Neural Acceleration of Scattering-Aware Color 3D Printing

Tobias Rittig[†,1], Denis Sumin[†,2], Vahid Babaei[2], Piotr Didyk[3], Alexey Voloboy[4], Alexander Wilkie[1], Bernd Bickel[5], Karol Myszkowski[2], Tim Weyrich[6], Jaroslav Křivánek[1]

[1]Charles University, Czech Republic [2]Max Planck Institute for Informatics, Germany [3]Università della Svizzera italiana, Switzerland
[4]Keldysh Institute of Applied Mathematics RAS, Russia [5]IST Austria, Austria [6]University College London, United Kingdom

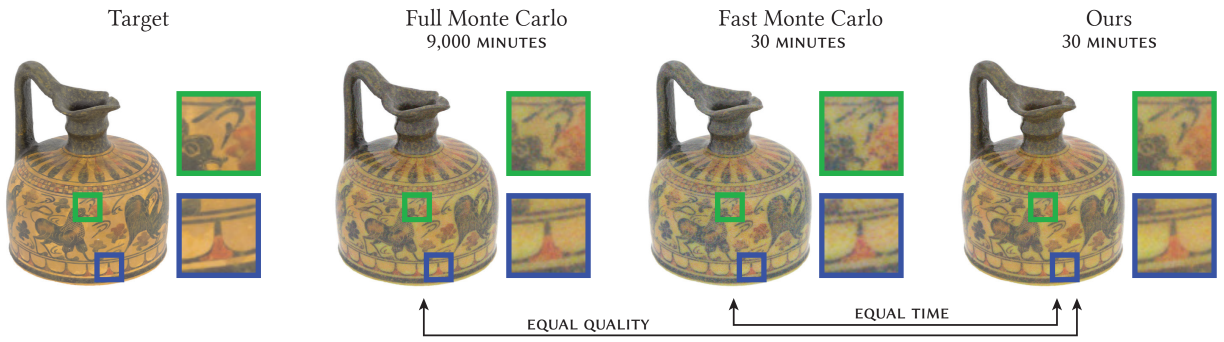


Figure 1: *We propose a neural scattering compensation for 3D color printing. Comparing to a method which uses noise-free Monte Carlo simulation our technique achieves* 300× *speedup in the above case while providing the same quality.*

Abstract

With the wider availability of full-color 3D printers, color-accurate 3D-print preparation has received increased attention. A key challenge lies in the inherent translucency of commonly used print materials that blurs out details of the color texture. Previous work tries to compensate for these scattering effects through strategic assignment of colored primary materials to printer voxels. To date, the highest-quality approach uses iterative optimization that relies on computationally expensive Monte Carlo light transport simulation to predict the surface appearance from subsurface scattering within a given print material distribution; that optimization, however, takes in the order of days on a single machine. In our work, we dramatically speed up the process by replacing the light transport simulation with a data-driven approach. Leveraging a deep neural network to predict the scattering within a highly heterogeneous medium, our method performs around two orders of magnitude faster than Monte Carlo rendering while yielding optimization results of similar quality level. The network is based on an established method from atmospheric cloud rendering, adapted to our domain and extended by a physically motivated weight sharing scheme that substantially reduces the network size. We analyze its performance in an end-to-end print preparation pipeline and compare quality and runtime to alternative approaches, and demonstrate its generalization to unseen geometry and material values. This for the first time enables full heterogenous material optimization for 3D-print preparation within time frames in the order of the actual printing time.

CCS Concepts • ***Computing methodologies*** → ***Reflectance modeling; Volumetric models;*** • ***Applied computing*** → ***Computer-aided manufacturing;***

Keywords: computational fabrication, appearance reproduction, Monte Carlo rendering, sub-surface light transport simulation, heterogeneous media, deep learning, machine learning

1. Introduction

Existing multi-material 3D printers are capable of depositing a broad spectrum of printing materials at very high spatial resolution. Apart from intricate geometries, such technologies can also be used to reproduce complex color variation on the object's surface. The recent advances in color and, more broadly, appearance reproduction [SRB*19, UTB*19] enabled quality, making these techniques suitable for a large number of applications ranging from a reproduction of cultural heritage [SBK*18] to medical prostheses [SVS*20].

The dominant technology used for both high-resolution printing and accurate appearance reproduction is the inkjet 3D printing using UV-curable materials [SARW*15, Str20]. In the context of color re-

† Shared first authorship.

production, the critical problem lies in undesired optical scattering properties which significantly impact the resulting texture sharpness and color accuracy. Addressing this problem, the most recent methods [ESZ*17, SRB*19] apply an iterative scattering-aware *refinement* that optimizes the arrangement of printing materials to counteract the undesired scattering and create an opaque look. The refinement procedure heavily relies on expensive Monte Carlo (MC) path tracing, which at every iteration predicts the appearance of the current volumetric materials arrangement. While using an accurate prediction leads to high-quality color reproduction, such a solution scales poorly with the number of voxels as the paths grow in length. The insufficient performance of these methods hampers their practical applicability, especially in the face of 3D printing's increasing build volume, material count, and resolution.

From a user's perspective, preparing a 3D printout is an iterative trial-and-error process due to inherent hardware limitations. A long tradition in 2D printing is providing the user with a virtual preview of the final print. If implemented accurately, this so-called *soft proofing* allows for gauging any problems on the preparation side and predicting the effect of limitations on the hardware side. In 3D printing so far, this has only been accurately possible with a full Monte Carlo simulation of light transport which can take many hours for a small object on a high-end desktop system.

In this work, we address the significant bottleneck of state-of-the-art color 3D printing techniques by proposing strategies to speed up the included light transport simulation. We start by analyzing the impact of a reduced number of samples used in the MC simulation. Our experiments demonstrate the robustness of the existing refinement techniques to noisy appearance predictions and, consequently, significant speed benefits at only moderate quality loss. To further improve the quality of the achieved results, we leverage advances in machine learning and propose a data-driven forward appearance model using a deep neural network.

We rely on key insights from previous work in both MC-based color 3D printing and neural volume rendering. Based on a learning framework devised for approximating multiple-scattering in clouds [KMM*17], we build a full replacement for the MC forward prediction inside an appearance refinement loop [SRB*19]. We depart from querying the learned predictor inside a MC estimator and obtain the final low-variance estimate from a single inference. Furthermore, we propose modifications to the architecture in the form of weight sharing, that relax the cloud-specific scope.

Our improvements offer two benefits. First, our solution generalizes better from simple geometries like cubes and spheres to more complex 3D geometries. Second, the network is able to predict a wider continuum of scattering parameters just from a sparse discrete training set. We train our neural network to predict light transport in heterogeneous volumes akin to composite multi-material prints with spatially varying albedo *and* density using data from MC renderings with complex volumetric textures. Our dataset allows the network to learn correct predictions for more general material arrangements than those typically found in 3D printouts. We argue that the dataset is critical for reliable execution of the iterative refinement as it encounters differing intermediate solutions. Moreover, as we will show, the neural predictor is also suitable for (spectral) soft-proofing during 3D print preparation.

With the new prediction integrated into the color 3D printing pipeline, and running on a single, GPU-equipped workstation, our approach outperforms previous work running on a multi-core workstation of similar caliber by around two orders of magnitude in speed. As a result, we enable the quality of previous methods at speeds suitable for more practical applications, without having to resort to massively parallel compute clusters. Finally, we also provide a comparison of heuristic backward refinement to recent differentiable rendering methods. While both perform on a comparable quality level, the heuristic shows a distinct convergence benefit.

Our contributions include:

- Fast appearance prediction for fully heterogeneous volumes using machine learning.
- Up to 300× speedup with constant quality compared to existing 3D printing pipelines.
- A synthetic dataset of scattering volumes representing a wide gamut of material arrangements.
- Vital architecture improvements that prepare the existing cloud-specific framework for broader arrangements.
- A comparison of differentiable rendering and heuristic material arrangement refinement for 3D printing volumes.

2. Background

Our work is related to multiple subfields in computer graphics from computational fabrication and machine learning (ML) to light transport simulation with participating media and color science. In the following, we will characterize relevant publications in these categories and how our work links to them.

2.1. Related Work

Color 3D Printing The technological advances in multi-material inkjet 3D printing [Str20] opened up new lines of investigation within appearance fabrication, especially on color 3D printing. [BAU15] introduced an error-diffusion approach to halftone continuous color over 3D surfaces to enable full-color printing with discrete primary materials. [BVF*17] showed how layering primaries with different thicknesses (contoning) can be used to control surface color and avoid halftoning artefacts. [BATU18] showed how volumes can be constructed to meet joint color and transparency goals on the surface. All these methods are *local* color predictors as they do not consider the effect of neighboring volume on the color, which results in considerable blur.

[ESZ*17] address that problem of blur on planar slabs due to the subsurface scattering of printing materials. [SRB*19] extended this method to 3D surfaces with complex geometry, such as thin features. Both methods are slow as they rely on expensive MC light transport simulation to predict the appearance of printouts from the volumetric material assignments. We build on the pipeline proposed by [SRB*19] and considerably improve the overall speed by replacing the forward renderer with a data-driven method.

Machine learning has been applied in 3D printing to predict the spectral reflectance of contone ink stacks [SBK*18]. Similar to previous local methods, this approach neglects the effects of lateral scattering on surface sharpness and color accuracy.

Neural Rendering Machine learning has recently become a popular technique for image synthesis and has been applied to many problems in computer graphics. We refer the reader to the recent survey by [TFT*20] for a complete overview of general neural rendering. In Monte Carlo rendering, machine learning has been used for varying lighting in global illumination [RWG*13], path guiding in primary sample space [ZZ19, MMR*19], importance sampling of first-bounce contribution [BMDS19, HWZ*20], and image denoising [JK21, KBS15, BVM*17, CKS*17, VRM*18, GLA*19]. Here, we particularly focus on data-driven approaches in conjunction with volume rendering [KKN*18].

Neural networks have replaced some components, such as path computation, inside the rendering loop for compute-heavy problems like volume scattering. [VKJ19] propose a learned subsurface scattering model for homogeneous media that samples an exitant location over a polynomial approximation of the local shape and infers a corresponding path attenuation. Long light paths in heterogeneous, forward-scattering media like clouds are the motivation for [KMM*17] to approximate multiple-scattering using a neural network. They learn the contribution of a directional light source towards a point inside the medium determined by MC single-scattering. Recently, [PN19] showed a 2.5× speed improvement over the latter work using spatial sharing of precomputed partial results.

