) and **text** (Figure 4.6) prompts are shown, together with the rightmost MERL reference material. The results are promising and showcase the usage for multi-modal diffusion-based data-driven BRDF generation.

(a) Cherry-235 (b) Alum

Figure 4.5: Category-conditioned synthetic materials | reference.

(a) Pure oxidized-steel (b) Pure fabric

Figure 4.6: Text-conditioned synthetic materials | reference.

Image-conditioned three generated samples with rightmost reference are shown in Figure 4.7. It could be observed that ResNet embedding alone is not a colour-aware condition. More results are detailed in Appendix B.3.

Synthesis novelty. By measuring the pairwise distance of materials, the closest neighbour in the reference set of each synthetic sample is visualized in Figure 4.9b. From a larger heatmap, the model is not just overfitting to the training set, but also generalizes well to new materials.

§ 4.3.3 Proposed metrics validation

In this section, I explore the effectiveness of the proposed novel generative metrics for tabulated BRDF. I visualize them by plotting the **heatmap** of pairwise distance $d(f, f^\star)$ and confusion matrix (Figure 4.8). The confusion matrix is classified on which set the neighbour belongs to. The nearest reference (blue) neighbour for each synthetic (purple) material is circled (right subfigure). It clearly shows the effectiveness of the proposed metrics in matching the closest BRDF with respect to the reflective behaviour.

I further illustrate the nearest neighbour in the test set (Figure 4.9a) for each unconditional synthetic sample (Figure 4.9b), which is picked for measuring the coverage between two sets. Through the two figures, it can be observed that the proposed metrics are capable of matching the closest reference BRDF in terms of reflective behaviours. I argue that proposed metrics

(a) Alumina oxide (b) Aluminium

Figure 4.7: Image-conditioned synthetic materials | reference.

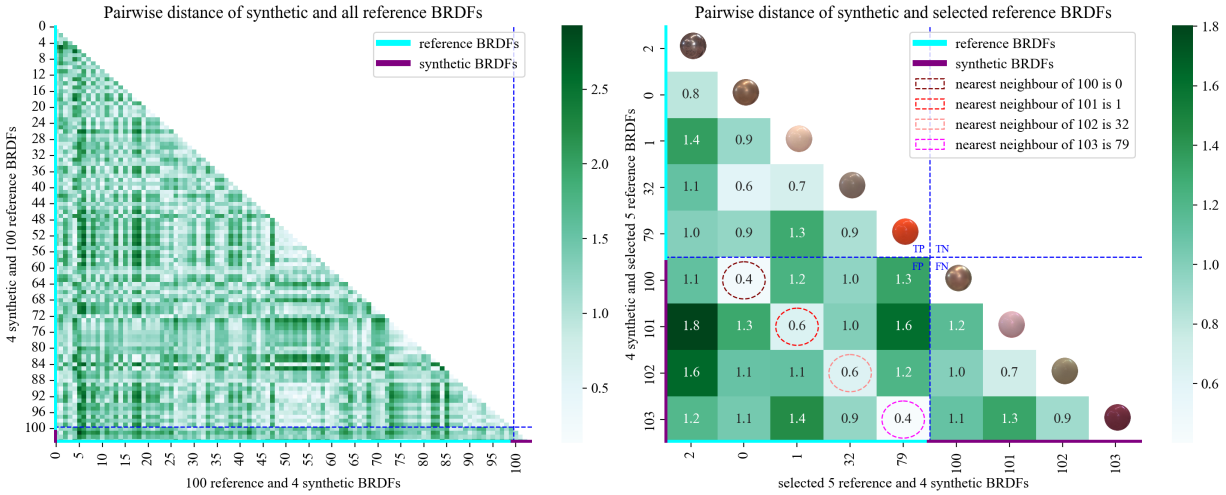
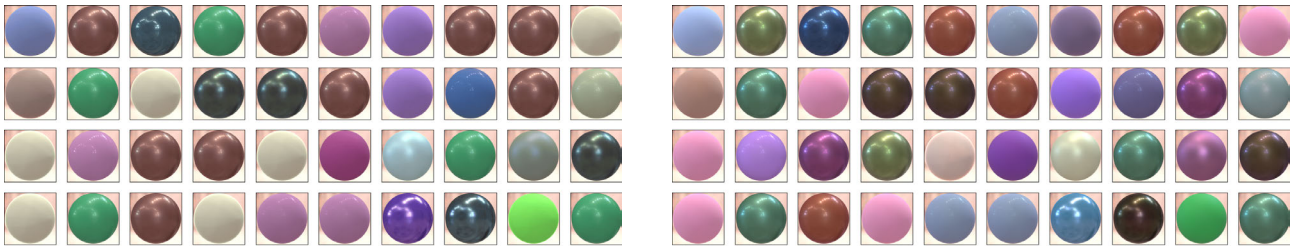


Figure 4.8: Confusion matrix for generative metrics.

based on distance $d_D = MAE(|\log[1 + f_1(p_s)] - \log[1 + f_2(p_s)]|)$ are ideal for tabulated BRDF-space evaluation, especially given a larger reference set.



(a) Unconditional sample set. (b) Nearest neighbour reference.

Figure 4.9: Closest reference for synthetic samples by proposed metric.

Based on the above two observations of heatmap and matched materials visualization, I argue that the proposed novel metrics are appropriate for tabulated BRDF-space evaluation. It serves as a benchmark for further research in material synthesis and analysis, or to be transferred to other domains as well.

§ 4.4 Summary

Firstly, my proposed NBRDF model not only excels in image quality, with competitive rendering speed and memory usage, but also showcases its utility in various research domains, including super-resolution for low-density BRDF, material synthesis and statistically constrained synthesis.

Secondly, the adapted multi-modal diffusion-based material generative model with transformer backbone excels in training cost, coverage and fidelity, when compared against baseline methods quantitatively and qualitatively. The proposed data augmentation method significantly

improves the model performance as well, according to the ablation study.
Last but not least, my formulated generative evaluation metrics are validated and shown to be effective in both tabulated BRDF-space and rendered image space. The results are promising and demonstrate the potential of NBRDF in the field of Computer Graphics and beyond.

5 Conclusions

Science is the process from confusion to understanding that's precise, predictive and reliable.

— Brian Greene

§ 5.1 Completed goals

In summary, all the core components have been completed and extension parts experimented. The dissertation is a comprehensive engineering and research project, as followed,

Rendering pipeline. I choose `Mitsuba3` (`Python` version) as backend and implement the specific renderer BRDF plugins for the selected BRDF format (e.g., `binary`), including data preprocessing pipeline and coordinate transformation. In addition, I implement analytical microfacet-based Cook-Torrance and adapt the PCA-based BRDF as baseline models (extension). After deriving six different importance sampling strategies, such as the uniform hemisphere, Blinn-Phong and microfacet-based, I implement them using appropriate class hierarchy.

Data and augmentation. A complete data processing pipeline is built. I utilize the data-driven BRDF datasets MERL and augment the scarce measured dataset for the BRDF synthesis (extension). The techniques used include RGB channel permutation and linear interpolation, which significantly improve the performance of generative models, as shown in the ablation study.

Neural BRDF design and implementation. I design and implement a neural network architecture (e.g., MLP) to capture the BRDF, mapping from spherical positional inputs to RGB reflectance outputs. I also write the training workflow from scratch, including the data loading pipeline, training/test split, shuffling, loss function and cross validation, etc, while maintaining the capability of the `Pytorch Lightning` module. In addition, the aforementioned workflow lays the foundation for the hyperdiffusion generation training and sampling.

Qualitative and quantitative evaluation on (1) image quality correctness, (2) speed and space effectiveness of the fitted neural BRDF and its utility for downstream tasks, e.g. (3) reconstruction given sparse positional samplings (*super-resolution*) and (4.1) unconditional, (4.2) constrained, (4.3) conditional material synthesis and (4.4) statistically constrained synthesis based on computer graphics theory.

Multi-modal generative model design and implementation. I design and implement the hyperdiffusion (NBRDF-space diffusion) model with multiple embedding layers for conditioning, in addition to different baseline generative models, i.e. PCA-based and VAE-based. I evaluated them thoroughly regarding their fidelity and coverage. As an extension, I design, measure and validate novel BRDF **domain-specific** generative evaluation metrics, filling in the gap in the material synthesis research community (§ 2.3).

Theoretical exploration around material, rendering and neural networks. This project has placed significant emphasis on the first principle and underlying relationship (e.g. literature review, rendering theory, microfacet theory, sampling theory, mathematical derivation and many more).

Hence, through my theoretical analysis and implementation with evaluation, I demonstrate the **correctness, effectiveness and utility** of a new neural representation for the BRDF which otherwise be memory-consuming and challenging to down-stream tasks due to high dimensionality.

§ 5.2 Lessons learnt

Implementing and evaluating the rendering pipeline and neural BRDF, including two distinct applications, have been a giant research and engineering project, with over 7291 lines of `Python` code and different advanced machine learning, sampling, rendering and microfacet theories. Throughout the development, I utilize a series of software engineering, documenting tools and management techniques. The neural representation provides a good tradeoff between memory and fidelity. This new model also facilitates the possibility of sparse sample reconstruction and multi-modal generation.

I perceive the importance of **early prototype with iterative enhancements**, balancing long-term usability with swift novel ideas. For example, I started with a simple MLP prototype. Given the results from Proof of Concept (PoC), I successively refine the UML design and codebase functionalities to address bottlenecks and integrate research explorations.

