Look-Ahead Training with Learned Reflectance Loss for Single-Image SVBRDF Estimation

XILONG ZHOU, Texas A&M University, USA
NIMA KHADEMI KALANTARI, Texas A&M University, USA

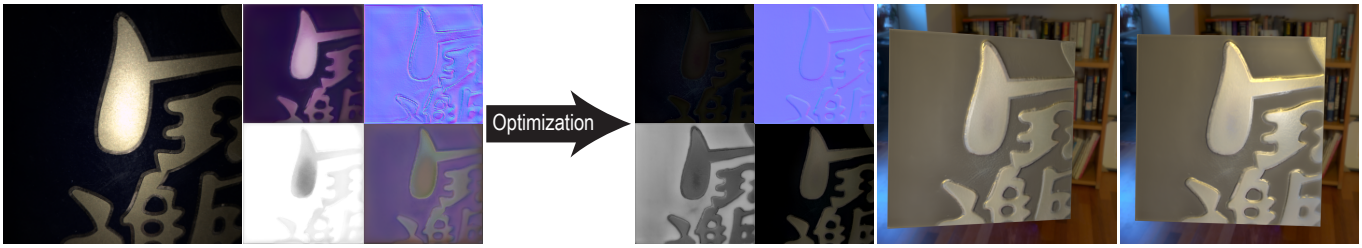

Fig. 1. We propose a novel optimization-based method for estimating the reflectance parameters given a single image. Test-time optimization for single image reflectance estimation is susceptible to overfitting as this is a highly ill-posed problem. To combat this issue, we introduce a novel approach that incorporates the test-time optimization into the training process. Specifically, we train our network by minimizing the error between the ground truth and network's output *after* the test-time optimization. Through this strategy, we ensures that our network learns a prior that is suitable for test-time optimization. Here, we show the results of our network before (initial) and after (optimized) a few iterations of test-time optimization, as well as the rerendered images using the optimized reflectance parameters. We provide comparisons against several state-of-the-art methods on a large number of scenes in the supplementary materials.

In this paper, we propose a novel optimization-based method to estimate the reflectance properties of a near planar surface from a single input image. Specifically, we perform test-time optimization by directly updating the parameters of a neural network to minimize the test error. Since single image SVBRDF estimation is a highly ill-posed problem, such an optimization is prone to overfitting. Our main contribution is to address this problem by introducing a training mechanism that takes the test-time optimization into account. Specifically, we train our network by minimizing the training loss after one or more gradient updates with the test loss. By training the network in this manner, we ensure that the network does not overfit to the input image during the test-time optimization process. Additionally, we propose a learned reflectance loss to augment the typically used rendering loss during the test-time optimization. We do so by using an auxiliary network that estimates pseudo ground truth reflectance parameters and train it in combination with the main network. Our approach is able to converge with a small number of iterations of the test-time optimization and produces better results compared to the state-of-the-art methods.

CCS Concepts: • **Computing methodologies** → **Rendering**.

Additional Key Words and Phrases: SVBRDF capture, machine learning

ACM Reference Format:
Xilong Zhou and Nima Khademi Kalantari. 2022. Look-Ahead Training with Learned Reflectance Loss for Single-Image SVBRDF Estimation. *ACM Trans. Graph.* 41, 6, Article 266 (December 2022), 12 pages. https://doi.org/10.1145/3550454.3555495

Authors' addresses: Xilong Zhou, zhouxilong199213@tamu.edu, Texas A&M University, College Station, USA; Nima Khademi Kalantari, nimak@tamu.edu, Texas A&M University, College Station, USA.

0730-0301/2022/12-ART266 $15.00
https://doi.org/10.1145/3550454.3555495

1 INTRODUCTION

Capturing the spatially-varying bidirectional reflectance distribution function (SVBRDF) of materials is an important problem in computer graphics and vision. While the materials can be accurately captured using specialized hardware [Dong et al. 2010; Ghosh et al. 2007, 2010; Kang et al. 2018], the acquisition process is time-consuming and the setups are often bulky and expensive. In recent years, there has been a significant interest in casual acquisition of the reflectance parameters using standard hardware (e.g., cellphone cameras). To this end, a large number of approaches estimate the material properties given one or a small number of images. While with multiple images, the problem becomes more constrained, this comes at the cost of requiring careful calibration of the input images, which is cumbersome for an average user.

To make the system more practical, we focus on estimating the reflectance parameters from a single image. The existing methods can generally be categorized into two classes of direct-estimation and optimization-based approaches. The approaches in the direct-estimation category [Deschaintre et al. 2018, 2019; Guo et al. 2021; Zhou and Kalantari 2021] train a neural network to directly estimate the SVBRDF maps from the input image. Through the training process, these networks are able to learn powerful priors and handle this highly ill-posed problem. However, these approaches may not properly reproduce the appearance of the material.

Optimization-based methods avoid this problem by performing a test-time optimization where the objective is to find reflectance parameters that minimize the error between the rerendered and the input image(s), known as the *rendering loss*. This problem, however, is highly ill-posed as there are many invalid SVBRDF maps

that can reproduce the appearance of the input image(s). Existing methods [Gao et al. 2019; Guo et al. 2020] propose various ways to constrain the optimization. These approaches are able to produce plausible results with multiple images, but with a single image, which is the focus of this paper, the optimization is extremely ill-posed, and thus their constraints become less effective.

Our goal is to design a system that enjoys the benefits of the two categories of methods, namely the learned powerful priors of the direct-estimation methods and the ability to reproduce the measured appearance through test-time optimization. At the first glance, it seems that our goal can be achieved in a straightforward manner by first training a network on a large dataset and then optimizing its parameters on the test image at hand. Unfortunately, this simple approach is not effective as the network overfits to the input image during the test-time optimization. We make two key observations to address the challenges with such a system: **1)** while by pre-training the network powerful priors can be learned, these priors are not able to avoid overfitting during the test-time optimization process, as the training and testing processes are disjoint, and **2)** the ill-posedness of the optimization comes from the mismatch between the test-time objective (typically the rendering loss), and the desired objective which is the error between the network's output and ground truth reflectance maps.

Inspired by the optimization-based meta-learning approaches [Finn et al. 2017], we propose to systematically address the first issue by incorporating the test-time optimization in the training process. Specifically, we train the network by minimizing the error between the estimated reflectance parameters, *after the test-time optimization*, and ground truth. At every iteration of training, we first perform the test-time optimization to obtain the updated network. This updated network is then used to generate the reflectance parameters. We then compute the loss between the estimated and ground truth reflectance parameters and use it to update the original network at this iteration. Essentially, we train the network by minimizing the training loss after "looking ahead" according to the test loss. Through this training, we basically learn a prior that is suitable for the test-time optimization process and ensures that the network produces optimal reflectance parameters *after* the test-time optimization.

To combat the second problem, we propose to minimize a learned *reflectance loss* in addition to the rendering loss, during the test-time optimization. To do so, we use an auxiliary network to estimate the reflectance parameters from a single image and compute the error between these pseudo ground truth SVBRDF parameters and the estimated parameters by the main network. In our system, this auxiliary network in combination with the main network is trained using our proposed training strategy. Through this process, the auxiliary network learns to estimate reflectance maps that are effective for optimizing the network during test-time optimization.

Once trained, our system is able to produce high-quality results (see Fig. 1) through only a small number of iterations (seven in our case) of the test-time optimization (as opposed to thousands in previous approaches [Gao et al. 2019; Guo et al. 2020]). We demonstrate that our system generates better results than the existing direct-estimation and optimization-based approaches, while being two orders of magnitude faster than the optimization-based methods.

In summary, our paper makes the following contributions:

- We propose a novel training strategy for single image SVBRDF estimation to achieve robustness to overfitting during the test-time optimization.
- We present a learned reflectance loss to compliment the rendering loss during the test-time optimization.
- We extensively compare our approach against several state-of-the-art methods on various datasets, including our new dataset of real images with ground truth.

2 RELATED WORK

Reflectance capture has been the subject of extensive research in the past decades and many powerful approaches have been developed. For brevity, here we only focus on approaches that estimate the material parameters using one or a small number of images, captured with standard cameras. We discuss these methods by classifying them into two categories of direct-estimation and optimization-based techniques. Since our method is inspired by model agnostic meta-learning [Finn et al. 2017], we also provide a brief overview of relevant approaches.

2.1 Direct-Estimation Methods

The approaches in this category train a machine learning model (e.g., neural network) on a large material dataset in an offline manner. During testing, the model is used to directly estimate the reflectance parameters from the input image(s). Several approaches [Deschaintre et al. 2018; Li et al. 2017, 2018; Ye et al. 2018] propose to train deep neural networks to estimate reflectance parameters from a single image. Deschaintre et al. [2019] propose a network architecture to handle an arbitrary number of images. A couple of methods [Vecchio et al. 2021; Zhou and Kalantari 2021] pose the problem as image to image translation and tackle it with a generative adversarial network. To properly handle the saturated highlights, Guo et al. [2021] propose a highlight-aware convolution operator. Since these approaches generate the reflectance maps by a forward pass through the network, their estimated SVBRDF may not accurately reproduce the appearance of the input image(s). We also train our network on large dataset, but we do so by taking the test-time optimization into account to be able to adapt our system to each example.