Our approach is inspired by the work by [KMM*17], but rather than evaluating the network for a single path sample, we replace MC integration altogether and predict the final radiance at once. Instead of directional lighting we assume the simpler case of smoother diffuse illumination, but with the added difficulty of a spatially-varying scattering albedo unlike the constant-albedo assumption of cloud modeling. Lastly, we improve the network architecture by a physically motivated weight sharing scheme that substantially reduces the network size.

Inverse Volume Rendering The ultimate goal of our work is to find a scattering volume given an input surface appearance, a task akin to inverse subsurface scattering. Inverse methods are used to capture scattering properties of homogeneous materials [GZB*13, CLZ*20]. In a recent work [CLZ*20] learn to predict *homogeneous* material coefficients from images using a renderer for training. We employ machine learning to accelerate the forward volume rendering and rely on a heuristic method [SRB*19] for the inverse reconstruction of a heterogeneous medium.

Differentiable volume renderers [NDVZJ19, ZWZ*19] are capable of reconstructing any volume parameter to match a target surface appearance using stochastic gradient descent (SGD). The recent publication of Radiative Backpropagation (RB) [NDSRJ20] allows for practical usage of differentiable rendering without significant time or memory overheads. We show in Section 5.4 how such a general tool is applicable in the context of 3D print preparation and compare against the heuristic inverse method [SRB*19] we chose for refinement.

2.2. Behavior of Volumetric Scattering

Volumetric light transport is typically a complex mapping between many volume points (voxels) and fewer surface points. While the space of possible volume assignments is wide in range, the gamut spanning their surface appearances is somewhat limited. As can be seen from the first two examples in Figure 2, vastly different volumes can exhibit comparable color saturation. That is because generally the influence of voxels decays with distance to the light source, i.e., the surface in this case. The deeper any absorptive material is embedded, the more it bleeds into the surrounding and affects local contrast, absolute brightness and texture sharpness on the surface, as can be observed across all examples in Figure 2. The overall absorptive power of a volume is physically limited by the geometric thickness and directly impacts the achievable color gamut.

2.3. Scattering-Aware Color 3D Printing

Previous work employed accurate MC rendering as a forward prediction in an appearance optimization loop for inkjet 3D printing [ESZ*17, SRB*19]. These works rely on heuristics that change an arrangement of printer materials for improved texture sharpness on the surface. The driving force in the optimization loop visualized in Figure 3 is the difference between the predicted appearance of the translucent materials and a given opaque target appearance. Notably, the optimized appearance solely covers the subsurface light transport, as the 3D printers do not (yet) allow for full control of surface reflectance. Thus, [SRB*19] assume uniform, omnidirectional incidence over the surface without self-shadowing or surface inter-reflections. This enables more general viewing conditions for the final 3D prints and prevents baking a certain lighting condition into the volume. In this work, we adopt the same lighting strategy.

Because these methods are proven to come close to the physical boundaries of overall texture sharpness and color reproduction in thin geometric regions, a practical application would be desirable. Unfortunately, practical adoption is considerably hindered by the computational power and runtime necessary. The authors report times comparable to thousands of core-hours, neccesitating the use of a compute cluster for a single object print preparation.

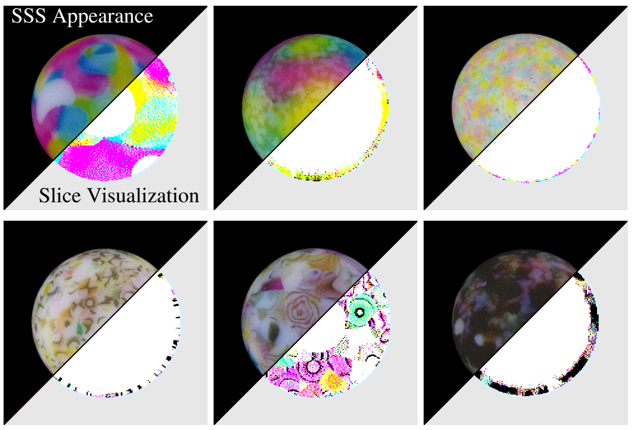


Figure 2: *Synthetic dataset examples: Monte Carlo predictions of generated volumes in a* 10 mm *sphere. The generator's output (Section 4.1) ranges from 3D Voronoi-like textures, to surface extrusions and thin surface coloring. Deeper voxels have less influence on color saturation but limit contrast and edge sharpness.*

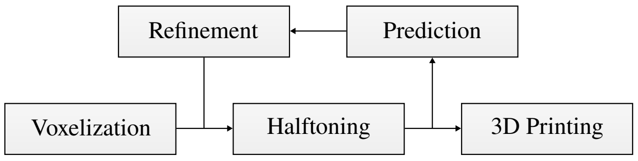
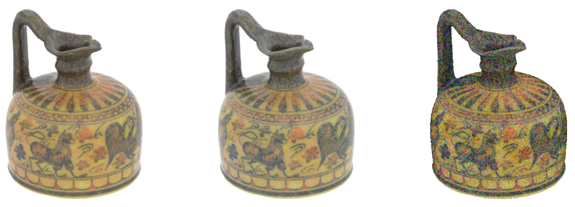


Figure 3: *Schema of a scattering-aware 3D print pipeline*



(a) *Optimized print MC 512 spp dE '00:* 8.43 **(b)** *Optimized print MC 4 spp dE '00:* 12.15 **(c)** *Volume preview MC 4 spp*

Figure 4: *High-sample MC previews showing the effect of reduced MC sampling-rate during refinement (a, b). A single 4 spp MC preview shows the level of quality the refinement is coping with (c).*

The reasons for this inefficiency of MC path tracing are manifold: with a high-albedo, forward-scattering material such as the white core of 3D printouts, light paths tend to grow in length while still contributing considerably. Thus, a high number of samples is required to reduce variance – [ESZ*17] report 500 spp. With increasing object volume, the MC estimator gets less efficient, as light paths grow longer and longer.

3. A First Approach to Fast Print Preparation

The major bottleneck in scattering-aware 3D printing [SRB*19] is the computationally demanding Monte Carlo prediction. A simple approach to reduce the computation time of any MC-based algorithm is to reduce the number of samples at the cost of higher variance. We experimented with preparing a print with only four samples per surface point and found that, against expectations, the approach of [SRB*19] is robust against a low-quality forward prediction. In Figure 4, we compare this object with a preparation time of less than 30 core hours against a reference object optimized with 512 spp taking over 3,000 core hours. At first glance, it seems like the refinement loop averages out the noise over iterations and still converges to a reasonable result. However, compared to the high-sample MC refinement result, the quality degradation is visible; the color difference values against the target, in terms of mean CIE dE 2000, are also noticeably worse.

Also visually, a low sampling rate is not suitable for soft proofing purposes because it exhibits distracting (color) noise as demonstrated in Figure 4c. Image denoising methods could arguably be of help in that case, but they are unsuitable for predictions inside the refinement loop. Denoising images heavily relies on regularly sampled neighborhood information and thus cannot trivially be applied to our 3D voxel representations of *surface* radiance. Alternatively, volume path guiding [HZE*19] appears promising to match the requirements for visual inspections but ultimately does not promise enough speed-up for the print preparation to become practical.

With the aspiration of an approximative but fast appearance prediction to the high-dimensional problem of volume scattering, we shift our focus to data-driven methods. These have been used in the past to accelerate light transport simulation in homogeneous [VKJ19] and even heterogeneous volumes [KMM*17]. The advantage of these methods is that they do not introduce high-frequency noise artifacts and have a constant cost per surface point, independent of scattering values and geometric features. Deviating from previous work, we do not employ (biased) machine learning inside an MC estimator. Instead, we eliminate the stochastic process completely and learn to predict the final, biased result.

4. Appearance Prediction Network

Our goal is to yield the full subsurface-scattering appearance of a surface point given a volumetric patch around that point using a Radiance Predicting Neural Network (RPNN) [KMM*17, PN19]. We assume uniform, omnidirectional incidence over the surface to account only for the subsurface part of light transport. The network ultimately replaces the MC forward prediction in an appearance optimization pipeline for 3D printing. Therefore, our learning ecosystem is targeted towards this application.

Problem Statement Formally speaking, we try to find a function $P\colon \mathcal{W}, \mathbb{R}_+^F \to \mathbb{R}_+$ with weights $\mathcal{W}$ that given a set of features $F_{\mathbf{x}_o}$ around point $\mathbf{x}_o$ results in the total in-scattered light from below the surface. This energy is described as the integral

$$L_o(\mathbf{x}_o, \omega_o) = \int_A \int_{\mathcal{H}^2} S(\mathbf{x}_o, \omega_o, \mathbf{x}_i, \omega_i) L_i(\mathbf{x}_i, \omega_i) |cos\,\theta_i| \, d\omega_i \, dA\,, \tag{1}$$

where S is the bidirectional scattering surface distribution function (BSSRDF) between an incident and outgoing point $\mathbf{x}$, and direction ω respectively. A denotes the object's surface area, $\mathcal{H}^2$ the hemisphere above the surface and L_i stands for the incoming illumination. Our target application replicates diffuse appearances and thus we are only interested in the outgoing radiance along the surface normal $\mathbf{n}_\mathbf{x}$. Also, the assumption of diffuse illumination allows for the inner integral to be integrated out, leaving

$$L_o(\mathbf{x}_o, \mathbf{n}_{\mathbf{x}_o}) = \int_A S(\mathbf{x}_o, \mathbf{n}_{\mathbf{x}_o}, \mathbf{x}_i) \, dA\,. \tag{2}$$

Learning an approximation of the remaining integral, we seek weights $\mathcal{W}$ of the network that minimize the L^2 distance between the RPNN prediction P and a ground truth Monte Carlo simulation L_o for all T training examples,

$$\arg\min_{\mathcal{W}} E(\mathcal{W}) = \frac{1}{T} \sum_{t=1}^{T} \left(L_o(\mathbf{x}_t, \mathbf{n}_{\mathbf{x}_t}) - P(\mathcal{W}, F_{\mathbf{x}_t}) \right)^2. \tag{3}$$

In the following, we will first describe the construction of a dataset and how features F are derived from it. Next, we detail our contributions to RPNNs, and finally, we describe their deployment. We drop the location index $\mathbf{x}$ for brevity.