I am also greatly grateful for all the precious advice from my supervisors, DoS and peers, etc. Without their help, I wouldn't have experienced and learnt so much throughout the year. The natural mixture of math, coding, engineering, proper demonstration techniques (e.g. use of tables, figures and diagrams) and research are the keys to novel fields, including this project and maybe any obstacles in my future career or life.

§ 5.3 Ethical considerations

Neural networks (e.g. generative models) for materials synthesis may cause potential ethical considerations. Firstly, the dataset and condition labels might include biases and misrepresentation, which need to be checked before public release. Secondly, fake and unrealistic information might be distributed by an untrusted person with bad intentions. Therefore, we urge for more regulations and detection algorithms. Last but not least, generative model risks replacing creative artists by artificial intelligence. On the other hand, if deployed properly, these tools benefit various industries with greater accessibility and lower entry barriers.

§ 5.4 Future work

MERL dataset contains only isotropic BRDF and I successfully compress and generate them by neural networks. Due to time constraints, there are more explorations worth considering. The next step is to experiment on anisotropic, spatially-varying BRDF and BTDF, with positional encoding. Afterwards, the complete set of BSDF could be represented and synthesized by neural networks effectively and efficiently. Textures are another potential future direction [KHH26].

The neural BRDF might not be physically accurate. One solution is physics-informed neural networks (PINN). Despite the conditioning capability of generative models, there are still misalignment. Hence, further dataset enlargement and model controlling are needed, e.g. with frequency rectification [ZHO25].

Bibliography

[ASRW17] P. Abrahamsson, Outi Salo, Jussi Ronkainen, and Juhani Warsta. Agile Software Development Methods: Review and Analysis. *ArXiv*, abs/1709.08439, 2017. URL: https://api.semanticscholar.org/CorpusID:18251549. 69

[Bli77] James F. Blinn. Models of light reflection for computer synthesized pictures. In *Proceedings of the 4th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '77, page 192–198, New York, NY, USA, 1977. Association for Computing Machinery. doi:10.1145/563858.563893. 10, 11, 16, 17

[BNC09] R. Baribeau, W. S. Neil, and É. Côté. Development of a robot-based gonioreflectometer for spectral BRDF measurement. *Journal of Modern Optics*, 56(13):1497–1503, 2009. doi:10.1080/09500340903045702. 15

[BP20] James C. Bieron and Pieter Peers. An Adaptive BRDF Fitting Metric. *Computer Graphics Forum*, 39, 2020. URL: https://api.semanticscholar.org/CorpusID:221082251. 25

[CT82] R. L. Cook and K. E. Torrance. A Reflectance Model for Computer Graphics. *ACM Trans. Graph.*, 1(1):7–24, January 1982. doi:10.1145/357290.357293. 11, 16, 17, 25

[DG01] Guido van Rossum David Goodger. Python Docstring Conventions. PEP 257, 2001. URL: https://peps.python.org/pep-0257/. 71

[Dox24] Doxygen documentation generator, 2024. URL: https://www.doxygen.nl/index.html. 71

[FFG12] A. Forés, J. Ferwerda, and J. Gu. Toward a perceptually based metric for BRDF modeling. *Color and Imaging Conference*, 1:142–148, 2012. 39

[GGGm16] Dar'ya Guarnera, Claudio Guarnera, Abhijeet Ghosh, and 2 more authors. BRDF Representation and Acquisition. *Computer Graphics Forum*, pages 625–650, 2016. 14, 15

[GHJV94] Erich Gamma, Richard Helm, Ralph Johnson, and John M. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Softwares*. Addison-Wesley Professional, 1994. 26, 28

[Git24a] Github action runner, 2024. URL: https://github.com/features/actions. 71

[Git24b] Github Kanban Board, 2024. URL: https://docs.github.com/en/issues/planning-and-tracking-with-projects/learning-about-projects/about-projects. 69

[GSZ+24] Fazilet Gokbudak, Alejandro Sztrajman, Chenliang Zhou, Fangcheng Zhong, Rafal Mantiuk, and Cengiz Oztireli. Hypernetworks for Generalizable BRDF Representation. 2024. 42

[HDMR21] Philipp Henzler, Valentin Deschaintre, Niloy J. Mitra, and Tobias Ritschel. Generative modelling of BRDF textures from flash images. *ACM Trans. Graph.*, 40(6), December 2021. doi:10.1145/3478513.3480507. 23

[HGC+20] Bingyang Hu, Jie Guo, Yanjun Chen, Mengtian Li, and Yanwen Guo. DeepBRDF: A Deep Representation for Manipulating Measured BRDF. *Computer Graphics Forum*, 2020. doi:10.1111/cgf.13920. 23

[HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models, 2020. URL: https://arxiv.org/abs/2006.11239, arXiv:2006.11239. 11, 19, 20, 32

[HRU+17] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6629–6640, Red Hook, NY, USA, 2017. Curran Associates Inc. 38

[HS22] Jonathan Ho and Tim Salimans. Classifier-Free Diffusion Guidance, 2022. URL: https://arxiv.org/abs/2207.12598, arXiv:2207.12598. 33

[HSK+12] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *ArXiv*, abs/1207.0580, 2012. URL: https://api.semanticscholar.org/CorpusID:14832074. 30

[HvDM+13] John F. Hughes, Andries van Dam, Morgan McGuire, David F. Sklar, James D. Foley, Steven K. Feiner, and Kurt Akeley. *Computer Graphics: Principles and Practice*. Addison-Wesley Professional, 3rd edition, 2013. © 2014 by Addison-Wesley. 15

[HZ10] Alain Horé and Djemel Ziou. Image Quality Metrics: PSNR vs. SSIM. In *2010 20th International Conference on Pattern Recognition*, pages 2366–2369, 2010. doi:10.1109/ICPR.2010.579. 37

[Jak10] Wenzel Jakob. Mitsuba Renderer, 2010. URL: http://www.mitsuba-renderer.org/index_old.html. 21

[JSRV22a] Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, and Delio Vicini. Dr.Jit: A Just-In-Time Compiler for Differentiable Rendering. *Transactions on Graphics (Proceedings of SIGGRAPH)*, 41(4), July 2022. URL: http://www.mitsuba-renderer.org/, doi:10.1145/3528223.3530099. 21

[JSRV22b] Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, and Delio Vicini. Pearson's chi-square test method for comparing Monte Carlo sampling strategy., July 2022. URL: https://mitsuba.readthedocs.io/en/stable/src/developer_guide/testing.html. 70

[KgLG+24] Zhifeng Kong, Sang gil Lee, Deepanway Ghosal, Navonil Majumder, Ambuj Mehrish, Rafael Valle, Soujanya Poria, and Bryan Catanzaro. Improving Text-To-Audio Models with Synthetic Captions, 2024. URL: https://arxiv.org/abs/2406.15487, arXiv:2406.15487. 11

[KHH26] Albert Kwok, Zheyuan Hu, and Dounia Hammou. Implicit neural representation of textures, 2026. URL: https://arxiv.org/abs/2602.02354, arXiv:2602.02354. 50

[KPH+21] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. DiffWave: A Versatile Diffusion Model for Audio Synthesis, 2021. URL: https://arxiv.org/abs/2009.09761, arXiv:2009.09761. 11

[KVG+19] Ivo Kondapaneni, Petr Vevoda, Pascal Grittmann, Tomáš Skřivan, Philipp Slusallek, and Jaroslav Křivánek. Optimal multiple importance sampling. *ACM Trans. Graph.*, 38(4), July 2019. doi:10.1145/3306346.3323009. 58

[KW22] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes, 2022. URL: https://arxiv.org/abs/1312.6114, arXiv:1312.6114. 18

[Lam60] Johann Heinrich Lambert. *Photometria, sive de mensura et gradibus luminis, colorum et umbrae*. Eberhard Klett, 1760. URL: https://books.google.co.uk/books?id=tbM6AAAAcAAJ. 11, 15, 17, 39

[LFTG97] Eric P. F. Lafortune, Sing-Choong Foo, Kenneth E. Torrance, and Donald P. Greenberg. Non-linear approximation of reflectance functions. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, page 117–126, USA, 1997. ACM Press/Addison-Wesley Publishing Co. doi:10.1145/258734.258801. 10, 11, 16

[LPO17] David Lopez-Paz and Maxime Oquab. Revisiting Classifier Two-Sample Tests. In *International Conference on Learning Representation (ICLR) 2017*, 2017. 39

[LZCS14] Mu Li, Tong Zhang, Yuqiang Chen, and Alexander J. Smola. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, page 661–670, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2623330.2623612. 30

[MCS23] Sean Memery, Osmar Cedron, and Kartic Subr. Generating Parametric BRDFs from Natural Language Descriptions. *Computer Graphics Forum*, 42(7):e14980,

2023. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14980, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14980, doi:10.1111/cgf.14980. 23

[ML06] Claus B. Madsen and Rune E. Laursen. A Comparison of Techniques for Approximating Full Image-Based Lighting. 2006. URL: https://api.semanticscholar.org/CorpusID:14189781. 40

[MPBM03] W. Matusik, H. Pfister, M. Brand, and L. McMillan. A Data-Driven Reflectance Model. *ACM Transactions on Graphics (TOG)*, 22(3):759–769, July 2003. URL: https://www.merl.com/publications/TR2003-83, doi:10.1145/882262.882343. 10, 11, 15, 26