2.2 Optimization-Based Methods

These methods estimate the material parameters by performing optimization on the test example at hand. Since the problem is highly ill-posed, various strategies have been proposed to constrain the optimization. For example, Hui et al. [2017] propose an optimization strategy with strong material prior by representing the materials as a linear combination of a set of material bases. Unfortunately, the hand-crafted prior is not universal and this method is restricted to materials that can be explained by the bases.

Aittala et al. [2016; 2015] propose an approach to estimate SVBRDF parameters from two- and one-shot photographs. Specifically, they perform the optimization on several tiles extracted from the input image. Zhao et al. [2020] optimize a network on patches of a single example and constrain the problem by ensuring the generated diffuse map is close to an initial rough diffuse estimate. Henzler et al. [2021a] propose to optimize a generator with image conditioned

latent space. These approaches considerably constrain and simplify the optimization by restricting the problem to stationary materials.

To handle general materials, Deschaintre et al. [2020] propose a fine-tuning approach to improve the results of a pre-trained network on a test example, but require multiple carefully captured examples of the same scene. Shi et al. [2020] propose to optimize the parameters of a procedural node graph with differentiable building blocks. The main limitation of this approach is that a node graph, corresponding to the input image, is required to achieve reasonable results. However, designing such a graph is not an easy task for a novice user.

A couple of methods [Gao et al. 2019; Guo et al. 2020] propose to perform the optimization in the latent space of a material generator. Specifically, Gao et al. [2019] first train an autoencoder on a large training dataset. They then optimize the latent space of this network to minimize the error between the rerendered and input images. Guo et al. [2020] use a similar strategy, but instead train the StyleGAN2 network [Karras et al. 2020] to act as the material generator. Unfortunately, since their trained material generator is not highly expressive, optimizing the latent space is not sufficient for producing high-quality results. Therefore, they often rely on post-refinement [Gao et al. 2019] or noise vector optimization [Guo et al. 2020] to be able to match the appearance of the input. However, such strategies reduce the effectiveness of the constraints, and thus the performance of these approaches significantly reduces when fewer images are used as the input. Moreover, these techniques require a large number of iterations at the test-time, making them computationally expensive. In contrast to these methods, we do not constrain the problem, but avoid overfitting by incorporating the test-time optimization into the training process.

2.3 Meta-Learning

Meta-learning or learning to learn refers to a group of methods that aim to train a machine learning model that is able to quickly learn new unseen tasks from a few labeled examples. Our work is most related to model agnostic meta-learning (MAML) [Finn et al. 2017] that formulates the meta-learning problem as two (inner and outer) nested optimizations. Specifically, the inner optimization "trains" the network using a small number of gradient updates, while the outer optimization ensures that the updated network performs well. This training obtains network parameters that are highly sensitive to the loss, i.e., a small change in the parameters using a few gradient updates leads to large improvements in the loss.

While our approach is inspired by MAML and its various extensions [Antoniou et al. 2019; Nichol et al. 2018; Rajeswaran et al. 2019], our goal and assumptions are fundamentally different from theirs. The goal of MAML is quick adaptation of the network to the target *domain* and its main assumption is that a few labeled datapoints in the target domain are available. Most of the current use cases of MAML are few-shot learning where the goal is to learn a new task (e.g., classifying unseen categories) with a few labeled data. On the other hand, our goal is to adapt the network to a *single* test example, where ground truth data is not available.

A few methods [Bergman et al. 2021; Sitzmann et al. 2020a; Tancik et al. 2021] adapt MAML with almost no modification for test-time optimization of the coordinate-based networks. These methods focus on applications where the objective is to overfit a coordinate-based network to a single example, e.g., an image. The main purpose of using MAML in these approaches is speeding up the optimization as they have access to the ground truth during the testing phase. In contrast, in our system, ground truth data for the test example is not available and our goal is to become less prone to overfitting during testing. Unlike these methods, faster convergence is merely a byproduct of our system and not our main objective.

3 ALGORITHM

Our goal is to estimate the reflectance parameters $\mathbf{F}$ from a single flash image $\mathbf{I}$ of a near-planar surface, captured using a colocated camera and flash. To represent reflectance, we use the Cook-Torrance model [Cook and Torrance 1982] with GGX [Walter et al. 2007] that describes the SVBRDF in terms of four feature maps: diffuse, normal, roughness and specular. To ease the need for calibration, we follow the existing techniques [Aittala et al. 2016, 2015; Henzler et al. 2021b] and assume that the input image is captured in a fronto-parallel setting. This way the direction from every point on the surface to the light is known. Similar to Henzler et al. citehenzler2021neuralmaterial, we assume a FOV of 45° to approximate smartphone cameras.

As mentioned, we would like to enjoy the benefits of the direct-estimation and optimization-based approaches, while avoiding their shortfalls. Specifically, direct-estimation methods learn powerful priors, but since their output is generated with a feedforward pass through the network, they may not be able to reproduce the appearance of the input. The optimization-based methods, on the other hand, are able to reproduce the measured appearance, but are susceptible to overfitting as the optimization using a single image is highly ill-posed.

A straightforward way to combine the two approaches is to first train a neural network f_θ on a large material dataset of N input images, $\mathbf{I}_1, \ldots, \mathbf{I}_N$, and corresponding ground truth reflectance parameters $\mathbf{F}_1, \ldots, \mathbf{F}_N$. The training can be done by performing the following optimization:

$$\theta_{\text{opt}} = \arg\min_\theta \sum_{n=1}^{N} E_{\text{train}}(f_\theta(\mathbf{I}_n), \mathbf{F}_n), \tag{1}$$

where E_{train} is the training loss and can be, for example, the $L2$ distance between the estimated and ground truth SVBRDF parameters (*reflectance loss*).

This pre-trained network can then be adapted to the test example at hand by forcing the network to reproduce the appearance of the test image. Specifically, this can be done by performing the following test-time optimization:

$$\theta^* = \arg\min_\theta E_{\text{test}}(f_\theta(\mathbf{I}), \mathbf{I}), \tag{2}$$

where E_{test} is the test loss. For example, it can be the $L2$ distance between the input and rerendered $R(f_\theta(\mathbf{I}))$ images, where $f_\theta(\mathbf{I})$ is the estimated reflectance parameters and R denotes the differentiable Cook-Torrance rendering model. Note that the initial θ is set to θ_{opt}, obtained from Eq. 1, as the goal of this process is to fine-tune the pre-trained network.

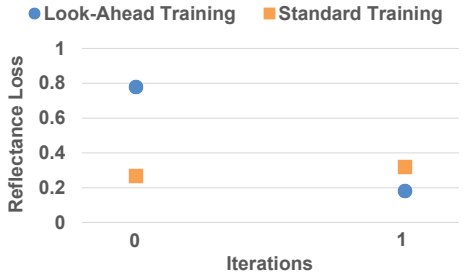

Fig. 2. We plot the reflectance loss (Eq. 11) during test-time optimization for the networks trained in a standard manner and using our look-ahead training strategy. All the values are obtained by averaging the results on a synthetic validation dataset containing 80 examples. Test-time optimization of the network, trained in a standard manner, increases the reflectance error as the network overfits to the input image. While the loss for our initial network is higher, our results after test-time optimization are significantly better than the alternative.

Unfortunately, minimizing the test loss does not necessarily improve the quality of the estimated reflectance maps with respect to the ground truth. As shown in Fig. 2 (standard training), the quality of the reflectance maps degrades with the test-time optimization.

In the next sections, we discuss our proposed look-ahead training and learned reflectance loss to address this issue. The overview of our method is provided in Fig. 3.

3.1 Look-Ahead Training

As discussed, by training the network using Eq. 1, we ensure that the network with parameters θ_{opt} produces optimal results compared to the ground truth training examples in terms of the E_{train} loss. However, with such a training, we will not be able to control the behavior of the network during the test-time optimization.

Our key observation is that this issue arises because of the disconnect between the expected outcome during training and testing; during training we focus on the direct output of the network, while during the test, we are interested in the output of the network after optimization. Inspired by the model-agnostic meta learning (MAML) approach [Finn et al. 2017], we propose to incorporate the test-time optimization into the training process. In our system, the training is done by minimizing the training loss generated by the network *after optimization* using the test loss. In an essence, we "look ahead" to observe the network in the future based on the test loss and maximize its performance according to the training loss.

We discuss our look-ahead training objective with one test loss gradient update, since we use this setting in our implementation and it simplifies the notation. However, extension to multiple gradient updates is straightforward. Our objective is defined as follows:

$$\theta^*_{\text{init}} = \arg\min_{\theta} \sum_{n=1}^{N} E_{\text{train}}\left(f_{\theta'_n}(\mathbf{I}_n), \mathbf{F}_n\right). \tag{3}$$

The key difference between this objective and the one in Eq. 1 is in the network weights. Instead of using the initial network weights θ, we first apply one gradient update based on the test loss on the n^{th} training example to obtain the updated network weights θ'_n. These weights are in turn used in Eq. 3 to evaluate the training loss. The updated network weights θ'_n are obtained as follows:

$$\theta'_n = \theta - \alpha \nabla_\theta E_{\text{test}}\left(f_\theta(\mathbf{I}_n), \mathbf{I}_n\right). \tag{4}$$

where α is the learning rate which controls the magnitude of the gradient step. Note that the computed training loss in Eq. 3 is back-propagated to update the network weights *before* taking the gradient updates using the test loss (θ), as shown in Fig. 3 (left).

