A Flexible Neural Renderer for Material Visualization

Aakash KT
IIIT Hyderabad

Parikshit Sakurikar
IIIT-H, DreamVu Inc.

Saurabh Saini
IIIT Hyderabad

P. J. Narayanan
IIIT Hyderabad

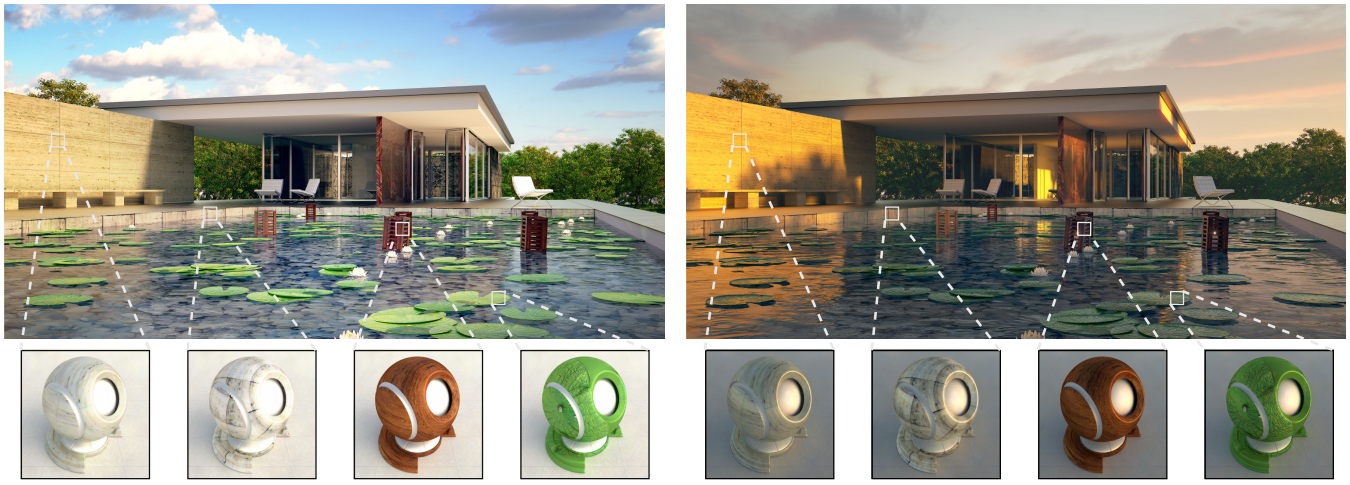

Figure 1: Shaderball visualizations of four selected materials produced by our network are shown for two lighting conditions: *Left* - Daylight and *Right* - Sunset. Scene adopted from ©eMirage (https://www.emirage.org/).

ABSTRACT

Photo realism in computer generated imagery is crucially dependent on how well an artist is able to recreate real-world materials in the scene. The workflow for material modeling and editing typically involves manual tweaking of material parameters and uses a standard path tracing engine for visual feedback. A lot of time may be spent in iterative selection and rendering of materials at an appropriate quality. In this work, we propose a convolutional neural network that quickly generates high-quality ray traced material visualizations on a shaderball. Our novel architecture allows for control over environment lighting which assists in material selection and also provides the ability to render spatially-varying materials. Comparison with state-of-the-art denoising and neural rendering techniques suggests that our neural renderer performs faster and better. We provide an interactive visualization tool and an extensive dataset to foster further research in this area.

CCS CONCEPTS

• **Computing methodologies** → **Rendering**; **Ray tracing**.