[NDR+22] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models, 2022. URL: https://arxiv.org/abs/2112.10741, arXiv:2112.10741. 11

[Nic65] Fred E. Nicodemus. Directional Reflectance and Emissivity of an Opaque Surface. *Appl. Opt.*, 4(7):767–775, Jul 1965. URL: https://opg.optica.org/ao/abstract.cfm?URI=ao-4-7-767, doi:10.1364/AO.4.000767. 14

[NJR15] Jannik Boll Nielsen, Henrik Wann Jensen, and Ravi Ramamoorthi. On Optimal, Minimal BRDF Sampling for Reflectance Acquisition. *ACM Transactions on Graphics (TOG)*, 34(6):186:1–186:11, November 2015. doi:10.1145/2816795.2818085. 18, 26

[PAG18] Ioannis Mitliagkas Panos Achlioptas, Olga Diamanti and Leonidas Guibas. Learning Representations and Generative Models for 3D Point Clouds. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018. 39

[PG17] Josh Patterson and Adam Gibson. *Understanding Learning Rates*. O'Reilly, 2017. 30

[Pho98] Bui Tuong Phong. *Illumination for computer generated pictures*, page 95–101. Association for Computing Machinery, New York, NY, USA, 1998. URL: https://doi.org/10.1145/280811.280980. 10, 11, 15, 17

[PJH16] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2016. 13

[Pyl24] Pylint: Python standard static code analyser, 2024. URL: https://pypi.org/project/pylint/. 71

[Rud16] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016. 29

[Rus98] Szymon M. Rusinkiewicz. A New Change of Variables for Efficient BRDF Representation. In George Drettakis and Nelson Max, editors, *Rendering Techniques '98*, pages 11–22, Vienna, 1998. Springer Vienna. 27

[SFH+25] Chong Su, Yingbin Fu, Zheyuan Hu, Jing Yang, Param Hanji, Shaojun Wang, Xuan Zhao, Cengiz Öztireli, and Fangcheng Zhong. CHOrD: Generation of Collision-Free, House-Scale, and Organized Digital Twins for 3D Indoor Scenes with Controllable Floor Plans and Optimal Layouts, 2025. URL: https://arxiv.org/abs/2503.11958, arXiv:2503.11958. 11

[SME22] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising Diffusion Implicit Models, 2022. URL: https://arxiv.org/abs/2010.02502, arXiv:2010.02502. 33

[SRRW21] Alejandro Sztrajman, Gilles Rainer, Tobias Ritschel, and Tim Weyrich. Neural BRDF Representation and Importance Sampling. *Computer Graphics Forum*, 2021. doi:10.1111/cgf.14335. 10, 11, 17

[TAM08] Naty Hoffman Tomas Akenine-Möller, Eric Haines. *Real-Time Rendering*. A K Peters/CRC Press, New York, 3rd edition, 2008. eBook Published 18 January 2019. doi:10.1201/9781315365459. 13

[vRWC01] Guido van Rossum, Barry Warsaw, and Nick Coghlan. Style Guide for Python Code. PEP 8, 2001. URL: https://www.python.org/dev/peps/pep-0008/. 69

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. 20, 21

[WBSS04] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. doi:10.1109/TIP.2003.819861. 37

[XLPD23] Xudong Xu, Zhaoyang Lyu, Xingang Pan, and Bo Dai. MATLABER: Material-Aware Text-to-3D via LAtent BRDF auto-EncodeR, 2023. URL: https://arxiv.org/abs/2308.09278, arXiv:2308.09278. 23

[ZHO25] Chenliang Zhou, Zheyuan Hu, and Cengiz Oztireli. FreNBRDF: A Frequency-Rectified Neural Material Representation, 2025. URL: https://arxiv.org/abs/2507.00476, arXiv:2507.00476. 50

[ZHS+24] Chenliang Zhou, Zheyuan Hu, Alejandro Sztrajman, Yancheng Cai, Yaru Liu, and Cengiz Oztireli. NeuMaDiff: Neural Material Synthesis via Hyperdiffusion, 2024. URL: https://arxiv.org/abs/2411.12015, arXiv:2411.12015. 23

[ZHS+26] Chenliang Zhou, Zheyuan Hu, Alejandro Sztrajman, Yancheng Cai, Yaru Liu, and Cengiz Oztireli. M3ashy: Multi-modal material synthesis via hyperdiffusion. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(16):13575–13583, Mar. 2026. URL: https://ojs.aaai.org/index.php/AAAI/article/view/38363, doi:10.1609/aaai.v40i16.38363. 23

[ZPX+24] Shangzhan Zhang, Sida Peng, Tao Xu, Yuanbo Yang, Tianrun Chen, Nan Xue, Yujun Shen, Hujun Bao, Ruizhen Hu, and Xiaowei Zhou. MaPa: Text-driven Photorealistic Material Painting for 3D Shapes, 2024. URL: https://arxiv.org/abs/2404.17569, arXiv:2404.17569. 11

A Importance sampling derivation

This chapter describes the theory of real-time rendering. It includes the derivation of sampling strategies for efficient computational integration convergence. Among these, *microfacet-based* sampling is the **state-of-the-art** (**SOTA**) in representing real-world BRDF materials.

§ A.1 Fundamental theory

This section discusses the main theory for importance sampling strategies. After that, I review the three kinds of basic hemisphere sampling methods and then explore the advanced microfacet-based type.

§ A.1.1 Rendering

Solid angle is denoted as $\vec{\omega}$ and its **spherical coordinates** are (r, ϕ, θ), where $r = 1$ for a unit sphere, azimuthal angle (polar angle with the bi-normal on the surface) $\phi \in [0, 2\pi]$ and zenith angle (polar angle with the surface normal) $\theta \in [0, \frac{\pi}{2}]$. It's a standard result from Calculus that the surface element $dS = r^2 \sin\theta \, d\theta \, d\phi$. By the *definition of solid angle* that $d\vec{\omega} = \frac{dS}{r^2}$, then

$$d\vec{\omega} = \sin\theta \, d\phi \, d\theta. \tag{A.1}$$

Plugging in (A.1), I get the relationship between the **joint** pdf for θ, ϕ and pdf for solid angle.

$$\text{pdf}(\phi, \theta) = \text{pdf}(\vec{\omega}) \sin\theta. \tag{A.2}$$

Given the incoming direction from camera $\vec{\omega}_i$, BRDF $f_r(\vec{\omega}_i, \vec{\omega}_o)$ and no emitted light at the hit point, $L_e(\vec{x}) = 0$, the main task of the renderer is to *sample the outgoing direction* $\vec{\omega}_o$ at each hit point $\vec{x}$ to compute the radiance $L_i(\vec{x})$ by the **rendering equation**,

$$L_i(\vec{x}) = \int_{H^2} f_r(\vec{\omega}_i, \vec{\omega}_o) L_o(\vec{x}) \cos\theta_o \, d\vec{\omega}_o. \tag{A.3}$$

To render scenes efficiently in real time, importance sampling would help the integrand **converges faster** in the limited Monte Carlo samples per pixel (spp).

§ A.1.2 Computational integration

Monte Carlo integration approximates the integral $F = \int_{x=a}^{b} f(x) \, dx$ by numerically computing the average of the integrand $f(x_k)$ at random samples x_k,

$$F \approx (b - a) \frac{1}{N} \sum_{k=1}^{N} f(x_k), \tag{A.4}$$

where the index k ranges from 1 to the total number of samples N.
It's an unbiased estimator of the integral with variance $\sigma^2 \propto \frac{1}{N}$, since the expected value of the estimator equals the true value of the integral,

$$\mathbb{E}[\frac{b - a}{N} \sum_{k=1}^{N} f(x_k)] = \int_{x=a}^{b} f(x) \, dx = F. \tag{A.5}$$

Importance sampling enhances the above numerical integration process with sampling den-

sity function $\text{pdf}(x)$, resulting in

$$\langle F^N \rangle = \frac{1}{N} \sum_{k=1}^{N} \frac{f(x_k)}{\text{pdf}(x_k)}. \tag{A.6}$$

The sampling density function $\text{pdf}(x)$ is typically chosen to be proportional to the integrand $f(x)$ for lower variance and thus faster convergence. Similarly, I *sample the outgoing direction* $\vec{\omega}_o$ with $\text{pdf}(\vec{\omega}_o)$ over the hemisphere H^2,

$$L_i(\vec{x}) \approx \langle L_i(\vec{x})^N \rangle = \frac{1}{N} \sum_{k=1}^{N} \frac{f(\vec{\omega}_{i,k}, \vec{\omega}_{o,k}) L_o(\vec{x}) \cos\theta_{o,k}}{\text{pdf}(\vec{\omega}_{o,k})}. \tag{A.7}$$

Multiple importance sampling (MIS) [KVG+19] incorporates N sub-sampling strategies, which are designed to match the different aspects of the integrand $F = \int_{x=a}^{b} f(x)\,dx$. The estimator is a weighted sum of these N sub-estimators,

$$\langle F^N \rangle^{\text{MIS}} = \frac{1}{N} \sum_{k=1}^{N} \frac{1}{n_k} \sum_{p=1}^{n_k} \omega_k(x_{k,p}) \frac{f(x_{k,p})}{\text{pdf}_k(x_{k,p})}. \tag{A.8}$$

To keep the estimator unbiased, the weight $w_k(x_k)$ is subject to the following constraints,

$$\begin{aligned} \omega_k(x) &= 0, \quad \text{if } \text{pdf}_k(x) = 0. \\ \sum_{i=1}^{N} \omega_k(x) &= 0, \quad \text{if } f(x) \neq 0. \end{aligned} \tag{A.9}$$

MIS is ideal for rendering glossy reflection with multiple light sources, where the integrand can be approximated from multiple importance sampling with respect to the BSDF f_s and light sources L_o.