4.1. Training Dataset

For our supervised regression-learning task, the datasets consist of pairs of volumes and rendered points on their surface. The volumes store trichromatic scattering and absorption coefficients in each voxel. The target values represent the surface reflectance at a given point in direction of its normal. We choose to work with a synthetic dataset of basic shapes filled with generated volume assignments. We summarize the properties of the dataset here and refer the reader to our supplementary material for more details.

Shape Shape is one of the two major components affecting a volume's appearance. Our dataset includes basic convex geometry in the form of a cube, a rotated cube, a sphere and a thin cuboid. The former three objects are instantiated at different sizes between 5 mm to 20 mm without additional rotation. The thin slab includes thicknesses from 1 mm to 2.5 mm. Despite such sparse sampling of geometric variation, we notice comparably little artifacts on arbitrary geometry thanks to our weight sharing approach (Section 4.3).

Texture The volumes are highly heterogeneous mixtures of discrete base materials. We generate procedural volumetric textures and intersect them with geometry to produce a single volume entity. Based on spatial, deterministic OpenSimplex noise [Kdo14], the textures are assembled using a variety of parametric functions. Through thresholding and manifold parameter dependency we build rich procedural volumes that exhibit multiple scales of frequencies, a wide dynamic range and different levels of chromaticity. Refer to Figure 2 for examples of generated volumetric distributions and their surface appearance.

Values The values of the dataset describe scattering coefficient σ_s and absorption coefficient σ_a in inverse path length [mm^{-1}]. These belong to five measured 3D printing materials [ESZ*17] each with three color channels. The measurements are fitted to a constant Henyey Greenstein phase function model with a forward scattering lobe of $g = 0.4$. We mix data from all channels together to train a single-channel network. For training stability, the values will also be rescaled as detailed in Section 4.4.

As the 3D printing materials vary in both extinction coefficient σ_t and albedo α, our approach is the first one to tackle fully heterogeneous volume scattering with machine learning.

4.2. Feature Descriptor

With light transport being an inherently global phenomenon, the outgoing radiance at each point on the object's surface depends on both local and global volumetric distribution of materials underneath. But in an ML setting, training on the full volumetric data becomes computationally intractable for larger objects and gigabytes of dense voxel grids. Motivated by these observations, [KMM*17] introduced a sparse, hierarchical descriptor that extracts the most important features to describe scattering through a point. This descriptor captures the high-resolution details within the immediate vicinity of a considered surface point but reduces (exponentially) to a sparser representation for the distant volume.

Similar to [KMM*17], our feature descriptor F extracts a set of volume *stencils* Σ^k at different levels $k \in \{1,\dots,K\}$. Higher (coarser) levels represent a larger portion of the volume but at lower resolution. Obtaining a single stencil Σ^k involves sampling the neighborhood of surface point $\mathbf{x}$ and extracting a set of scattering and absorption coefficients at $|\Sigma^k| = I$ locations q_i^k,

$$\Sigma^k = \left\{\sigma_\text{s}(q_1^k), \sigma_\text{a}(q_1^k), \dots, \sigma_\text{s}(q_I^k), \sigma_\text{a}(q_I^k)\right\}. \tag{4}$$

For each hierarchy *level*, the considered volume scales by 2^{k-1} with descriptor dimensions staying constant (I). With all stencils being centered around the same surface point $\mathbf{x}$ (Figure 5a), the resolution of the neighborhood effectively decreases exponentially. The final feature descriptor is a set of all stencils in the hierarchy $F = \Sigma = \bigcup_{k=1}^{K} \Sigma^k$. In practice, we use $K = 9$ levels and a stencil dimension of $I = 5\times3\times9$ with $[-2,-1,-4]$ and $[2,1,4]$ being corners of the index-space offsets from $\mathbf{x}$.

In contrast to [KMM*17]'s requirement for an elongated and rotated stencil towards a directional light source, we take advantage of diffuse illumination in our setting and simplify the data lookups. We sample the volume with stencils axis-aligned to the voxel grid and of cubic world-space size. Due to asymmetry in the voxel resolutions of our 3D printer ($600\times300\times940$ DPI), the actual cell numbers still differ per axis. Our stencil of $5\times3\times9$ samples then covers a volume of $211\,\mu\text{m}\times254\,\mu\text{m}\times243\,\mu\text{m}\cdot2^{k-1}$ and reaches 65 mm of the surrounding on the coarsest level. Depending on channel and material, this distance is 146 to 1560 mean-free-path-lengths.

Values in the coarser levels $k > 1$ are averaged from the lowest level to reduce aliasing; volume partials outside the objects are treated as zeros. This is especially important with our high-frequency data for 3D printing as opposed to smoother value changes in clouds. Our filtering approach, detailed in Section 4.4, differs from mip-mapping in order to align perfectly to the central point $\mathbf{x}$ on coarser levels.

4.3. Network Architecture

Our general network architecture is based on the proposal of [KMM*17]. Their architecture is based on a Multilayer Perceptron (MLP) and uses skip connections between consecutive blocks. One of their insights is, instead of passing all inputs from the top, each level of the descriptor Σ^k is passed into one RPNN block (Figure 5b). Each block consists of three dense layers ($\mathcal{D}_{[3,5]}$ in Figure 5c) in which residuals are mixed with knowledge from all previous levels (o^{k-1}). In the end, three dense layers $\mathcal{D}_{[6,8]}$ refine the gathered information and produce a single output value for P. Please refer to our supplemental for a complete specification.

In the remainder, this basic architecture is referred to as the *baseline* solution. In Section 5 we show that it already performs surprisingly well in our setting. This baseline explicitly differs from [KMM*17]'s method in the number of levels $K$and the per-ray inference with explicit MC single-scattering simulation. As our domain features distinct media (density, heterogeneity) and illumination (diffuse) from the cloud setting, we include straightforward adapations to avoid computational overhead.

For our architecture we incorporate further domain knowledge of volumetric light transport and enhance the baseline by strategically placed weight sharing ($\equiv$) that implements two physical invariants: isometric scaling and rotational invariance.

Isometric Scaling Intuitively, the network should compute the same function on all K levels of the hierarchical volume representation Σ,

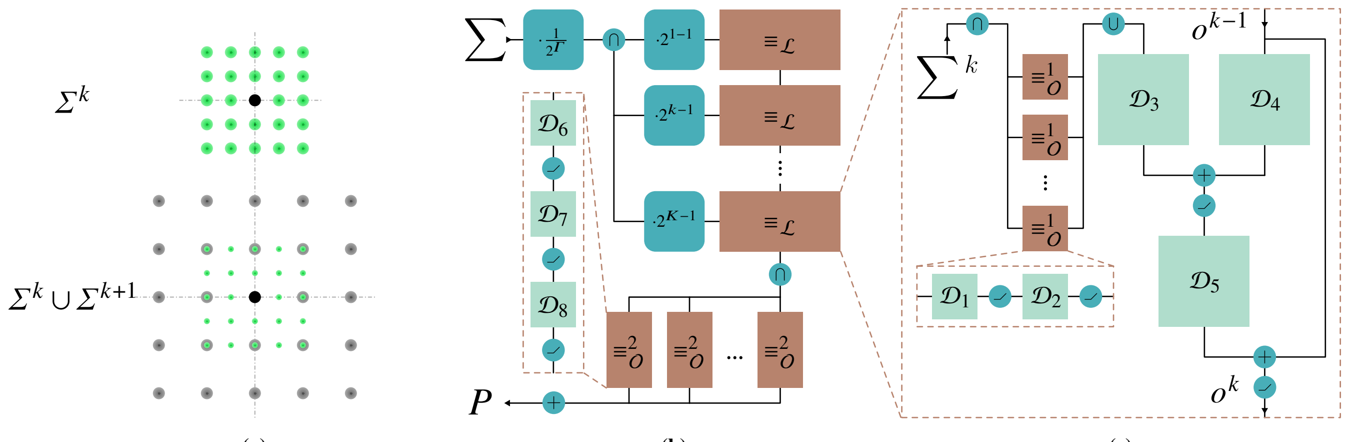
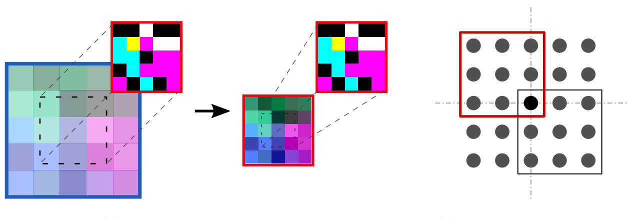


Figure 5: *Network Architecture: (a) visualizes the spatial relationship between levels inside one stencil Σ and the central point* **x**. *(b) Network overview where $\mathcal{D}$ blocks mark dense layers with trainable weights, $\equiv$ blocks indicate shared weights and round edges/circles signify arithmetic operations ($\cap$: slicing, $\cup$: concatenation) The factor $\frac{1}{2^T}$ keeps values in range for training stability. (c) Detail of one RPNN block.*



(a) *Unifying Level Size* **(b)** *Quadrants/Octants*

Figure 6: *Weight sharing. (a) Hierarchy levels are rescaled to the same size (with adjusted optical density) to enable processing with the same weights across levels ($\equiv_{\mathcal{L}}$). (b) On each level, the stencil is split in half per axis and we share weights ($\equiv_O$) over correctly rotated quadrants (octants in three dimensions) to enforce symmetry following from rotational invariance around the black central point* **x**.

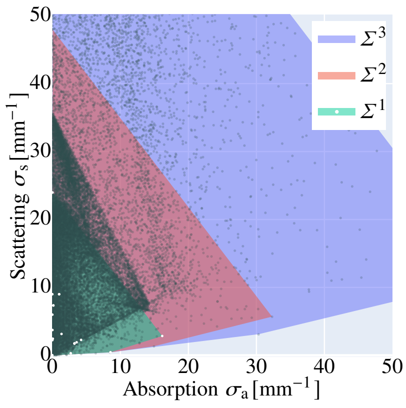


Figure 7: *Input values of three Σ levels during training: On the lowest level Σ^1 (white points), these are the 15 discrete values corresponding to CMYKW at three channels. Higher levels (with isometrically scaled density) also contain interpolated values according to the mixing ratios of base materials.*

that is, determine how much each voxel affects the central point of interest; the only difference is the geometric scale of the input representation. We enforce that identity by isometrically rescaling values according to the extent of each level of our multi-scale input.

In contrast to the well-known relationship for area and volume that scale with powers 2, and 3, respectively, scattering coefficients σ_s and σ_a scale with the power of -1 when the geometric reference frame changes. Figure 6a visualizes that shrinking a coarser level requires the optical density to be increased to keep the same appearance.