Our training is in contrast to standard training, used in direct-estimation methods, that learns a prior that is suitable for directly estimating the output. Such a prior has no impact on the test-time optimization. However, by minimizing the training loss of the optimized network, we essentially learn a prior that controls the behavior of the network during the test-time optimization and avoids overfitting to the input. An additional benefit of our training is that our system quickly converges to a high-quality solution as the training is performed with a single gradient update of the test loss.

We show the advantage of our training strategy compared to the standard training in Fig. 2. As seen, while our initial network (0 iteration) produces worse results than standard training, after one iteration of test-time optimization, we are able to produce significantly better results than the alternative. In contrast, test-time optimization of a network trained with standard training degrades the quality of the results.

Discussion. We note that our training can be thought of as a generalization of the standard training, which is used in the direct-estimation methods. This is because α, which controls the magnitude of the gradient step in Eq. 4, can be adjusted to increase or decrease the impact of the test-time optimization. In the limit, when α is equal to zero, our test-time optimization becomes ineffective and our training boils down to standard training.

3.2 Learned Pseudo Reflectance Loss

As discussed, since the ground truth reflectance parameters are not available during testing, our training and testing losses are different. We use a combination of rendering and reflectance errors as our training loss (see Eq. 9), but can only use the rendering loss for our testing objective. This mismatch between the training and testing losses could potentially create difficulty for our look-ahead training.

We illustrate this problem in Fig. 4. Ideally, our testing loss would be the same as our training loss. As shown, taking gradient step using the ideal testing loss (red arrows) moves us closer to the optimal parameters for each test example (θ^*_1 and θ^*_2). However, with the rendering error as the test loss, our optimization moves in a direction (green arrows) that could potentially be drastically different from the direction towards the optimal parameters.

To address the mismatch between the training and testing losses, we need to augment the rendering loss with the reflectance loss. Unfortunately, ground truth reflectance parameters are not available during testing. Therefore, we propose to estimate pseudo ground truth reflectance parameters using a separate auxiliary network. This network f_ψ takes the image as the input and estimates the pseudo ground truth reflectance parameters $\tilde{F}$. In our test loss (see Eq. 7), we minimize the distance between the estimated and pseudo ground truth parameters (*pseudo reflectance loss*) in addition to the rendering loss. The gradient of the pseudo reflectance loss (blue

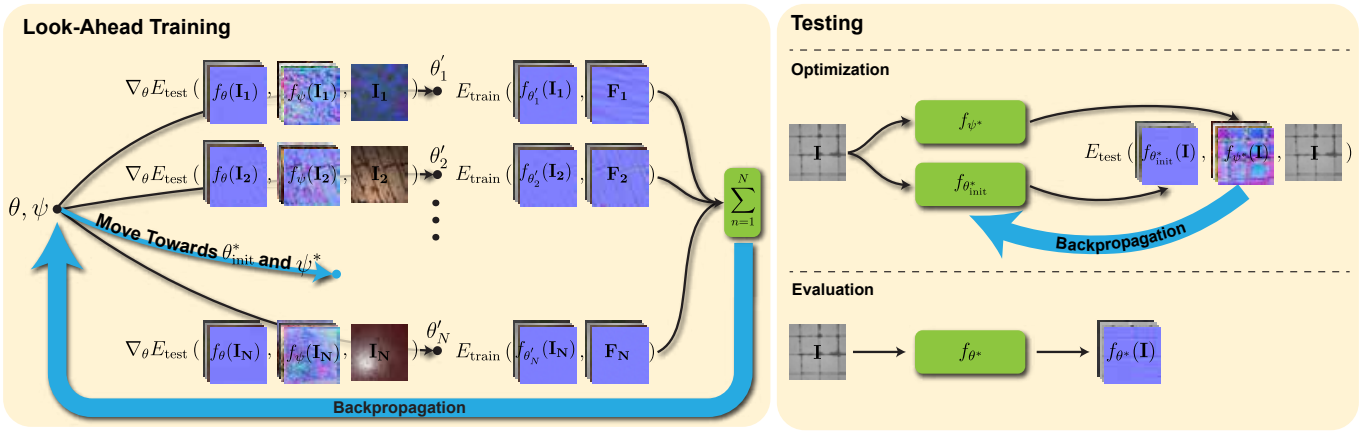

Fig. 3. On the left, we show an overview of our training strategy. Given the current weights for the main and auxiliary networks (θ and ψ), we take one gradient step using the test loss to obtain the updated main network parameters. This process is done for every example in the training data (or mini-batch) and results in a set of updated weights $\theta'_1 \ldots, \theta'_N$. These weights are then used to evaluate the training loss on the corresponding training examples. The training losses are then averaged and backpropagated to update both θ and ψ (shown with the blue arrow). Note that, we only update the main network using the test loss, but update both main and auxiliary network using the training loss. Moreover, the illustration is done for batch training (all N training examples), however, in practice we use mini-batches. On the right, we show an overview of the testing phase. We pass the the input image to both the main and auxiliary networks and evaluate the test loss. This loss is then backpropagated to update the parameters of the main network. Once converged, we use the optimized network to evaluate the final results (bottom-right).

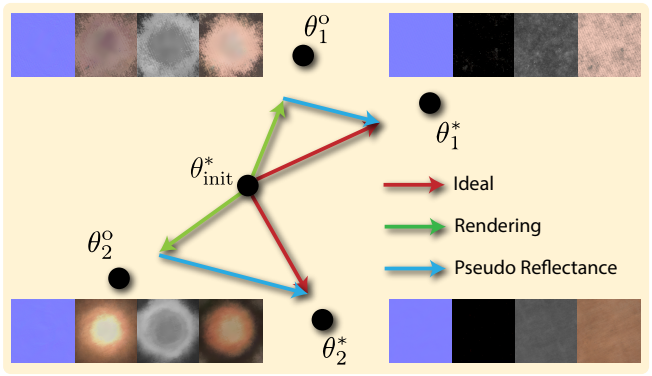

Fig. 4. Ideally, the training and testing losses are the same. In this case, the gradient update of the test loss for each example (red arrows), moves us from the initial network (θ^*_{init}) closer to the optimal network for that example (θ^*_1 or θ^*_2). However, since in practice ground truth reflectance maps are not available during testing, our training and testing losses are different. Unfortunately, the gradient update of the typically used rendering loss (green arrows) moves us in the direction of the optimal parameters based on this loss (θ^o_1 and θ^o_2). To address this mismatch, we propose a learned reflectance loss which approximates the difference in gradient between the rendering and ideal test loss (blue arrows).

arrows in Fig. 4) compensates the difference in gradients of the rendering and the ideal test losses.

We optimize the weights of the auxiliary network ψ along with the weights of the main network θ during the look-ahead training process. Specifically, our final training objective is as follows:

$$\theta^*_{\text{init}} = \arg\min_{\theta,\psi} \sum_{n=1}^{N} E_{\text{train}}\left(f_{\theta'_n}(\mathbf{I}_n), \mathbf{F}_n\right), \tag{5}$$

where θ'_n is calculated as follows:

$$\theta'_n = \theta - \alpha \nabla_\theta E_{\text{test}}\left(f_\theta(\mathbf{I}_n), f_\psi(\mathbf{I}_n), \mathbf{I}_n\right). \tag{6}$$

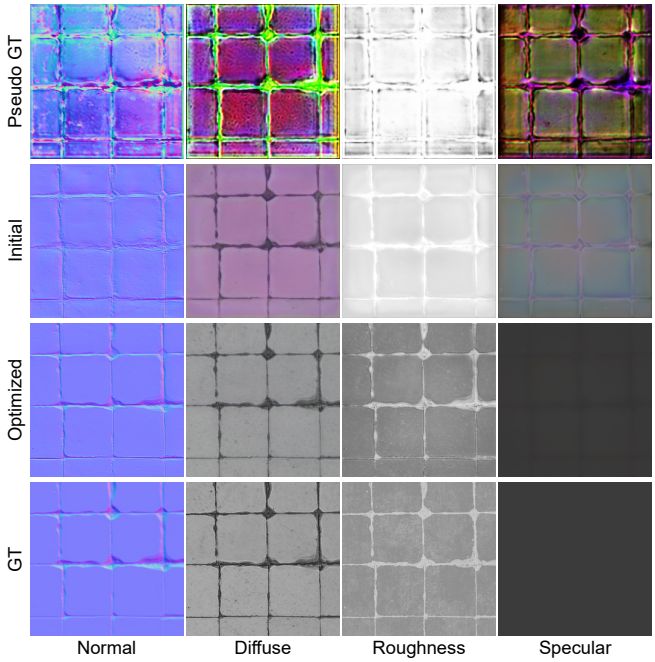

Fig. 5. On the top two rows, we show the output of the auxiliary network ("Pseudo GT") as well as the result of our initial network ($f^*_{\theta^*\text{init}}$). Note that while the pseudo ground truth maps are substantially different from the ground truth, they properly approximate the required gradients for pushing the optimization towards the ground truth.