§ A.1.3 Probability theory

The density function has the property of **"sum to one"** over the domain, the latter is by transformation (A.1),

$$\int_{H^2} \text{pdf}(\vec{\omega}) d\vec{\omega} = \int_{\phi=0}^{2\pi} \int_{\theta=0}^{\frac{\pi}{2}} \text{pdf}(\phi, \theta)\, d\phi\, d\theta = 1. \tag{A.10}$$

Joint probability density function (pdf) for θ and ϕ are defined in (A.2), I could derive the **marginal** probability density function (pdf) for one of them by integrating over the other,

$$\text{pdf}(\theta) = \int_{0}^{2\pi} \text{pdf}(\phi, \theta)\, d\phi, \text{ for } \theta \in [0, \frac{\pi}{2}]. \tag{A.11}$$

The **conditional** probability density function (pdf) for ϕ conditioned on *theta* could be derived from the *Bayes' rule*,

$$\text{pdf}(\phi|\theta) = \frac{\text{pdf}(\phi, \theta)}{\text{pdf}(\theta)}, \text{ for } \phi \in [0, 2\pi]. \tag{A.12}$$

To find the `sample()` function for θ, ϕ, I set the **cumulative distribution function (cdf)** of them to a random number $\xi \in [0, 1]$ and get the inverted formula for the spherical coordinate variables,

$$P(\theta' \leq \theta) = \int_{0}^{\theta} \text{pdf}(\theta)\, d\theta = \xi_0, \text{ for } \theta \in [0, \frac{\pi}{2}]. \tag{A.13}$$

§ A.1.4 Microfacet Normal Distribution Function (NDF)

For microfacet-based BRDFs, given the halfway vector $\vec{h}$, it should satisfy the following constraint, where θ is the angle between the normal vector and $\vec{h}$.

$$\int_{H^2} D(\vec{h}) \cos\theta \, dh = 1. \tag{A.14}$$

Considering (A.10), the sampling is defined to be proportional to the NDF $D(\vec{h})$.

$$\text{pdf}(\vec{\omega}_o) = D(\vec{h}) \cos\theta. \tag{A.15}$$

Microfacet	Normal Distribution Function
Blinn-Phong	$D(\vec{h}) = \frac{(\alpha+2)\cos\theta^\alpha}{2\pi}$
Beckmann	$D(\vec{h}) = \frac{1}{\alpha^2 \cos^4\theta} e^{\frac{-\tan^2\theta}{\alpha^2}}$
GGX	$D(\vec{h}) = \frac{\alpha^2}{\pi \cos^4\theta(\alpha^2+\tan^2\theta)^2}$

§ A.2 Basic sampling derivation

There are three hemisphere sampling strategies widely used as entry level, namely uniform, cosine-weighted and power cosine-weighted sampling.

§ A.2.1 Uniform hemisphere sampling

The probability **density** function for solid angle $\vec{\omega}_o \in H^2$ is a constant κ, by (A.10) and $\text{pdf}(\vec{\omega}_o) = \kappa$, $\int_{H^2} \kappa \, d\vec{\omega} = 1$. Hence, $\kappa = \frac{1}{\int_{H^2} d\vec{\omega}}$. The surface area of a hemisphere is $S = 2\pi r^2$. By definition of solid angle, the denominator is $\int_{H^2} d\vec{\omega} = \frac{S}{r^2} = 2\pi$. Hence, $\kappa = \frac{1}{2\pi}$ and $\text{pdf}(\vec{\omega}_o) = \frac{1}{2\pi}$.

Plugging the solid angle pdf into (A.2), the joint $\text{pdf}(\phi, \theta) = \text{pdf}(\vec{\omega}_o) \sin\theta = \frac{\sin\theta}{2\pi}$. The **marginal** $\text{pdf}(\theta) = \int_0^{2\pi} \text{pdf}(\phi, \theta) \, d\phi = \int_0^{2\pi} \frac{\sin\theta}{2\pi} \, d\phi = \sin\theta$, for $\theta \in [0, \frac{\pi}{2}]$, by (A.11). Bayes' rule (A.12) says that **conditional** $\text{pdf}(\phi|\theta) = \frac{\text{pdf}(\phi,\theta)}{\text{pdf}(\theta)} = \frac{1}{2\pi}$, for $\phi \in [0, 2\pi]$.

To find the `sample()` function for θ, ϕ, I set the cdf of them, by (A.13), to random numbers $\xi_i \in [0, 1]$,

$$P(\theta' \leq \theta) = \int_0^\theta \sin\theta' \, d\theta' = 1 - \cos\theta = \xi_0. \tag{A.16}$$

$$P(\phi|\theta') = \int_0^\phi \frac{1}{2\pi} \, d\phi = \frac{\phi}{2\pi} = \xi_2. \tag{A.17}$$

Hence, I could invert the above formula to get the `sample()` function needed by assigning a new random number $\xi_1 = 1 - \xi_0$,

$$\vec{\omega}_o = (\theta, \phi) = (\arccos(\xi_1), 2\pi\xi_2). \tag{A.18}$$

§ A.2.2 (Power) cosine-weighted sampling

The latter two sampling strategies are used for *diffuse objects*. They could be combined, where the cosine-weighted sampling is a special case when the power coefficient $\alpha = 1$.

Diffuse object has a constant brdf in the rendering equation (A.3), then probability **density** function $\text{pdf}(\vec{\omega}_o) \propto \cos^\alpha\theta$ would be a wiser choice fitting the integrand.

By (A.10) and $\text{pdf}(\vec{\omega}_o) = \kappa \cos^\alpha \theta$, $\int_{\phi=0}^{2\pi} \int_{\theta=0}^{\frac{\pi}{2}} \kappa \cos^\alpha \theta \sin\theta \, d\phi \, d\theta = 1$. Hence, $\kappa = \frac{-1}{2\pi \int_{\theta=0}^{\frac{\pi}{2}} \kappa \cos^\alpha \theta \, dcos\theta}$ $= \frac{-(alpha+1)}{2\pi[\cos^{\alpha+1}\theta]_0^{\frac{\pi}{2}}} = \frac{(\alpha+1)}{2\pi}$ and $\text{pdf}(\vec{\omega}_o) = \frac{(\alpha+1)}{2\pi} \cos^\alpha \theta$.

Plugging the solid angle pdf into (A.2), the joint $\text{pdf}(\phi, \theta) = \text{pdf}(\vec{\omega}_o) \sin\theta = \frac{(\alpha+1)}{2\pi} \cos^\alpha \theta \sin\theta$. The **marginal** $\text{pdf}(\theta) = \int_0^{2\pi} \frac{(\alpha+1)}{2\pi} \cos^\alpha \theta \sin\theta \, d\phi = (\alpha+1) \cos^\alpha \theta \sin\theta$, for $\theta \in [0, \frac{\pi}{2}]$, by (A.11). Bayes' rule (A.12) says that **conditional** $\text{pdf}(\phi|\theta) = \frac{\text{pdf}(\phi,\theta)}{\text{pdf}(\theta)} = \frac{1}{2\pi}$, for $\phi \in [0, 2\pi]$, which remains the same as above.

For `sample()`, I set the cdf of θ, by (A.13), to random numbers $\xi_i \in [0, 1]$, while for ϕ, the steps are the same,

$$P(\theta' \leq \theta) = \int_0^\theta (\alpha+1) \cos^\alpha \theta' \sin\theta' \, d\theta' = 1 - \cos^{\alpha+1}\theta = \xi_0. \tag{A.19}$$

Hence, by inverting the above formula and assigning a new random number $\xi_1 = 1 - \xi_0$,

$$\vec{\omega}_o = (\theta, \phi) = (\arccos(\xi_1^{\frac{1}{\alpha+1}}), 2\pi\xi_2). \tag{A.20}$$

§ A.2.3 Summary

With two random numbers $\xi_1, \xi_2 \in [0, 1]$ and functions `sample()` defined in the below table, the outgoing directions are sampled by

$$\vec{\omega}_o = (\theta, \phi) = (sample(\xi_1), 2\pi\xi_2). \tag{A.21}$$

Table A.2: Basic hemisphere sampling functions.

Hemisphere	Solid angle pdf	Spherical sample
Uniform	$\text{pdf}(\vec{\omega}_o) = \frac{1}{2\pi}$	$\theta = \arccos(\xi_1)$
Cosine-weighted	$\text{pdf}(\vec{\omega}_o) = \frac{\cos\theta}{\pi}$	$\theta = \arccos(\sqrt{\xi_1})$
Power cosine	$\text{pdf}(\vec{\omega}_o) = \frac{(\alpha+1)\cos^\alpha\theta}{2\pi}$	$\theta = \arccos(\xi_1^{\frac{1}{\alpha+1}})$

§ A.3 Microfacet sampling derivation

As introduced in (A.15), the probability **density** function for solid angle is defined to be $\text{pdf}(\vec{\omega}_o) = D(\vec{h}) \cos\theta$.