KEYWORDS

Ray Tracing, Global Illumination, Deep Learning, Neural Rendering

ACM Reference Format:
Aakash KT, Parikshit Sakurikar, Saurabh Saini, and P. J. Narayanan. 2019. A Flexible Neural Renderer for Material Visualization. In *SIGGRAPH Asia 2019 Technical Briefs (SA '19 Technical Briefs), November 17–20, 2019, Brisbane, QLD, Australia.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3355088.3365160

1 INTRODUCTION

Ray tracing has emerged as the industry standard for creating photo realistic images and visual effects, which is achieved by accurate modeling of the behavior and traversal of light along with physically accurate material models. Achieving photo realism is however a tedious process. Ray tracing is a computationally expensive operation while physically accurate material modeling requires expertise in fine-tuning of parameters. Visualization of edits during fine-tuning is therefore a time consuming process. An artist might thereby end up spending a lot of time in a slow and iterative visualization loop.

In this paper, we present a neural network architecture that can quickly output high-quality ray-traced visualizations to assist in material selection. Our work extends the state-of-the-art in neural rendering by providing the ability to deal with a large range of uniform as well as spatially-varying materials with flexible environment lighting. We render on a fixed shaderball geometry which is complex enough to encode fine interactions between light and the underlying material.

We evaluate our system quantitatively and also compare qualitative render quality with existing neural rendering frameworks. We also conduct a user study to illustrate the benefit of providing

SA '19 Technical Briefs, November 17–20, 2019, Brisbane, QLD, Australia

ACM ISBN 978-1-4503-6945-9/19/11…$15.00
https://doi.org/10.1145/3355088.3365160

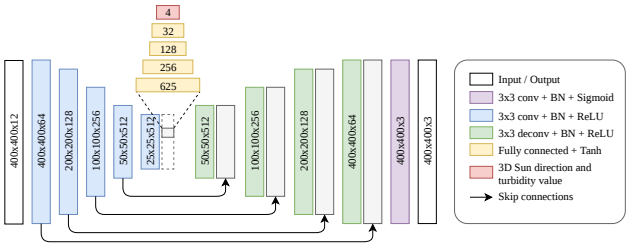

Figure 2: Our neural renderer that uses an autoencoder inspired from U-Net which renders the shaderball under a given environment lighting specified by the sun direction and turbidity value. The material of the shaderball is specified by four screen space material maps.

control over lighting for material selection. We show that our proposed system is fast and therefore helps in real-time visualization of materials. Our method also compares favourably with denoising frameworks in producing faster and better rendered images. In summary, the following are the contributions of our work:

- A neural renderer for rendering uniform and spatially-varying materials, with control over environment lighting.
- An interactive tool[1]for material visualization and editing and a large-scale dataset of uniform and spatially-varying material parameters with corresponding ground truth ray-traced images.

2 RELATED WORK

The use of neural networks for rendering has gained popularity in the recent past. Existing approaches to neural rendering deal with denoising low sample-count Monte-Carlo renders, image-based relighting or neural rendering of materials on a fixed geometry, which we briefly discuss in the context of our contributions.

Data-driven methods for denoising Monte-Carlo (MC) renders have been used to produced ray-traced quality images in real-time [Chaitanya et al. 2017; Lehtinen et al. 2018]. We show that our approach produces better visualizations than denoising a low sample count image of the material. Furthermore, our network has a better overall run-time than a denoiser, which is limited by the speed of the low sample count render.

Neural networks have also been used for relighting an image [Ren et al. 2015; Xu et al. 2018]. While not directly related to our work, we take inspiration from such architectures to provide control over environment lighting in the rendered material output.

Neural rendering for material visualization as proposed by Zsolnai-Fehér et al. [2018] is closely related to our work. We extend this work by enabling the use of spatially-varying materials, which is necessary for photo realism, along with control over environment lighting. Additionally, our proposed network is much smaller and hence runs faster in a real-time setting.

We refer the reader to our arXiv paper [KT et al. 2019] for more details on our proposed method, the user study and additional results and comparisons. The source code and training dataset can be found at the project page[1].