As shown in Figure 5b, our network scales all levels Σ^k to the size of the lowest (finest) level, increases optical density ($\cdot 2^{k-1}$) and processes them using the same weights ($\equiv_{\mathcal{L}}$). While this weight sharing primarily reduces the network size, it also, implicitly, augments the discrete-valued training dataset in two ways:

1. Averaging of values for coarser levels ($k > 1$) creates more intermediate data points within the convex hull and mitigates the limitation to discrete materials.
2. Increasing the density then uniformly scales the convex hull of $(\sigma_\mathrm{s}, \sigma_\mathrm{a})$ to larger value ranges.

Figure 7 shows the convex hulls of 10^6 exemplary training points on different levels. Without the isometric scaling, the network would *only* see values from the lowest level (green hull, white discrete points). Using our architecture, the network is forced to learn a consistent function for $\equiv_{\mathcal{L}}$ across a larger range of values. Note how averaging with non-scattering "air" on coarser levels also produces less dense data points towards the coordinate origin.

Rotational Invariance Object rotation against the grid axes should not impact the appearance. We can enforce this invariance by sharing weights between symmetrical parts within a grid around our central point of interest **x**. As illustrated in Figure 6b, we split the data in all three dimensions in the middle (with potentially one row/column overlap), and process all eight parts (*octants*), after correct alignment, using the same filters ($\equiv_O$). We apply this concept twice: First, to each input stencil Σ^k within the already shared level $\equiv_{\mathcal{L}}$ as depicted in Figure 5c ($\equiv_O^1$). Second, at the end of the network ($\equiv_O^2$ in Figure 5b) to produce a single output value per octant. This way, one can weakly enforce rotational invariance without resampling data or using a gridless learning framework. It also implicitly augments the data without explicitly storing duplicates.

Overall, our network consists of only eight trainable layers ($\mathcal{D}_{[1,8]}$) which greatly reduces the number of parameters and improves training speed. We exploit domain knowledge with our weight-sharing and support generalization of the network to unseen volume arrangements and material values.

4.4. Training Protocol

Our implementation is based on Tensorflow [AAB*15] and is using the Adam optimization algorithm [KB14] with a 0.001 learning rate. The networks are trained on dual-socket Intel Xeon CPU E5-2680 machines (40 threads) with 256 GB memory and a Nvidia Tesla K40m GPU (11 GB memory). Both, the implementation and the dataset to this paper are published alongside the paper [RSB*21].

Dataset Loading For optimization with stochastic gradient estimation, the order of training samples needs to be shuffled. With the scale of volumetric data we are working with, the datasets typically do not fit into memory. Despite one stencil Σ being only 11 kB, with 200×10^6 surface points a dataset easily reaches into the order of 2 TB. These are sizes that cannot even be streamed from storage quick enough for GPU processing. Thus, we employ an out-of-core method to process larger datasets using a sliding window approach over volumes within which we randomize samples. Channels of multi-channel (RGB) volumes are mixed to train a single-channel network as light transport is channel-independent.

Training on 340 volumes (sliding window of 110 volumes) took from 32 to 44 hours for the different networks variants. In our setup, the epoch size was set to 2M data points, and the whole training converged after 500 epochs.

Stencil Generation With memory consumption being a concern, we trade storage for computation and do not store explicit lower resolution representations of volumes. Instead, we repeatedly downscale particular neighborhoods of a requested sample point $\mathbf{x}$ using a box filter that doubles in footprint for each level as described in Section 4.2. Efficiently extracting spatial averages on the CPU is key to provide data points quickly for training on the GPU.

We chose to work with Summed Area Tables (SATs) that we implement in a two-level hierarchical way to avoid issues with numerical precision on larger volumes as well as reduce the memory consumption. A hierarchical split allows for storing a sparser representation of the SAT in constant areas by replacing their leaf-level tile with a constant value. Using SATs, it takes a constant-time lookup of eight values to calculate the average for any axis-aligned box in the volume. When querying for a training sample we build a hierarchy of volumetric patches Σ around a surface point $\mathbf{x}$. In contrast to a precomputed downsampling approach, this aligns perfectly for each $\mathbf{x}$ and avoids interpolation artifacts on coarser levels. Avoiding the exploding gradient problem, we rescale all input values to the network by a constant $\frac{1}{2\Gamma}$ with $\Gamma = 6$ as visualized in Figure 5b.

4.5. Inference

Unlike previous work [KMM*17, PN19], our RPNN is not integrated with a path tracer. Instead, it is a full replacement of a renderer within a scattering-aware optimization loop for fabricating textured objects with a color 3D printer [SRB*19]. The RPNN is able to predict the full subsurface light transport in one go without a separate single-scattering simulation as [KMM*17] propose. We are able to do that, because the radiance function is smoother in our setup for two reasons. First, the lighting setup is uniform incidence over the surface, whereas [KMM*17] account for sharp directional illumination. Second, our estimates are not sampling-based, i.e., integrated within the loop of a path tracing renderer as previous work [KMM*17, VKJ19, PN19] did. Thus, the network is only asked to predict the averaged final appearance of the subsurface scattering, not individual paths.

On inference, we extract volumetric patches around each surface point as described in Section 4.4 and process them with the network's computational graph in parallel batches. Each color channel / spectral band is treated individually. This results in a linear runtime dependency on the number of surface points and color channels. Differing from MC rendering, the prediction quality stays constant without requiring multiple evaluations to improve the noise level.

5. Results

We evaluate our results on geometries and textures that have not been part of the training or validation dataset. These objects (shown in Figs. 9 to 12) are chosen to exhibit certain properties we deem important for our application.

WEDGE (6 cm) has varying thickness from 0.5 mm to 6.5 mm. The mirrored texture in complementary colors on each side is designed to show cross-talk at different spatial frequencies and thicknesses.
SHOWCASE (3 cm) features positive and negative curvature. The white side enables judgment of the scattering on thick geometry.
OCTOSTAR (3 cm) has constant thin geometry in various orientations. This object is particularly challenging as it contains loops.
CORK BONE (4 cm) has a high-frequency texture over a wide range of curvature, thickness and axis-orientation.
YELLOW VASE (4 cm) has a rich texture and a thin handle.
RED VASE (5 cm) is partially hollow with average positive and negative curvature.
CAT (6 cm) features thin tubes in various orientations.
THIN PLANE (3 cm) is a 0.5 mm slab with textures on both sides.

The volumetric arrangement of printing materials is achieved either by extruding a mixture of surface color into the depth or through refinement with the method by [SRB*19].

5.1. Scattering Compensation

We integrate our RPNN into the core of an optimization loop that refines volumetric distributions of materials for texture sharpness and contrast [SRB*19]. Such scenario involves repeatedly predicting the surface appearance in order to gauge if adjustments to the volume have shown its desired effect. Any potential systematic errors would get amplified over more iterations and manifest themselves in the material arrangement.

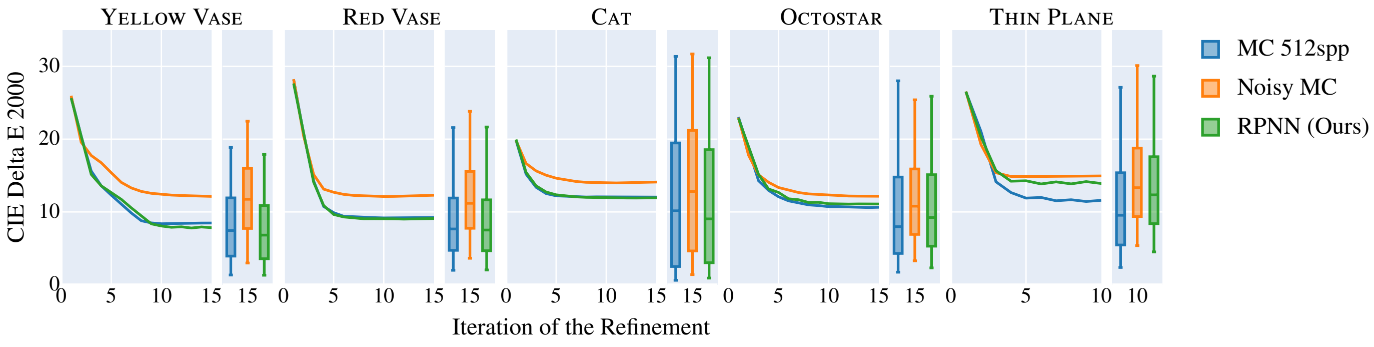


Figure 8: *Convergence plots for the five fabricated objects in Figure 9 using three prediction methods: MC-based refinement with 512 spp, our network-based solution and an equal-time MC prediction. Average CIE dE 2000 metric values against the target object are derived from a reference MC (re-)prediction (512 spp) for all methods. Boxplots show the 0.05, 0.25, 0.5, 0.75, and 0.95 quantiles of the last iteration.*

In a typical run, it requires 8 to 12 iterations for a refinement to converge. Figure 8 plots the perceptual color differences between target and simulated objects in terms of CIE dE 2000 [SWD05] at each iteration. The figure compares a reference refinement (MC 512 spp) to our RPNN-based method and an equal-time MC rendering (4 to 16 spp, depending on the object's surface area). For comparability, the volumes of all three methods have been re-rendered using MC 512 spp to obtain the difference values with equal noise-level. The comparably high level of CIE dE 2000 errors for each model reflects the fact that exact texture reproduction is often impossible in the presence of significant scattering [SRB*19] for the given amount of high frequencies in the texture or the given geometry. Note that the error is calculated against a non-gamut-mapped target.

In most cases, our network-based refinement is on par with the established method of a 512 spp Monte Carlo rendering. For the challenging case of the THIN PLANE our solution performs more comparably to a low-quality MC prediction. This is explainable from the training data, as the thickness (0.5 mm) lies outside the range of our training set (1 mm to 20 mm).

Figure 9 depicts MC preview renderings of the objects obtained from these refinement runs. Overall, reproductions with all three methods show acceptable performance. The results of our RPNN-based method are hard to distinguish from those of the MC 512 spp. Despite remarkable performance of the low-sample MC approach given the very noisy predictions, it shows systematically lower quality than the two other methods. For example, for this method, we observe noise patterns on THIN PLANE and YELLOW VASE, reduced sharpness on RED VASE, and color shifts (within yellowish patches) on CAT. Visually, the distances between methods on THIN PLANE match the differences from the convergence graph in Figure 8.