§ A.3.1 Blinn-Phong

By (A.15), $\text{pdf}(\vec{\omega}_o) = \frac{(\alpha+2)\cos^{\alpha+1}\theta}{2\pi}$. The joint $\text{pdf}(\phi, \theta) = \text{pdf}(\vec{\omega}_o) \sin\theta$, by (A.2).

The **marginal** $\text{pdf}(\theta) = \int_0^{2\pi} \text{pdf}(\phi, \theta) \, d\phi = (\alpha+2) \cos^{\alpha+1}\theta \sin\theta$, for $\theta \in [0, \frac{\pi}{2}]$, by (A.11). Bayes' rule (A.12) says that **conditional** $\text{pdf}(\phi|\theta) = \frac{\text{pdf}(\phi,\theta)}{\text{pdf}(\theta)} = \frac{1}{2\pi}$, for $\phi \in [0, 2\pi]$.

To find the `sample()` function for θ, ϕ I set the cdf of it, by (A.13), to random number $\xi_i \in [0, 1]$,

$$P(\theta' \leq \theta) = \int_0^\theta (\alpha+2) \cos^{\alpha+1}\theta \sin\theta \, d\theta' = 1 - \cos^{\alpha+2}\theta = \xi_0, \tag{A.22}$$

$$P(\phi|\theta') = \int_0^\phi \frac{1}{2\pi} \, d\phi = \frac{\phi}{2\pi} = \xi_2. \tag{A.23}$$

Hence, I could invert the above formula to get the `sample()` function needed by assigning a

new random number $\xi_1 = 1 - \xi_0$,

$$\vec{\omega}_o = (\theta, \phi) = (\arccos(\xi_1^{\frac{1}{\alpha+2}}), 2\pi\xi_2). \tag{A.24}$$

§ A.3.2 Beckmann

$\text{pdf}(\vec{\omega}_o) = \frac{1}{\alpha^2 \cos^3\theta} e^{\frac{-\tan^2\theta}{\alpha^2}}$, by (A.15). The joint $\text{pdf}(\phi, \theta) = \text{pdf}(\vec{\omega}_o) \sin\theta$, by (A.2).
The **marginal** $\text{pdf}(\theta) = \int_0^{2\pi} \text{pdf}(\phi, \theta)\, d\phi = \frac{2\pi \sin\theta}{\alpha^2 \cos^3\theta} e^{\frac{-\tan^2\theta}{\alpha^2}}$, for $\theta \in [0, \frac{\pi}{2}]$, by (A.11). **Conditional** $\text{pdf}(\phi|\theta) = \frac{\text{pdf}(\phi,\theta)}{\text{pdf}(\theta)} = \frac{1}{2\pi}$ remains the same, for $\phi \in [0, 2\pi]$.
To find the `sample()` function for θ, I set the cdf of it, by (A.13), to random number $\xi_i \in [0, 1]$, notice that $d[-e^{\frac{-\tan^2\theta}{\alpha^2}}] = 2\frac{1}{\alpha^2} e^{\frac{-\tan^2\theta}{\alpha^2}} \tan\theta \cdot d\tan\theta$ and $d\tan\theta = \frac{1}{\cos^2\theta}$,

$$P(\theta' \le \theta) = \int_0^{\theta} \frac{2\pi \sin\theta}{\alpha^2 \cos^3\theta} e^{\frac{-\tan^2\theta}{\alpha^2}}\, d\theta' = \pi(1 - e^{\frac{-\tan^2\theta}{\alpha^2}}) = \xi_1. \tag{A.25}$$

Hence, the above formula is inverted to get the `sample()` function,

$$\vec{\omega}_o = (\theta, \phi) = (\arctan(\sqrt{-\alpha^2 \ln(1 - \frac{\xi_1}{\pi})}), 2\pi\xi_2). \tag{A.26}$$

§ A.3.3 GGX

$\text{pdf}(\vec{\omega}_o) = \frac{\alpha^2}{\pi \cos^3\theta(\alpha^2 + \tan^2\theta)^2}$, by (A.15). The joint $\text{pdf}(\phi, \theta) = \text{pdf}(\vec{\omega}_o) \sin\theta$, by (A.2).
The **marginal** $\text{pdf}(\theta) = \int_0^{2\pi} \text{pdf}(\phi, \theta)\, d\phi = \frac{2\alpha^2 \sin\theta}{\cos^3\theta(\alpha^2 + \tan^2\theta)^2}$, for $\theta \in [0, \frac{\pi}{2}]$, by (A.11). **Conditional** $\text{pdf}(\phi|\theta) = \frac{\text{pdf}(\phi,\theta)}{\text{pdf}(\theta)} = \frac{1}{2\pi}$ remains the same, for $\phi \in [0, 2\pi]$.
To find the `sample()` function for θ, I set the cdf of it, by (A.13), to random number $\xi_i \in [0, 1]$, notice that $d[(\alpha^2 + \tan^2\theta)^{-1}] = \frac{-2\tan\theta \cdot d\tan\theta}{(\alpha^2 + \tan^2\theta)^2}$ and $d\tan\theta = \frac{1}{\cos^2\theta}$,

$$P(\theta' \le \theta) = \int_0^{\theta} \frac{2\alpha^2 sin\theta}{\cos^3\theta(\alpha^2 + \tan^2\theta)^2}\, d\theta' = 1 - \frac{\alpha^2}{\alpha^2 + \tan^2\theta} = \xi_1. \tag{A.27}$$

Hence, the above formula is inverted to get the `sample()` function,

$$\vec{\omega}_o = (\theta, \phi) = (\arctan(\frac{\alpha\sqrt{\xi_1}}{\sqrt{1 - \xi_1}}), 2\pi\xi_2). \tag{A.28}$$

§ A.3.4 Summary

With two random numbers $\xi_1, \xi_2 \in [0, 1]$ and functions `sample()` defined in the below table, the outgoing directions are sampled by equation (A.21).

Table A.3: Microfacet sampling functions.

Microfacet	Solid angle pdf	Spherical sample
Blinn-Phong	$\text{pdf}(\vec{\omega}_o) = \frac{(\alpha+2)\cos^{\alpha+1}\theta}{2\pi}$	$\theta = \arccos(\xi_1^{\frac{1}{\alpha+2}})$
Beckmann	$\text{pdf}(\vec{\omega}_o) = \frac{1}{\alpha^2 \cos^3\theta} e^{\frac{-\tan^2\theta}{\alpha^2}}$	$\theta = \arctan(\sqrt{-\alpha^2 \ln(1 - \frac{\xi_1}{\pi})})$
GGX	$\text{pdf}(\vec{\omega}_o) = \frac{\alpha^2}{\pi \cos^3\theta(\alpha^2 + \tan^2\theta)^2}$	$\theta = \arctan(\frac{\alpha\sqrt{\xi_1}}{\sqrt{1-\xi_1}})$

B Additional results

§ B.1 RGB permutation

RGB-channel permutation and interpolation augmentation are shown respectively in the first six and last two mosaic images of Figure B.1.

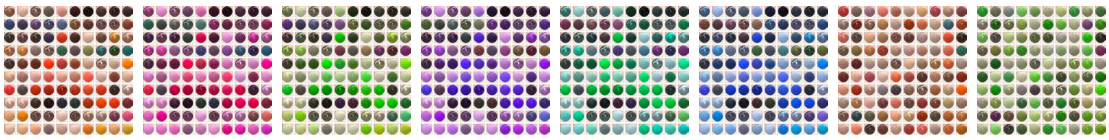


Figure B.1: Augmented MERL dataset § 3.2.2.

§ B.2 Unconditional generation evaluation details

I include the full statistics for different unconditional material generation methods. The diversity and fidelity are measured by FID, averaged MMD from the reference set (MMD-ref) and from the sample set (MMD-sam), COV and 1-NNA between the sample and reference set[1]. The pairwise distance is measured by two types of metrics, namely captured BRDF-space $\mathbb{R}^{90\times90\times180}$ and image-space $\mathbb{R}^{256\times256\times3}$. The former distances include $\mathcal{L}_{L1} = |f_1 - f_2|$, $\mathcal{L}_{L1-log} = |\log(1+f_1) - \log(1+f_2)|$ and $\mathcal{L}_{L1-log-cos} = |\log(1+f_1 cos\theta_i) - \log(1+f_2 cos\theta_i)|$, while the PSNR and SSIM (↑) scores are negated for shortest distance comparison.

§ B.2.1 Proposed methods

For hyperdiffusion (200 epochs), the FID value is 0.4395 and the other results are measured in Table B.1.

Table B.1: Unconditional hyperdiffusion-200 performance on test set.

Distance	MMD-ref	MMD-sam	COV (↑)	1-NNA(50%)
L1-log-cos (↓)	0.7184	0.7139	46.67%	99.17%
L1-log (↓)	1.1532	1.2195	45.00%	100.0%
L1 (↓)	4019.1	3070.0	50.83%	80.00%
RMSE (↓)	0.0934	0.0843	49.17%	59.17%
PSNR (↑)	25.354	25.790	50.00%	60.42%
SSIM (↑)	0.9396	0.9456	51.67%	61.67%

For hyperdiffusion (700 epochs), the FID value is 0.5634 and the other results are measured in Table B.2.

[1]Following Computer Vision convention, standard deviation is not reported for generative models.

Table B.2: Unconditional hyperdiffusion-700 performance on test set.