[1]https://aakashkt.github.io/neural-renderer-material-visualization.html

3 A NEURAL RENDERING MODEL

The incoming radiance at each pixel $\mathbf{x}$ of a ray-traced 2D image can be modeled as:

$$I(x) = \int_{\Omega} f_r(p_x, \omega_i, \omega_x) L(p_x, \omega_i)(\omega_i \cdot n) d\omega_i, \tag{1}$$

where p_x is the 3D point corresponding to the 2D pixel $\mathbf{x}$, ω_i is the incoming light direction at $\mathbf{x}$, ω_x is the direction towards pixel $\mathbf{x}$ from point p_x, $L(p_x, \omega_i)$ is the radiance of incoming light at point p_x from direction ω_i, Ω is the set of directions on the upper hemisphere and f_r is the Bi-directional Reflectance Distribution Function (BRDF). The choice of f_r determines the material model in use. The hyperparameters of f_r describe the surface and material properties of the geometry, which we refer to as the *material parameters* (m_f).

We use the Cook-Torrance material model (f_r) that is based on the microfacet theory and accurately models surface properties. We make use of a large SVBRDF dataset for the Cook-Torrance model which is publicly available [Deschaintre et al. 2018].

We parameterize the environment lighting in the scene using the [Hosek and Wilkie 2012] sky model. Such a sky model simulates realistic and plausible environment lighting given only the sun direction ω_s and turbidity (cloudiness) c as input. It encodes large variations in outdoor lighting using only four parameters.

Given the Cook-Torrance material parameters m_f, the incoming sun direction ω_s and turbidity c, the solution of Eq. (1) is estimated by a convolutional neural network ϕ as:

$$I(x) = \phi(x, m_f, \omega_s, c). \tag{2}$$

4 IMPLEMENTATION DETAILS

4.1 Training Dataset

We generate a synthetic dataset of 50,000 material parameter maps and ground truth render pairs, containing equal number of spatially-varying and uniform parameter maps. For uniform maps, we randomly choose one value for all the four parameters (Diffuse, Specular, Roughness, Normal), and replicate it along the width and height (400x400) to get a uniform parameter map. We use SVBRDF textures from dataset of [Deschaintre et al. 2018] for spatially-varying maps. For each material map, we sample 5 random sun directions on the upper hemisphere with random turbidity value, and render the scene at 150 samples-per-pixel (spp).

The input to the network is the screen space map of each material parameter map, along with a 3D vector encoding the sun direction and turbidity. We construct these screen-space maps by UV-mapping the shaderball, and rasterize the scene with each material map as the base texture (Fig. 3). We evaluate the loss of the network's output with respect to the ray-traced 150 spp ground truth for the given environment lighting.

4.2 Network Architecture

Fig. 2 shows our neural rendering architecture, inspired from the U-net-style autoencoder. The encoder takes 400x400 screen space maps of the material parameters: *Diffuse*, *Specular*, *Roughness* and *Normal*, and reduces them to 25x25x512 dimensional feature map. The sun direction, which is specified as a 3D vector in the upper hemisphere, along with the turbidity value, is encoded using a separate fully-connected network, which expands this 4D vector to

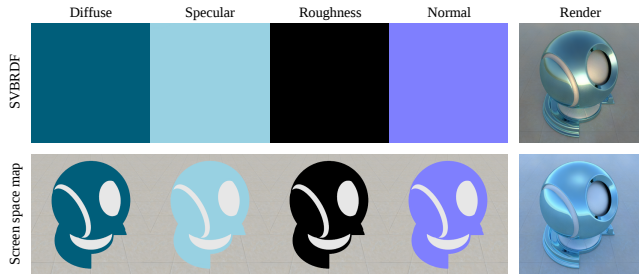

Figure 3: Constructing a screen space material map from the input material. Each map is UV mapped to the shaderball, and the scene is rasterized with that map as base texture.

a 625-dimensional feature. This feature vector is then reshaped to a 25x25 dimensional feature map, replicated 128-times along the channel dimension, and appended to the bottleneck of the auto-encoder. The decoder takes this concatenation of the low resolution feature map and the encoded environment lighting, and outputs the target 400x400 rendered RGB image. We use skip connections between the encoder-decoder pair, for accurate high frequency detail recovery.