The computational time required to run the compared methods is listed in Table 1. Our method was run on a desktop-grade workstation with a single Nvidia RTX 2070 GPU and 8-thread Intel CPU. Due to the heterogeneous hardware requirements, we are forced to convert the high-quality method's runtime, originally on a large compute cluster, to a comparable competitor. Again, low-quality MC predictions are based on equal time to RPNN-based prediction and thus not explicitly listed. Our method performs significantly faster with two orders of magnitude speedup for most objects. This ratio is depending on object geometry and grows with geometric volume. The RPNN predictor has a constant runtime per surface point, independent of underlying curvature, geometric volume or medium values. In contrast, with forward scattering materials and high albedo, a MC path-tracer's runtime is dependent on the volume underneath each surface point, as paths can scatter more and deeper into the medium. In our supplementary material, we provide additional speed comparisons with variants of Mitsuba2 [NDVZJ19] including GPU path-tracing.

Table 1: *Prediction timings (per object, per iteration) for results in Figure 9 in the equal-quality setting. MC 512 spp is recalculated to a single 32-thread CPU system – NN is evaluated on a 8-thread CPU + GPU system. Low-spp MC is equal-time to RPNN.*

Object	Surface points	MC 512 spp	RPNN (Ours)	Speedup
YELLOW VASE	1.089×10^6	38 745 s	128 s	303×
RED VASE	1.092×10^6	39 206 s	140 s	280×
CAT	0.637×10^6	20 987 s	75 s	280×
OCTOSTAR	1.833×10^6	29 059 s	236 s	123×
THIN PLANE	0.348×10^6	428 s	16 s	27×

Finally, we also show photographs of actual 3D printouts prepared by both high-quality methods in Figure 10. These are fabricated on a Stratasys J750 printer using the VeroOpaque material family, sanded and varnished with a clear-coat. On all fabricated samples, once again, the two high-quality methods are difficult to discern. Exception to this trend, as mentioned before, is the fabricated THIN PLANE.

5.2. Network Generalization

In the last subsection, we showed our predictor's performance averaged over many iterations. Here, we focus on single forward predictions and show raw network output values by visualizing the surface voxel values.

Evaluating our improvements to the Radiance Predicting Neural Network (RPNN) framework [KMM*17, PN19], we compare different architecture variants within our setting. Because of our weight

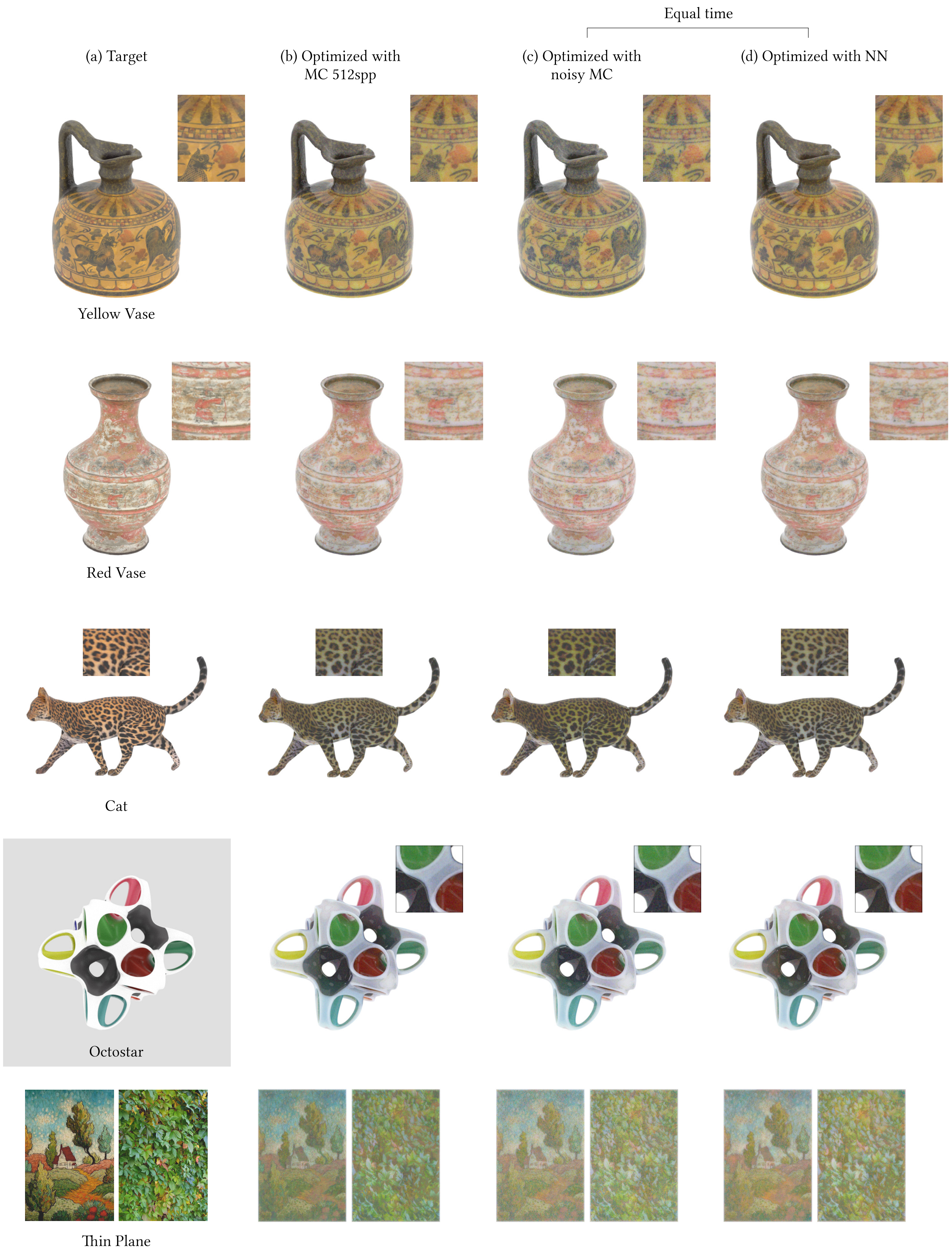


Figure 9: *Comparison of prediction methods inside a refinement loop: (a) target models (b) using 512 spp MC prediction, (c) using equal-time MC prediction with 4 to 16 spp, and (d) our proposed solution using RPNN-based predictions. Refinements with low-sampling MC prediction produce lower quality of texture reproduction compared to the other two. Timings are reported in Table 1.*

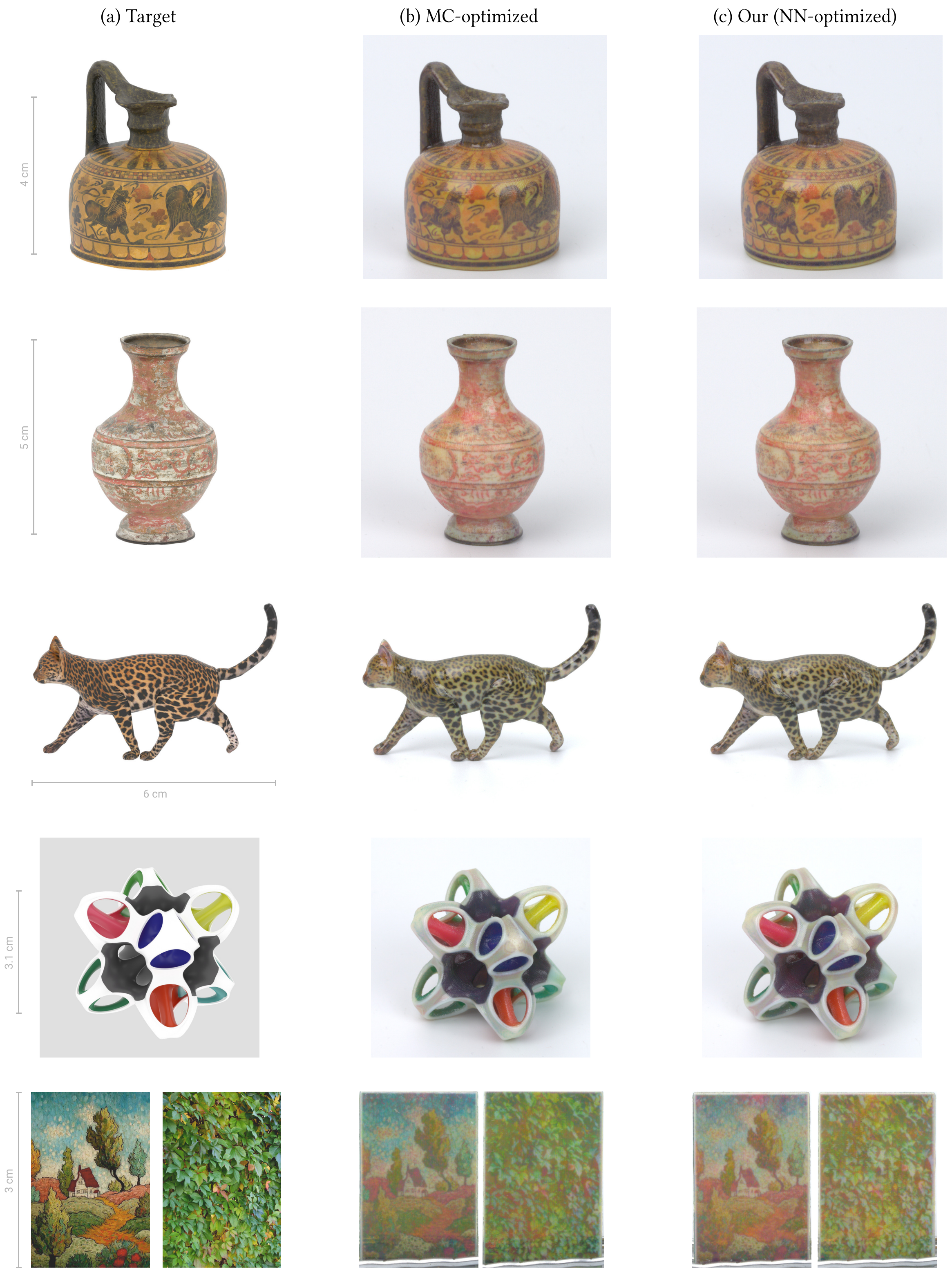


Figure 10: *Comparison of photographed printouts to the rendered target models (a): refinement solution [SRB*19] with MC-based predictions (b), our proposed solution with RPNN-based predictions (c). The bottom object is a thin planar slab* (0.5 mm *thick) with two different textures on its front and back sides.*