Distance	MMD-ref	MMD-sam	COV (↑)	1-NNA(50%)
L1-log-cos (↓)	0.7653	0.6651	27.50%	99.58%
L1-log (↓)	1.2048	1.1465	27.50%	100.0%
L1 (↓)	4146.7	2085.8	29.17%	84.17%
RMSE (↓)	0.1062	0.0841	32.50%	77.083%
PSNR (↑)	24.293	25.859	32.50%	77.50%
SSIM (↑)	0.9249	0.9539	28.33%	78.33%

Ablation study for the dataset augmentation methods is shown in Table B.3, where a hyperdiffusion (200 epochs) with only the unaugmented MERL dataset is trained and evaluated. The FID value is 7.5558. The results are consistent with the intuition that when the augmentation methods are applied, the better the performance is.

Table B.3: Unconditional hyperdiffusion without dataset augmentation performance on test set.

Distance	MMD-ref	MMD-sam	COV (↑)	1-NNA(50%)
L1-log-cos (↓)	0.7496	0.6222	21.67%	98.75%
L1-log (↓)	1.1622	0.9748	22.50%	100.0%
L1 (↓)	4304.1	19510.3	28.33%	92.50%
RMSE (↓)	0.1338	0.0976	25.00%	84.58%
PSNR (↑)	22.552	25.921	25.00%	84.58%
SSIM (↑)	0.8274	0.8731	22.50%	86.25%

§ B.2.2 Baseline methods

The detailed results for all four baseline methods are as below.

For PCA-brdf, the FID value is 10.8572 and the other results are shown in Table B.4. I present some of its interesting synthetic samples in Figure B.2.

Table B.4: Unconditional PCA-brdf performance on test set.

Distance	MMD-ref	MMD-sam	COV (↑)	1-NNA(50%)
L1-log-cos (↓)	2.3461	2.4451	8.33%	100.0%
L1-log (↓)	2.7236	2.8769	8.33%	100.0%
L1 (↓)	9048.3	584438.0	2.50%	100.0%
RMSE (↓)	0.3329	0.3162	18.33%	96.25%
PSNR (↑)	13.8528	15.1259	18.33%	94.17%
SSIM (↑)	0.6737	0.5217	23.33%	96.25%

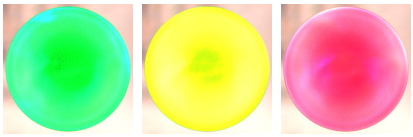


Figure B.2: PCA-brdf synthetic samples.

For PCA-mlp, the FID value is 23.8291 and the other results are shown in Table B.5. I present some of its interesting synthetic samples in Figure B.3.

Table B.5: Unconditional PCA-mlp performance on test set.

Distance	MMD-ref	MMD-sam	COV (↑)	1-NNA(50%)
L1-log-cos (↓)	1.3465	9.0389	30.83%	90.42%
L1-log (↓)	1.6574	10.5269	30.0%	92.08%
L1 (↓)	9223.50	∞	30.0%	95.41%
RMSE (↓)	0.3017	0.4682	28.33%	93.75%
PSNR (↑)	14.7537	12.1484	28.33%	89.99%
SSIM (↑)	0.6289	0.3507	16.67%	96.66%

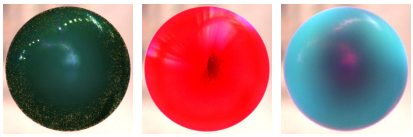


Figure B.3: PCA-mlp synthetic samples.

For VAE-mlp, the FID value is 10.0062 and the other results are shown in Table B.6. The VAE architecture contains MLP-based encoder and decoder, each with 4 layers, input dimension $\mathbb{R}^{675}$, hidden dimension $\mathbb{R}^{300}$ and latent space dimension $\mathbb{R}^{300}$. The likelihood between synthetic and input data used is measured by mean square error. The model is trained with batch size 16, learning rate 1e-3 after half a hundred epochs until convergence.

Table B.6: Unconditional VAE-mlp performance on test set.

Distance	MMD-ref	MMD-sam	COV (↑)	1-NNA(50%)
L1-log-cos (↓)	0.8573	2.5035	21.67%	99.58%
L1-log (↓)	1.2664	3.0055	22.50%	99.58%
L1 (↓)	5827.333	5181702.5	20.83%	96.66%
RMSE (↓)	0.1552	0.1898	16.67%	93.33%
PSNR (↑)	20.9926	21.7842	18.33%	93.33%
SSIM (↑)	0.6860	0.6539	17.50%	93.75%

Some of the interesting rendered outputs are shown in Figure B.4, where over half of the generated materials are undesirably black among the 120 samples.

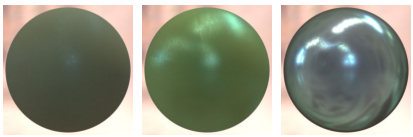


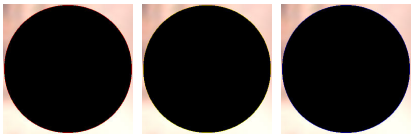
Figure B.4: VAE-mlp synthetic samples.

For VAE-brdf, the FID value is 26.0873 and the other results are shown in Table B.7. To reduce the complexity of VAE-brdf, I downsample the input to $\mathbb{R}^{45\times45\times90}$ and upscale the synthetic results using nearest neighbour. For VAE architecture, input dimension $\mathbb{R}^{45\times45\times90}$, hidden dimension $\mathbb{R}^{256}$ and latent space dimension $\mathbb{R}^{300}$. The model is trained with batch size 16, learning rate 5e-3 until convergence.

Table B.7: Unconditional VAE-brdf performance on test set.

Distance	MMD-ref	MMD-sam	COV (↑)	1-NNA(50%)
L1-log-cos (↓)	3.1249	1.8517	0.83%	100.0%
L1-log (↓)	3.7000	2.2994	1.67%	100.0%
L1 (↓)	9091.9883	60.6147	0.83%	100.0%
RMSE (↓)	0.6368	0.2437	0.83%	100.0%
PSNR (↑)	8.2503	18.6596	0.83%	100.0%
SSIM (↑)	0.2676	0.2873	0.83%	100.0%

I present the visual rendered images in Figure B.5. There's only a dim colour at the border of the sphere, labeled by the figure caption.



(a) Red (b) Yellow (c) Blue

Figure B.5: VAE-brdf synthetic samples.

§ B.2.3 Reference evaluation

I measure the selected training set performance with respect to the reference set by picking random samples from the former and keeping them of the same number, as fitting all training high dimensional data-driven BRDF requires huge memory and computation time. This process is kept the same across all measurements as those synthetic samples. The performance between the generated 120 materials and 120 test set is measured, where FID = 0.18734 and the rest of results are shown in Table B.8.

Table B.8: Training data performance on test set.

Distance	MMD-ref	MMD-sam	COV (↑)	1-NNA(50%)
L1-log-cos (↓)	0.6197	0.6188	64.16%	93.33%
L1-log (↓)	1.0475	1.0639	63.33%	98.33%
L1 (↓)	2505.3	2658.5	60.83%	58.75%
RMSE (↓)	0.0754	0.0744	55.83%	44.58%
PSNR (↑)	28.7426	29.0334	56.67%	45.00%
SSIM (↑)	0.9553	0.9487	59.16%	42.50%

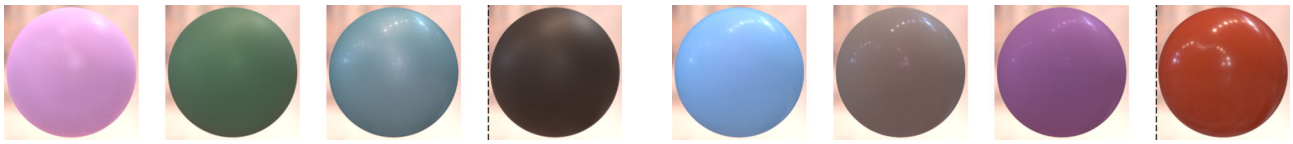
I also randomly interpolate three MERL materials fitted by PCA with random colour channels permutation (Table B.9). The performance between 120 generated materials and 120 test set is measured, where FID = 1.2531. I argue that due to the random selection process, which is intrinsically different from the proposed methods, PCA scores high marks on coverage-related metrics.

Table B.9: PCA based conditional models performance on test set.

Distance	MMD-ref	MMD-sam	COV (↑)	1-NNA(50%)
L1-log-cos (↓)	0.4157	0.4281	43.33%	78.33%
L1-log (↓)	0.4475	0.4745	44.17%	78.33%
L1 (↓)	3071.2	1954.1	44.99%	62.08%
RMSE (↓)	0.1027	0.1001	45.83%	70.83%
PSNR (↑)	24.513	25.605	46.67%	70.42%
SSIM (↑)	0.9387	0.9175	51.67%	70.83%

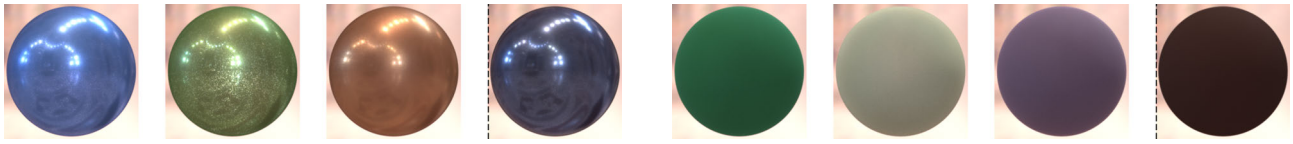
§ B.3 Conditional generation evaluation details

Three generated samples with category / text / image prompts are shown in Figure B.6 / B.7 / B.8, together with the rightmost MERL reference material. Despite the non-colour-aware ResNet embedding, the results are promising and showcase the usage for multi-modal diffusion-based data-driven BRDF generation.