The main difference between these equations and the ones in Eqs. 3 and 4 are two-folds: **1)** the minimization in Eq. 5 is done over the weights of both the main and auxiliary networks, and **2)** the test loss in Eq. 6 incorporates the pseudo ground truth reflectance parameters $f_\psi(I_n)$. Note that only the main network parameters are

Table 1. Numerical comparison on a set of 52 synthetic test images. We evaluate the quality of renderings both in terms of RMSE and LPIPS [Zhang et al. 2018], a perceptual metric, but the four reflectance parameters (normal "N", diffuse "D', roughness "R", and specular "S") are evaluated using RMSE. "Ren" refers to renderings for which the numerical values are obtained on a set of 20 images of each scene under different lights.

	RMSE					LPIPS
	N	D	R	S	Ren	Ren
Des18	0.060	0.0963	0.262	0.134	0.108	0.262
Des19	0.070	0.134	0.172	**0.076**	0.130	0.226
Zhou21	0.065	0.108	0.153	0.079	0.080	0.179
Guo21	0.073	0.105	0.144	0.092	0.100	0.240
Gao19	0.068	0.101	0.137	0.092	0.074	0.142
Guo20	0.070	0.095	0.159	0.094	0.081	0.165
Ours	**0.058**	**0.078**	**0.124**	0.089	**0.061**	**0.139**

updated in Eq. 6, as the auxiliary network is only used to calculate a better loss and does not need to be optimized during the testing.

Discussion. One might think that to truly resolve the mismatch between the training and testing losses, the pseudo ground truth parameters (auxiliary network's output) should match the ground truth reflectance maps. This conclusion can also be drawn based on our pseudo reflectance loss in Eq. 8; since we minimize the *L2* distance between the output of the main and auxiliary networks, it seems like the optimal results can only be achieved if the auxiliary network produces reflectance maps that are similar to the ground truth. Although this would be the case if we optimize our test loss (see Eqs. 7 and 8) until convergence, in our formulation the test loss is only optimized with a *single* gradient update (see Eq. 6). This update moves the main network towards a solution that is optimal in terms of the training objective, but not the test loss. Therefore, while the output of the main network after one gradient update is similar to the ground truth reflectance maps, it could be substantially different from the auxiliary network output, as shown in Fig. 5. Note that because we do not explicitly enforce the auxiliary network to produce reflectance maps that resemble the ground truth, this network learns to approximate the difference in gradient between the rendering and the ideal losses. The approximated gradients, which lie in a high dimensional space, do not necessarily map to interpretable reflectance maps.

We note that our learned reflectance loss cannot be directly used to improve the test-time optimization of the existing optimization-based methods. This is because this loss provides gradients that are suitable for one step optimization with our main network (θ^*_{init}). If the auxiliary network, trained within our system, is directly used with other systems, the gradients produced by the such a reflectance loss could potentially even hurt their optimization.

4 IMPLEMENTATION

In this section, we discuss the necessary details to implement our approach, including the network architecture, loss functions, and the training process. The source code, trained networks, and dataset can be found at: https://people.engr.tamu.edu/nimak/Papers/SIGAsia2022_MatCap/.

Table 2. Numerical comparison on a set of 33 real test scenes from Guo et al. [Guo et al. 2020] and 76 scenes from our dataset. In both datasets, each scene contains 9 images. For each scene, we use one image as the input and the remaining 8 images are used as ground truth.

	MaterialGAN		Ours	
	LPIPS	RMSE	LPIPS	RMSE
Des18	0.391	0.140	0.316	0.102
Des19	0.383	0.188	0.310	0.129
Zhou21	0.314	0.154	0.266	0.132
Guo21	0.391	0.161	0.303	0.103
Gao19	0.361	0.158	0.290	0.110
Guo20	0.316	0.153	0.256	0.113
Ours	**0.286**	**0.133**	**0.216**	**0.093**

4.1 Network architecture

Our system includes an auxiliary and main network. For the auxiliary network, we directly use the architecture proposed by Deschaintre et al. [2019]. For our main network, we use a small conditional coordinate-based neural network. This is because our training is memory intensive as it requires backpropagating the training loss through all the test gradient updates. This means that we need to compute the second order gradients through multiple networks which in turn requires saving all the updated main networks (for each gradient update and each training example in the mini-batch) during training.

Our main network has two primary components. The first component extracts a set of features from the input image. The second component takes the extracted features along with the coordinates (in x and y) at each pixel and generates the reflectance parameters at that pixel. For the first component, we use a simple UNet [Ronneberger et al. 2015] with two downsampling and upsampling layers. This network takes a 3 channel RGB image as the input and outputs a 128 channel feature map (3 -> 32 -> 64 -> 128 -> 128). All the layers are convolutional with a kernel size of 4. As for the second component, we use the SIREN network [Sitzmann et al. 2020b], but modify the number of output channels to 8. Specifically, we use 1 channel for height map, 3 channels for diffuse, 1 channel for roughness, and 3 channels for the specular maps. Note that our network estimates a single channel height map, but we convert it to a 3 channel normal for renderings using central finite differences with circular boundaries. We also experimented with directly outputting the 3 channel normal map, but since the results were similar we estimate the height map for memory efficiency.

4.2 Loss function

Test loss. As discussed, we use a combination of the rendering and pseudo reflectance losses for the test. Specifically, our test loss is defined as:

$$E_{\text{test}}\left(f_\theta(\mathbf{I}), f_\psi(\mathbf{I}), \mathbf{I}\right) = \mathcal{L}_{\text{ren}} + \lambda \mathcal{L}_{\text{pseudo}}, \tag{7}$$

where λ defines the weight of the pseudo reflectance loss and is equal to 1 in our implementation. Moreover, the rendering and pseudo reflectance losses are defined as follows:

$$\mathcal{L}_{\text{ren}} = \|R(f_\theta(\mathbf{I})) - \mathbf{I}\|_2, \quad \mathcal{L}_{\text{pseudo}} = \|f_\theta(\mathbf{I}) - f_\psi(\mathbf{I})\|_2, \tag{8}$$

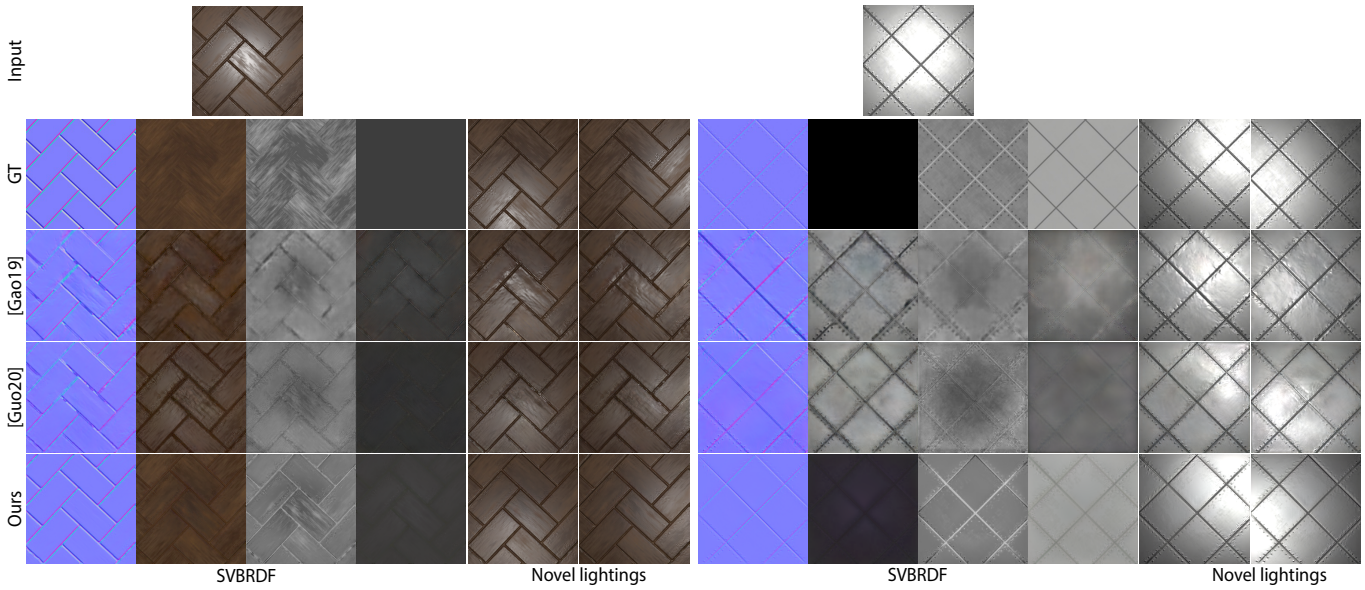

Fig. 6. We show comparison of our approach against two optimization-based approaches by Gao et al. [2019] and Guo et al. [2020] on two synthetic scenes. For each scene, we compare the reflectance maps along with three rerendered images under novel lightings.