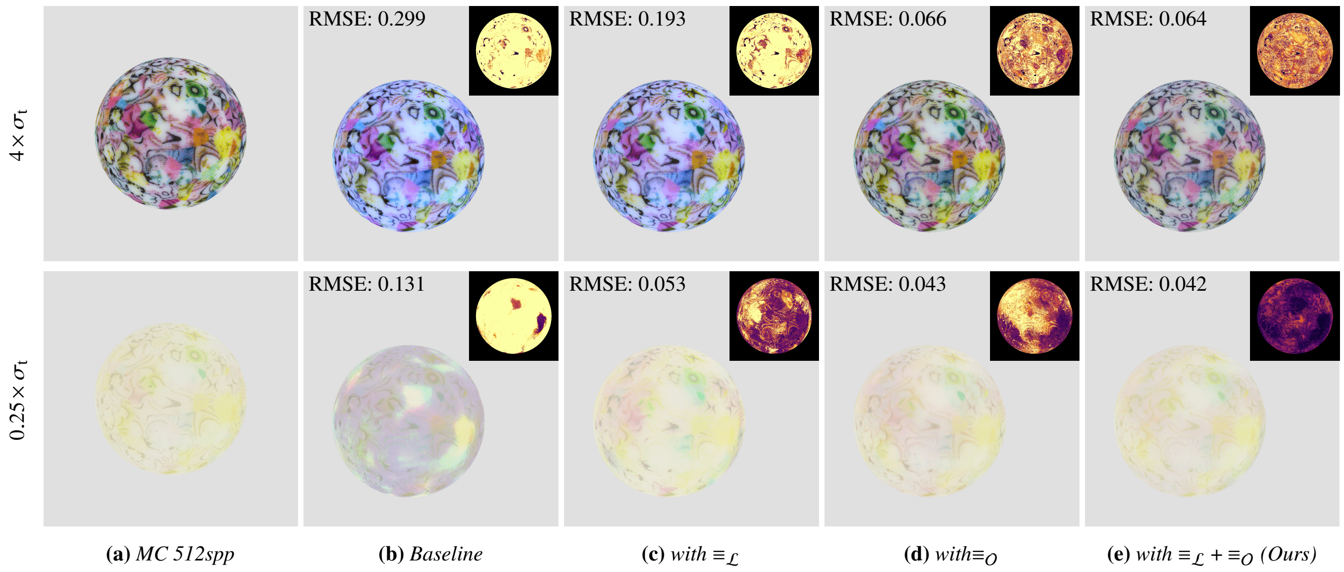


Figure 11: *Single forward predictions comparing the generalization of network architectures to new values. The* 15 mm *sphere is a training example with adjusted density. The top row is optically four times thicker, whereas the bottom row is four times thinner. Our weight-shared architecture handles values outside the training range with less artifacts. CIE dE 2000: 0* ▬ *10*

sharing (≡) in multiple key locations, the number of trainable parameters, i.e., the learning capacity, varies between all four architectures as illustrated in Table 2. Against intuition, the variant with octant weight-sharing $\equiv_O$ increases the number of weights because it adds one overlapping column in each dimension (see Figure 6b) for every layer.

Table 2: *Number of trainable parameters for each architecture given the same stencil dimensions* $I = 5\times3\times9$.

Architecture	**Baseline**	**with** $\equiv_{\mathcal{L}}$	**with** $\equiv_O$	**with** $\equiv_{\mathcal{L}} + \equiv_O$ **(Ours)**
Parameters	2 046 871	365 851	3 767 341	706 861

Values First, we compare generalization over new scattering and absorption values outside the range of the 15 discrete ones in the training dataset. We take a training example and rescale its density to simulate different scattering materials. This way, geometry and volume arrangement have been seen by the networks before, but the values are in a different range. Figure 11 depicts predictions for all four variants. For both rows, the baseline architecture exhibits mispredictions of the appearance, whereas ours handles this unfamiliar environment well. This demonstrates the benefits of our weight sharing over levels ($\equiv_{\mathcal{L}}$), as they implicitly augment the discrete training values to greater ranges as illustrated in Figure 7. In the baseline architecture, each level overfits to a distinct set of (interpolated) values in the lowest level's range.

Shape The results in Section 5.1 suggest that the network prediction performs well for arbitrary geometry. Here, we further verify this and dissect this generalization over shape per architecture. Figure 12 shows the quality level of predictions on general geometry. Despite training only on a very simplistic set, all architectures perform acceptably on concave, thin regions and arbitrary surface orientation against the stencil grid.

While RMSE error values are rather comparable over the architectures, the baseline exhibits distinct bias in multiple cases. On the white face of the SHOWCASE object and around the blue+green cone where it intersects the coordinate axes, the baseline has banding artifacts. For the WEDGE it fails to predict the crosstalk in the thin region correctly.

On the CORK BONE model the bias increases for the thinner parts and slightly less of the weight-shared architecture. This can be attributed to the octant weight sharing $\equiv_O$, as it is less prone to overfitting on specific voxel arrangements in particular voxel positions. Because the weights are shared across all eight octants in the coordinate grid, training for individual voxel arrangements and occupancies in every voxel index is not necessary. Instead, training data from all eight directions contributes to the quality in all directions. Achieving true rotational invariance would require a spherical architecture and representation – a potential topic for future research.

On the challenging case of OCTOSTAR, the level sharing $\equiv_{\mathcal{L}}$ introduces considerable bias on the hollow bridge regions between colored corners. Architecture variants without this feature, such as the baseline, exhibit this problem less pronounced. We discuss this limitation in further detail in Section 6.

5.3. Application: Spectral Prediction

Our data-driven method can also be used for spectral volume predictions in a relatively straightforward manner. This application has important implications, as spectral 3D printing has shown promising results [SBK*18]. While, in this paper, we show only spectral pre-

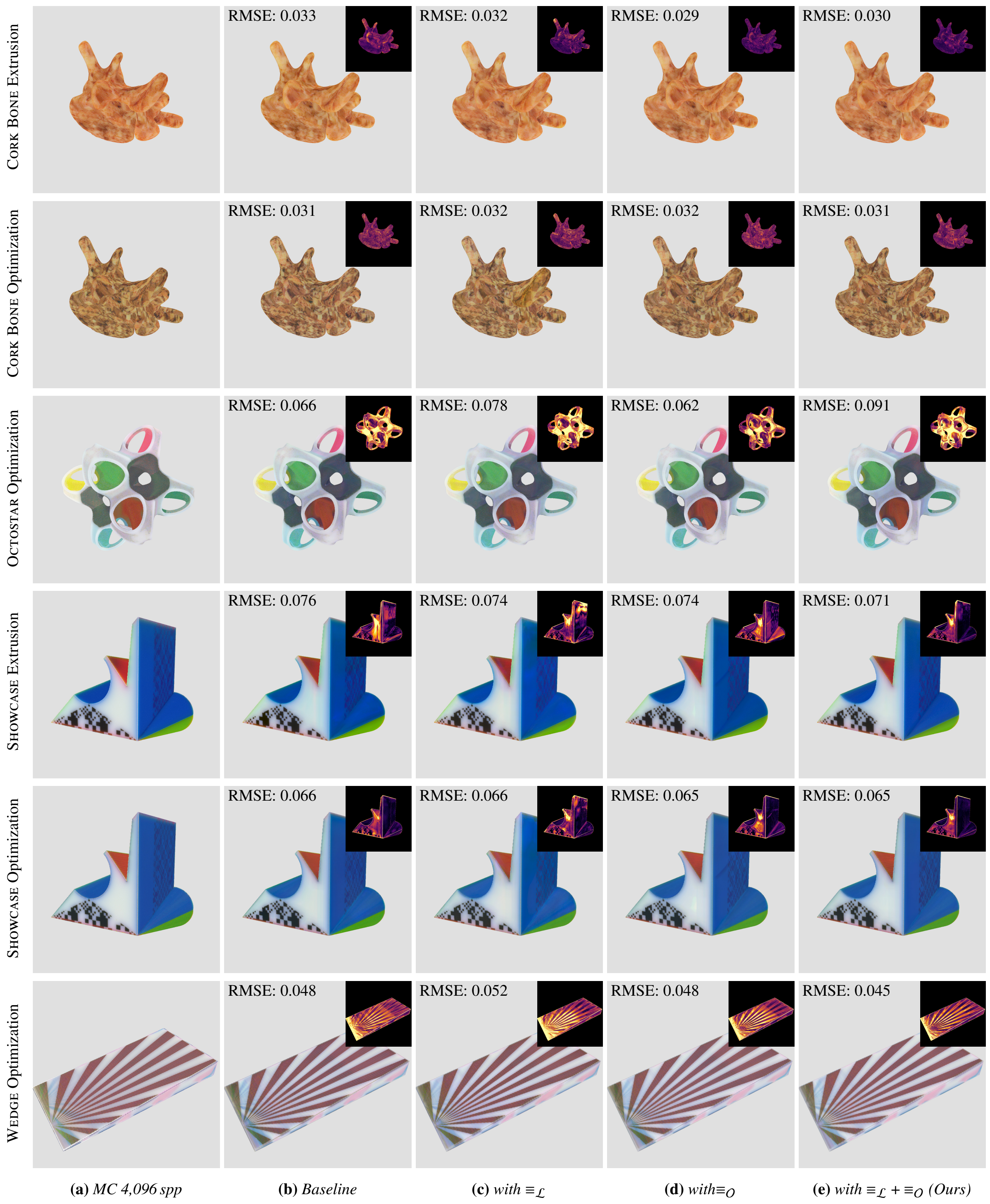


Figure 12: *Single forward predictions comparing four architecture variants over prediction error (RMSE) for general geometry. The baseline, a basic adaptation of [KMM*17]'s RPNN, is compared against our weight sharing over levels $\equiv_{\mathcal{L}}$ and octants $\equiv_{\mathcal{O}}$. The objects' unique features are discussed in Section 5. CIE dE 2000: 0 ▬ 10*

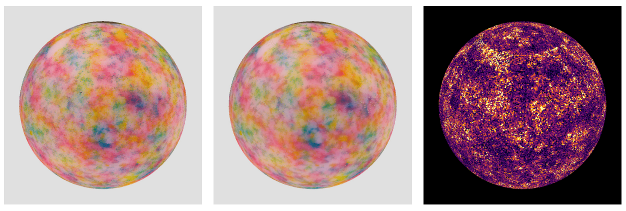


Figure 13: *Spectral predictions for an object formed by virtual spectrally-defined materials. Our network generalizes well over scattering parameters from only 15 discrete values and can accurately predict continuous spectral curves. CIE dE 2000: 0* ▬ *10*

dictions, spectral scattering compensation for 3D printing remains an exciting direction for future work.