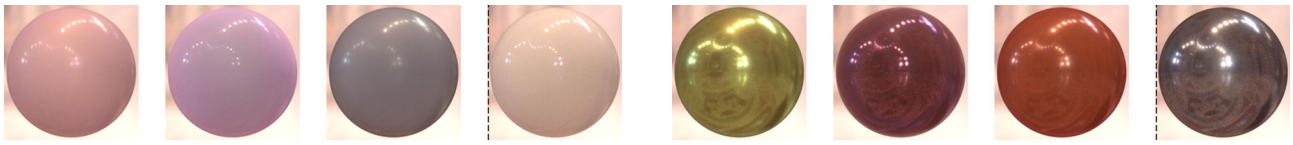
(a) Oxidized steel (b) Phenolic

Figure B.6: Category-conditioned synthetic materials | reference.



(a) Pure metallic-paint (b) Pure fabric

Figure B.7: Text-conditioned synthetic materials | reference.

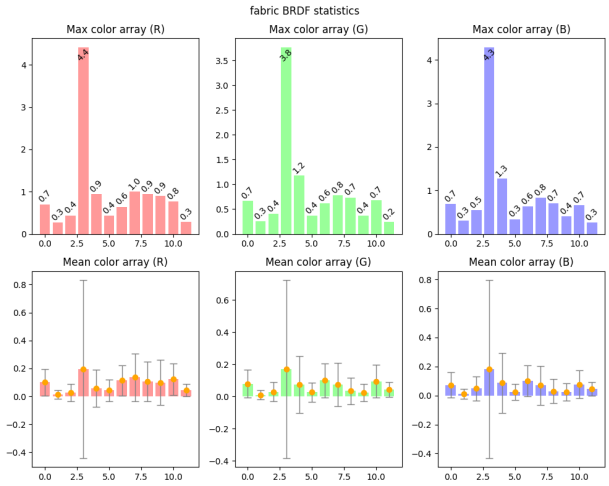


(a) Alumina oxide (b) Aluminium

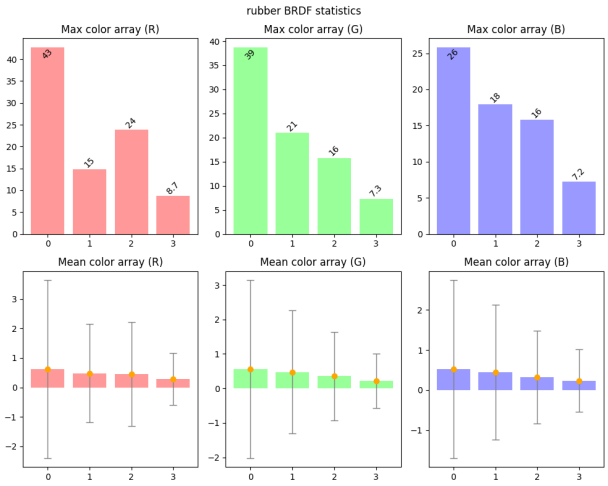
Figure B.8: Image-conditioned synthetic materials | reference.

§ B.4 MERL statistical analysis details

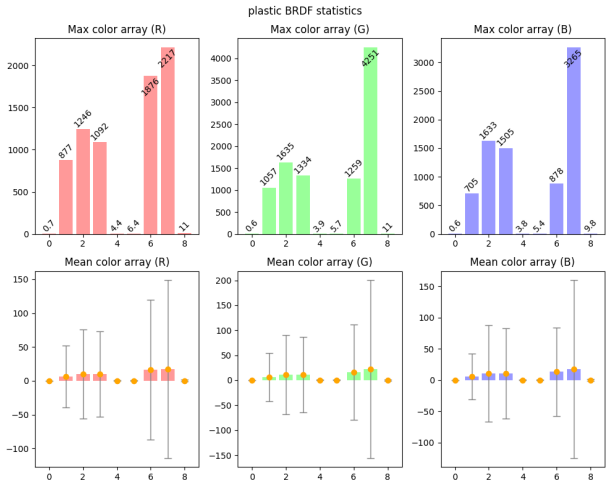
I perform the statistical analysis on the most data-rich MERL categories obtained from material filename. In addition to the per-category statistical summary 3.7a, I attach the individual details of selected MERL BRDF samples, including the max and mean reflectance values across all colour channels, in Figure B.9.



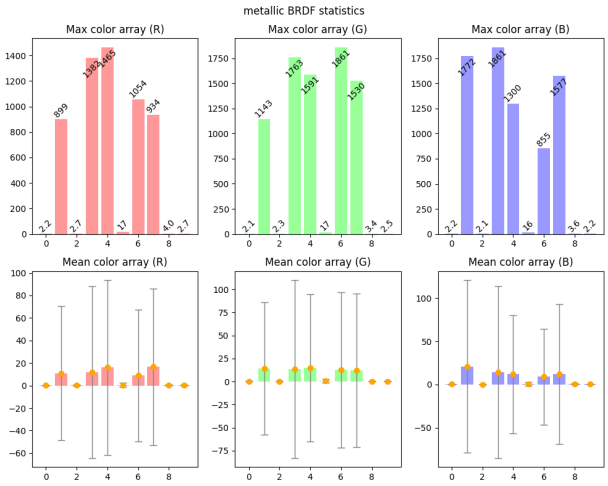
(a) Fabric BRDF statistics



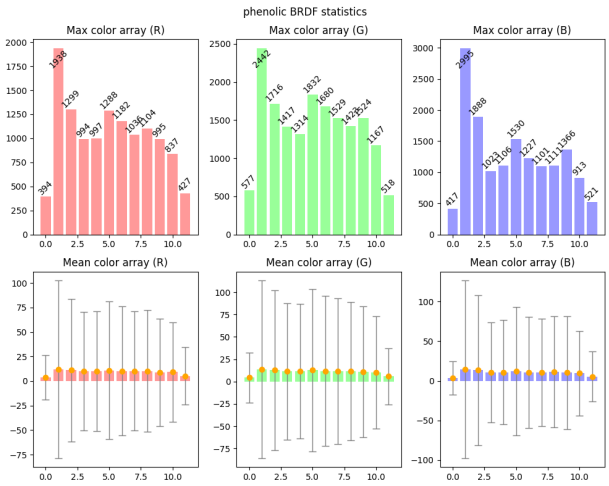
(b) Rubber BRDF statistics



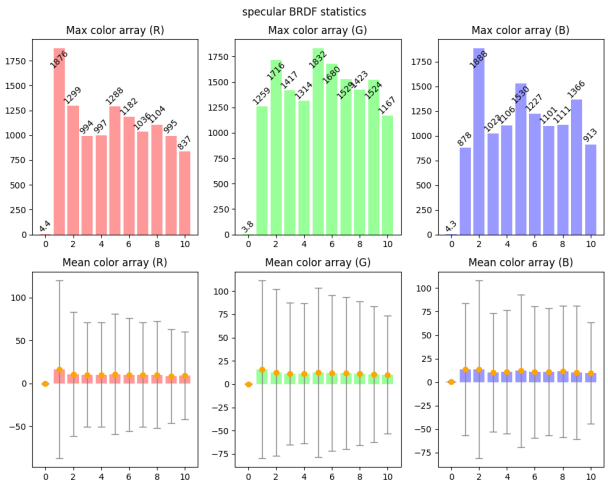
(c) Plastic BRDF statistics



(d) Metallic BRDF statistics



(e) Phenolic BRDF statistics



(f) Specular BRDF statistics

Figure B.9: MERL BRDF category statistical analysis.

C Miscellany

§ C.1 Software engineering tools and techniques

For completeness, this chapter describes the software engineering tools and techniques used in the project, including the development methodology, languages and libraries chosen, version control and testing strategies, automation and documentation.

§ C.1.1 Development methodology

The project's cross-disciplinary nature suggests deploying the agile methodology [ASRW17]. The work is broken down into small subtasks that reduce advanced planning, namely in the field of data processing, Computer Graphics and Machine Learning. Every two to four weeks, I reflect on the accomplished parts with extensive testing and adjust the plan (Figure C.1). For the extensions, the exploratory research results motivate iterative planning and development. The final evaluation is delayed until the extensions are wrapped up.

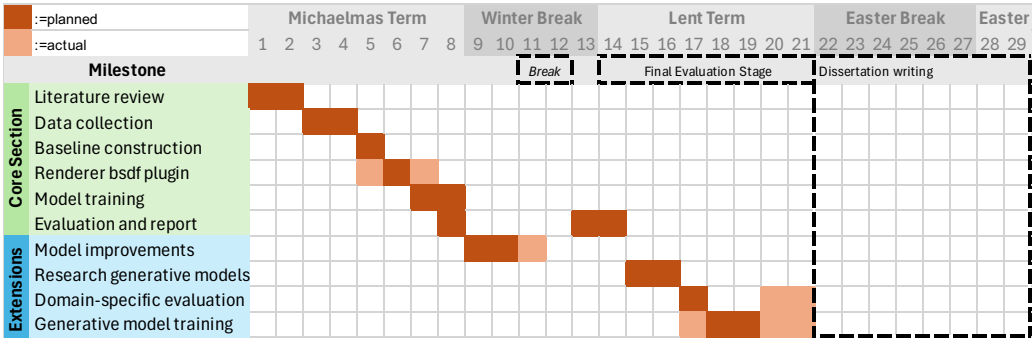


Figure C.1: Gantt graph of the project timeline.