4.3 Training Loss

The task of material visualization requires that the perceptual visual quality of the rendered images is impeccable. Cost functions based on Euclidean (L_2) distance are known to be prone to blurring and pixel degradation. We therefore use a loss term which evaluates the perceptual quality of the rendered image, along with L_1 loss for training.

Specifically, we use the feature reconstruction loss from a pre-trained VGG16 network, which is given by :

$$\mathcal{L}^{j}_{feat}(y', y) = \frac{1}{C_j H_j W_j} \left\| \phi_j(y') - \phi_j(y) \right\|_2^2, \tag{3}$$

where ϕ_j is the activation of the *jth* convolutional layer with dimensions $C_j \times H_j \times W_j$ representing number of channels, width and height of the feature map respectively. Here, y denotes the predicted output and y' is the ground truth. We use the *relu_3_3* (*j=relu_3_3*) feature representation in our experiments. Our composite loss function is given by:

$$\mathcal{L}_{train} = L_1 + \mathcal{L}^{relu\_3\_3}_{feat}. \tag{4}$$

We train our network on an NVIDIA GTX 1080Ti, with a batch size of six using the Adam Optimizer ($lr = 10^{-2}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$). We initialize all weights using Glorot-initialization. The network is trained for 30 epochs and takes around 90 hours to train.

5 EVALUATION

We compare our performance with two contemporary paradigms for neural rendering: (1) Neural rendering based on denoising [Chaitanya et al. 2017]; (2) Direct neural rendering [Zsolnai-Fehér et al. 2018]. We also justify the design choices of our network using an ablation study and we conduct a user study for perceptual evaluation. More results and comparisons can be found on our arXiv paper [KT et al. 2019].

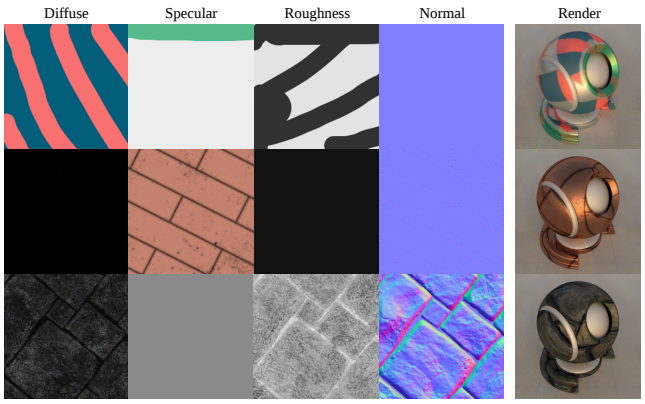

Figure 4: Shaderball visualizations produced by our network corresponding to a variety of SVBRDF maps.

5.1 Comparison with previous work

We show qualitative and run-time comparisons with the recurrent denoiser for Monte Carlo images [Chaitanya et al. 2017]. We implement their network in PyTorch and train on our dataset with 2spp render inputs and 150spp ground truth, consisting of both uniform and spatially-varying materials. The denoiser fails to recover accurate details, which are essential for material visualization (Fig 5). In terms of run-time, rendering a 2spp image and denoising it requires a lot more time than what is required by our network, even with the additional overhead of UV-mapping (Table 1).

We also compare our results with a previously proposed neural renderer for material visualization [Zsolnai-Fehér et al. 2018]. We implement their network in PyTorch and train on our dataset of uniform material parameter maps. For a fair comparison, we compare results over a fixed environment lighting. Our network produces better results than theirs both in terms of render quality and PSNR (Fig. 5, Table 1). Since the number of parameters of our network is lesser than theirs by a factor of 10, the run-time of our network is also better.

Table 1: Quantitative and run-time comparison (in milliseconds). The network of [Chaitanya et al. 2017] has a pre-process time for rendering the 2spp image while our network has a pre-process time for UV-mapping. Run-time values are evaluated on a workstation with 40 CPU cores and one NVIDIA GTX 1080Ti GPU.