Training loss. As is common in existing techniques [Deschaintre et al. 2019; Guo et al. 2021], we use a combination of the rendering and reflectance losses for training. Our loss is defined as follows:

$$E_{\text{train}}\left(f_\theta(\mathbf{I}), \mathbf{I}, \mathbf{F}\right) = \beta \mathcal{L}_{\text{p-ren}} + \gamma \mathcal{L}_{\text{ref}}, \tag{9}$$

where β and γ define the weights of the rendering and reflectance losses and are set to 10 and 0.25, respectively. Moreover, $\mathcal{L}_{\text{p-ren}}$ is a perceptual rendering loss which is defined as:

$$\mathcal{L}_{\text{p-ren}} = \|R(f_\theta(\mathbf{I})) - \mathbf{I}\|_2 + \eta \mathcal{L}_{\text{style}}(R(f_\theta(\mathbf{I})), \mathbf{I}), \tag{10}$$

where $\mathcal{L}_{\text{style}}$ is the style loss based on the Gram matrix [Gatys et al. 2015] of the VGG features [2015]. Furthermore, η defines the weight of the style loss and is set to 1. Note that we compute the $L2$ loss between downsampled (to 16×16) rerendered and ground truth images. Finally, the reflectance loss $\mathcal{L}_{\text{ref}}$ is defined as:

$$\begin{aligned}\mathcal{L}_{\text{ref}} =& \lambda_n \|\hat{\mathbf{n}} - \mathbf{n}\|_2 + \lambda_d \|\hat{\mathbf{d}} - \mathbf{d}\|_2 + \lambda_r \|\hat{\mathbf{r}} - \mathbf{r}\|_2 \\ &+ \lambda_s \|\hat{\mathbf{s}} - \mathbf{s}\|_2 + \lambda_{dv} \mathcal{L}_{\text{vgg}}(\hat{\mathbf{d}}, \mathbf{d}) + \lambda_{sv} \mathcal{L}_{\text{vgg}}(\hat{\mathbf{s}}, \mathbf{s}),\end{aligned} \tag{11}$$

where $\hat{\mathbf{n}}$, $\hat{\mathbf{d}}$, $\hat{\mathbf{r}}$, and $\hat{\mathbf{s}}$ are the estimated normal, diffuse, roughness, and specular, while $\mathbf{n}$, $\mathbf{d}$, $\mathbf{r}$, and $\mathbf{s}$ are their ground truth counterpart. Furthermore, $\mathcal{L}_{\text{vgg}}$ is the VGG-based perceptual loss, proposed by Chen et al. [Chen and Koltun 2017]. Finally, λ_n, λ_d, λ_r, λ_s, λ_{dv}, and λ_{sv} define the weight of each term and we set them to 80, 1, 1, 5, 2.5×10^{-2}, and 2.5×10^{-2}, respectively.

4.3 Training

We use the synthetic SVBRDFs dataset provided by Deschaintre et al. [2018] to train our system. We use the Xavier approach [Glorot and Bengio 2010] to initialize the feature extractor in our main network, but use the proposed strategy by Sitzmann et al. [Sitzmann et al. 2020b] to initialize SIREN. For the auxiliary network, we start by the pre-trained network provided by Deschaintre et al. [2019]. We set the learning rate for our test-time optimization (α) to 1×10^{-3}. For training, we use the Adam optimizer [Kingma and Ba 2014] with the internal parameters β_1 and β_2 set to 0.5 and a batch size of 2. For the main network, we use an initial learning rate of 1×10^{-6} and reduce it by a factor of 2 after every 500,000 iterations. On the other hand, we use a fixed learning rate of 1×10^{-5} for the auxiliary network. Note that the test-time gradient update is performed using stochastic gradient descent (SGD) following Eq. 6, as calculating the gradients through an Adam optimizer is significantly more challenging. We train our system, implemented in PyTorch, for 1.3M iterations which takes roughly 5 days on a GeForce 2080 Ti GPU.

4.4 Testing

Our network is trained to produce optimal results after one iteration of test-time optimization using SGD with a learning rate of 1×10^{-3}. However, by evaluating the results on a validation dataset containing 80 examples, we observed that the training loss (reflectance + rendering) still decreases for 6 more iterations beyond the initial update . Therefore, we generate all the results with 7 (1 + 6) iterations of test-time optimization. Note that, since our pseudo reflectance loss is trained with one gradient update, using it after the first iteration would hurt the system. Therefore, we perform the remaining iterations using the rendering loss in Eq. 10. We perform the first update using SGD with a learning rate of 1×10^{-3}, but the remaining 6 iterations are done using Adam with a learning rate of 2×10^{-7}.

5 RESULTS

Throughout this section, we compare our approach against several direct-estimation and optimization-based approaches. Specifically, we compare against the direct-estimation approaches by Deschaintre et al. [2018; 2019], Zhou et al. [2021], and Guo et al. [2021], as well as the optimization-based methods of Gao et al. [2019] and

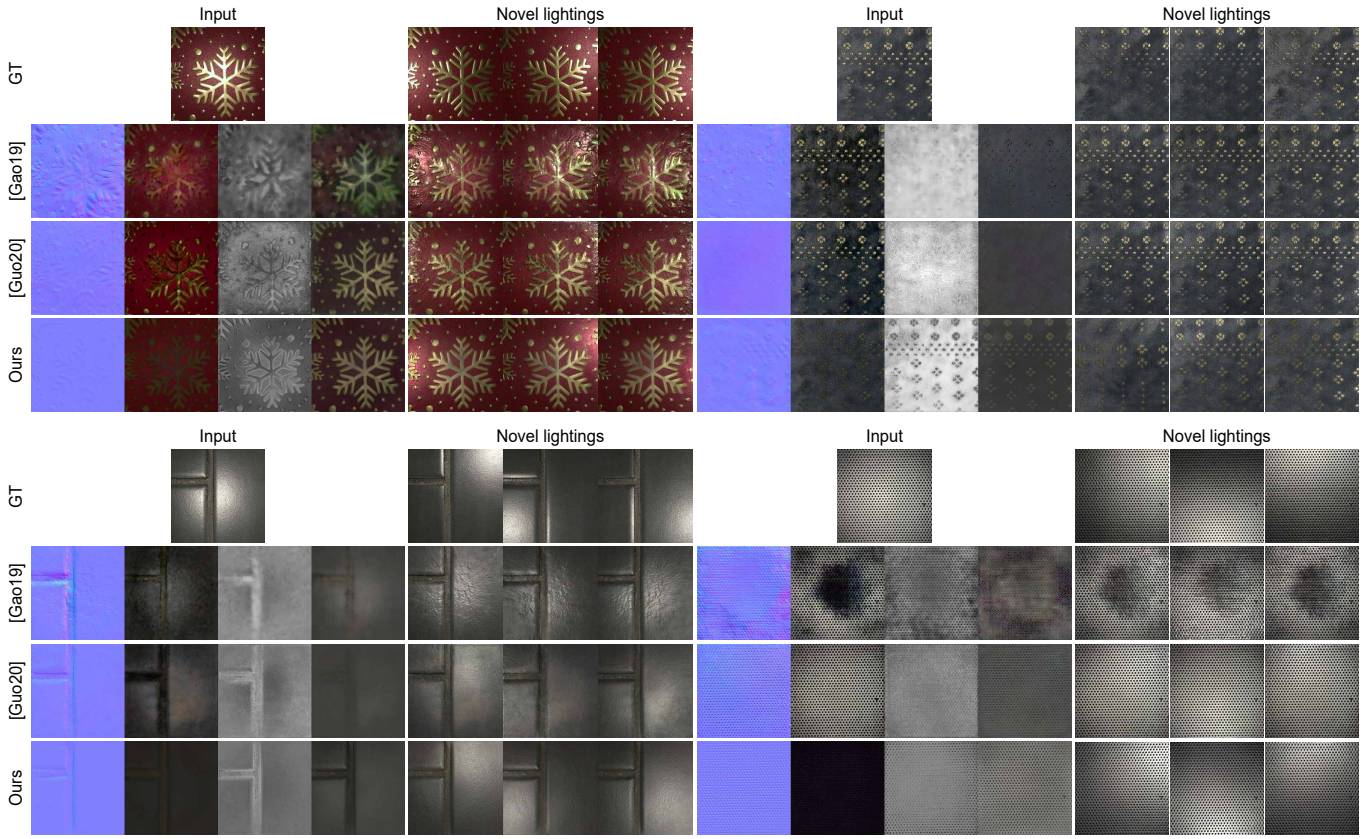

Fig. 7. We compare our approach against the other methods on four real scenes with ground truth renderings. Note that the ground truth reflectance parameters are not available in this case.

Guo et al. [2020]. Since the source code of the approach by Guo et al. [2021] is not publicly available, we asked the authors to run their code on our images. For all the other methods, we use the source code provided by the authors. Moreover, we use a single image as the input to all the approaches, even the ones with the ability to handle multiple images [Deschaintre et al. 2019; Gao et al. 2019; Guo et al. 2020], to have a fair comparison. We include the numerical comparisons against all the approaches, but only provide visual comparisons against the optimization-based methods of Gao et al. [2019] and Guo et al. [2020]. *The complete visual comparisons against all the approaches on a large set of scenes are provided in the supplementary materials.*

Synthetic Images. We begin by numerically comparing our approach against the other methods on a set of 52 synthetic scenes, gathered from Deschaintre et al. [2019] and Guo et al. [Guo et al. 2020], in Table 1. We evaluate the quality of the reflectance parameters in terms of root mean squared error (RMSE), while the rerendered images are evaluated using both RMSE and learned perceptual image patch similarity (LPIPS) [Zhang et al. 2018]. Overall, our method produces better results than the other approaches, except for the specular maps which are slightly worse than Deschaintre et al.[2019] and Zhou et al. [2021].