For project management, I use a Github **Kanban board** [Git24b] (an adaptable task-board tool), where around 30 subtasks are filed, with integration with issues and milestones. I record all **key decisions** throughout the project. Different branches are tracked for divergent novel explorations, with tested features merging into the main branch. After Proof of Concept (PoC), the structure of the final implementation is designed with Unified Modelling Language (UML) diagrams, i.e. clear class hierarchies.

§ C.1.2 Languages and libraries

The project is written in `Python 3.10`, the main-stream language for machine learning. The codebase complies with the Coding Style Guide `PEP 8` [vRWC01], with the help of autoformatters. The code is tested on MacOS and Linux.

Table C.1 summarizes the main[1] third-party libraries used, together with their licenses. `Pytorch` (`torch`) provides the basic deep learning development kits, reducing the incidental complexity. The `Mitsuba3` supports Monte Carlo sampling and real-time rendering. Most reference Graphics codes are written in `C/C++`. For smoother portability to training, I extended its *BRDF*

[1]The full attribution list is in the License file of the repository.

interface in `Python` instead.

Table C.1: The core libraries used in the project.

Library	Motivation	License
`Doxygen`	Documentation generator tool.	GNU
`Mitsuba3`	Renderer backbone with extensible components.	BSD
`torch`	Neural network models, training and inference.	BSD
`numpy`	Scientific computing for multi-dimensional arrays.	BSD
`matplotlib`	Scientific plotting and visualization.	PSF-2.0
`argparse`	User-friendly command-line interfaces	PSF-2.0

§ C.1.3 Version control and testing

Git tracks the whole project workspace, which is hosted on the Github repository. The remote backup improves fault tolerance. I utilize the **trunk-based workflow**. All source code files are committed to their respective *development* branches, supporting multiple *concurrent diverging* features. The conflicts are resolved when applying patches[2] or merging. After passing all essential tests, I pull them into the *main* branch.

I adopt the **Test Driven Development (TDD)** methodology. One example is the six importance sampling strategies (§ 2.1.5). Before the actual coding, I deploy two unit tests on my Monte Carlo `sample()` distribution O and the expected `pdf()` E (§ 3.1). Firstly, by Appendix A.1, the former histogram sum around 1 and the latter numerical integration should be 1 (C.1).

$$1 = \int_{H^2} \text{pdf}(\vec{\omega})\, d\vec{\omega} \approx \sum_{k=0}^{n} sample(\vec{\omega}_k). \tag{C.1}$$

Secondly, I conduct a **significance test** on the above two distributions. The null hypothesis is that they are matched. Both distributions and their difference are visualized in Figure C.2.

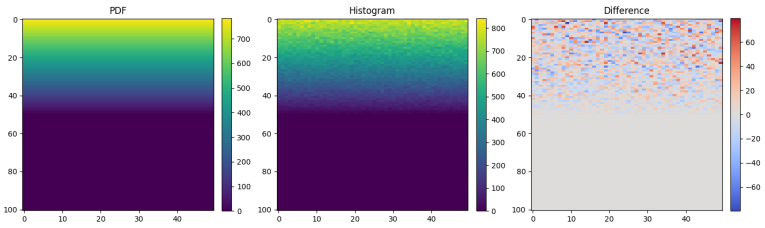


Figure C.2: Cosine-weighted sampling PDF and Histogram.

Pearson χ^2 goodness of fit test [JSRV22b] is used. With a degree of freedom (d.o.f)[3] and significance value $\alpha = 0.01$, the critical p-value is calculated. By comparing the p-value with the chi-square value $\chi^2 = \frac{1}{d}\sum_{k=1}^{n} \frac{(O_k - E_k)^2}{E_k}$, the null hypothesis can then be rejected or not. The above two tests ensure that all functions are implemented correctly.

In general, the **unit tests** frameworks used are *unit-test* in `Python`. Among which, I adopt assertion testing with around 24 `assert` statements over 11 files, which helps catch violation of

[2]Patches are created via `git diff > changes.patch`.
[3]d.o.f is defined to be the number of groups minus one.

invariants early during development. These assertions are removed during the final evaluation and deployment, for better performance. For **regression tests**, the *expect test paradigm* is adopted because of its reduction of incidental complexity. It saves the expected output of the function during the first validated run and compares it with the actual output.

Integration tests play a crucial role in the system verification. For instance, I *reconcile* my implementation of the Python plugin with the official MERL C++ parser, which is termed as *end-to-end output verification*. It ensures that the IO utilities, coordinate transformation and RGB scaling work seamlessly together.

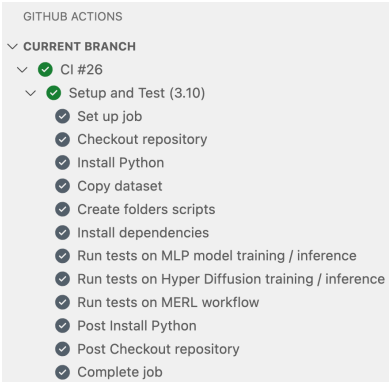
In total, there are over forty tests, covering all five keys modules at 87% coverage rate[4].

Type annotations and runtime checking are added for type-agnostic Python. Dynamically-typed language is prone to runtime errors. Hence, I adopt `jaxtyping` module to annotate the `NumPy` array or `PyTorch` tensor. The benefit not only includes finer control over data types and sizes, but it also helps document the codebase, which is ideal for future maintenance and collaboration. During real-time execution, such annotations could be removed to speed up the performance.

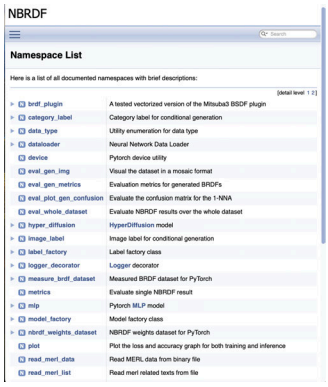
§ C.1.4 Automation and documentation

For automation, **code linter** [Pyl24] checks the potential semantic and stylistic problems before each commit. The Python `argparse` module helps parse the command line arguments for user-friendly interaction and deployment. I setup a **Continuous Integration (CI)** workflow[5] in the remote Github Action Runner [Git24a], such that the bash scripts, including all test suites, are executed after each major commit (Figure C.3a).

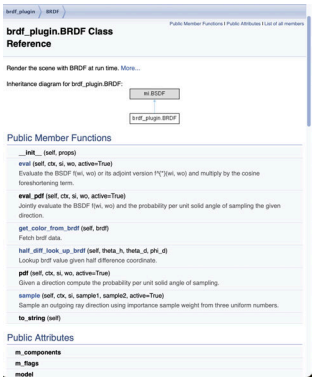
For documentation, I choose Doxygen [Dox24] and follow `PEP 257` docstring convention [DG01] consistently (Figure C.3b, C.3c). The generator combines them with markdown files[6]. The final web-based documentation helps me structure the codebase wisely and improves maintainability.



(a) CI workflow.



(b) Namespace doc.



(c) Class doc.

Figure C.3: CI workflow and documentation.

[4]The coverage report is generated from `python3 -m coverage` and statistics of the assertion tests. The uncovered lines are mainly the unexecuted branches or error handling.

[5]For details, please refer to `.github/workflows/ci.yml` in the submitted repository.

[6]For details, please refer to `docs/doxygen/html/index.html` in the submitted repository.

Acronyms

1-NNA 1-Nearest-Neighbour Accuracy. 39, 44, 62

BRDF Bidirectional Reflectance Distribution Function. 3, 10, 13, 14

BSDF Bidirectional Scattering Distribution Function. 13, 50

BTDF Bidirectional Transmittance Distribution Function. 13, 50

cdf cumulative distribution function. 58–61

CI Continuous Integration. 71

COV Coverage. 39, 44, 62

CVPR Computer Vision and Pattern Recognition Conference. 23

FFN Feed-Forward Network. 21, 31

FID Fréchet Inception Distance. 38, 44, 62

GI Global Illumination. 10

HDR High Dynamic Range. 15

IBL Image-based lighting. 40

MAE Mean Absolute Error. 39, 44

MLP Multi-Layer Perceptron. 21, 24

MMD Minimum Matching Distance. 39, 44, 62

MSE Mean Squared Error. 37, 39, 40

NBRDF Neural-BRDF. 11, 12, 17, 24, 27

NLL Negative Log-Likelihood. 19, 20

PBR Physically-Based Rendering. 10

PCA Principal Component Analysis. 11, 13, 18, 25, 26, 33, 40, 42, 44, 49, 64

pdf probability density function. 26, 57–60

PoC Proof of Concept. 50, 69

PSNR Peak Signal-to-Noise Ratio. 12, 37, 39, 40

RMSE Root Mean Squared Error. 37

RT Ray Tracing. 10

SOTA state-of-the-art. 23, 35, 57

spp samples per pixel. 18, 57

SSIM Structural Similarity Index. 12, 37, 39, 40, 43

TDD Test Driven Development. 70

UML Unified Modelling Language. 24, 28, 50, 69

VAE Variational Autoencoder. 13, 18, 33, 44, 49, 65