Because the network is trained on a single color channel only, it can be repeatedly applied to predict individual spectral bands. In Figure 13, we demonstrate that our method matches the quality of an MC spectral rendering. This object is filled with *virtual* materials whose scattering properties are hand-crafted to mimic the spectral properties of a CMYKW ink set. The network predicts, just like the MC renderer, 31 spectral bins, from 400 to 700 nm, which are converted into RGB using standard procedures [SG31]. For predicting arbitrary spectral curves, a good generalization over scattering values is important. Our network architecture achieves that, as we demonstrate in the previous subsection. As with any spectral image, one can easily vary the diffuse illumination spectrum afterwards and predict the look under different illuminants. This result also highlights the benefits of using a network for soft proofing, because the runtime speedup we report above leaves room for more accurate colorimetric prediction within a practical timeframe.

5.4. Differentiable Rendering Comparison

With the recent publication of Radiative Backpropagation (RB) [NDSRJ20], time and memory costs of differentiable rendering methods become viable for inverse reconstruction of heterogeneous media. The paper contains experiments that hint at a potential application in the field of 3D print preparation. Naturally, the question follows how a heuristic-based iterative scheme [SRB*19] compares to such a generic optimization framework. The available implementation of the experiments require us to recreate their specialized setup and compare both inverse methods in a more simplified environment. As the backward heuristic is independent of the used forward model (MC or RPNN) we chose MC for consistency.

The setup presented in [NDSRJ20] differs from a traditional 3D printing setup in the following attributes:

- Constant density across the volume.
- Isotropic phase function.
- No material discretization (halftoning) in the iteration loop.

This is important to consider for speed (density) and quality (latter two) comparisons. Please refer to our supplemental material for a detailed description of the scenario, a discussion on timings and more results. Here, we summarize the outcome:

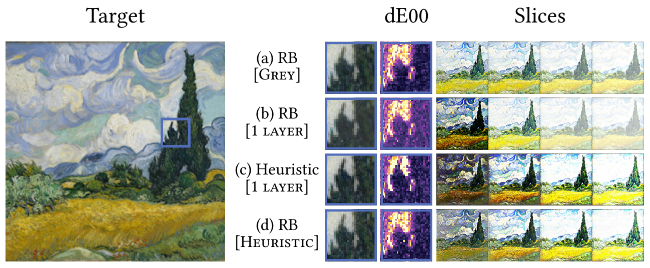


Figure 14: *Comparing Radiative Backpropagation (RB) [NDSRJ20] with heuristic refinement [SRB*19] on different initializations (a-d). The best result for RB (a) matches the quality of (c). On the same initialization (b) details are less pronounced. When continuing (c) with RB, the SGD optimization moves only slightly in the local minimum (d). CIE dE 2000: 0* ▬ *10*

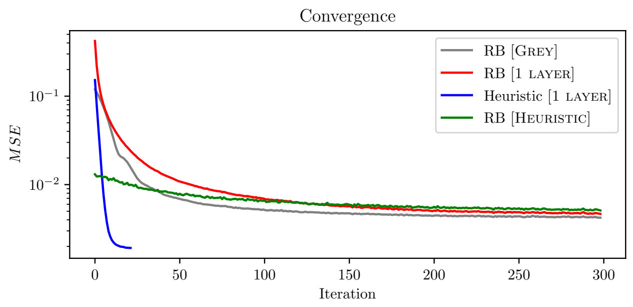


Figure 15: *Convergence behavior of Radiative Backpropagation (RB) [NDSRJ20] and heuristic refinement [SRB*19] for results shown in Figure 14. The heuristic converged after* 24 *iterations and visually matches the quality of RB [Grey] after* 300 *iterations.*

With stochastic gradient descent (SGD) being a local optimization method, the starting condition is important. Figure 14 compares cutouts of four optimizations. Choosing from several initializations, the best result for RB (a) compares in visual quality to the heuristic refinement (c) with (c) being slightly sharper. For the same initialization as (c), RB leaves texture details washed out (b). The most interesting result is (d) where RB is initialized with the result from (c). There, it minorly adjusts edge sharpness, but overall stays visually constant in the same local minimum.

Looking at the resulting volumetric distribution, slices are very different for each method. The heuristic (c) prioritizes compression of absorptive material close to the surface while RB finds different local minima depending on the initialization. Being designed for a particular initial condition the heuristic will not work stable with other initializations.

In conclusion, differentiable rendering is a general tool that requires further research to be applicable for practical 3D print preparation. Besides the required technical updates (proper volume modeling and material discretization), we see that the generic SGD optimization applied to the translucent printing materials produces a less crispy appearance. SGD's convergence behavior is sensitive to initialization and visual quality depends on the used error metric. A simple metric like MSE is not indicative of perceptual considerations and more perceptual metrics have yet to be developed for surfaces in voxel representation.

Most importantly, in the presence of long mean-free paths, the SGD optimization requires much more iterations to converge. As can be seen in Figure 15, the heuristic converges after 24 iterations whereas SGD optimization on MSE requires a long-tailed optimization to increase detail fidelity. When running both methods with the same forward predictor, which is the dominant factor, the absolute speed difference follows the same trend.

6. Discussion

Overall our results confirm that double (nested) weights sharing is a good way to regularize training, reduce weights, increase generalization and inject domain knowledge in an RPNN architecture.

With the generalization over values our published, pre-trained network can be useful even for future materials without re-rendering or re-training, given their measured scattering and absorption coefficients lie within the range illustrated in Figure 7. In case of vastly different value ranges one can still leverage the published dataset by re-rendering and subsequent re-training.

The results also show limitations of the method. When it comes to very thin objects like our THIN PLANE, the network does not extrapolate well from the training data. More examples with reduced thickness would be a first solution to solve that. Also, the architecture exhibits artifacts with highly-concave and closed-loop geometry like our OCTOSTAR and parts on the SHOWCASE. In an attempt to find the root-cause, we included similar objects to the training dataset, but the train- and test-time metrics didn't improve. We suspect the stencil representation being ambiguous between a completely solid object with air around and an open loop on higher levels.

Compared to MC rendering, our method provides faster evaluation times, which is independent of scene complexity and density of the medium. Besides practicality of the print preparation pipeline, it is also an important step for learning to generate a material distribution for a target surface appearance. Only with a fast forward predictor one can prepare enough training data to approach the backward step with machine learning.

Our training data is obtained by means of virtual simulations with an idealized voxel model. In reality, the printing process might induce partial voxel mixing, non-smooth interfaces and anisotropic scattering effects w.r.t. the layer orientation. Thus, a machine learning approach based on extensively measured data seems preferable, but requires a set of costly fabricated samples.

In a comparison with differentiable rendering, heuristic refinement performs on an equal quality level, partially with perceptually slightly "crisper" reproduction. But in terms of convergence, a naïve implementation requires ten times more iterations. An exciting trait of our RPNN forward model is that it is also differentiable and can deliver surface gradients w.r.t. volume patches. Untangling gradients shaped like our stencil Σ for a volumetric reconstruction is left to future work.

7. Conclusion

In this work we proposed a high-fidelity 3D printing pipeline that matches previous work in quality, while typically being 100 to 300 times faster. The key to this was combining the strengths of a state-of-the-art heuristic iterative optimizer for 3D color print preparation with a neural predictor for surface appearance from heterogeneous subsurface scattering.

As we have shown, despite the limited generality of any data-driven model, the network generalizes well to unseen geometry and material values. This robustness lends our solution for real-world deployment. In effect, this is the first time that full heterogenous material optimization for 3D-print preparation becomes possible within time frames in the order of the actual printing time.

Acknowledgments

We thank Sebastian Cucerca for processing and capturing the physical printouts. This work was supported by the Charles University grant SVV-260588 and Czech Science Foundation grant 19-07626S. This project has received funding from the European Union's Horizon 2020 research and innovation programme, under the Marie Skłodowska-Curie grant agreements No 642841 (DISTRO) and No 765911 (RealVision), and under the European Research Council grant agreement No 715767 (MATERIALIZABLE).

References

[AAB*15] ABADI M., AGARWAL A., BARHAM P., BREVDO E., CHEN Z., CITRO C., CORRADO G. S., DAVIS A., DEAN J., DEVIN M., GHEMAWAT S., GOODFELLOW I., HARP A., IRVING G., ISARD M., JIA Y., JOZEFOWICZ R., KAISER L., KUDLUR M., LEVENBERG J., MANÉ D., MONGA R., MOORE S., MURRAY D., OLAH C., SCHUSTER M., SHLENS J., STEINER B., SUTSKEVER I., TALWAR K., TUCKER P., VANHOUCKE V., VASUDEVAN V., VIÉGAS F., VINYALS O., WARDEN P., WATTENBERG M., WICKE M., YU Y., ZHENG X.: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. URL: https://tensorflow.org. 7

[BATU18] BRUNTON A., ARIKAN C. A., TANKSALE T. M., URBAN P.: 3D Printing Spatially Varying Color and Translucency. *ACM Transactions on Graphics (Proc. SIGGRAPH) 37*, 4 (July 2018), 157:1–157:13. doi:10.1145/3197517.3201349. 2

[BAU15] BRUNTON A., ARIKAN C. A., URBAN P.: Pushing the Limits of 3D Color Printing: Error Diffusion with Translucent Materials. *ACM Transactions on Graphics 35*, 1 (Dec. 2015), 4:1–4:13. doi:10.1145/2832905. 2

[BMDS19] BAKO S., MEYER M., DEROSE T., SEN P.: Offline Deep Importance Sampling for Monte Carlo Path Tracing. *Computer Graphics Forum 38*, 7 (Oct. 2019), 527–542. doi:10.1111/cgf.13858. 3

[BVF*17] BABAEI V., VIDIMČE K., FOSHEY M., KASPAR A., DIDYK P., MATUSIK W.: Color Contoning for 3D Printing. *ACM Transactions on Graphics (Proc. SIGGRAPH) 36*, 4 (July 2017), 124:1–124:15. doi:10.1145/3072959.3073605. 2