Next, we compare our approach against the other optimization-based methods [Gao et al. 2019; Guo et al. 2020] on two synthetic scenes in Fig. 6. The other techniques overfit to the input image through the test-time optimization as their system is not sufficiently constrained. Therefore, they bake in the input highlight in their estimated parameters (particularly diffuse and roughness), which severely degrades the quality of their rerendered images under novel lighting. With our proposed look-ahead training, the overfitting is significantly mitigated and we are able to produce results that better match the ground truth. Note that while other approaches produce sharp normal maps, their normals contain unnecessary details that do not exist in the ground truth.

Real Images with Ground Truth Renderings. To numerically compare our approach against the other methods on real images, we capture 76 scenes, each with 9 images with calibrated lightings. Similar to the previous methods [Deschaintre et al. 2019; Guo et al. 2020; Hui et al. 2017], we capture colocated camera and flash images using a cellphone and use a checkerboard frame to calibrate the images. For each scene, we capture 9 images where one is captured in a fronto-parallel manner and is used as the input, while the rest are used as ground truth. In addition to our dataset, we also compare our method against the other approaches on MaterialGAN dataset [Guo et al. 2020] with 33 scenes. For approaches by Gao et al. [2019] and Guo et al. [2020], we use the exact calibrated light position. However, we use a fixed lighting by assuming an FOV of 45° for our optimization. Note that, for some scenes, there are no

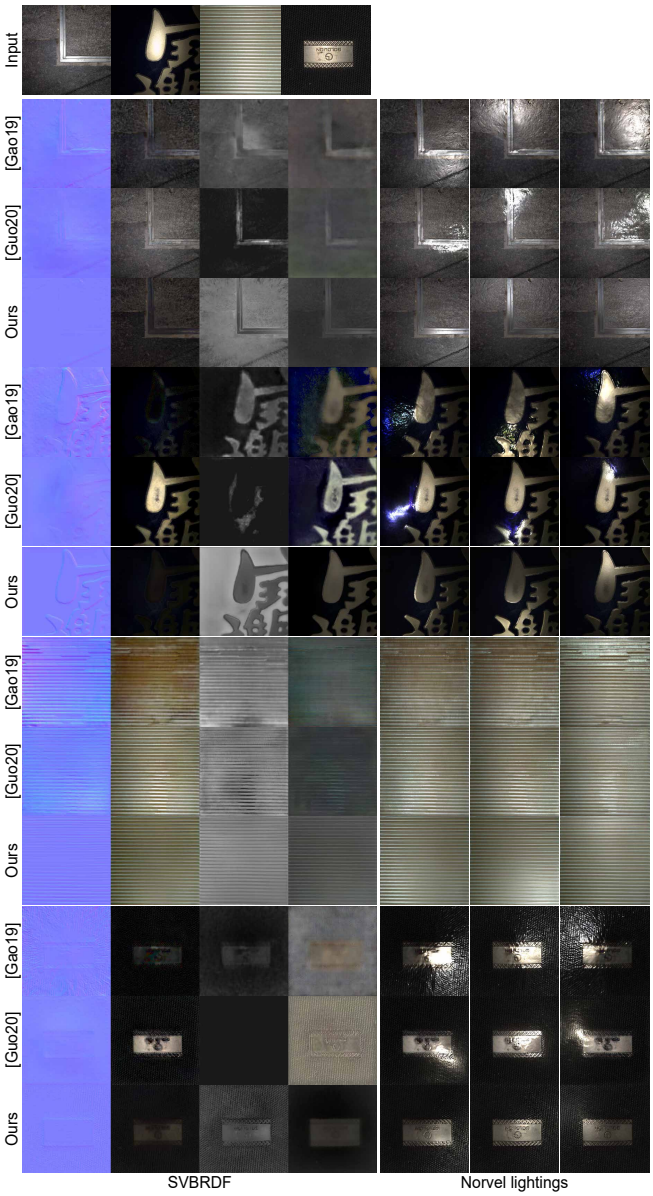

Fig. 8. We show comparison of our approach agains the other methods on four casually captured scenes. Note that ground truth renderings for these images are not available.

images that are completely fronto-parallel. In these cases, we use the simple approach proposed by Aittala et al. [2016] to transform the image. We adjust the calibrated light position for the approaches by Gao et al. and Guo et al. by taking into account this transformation. In Table 2, we show the results of this comparison in terms of RMSE and LPIPS. As seen, our method significantly outperforms other approaches both in terms of RMSE and LPIPS.

We further show visual comparison against the methods by Gao et al. [2019] and Guo et al. [2020] on four scenes from the two datasets in Fig. 7. The two scenes on the left are from the MaterialGAN dataset [Guo et al. 2020], while the ones on the right are from our dataset. Overall, our approach produces cleaner reflectance

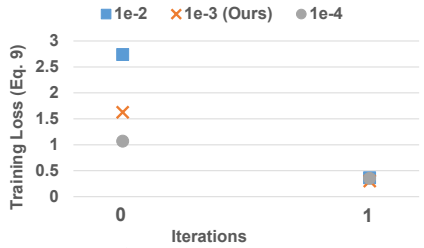

Fig. 9. We evaluate the effect of the test-time optimization learning rate (α) quantitatively. The initial results with larger α are worse, but the results after one step for all the learning rates are similar.

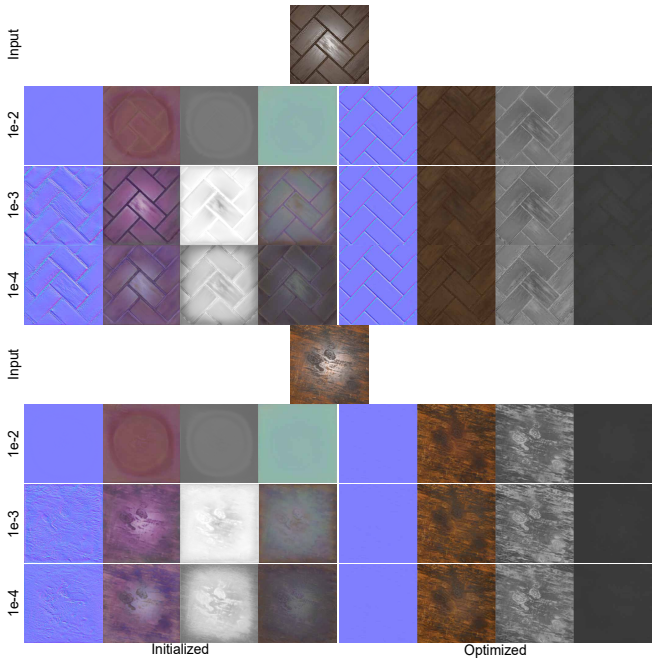

Fig. 10. We show the results of our network before (left) and after (right) test-time optimization for different test-time learning rates (α). With larger learning rates, the initial results become worse, but the improvement after the first step becomes larger. All the experiments converge to similar results after one iteration, demonstrating the robustness of our approach to the choice of learning rate.

maps and rerendered images and is more robust to overfitting. For example, only our approach does not bake in the highlights on the golden patterns of the top right scene in the diffuse map. Similarly, in the bottom right example, the other approaches either produce results with sever artifacts [Gao et al. 2019] or incorrectly bake in the highlights in the diffuse map.

Images Captured In the Wild. To further show the robustness of our approach, we provide comparisons against other approaches on casually captured images in Fig. 8. The top two scenes are from Guo et al. [2021], while we captured the rest. Note that for these scenes ground truth images under novel lightings are not available. Moreover, since the images are casually captured and not calibrated, we apply the same light settings as ours for the approaches by Gao et al. [2019] and Guo et al. [Guo et al. 2020]. Other algorithms produce results with visible artifacts, while our method generates cleaner

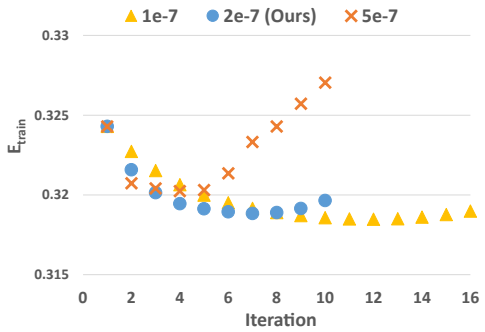

Fig. 11. We evaluate the training loss (Eq. 9) for different number of additional iterations after the first gradient update. We show the results for different learning rates. Note that we have excluded the error of the initial network (1.23 at iteration 0), as it would make it difficult to see the differences between the three plots. The error at the first iteration for all the cases is the same, since we use a learning rate of 1×10^{-3} for the first update (configuration used for training).