[BVM*17] BAKO S., VOGELS T., MCWILLIAMS B., MEYER M., NOVÁK J., HARVILL A., SEN P., DEROSE T., ROUSSELLE F.: Kernel-predicting convolutional networks for denoising Monte Carlo renderings. *ACM Transactions on Graphics (Proc. SIGGRAPH) 36*, 4 (July 2017), 97:1–97:14. doi:10.1145/3072959.3073708. 3

[CKS*17] CHAITANYA C. R. A., KAPLANYAN A. S., SCHIED C., SALVI M., LEFOHN A., NOWROUZEZAHRAI D., AILA T.: Interactive reconstruction of Monte Carlo image sequences using a recurrent denoising autoencoder. *ACM Transactions on Graphics (Proc. SIGGRAPH) 36*, 4 (July 2017), 98:1–98:12. doi:10.1145/3072959.3073601. 3

[CLZ*20] Che C., Luan F., Zhao S., Bala K., Gkioulekas I.: Towards Learning-based Inverse Subsurface Scattering. In *Proc. IEEE International Conference on Computational Photography* (Apr. 2020), IEEE Computer Society, pp. 1–12. doi:10.1109/ICCP48838.2020.9105209. 3

[ESZ*17] Elek O., Sumin D., Zhang R., Weyrich T., Myszkowski K., Bickel B., Wilkie A., Křivánek J.: Scattering-aware Texture Reproduction for 3D Printing. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 36*, 6 (Nov. 2017), 241:1–241:15. doi:10.1145/3130800.3130890. 2, 3, 4, 5

[GLA*19] Gharbi M., Li T.-M., Aittala M., Lehtinen J., Durand F.: Sample-based Monte Carlo denoising using a kernel-splatting network. *ACM Transactions on Graphics (Proc. SIGGRAPH) 38*, 4 (July 2019), 125:1–125:12. doi:10.1145/3306346.3322954. 3

[GZB*13] Gkioulekas I., Zhao S., Bala K., Zickler T., Levin A.: Inverse volume rendering with material dictionaries. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 32*, 6 (Nov. 2013), 162:1–162:13. doi:10.1145/2508363.2508377. 3

[HWZ*20] Huo Y., Wang R., Zheng R., Xu H., Bao H., Yoon S.-E.: Adaptive Incident Radiance Field Sampling and Reconstruction Using Deep Reinforcement Learning. *ACM Transactions on Graphics 39*, 1 (Jan. 2020), 6:1–6:17. doi:10.1145/3368313. 3

[HZE*19] Herholz S., Zhao Y., Elek O., Nowrouzezahrai D., Lensch H. P. A., Křivánek J.: Volume Path Guiding Based on Zero-Variance Random Walk Theory. *ACM Transactions on Graphics 38*, 3 (June 2019), 25:1–25:19. doi:10.1145/3230635. 4

[JK21] Jiang G., Kainz B.: Deep radiance caching: Convolutional autoencoders deeper in ray tracing. *Computers & Graphics 94* (Feb. 2021), 22–31. doi:10.1016/j.cag.2020.09.007. 3

[KB14] Kingma D. P., Ba J.: Adam: A method for stochastic optimization. *arXiv:1412.6980 [cs]* (Dec 2014), 1–11. arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980. 7

[KBS15] Kalantari N. K., Bako S., Sen P.: A Machine Learning Approach for Filtering Monte Carlo Noise. *ACM Transactions on Graphics (Proc. SIGGRAPH) 34*, 4 (July 2015), 122:1–122:12. doi:10.1145/2766977. 3

[Kdo14] KdotJPG: OpenSimplex Noise, Sept. 2014. Accessed 2021-02-15. URL: https://uniblock.tumblr.com/post/97868843242/noise. 5

[KKN*18] Keller A., Křivánek J., Novák J., Kaplanyan A., Salvi M.: Machine learning and rendering. In *ACM SIGGRAPH 2018 Courses* (Aug. 2018), ACM, pp. 1–2. doi:10.1145/3214834.3214841. 3

[KMM*17] Kallweit S., Müller T., Mcwilliams B., Gross M., Novák J.: Deep Scattering: Rendering Atmospheric Clouds with Radiance-Predicting Neural Networks. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 36*, 6 (Nov. 2017), 231:1–231:11. doi:10.1145/3130800.3130880. 2, 3, 4, 5, 7, 8, 12

[MMR*19] Müller T., Mcwilliams B., Rousselle F., Gross M., Novák J.: Neural Importance Sampling. *ACM Transactions on Graphics 38*, 5 (Oct. 2019), 145:1–145:19. doi:10.1145/3341156. 3

[NDSRJ20] Nimier-David M., Speierer S., Ruiz B., Jakob W.: Radiative Backpropagation: An Adjoint Method for Lightning-Fast Differentiable Rendering. *ACM Transactions on Graphics (Proc. SIGGRAPH) 39*, 4 (July 2020), 146:1–146:15. doi:10.1145/3386569.3392406. 3, 13

[NDVZJ19] Nimier-David M., Vicini D., Zeltner T., Jakob W.: Mitsuba 2: A Retargetable Forward and Inverse Renderer. *ACM Transactions on Graphics 38*, 6 (Nov. 2019), 203:1–203:17. doi:10.1145/3355089.3356498. 3, 8

[PN19] Panin M., Nikolenko S.: Faster RPNN: Rendering Clouds with Latent Space Light Probes. In *SIGGRAPH Asia 2019 Technical Briefs* (Nov. 2019), SA '19, pp. 21–24. doi:10.1145/3355088.3365150. 3, 4, 7, 8

[RSB*21] Rittig T., Sumin D., Babaei V., Didyk P., Voloboy A., Wilkie A., Bickel B., Myszkowski K., Weyrich T., Křivánek J.: Neural Scattering Prediction: Code and Dataset, 2021. URL: https://github.com/denis-sumin/neural-scattering-prediction. 7

[RWG*13] Ren P., Wang J., Gong M., Lin S., Tong X., Guo B.: Global Illumination with Radiance Regression Functions. *ACM Transactions on Graphics (Proc. SIGGRAPH) 32*, 4 (July 2013), 130:1–130:12. doi:10.1145/2461912.2462009. 3

[SARW*15] Sitthi-Amorn P., Ramos J. E., Wangy Y., Kwan J., Lan J., Wang W., Matusik W.: MultiFab: A Machine Vision Assisted Platform for Multi-material 3D Printing. *ACM Transactions on Graphics (Proc. SIGGRAPH) 34*, 4 (July 2015), 129:1–129:11. doi:10.1145/2766962. 2

[SBK*18] Shi L., Babaei V., Kim C., Foshey M., Hu Y., Sitthi-Amorn P., Rusinkiewicz S., Matusik W.: Deep Multispectral Painting Reproduction via Multi-Layer, Custom-Ink Printing. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 37*, 6 (Dec. 2018), 271:1–271:15. doi:10.1145/3272127.3275057. 1, 2, 11

[SG31] Smith T., Guild J.: The C.I.E. colorimetric standards and their use. *Transactions of the Optical Society 33*, 3 (Jan 1931), 73–134. doi:10.1088/1475-4878/33/3/301. 13

[SRB*19] Sumin D., Rittig T., Babaei V., Nindel T., Wilkie A., Didyk P., Bickel B., Křivánek J., Myszkowski K., Weyrich T.: Geometry-Aware Scattering Compensation for 3D Printing. *ACM Transactions on Graphics (Proc. SIGGRAPH) 38*, 4 (July 2019), 111:1–111:14. doi:10.1145/3306346.3322992. 1, 2, 3, 4, 7, 8, 10, 13

[Str20] Stratasys: Stratasys ultimate full-color multi-material 3D printer. https://www.stratasys.com/3d-printers/j8-series, 2020. Accessed 15-09-2020. 2

[SVS*20] Sedlák J., Vocílka O., Slaný M., Polzer A., Varhaník M.: Design and Production of Eye Prosthesis Using 3D Printing. *MM Science Journal* (Mar. 2020), 3806–3812. doi:10.17973/MMSJ.2020_03_2019127. 1

[SWD05] Sharma G., Wu W., Dalal E. N.: The CIEDE2000 Color-Difference Formula: Implementation Notes, Supplementary Test Data, and Mathematical Observations. *Color Research & Application 30*, 1 (2005), 21–30. doi:10.1002/col.20070. 8

[TFT*20] Tewari A., Fried O., Thies J., Sitzmann V., Lombardi S., Sunkavalli K., Martin-Brualla R., Simon T., Saragih J., Niessner M., Pandey R., Fanello S., Wetzstein G., Zhu J.-Y., Theobalt C., Agrawala M., Shechtman E., Goldman D. B., Zollhöfer M.: State of the Art on Neural Rendering. *Computer Graphics Forum (Eurographics State of the Art Reports) 39*, 2 (July 2020), 701–727. doi:https://doi.org/10.1111/cgf.14022. 3

[UTB*19] Urban P., Tanksale T. M., Brunton A., Vu B. M., Nakauchi S.: Redefining A in RGBA: Towards a Standard for Graphical 3D Printing. *ACM Transactions on Graphics 38*, 3 (May 2019), 21:1–21:14. doi:10.1145/3319910. 1

[VKJ19] Vicini D., Koltun V., Jakob W.: A Learned Shape-Adaptive Subsurface Scattering Model. *ACM Transactions on Graphics (Proc. SIGGRAPH) 38*, 4 (July 2019), 127:1–127:15. doi:10.1145/3306346.3322974. 3, 4, 7

[VRM*18] Vogels T., Rousselle F., Mcwilliams B., Röthlin G., Harvill A., Adler D., Meyer M., Novák J.: Denoising with kernel prediction and asymmetric loss functions. *ACM Transactions on Graphics (Proc. SIGGRAPH) 37*, 4 (July 2018), 124:1–124:15. doi:10.1145/3197517.3201388. 3

[ZWZ*19] Zhang C., Wu L., Zheng C., Gkioulekas I., Ramamoorthi R., Zhao S.: A differential theory of radiative transfer. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 38*, 6 (Nov. 2019), 227:1–227:16. doi:10.1145/3355089.3356522. 3

[ZZ19] Zheng Q., Zwicker M.: Learning to Importance Sample in Primary Sample Space. *Computer Graphics Forum 38*, 2 (June 2019), 169–179. doi:https://doi.org/10.1111/cgf.13628. 3