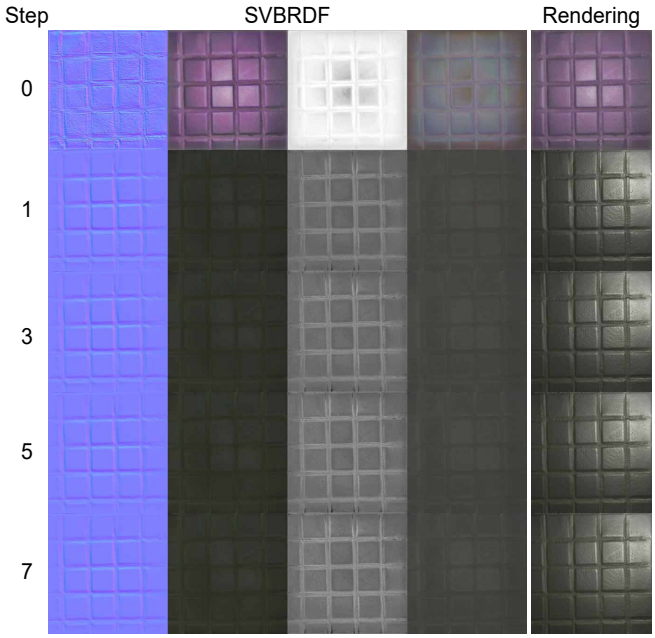

Fig. 12. We show the results after different number of gradient updates during testing. The majority of the improvement comes from the initial gradient update. While the additional gradient updates further improve the quality, their impact is relatively small.

reflectance maps and consequently rerendered images. For example, the rough stone in the top example is flat, but other approaches produce unnecessary details in the normal map which negatively affects the quality of their renderings. Moreover, note that only our approach is able to properly estimate the difference between the stone and metal's roughness.

5.1 Ablation Studies

Effect of α. We evaluate the effect of test-time optimization learning rate in Figs. 9 and 10. For each learning rate, we show the error plots (Fig. 9) and visual results (Fig. 10) of before and after 1 iteration

Table 3. Numerical evaluation of the effect of the different loss terms in our testing objective.

	Synthetic		MaterialGAN		Ours	
	LPIPS	RMSE	LPIPS	RMSE	LPIPS	RMSE
Only Rendering	0.150	0.064	0.296	0.137	0.223	0.096
Only Pseudo Reflectance	**0.136**	0.063	0.290	0.133	0.218	0.095
Fixed Pseudo Reflectance	0.146	0.064	0.304	**0.130**	0.229	0.094
Add $\mathcal{L}_{\text{style}}$	0.139	0.062	0.296	0.135	0.22	0.095
Standard Training	0.184	0.081	0.358	0.159	0.278	0.113
Ours	0.139	**0.061**	**0.286**	0.133	**0.216**	**0.093**

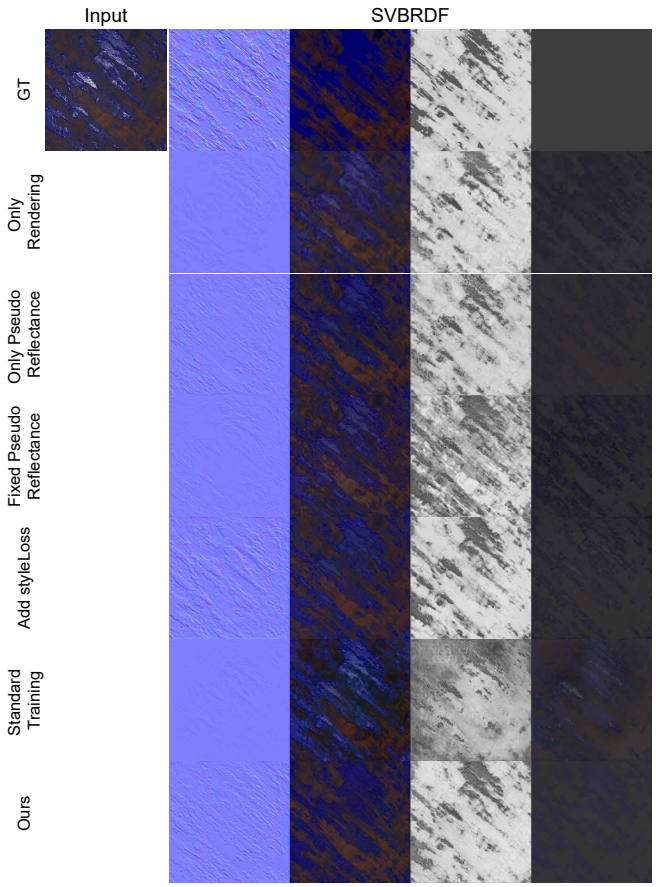

Fig. 13. We show the visual results for the experiments corresponding to Table. 3 on one example.

of the test-time optimization. Note that we train the network using our strategy for each test-time learning rates. With larger learning rates, the network produces worse initial results, but the improvement after one iteration of test-time optimization becomes larger. This is expected as with a larger learning rate, the test-time gradient update becomes bigger, and thus the initial results should be further away from the optimal ones. The three scenarios converge to similar results after optimization, although with 1×10^{-3} we achieve slightly better results numerically. Note that, as discussed, with an extremely small test-time learning rate, our look-ahead training becomes similar to standard training.

Table 4. Numerical evaluation of the effect of the different loss terms in our training objective.

	Synthetic		MaterialGAN		Ours	
	LPIPS	RMSE	LPIPS	RMSE	LPIPS	RMSE
no $\mathcal{L}_{\text{vgg}}$	0.137	0.062	0.289	0.136	0.217	0.096
no $\mathcal{L}_{\text{style}}$	0.216	0.061	0.329	**0.131**	0.258	0.093
256x256	**0.134**	**0.060**	0.294	0.136	0.217	0.094
64x64	0.139	0.062	0.288	0.133	0.22	0.096
Ours	0.139	0.061	**0.286**	0.133	**0.216**	**0.093**

Additional Gradient Updates. As is discussed in Sec. 4.4, after the first update with a learning rate of 1×10^{-3} (the configuration used during training), we further optimize our network for 6 additional iterations with a learning rate of 2×10^{-7}. To justify our choice for number of additional iterations, we evaluate the training loss (reflectance + rendering) after additional iterations on a validation dataset containing 80 examples in Fig. 11 (blue dots). As seen, the loss still decreases for 6 more iteration beyond the first update. Note that, as shown in Fig. 12, the majority of the improvement comes from the first update and the remaining iterations have a small impact on the quality of the results. We further justify our choice of learning rate for the additional iterations by showing the training error for additional iterations with different learning rates in Fig. 11. We achieve the minimum error of 0.3202, 0.3188, and 0.3185 with the learning rates of 5×10^{-7}, 2×10^{-7}, and 1×10^{-7}, respectively. While the learning rate of 1×10^{-7} produces results with a slightly lower error, this is obtained with 11 additional iterations (compared to 6 in our case). Therefore, we choose 2×10^{-7} as the learning rate for the additional gradient updates.

Effect of Testing Loss Terms. We evaluate the effect of various losses in our testing objective in Table 3 (numerically) and Fig. 13 (visually). Our system using only the rendering loss, produces results that are worse than our method with the full loss. As shown in Fig. 13, our results with only the rendering loss, have weaker normals and slightly bake the highlights into the diffuse map. As discussed in Sec. 3.2 this is because the gradient mismatch between the rendering and ideal test losses (see Fig. 4). Interestingly, our system using only the learned pseudo reflectance loss produces comparable results to our method using the full loss (both rendering and pseudo reflectance losses). This is because in this case, auxiliary network learns to estimate pseudo ground truth reflectance maps that directly approximate the gradients of the ideal test loss (red arrows in Fig. 4).

We also evaluate the importance of *learning* our pseudo reflectance loss, by examining a variation of our approach where the pseudo reflectance loss is fixed. To this end, we use the pre-trained network by Deschaintre et al. [2019] as our auxiliary network and keep its weights fixed throughout our training. Compared to only the rendering loss, we observe that there is not a significant benefit in adding the pseudo reflectance loss, if it is not learned. This is expected as gradients provided by the fixed pseudo reflectance loss may not properly approximate the difference in gradients between the rendering and ideal losses. Finally, we experiment with adding the style loss between the rerendered and input images to our testing loss similar to the one in the training loss (see $\mathcal{L}_{\text{style}}$ in Eq. 10). As seen in Table 3 and Fig. 13, adding the style loss to the testing objective does not significantly impact the quality of the results. Since adding this loss substantially increases the computational complexity of the training, we do not include it in our final testing objective.

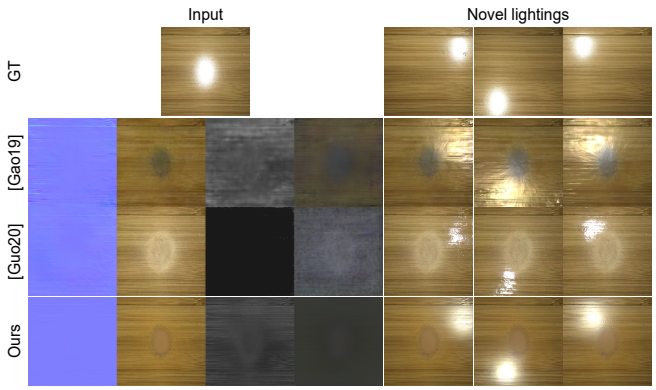

Fig. 14. We compare our method against the other techniques on an example with a strong highlight. Our method is not able to properly hallucinate the missing details, producing results with a slight burn-in effects. However, our results are still better than the ones produced by the other methods.

Effect of Training Loss Terms. In Table 4, we show the effect of different terms in our training objective (Eq. 9). Specifically, we experiment with removing the VGG losses from Eq. 11, and style loss between the rerendered and ground truth images from Eq. 10. We also varied the resolution of the images, used to compute the $L2$ loss in Eq. 10. Overall, the style loss has the biggest impact on the quality of the results, especially perceptually as measured by LPIPS. While the other losses do not substantially affect the quality of the results, our combination of losses produces slightly better results.

Effect of Look-Ahead Training. To demonstrate the importance of our look-ahead training strategy, we compare our results against the result of our network trained in a standard manner. As seen in Table 3 and Fig. 13, the network trained in a standard way produces significantly worse result than ours both numerically and visually.

6 LIMITATIONS AND FUTURE WORK

Single image SVBRDF estimation is a notoriously challenging problem. Although our approach is able to produce high-quality results, it has some limitations. For example, in cases with extremely strong highlights in textured regions, our method may not be able to properly hallucinate the missing information in the input image. Fig. 14 shows an example of such a case where the estimated reflectance maps exhibit slight burn-in effect. Nevertheless, our results are significantly better than the other approaches.

While we demonstrated high-quality results by training our system with just one gradient update of the test loss, we could potentially benefit from multiple gradient updates during training. However, as discussed in Sec. 4.1, our training is memory intensive. With more gradient updates, the training becomes even more memory intensive and additionally unstable. In the future, we plan to investigate ways to improve the stability of the training and explore the possibility of increasing the number of test-time iterations.

Look-ahead training strategy is general and can have applications beyond reflectance estimation. In general, our system can be used in any application with differentiable test-time optimization to adapt the network to the individual test examples. In the future, we plan to investigate the possibility of using our training for such applications.

7 CONCLUSION

We have presented a novel optimization-based approach for estimating the SVBRDF parameters from a single image. To avoid overfitting to the input image during optimization, we incorporate the test-time optimization into the training process. Through the training process, we minimize the training loss of the network after the test-time optimization. To further improve the robustness to overfitting, we augment the typically used rendering loss with a pseudo ground truth reflectance loss. We do this by using an auxiliary network to estimate pseudo ground truth reflectance parameters from the input image. We train both the main and auxiliary networks using our proposed training strategy. We demonstrate that our approach produces better results than the state-of-the-art while being significantly faster than the optimization based methods.

REFERENCES

Miika Aittala, Timo Aila, and Jaakko Lehtinen. 2016. Reflectance modeling by neural texture synthesis. *ACM Transactions on Graphics (ToG)* 35, 4 (2016), 1–13.

Miika Aittala, Tim Weyrich, Jaakko Lehtinen, et al. 2015. Two-shot SVBRDF capture for stationary materials. *ACM Trans. Graph.* 34, 4 (2015), 110–1.

Antreas Antoniou, Harrison Edwards, and Amos Storkey. 2019. How to train your MAML. In *International Conference on Learning Representations*. https://openreview.net/forum?id=HJGven05Y7

Alexander William Bergman, Petr Kellnhofer, and Gordon Wetzstein. 2021. Fast Training of Neural Lumigraph Representations using Meta Learning. In *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (Eds.). https://openreview.net/forum?id=XCaZKu00a_D

Qifeng Chen and Vladlen Koltun. 2017. Photographic image synthesis with cascaded refinement networks. In *Proceedings of the IEEE international conference on computer vision*. 1511–1520.

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

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

Valentin Deschaintre, Miika Aittala, Frédo Durand, George Drettakis, and Adrien Bousseau. 2019. Flexible SVBRDF Capture with a Multi-Image Deep Network. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library, 1–13.

Valentin Deschaintre, George Drettakis, and Adrien Bousseau. 2020. Guided Fine-Tuning for Large-Scale Material Transfer. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 91–105.

Yue Dong, Jiaping Wang, Xin Tong, John Snyder, Yanxiang Lan, Moshe Ben-Ezra, and Baining Guo. 2010. Manifold bootstrapping for SVBRDF capture. *ACM Transactions on Graphics (TOG)* 29, 4 (2010), 1–10.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*. PMLR, 1126–1135.

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

Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. 2015. A Neural Algorithm of Artistic Style. arXiv:1508.06576

Abhijeet Ghosh, Shruthi Achutha, Wolfgang Heidrich, and Matthew O'Toole. 2007. BRDF acquisition with basis illumination. In *2007 IEEE 11th International Conference on Computer Vision*. IEEE, 1–8.

Abhijeet Ghosh, Tongbo Chen, Pieter Peers, Cyrus A Wilson, and Paul Debevec. 2010. Circularly polarized spherical illumination reflectometry. In *ACM SIGGRAPH Asia 2010 papers*. 1–12.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*. 249–256.

Jie Guo, Shuichang Lai, Chengzhi Tao, Yuelong Cai, Lei Wang, Yanwen Guo, and Ling-Qi Yan. 2021. Highlight-Aware Two-Stream Network for Single-Image SVBRDF Acquisition. *ACM Trans. Graph.* 40, 4, Article 123 (jul 2021), 14 pages. https://doi.org/10.1145/3450626.3459854

Yu Guo, Cameron Smith, Miloš Hašan, Kalyan Sunkavalli, and Shuang Zhao. 2020. MaterialGAN: Reflectance Capture using a Generative SVBRDF Model. *ACM Trans. Graph.* 39, 6 (2020), 254:1–254:13.

Philipp Henzler, Valentin Deschaintre, Niloy J. Mitra, and Tobias Ritschel. 2021a. Generative Modelling of BRDF Textures from Flash Images. *ACM Trans. Graph.* 40, 6, Article 284 (dec 2021), 13 pages. https://doi.org/10.1145/3478513.3480507

Philipp Henzler, Valentin Deschaintre, Niloy J Mitra, and Tobias Ritschel. 2021b. Generative Modelling of BRDF Textures from Flash Images. *ACM Trans Graph (Proc. SIGGRAPH Asia)* 40, 6 (2021).

Zhuo Hui, Kalyan Sunkavalli, Joon-Young Lee, Sunil Hadap, Jian Wang, and Aswin C Sankaranarayanan. 2017. Reflectance capture using univariate sampling of BRDFs. In *Proceedings of the IEEE International Conference on Computer Vision*. 5362–5370.

Kaizhang Kang, Zimin Chen, Jiaping Wang, Kun Zhou, and Hongzhi Wu. 2018. Efficient reflectance capture using an autoencoder. *ACM Trans. Graph.* 37, 4 (2018), 127–1.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. 2020. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8110–8119.

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

Xiao Li, Yue Dong, Pieter Peers, and Xin Tong. 2017. Modeling surface appearance from a single photograph using self-augmented convolutional neural networks. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–11.

Zhengqin Li, Kalyan Sunkavalli, and Manmohan Chandraker. 2018. Materials for masses: SVBRDF acquisition with a single mobile phone image. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 72–87.

Alex Nichol, Joshua Achiam, and John Schulman. 2018. On First-Order Meta-Learning Algorithms. https://doi.org/10.48550/ARXIV.1803.02999

Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. 2019. Meta-Learning with Implicit Gradients. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/072b030ba126b2f4b2374f342be9ed44-Paper.pdf

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi (Eds.). Springer International Publishing, Cham, 234–241.

Liang Shi, Beichen Li, Miloš Hašan, Kalyan Sunkavalli, Tamy Boubekeur, Radomir Mech, and Wojciech Matusik. 2020. MATch: differentiable material graphs for procedural material capture. (2020).

Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*.

Vincent Sitzmann, Eric R. Chan, Richard Tucker, Noah Snavely, and Gordon Wetzstein. 2020a. MetaSDF: Meta-Learning Signed Distance Functions. In *Proc. NeurIPS*.

Vincent Sitzmann, Julien NP Martel, Alexander W Bergman, David B Lindell, and Gordon Wetzstein. 2020b. Implicit neural representations with periodic activation functions. *arXiv preprint arXiv:2006.09661* (2020).

Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P. Srinivasan, Jonathan T. Barron, and Ren Ng. 2021. Learned Initializations for Optimizing Coordinate-Based Neural Representations. In *CVPR*.

Giuseppe Vecchio, Simone Palazzo, and Concetto Spampinato. 2021. SurfaceNet: Adversarial SVBRDF Estimation From a Single Image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 12840–12848.

Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. 2007. Microfacet Models for Refraction through Rough Surfaces. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques* (Grenoble, France) *(EGSR'07)*. Eurographics Association, Goslar, DEU, 195–206.

Wenjie Ye, Xiao Li, Yue Dong, Pieter Peers, and Xin Tong. 2018. Single image surface appearance modeling with self-augmented cnns and inexact supervision. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 201–211.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 586–595.

Yezi Zhao, Beibei Wang, Yanning Xu, Zheng Zeng, Lu Wang, and Nicolas Holzschuch. 2020. Joint SVBRDF Recovery and Synthesis From a Single Image using an Unsupervised Generative Adversarial Network. (2020).

Xilong Zhou and Nima Khademi Kalantari. 2021. Adversarial Single-Image SVBRDF Estimation with Hybrid Training. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 315–325.