	Quantitative		Run-time		
Algorithm	**PSNR(dB)**	**SSIM**	**Pre-proc.**	**Network**	**Total**
Zsolnai-Fehér et al. Params: 5,374,75,643	36.105	0.992	-	13.866	13.866
Chaitanya et al. Params: 15,05,453	30.437	0.965	70.000	2.510	72.510
Ours Params: 117,52,404	**37.656**	0.985	0.002	2.715	**2.717**

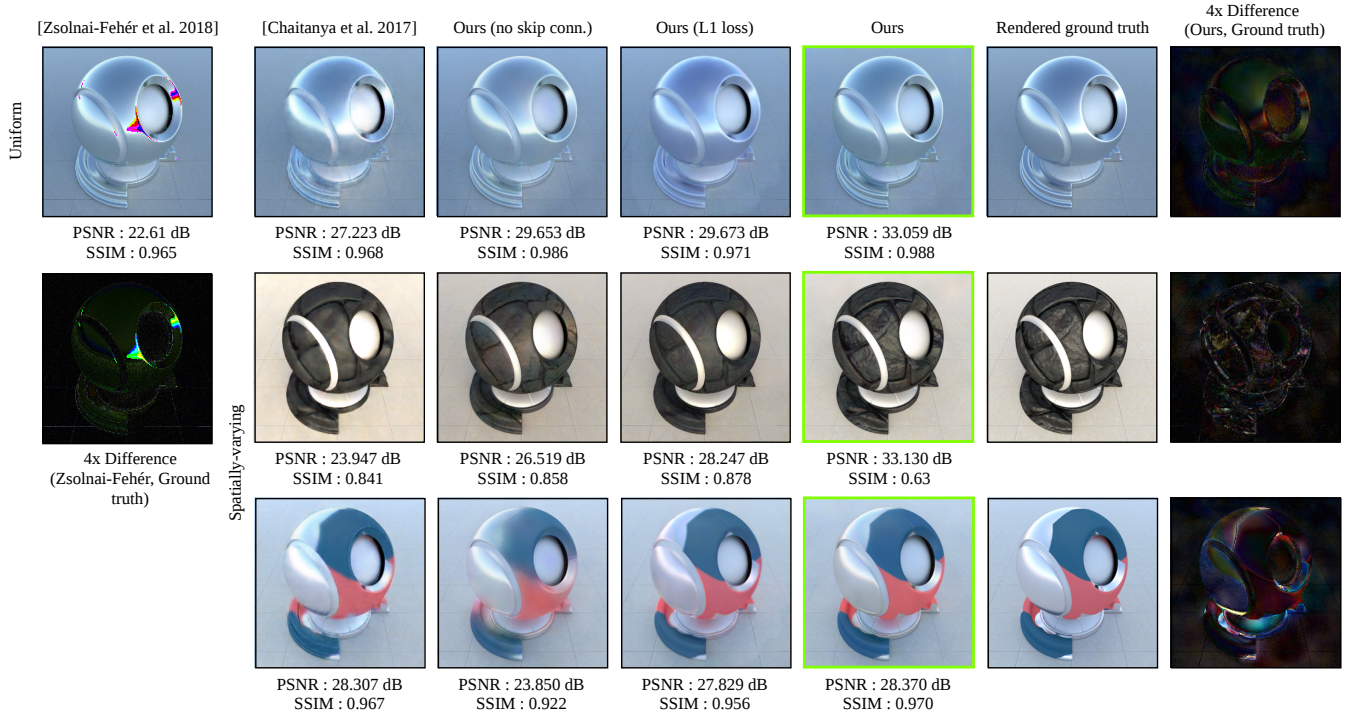

Figure 5: Average case and ablation study comparisons with [Zsolnai-Fehér et al. 2018] and [Chaitanya et al. 2017].

5.2 Quantitative Results and User Study

We justify the impact of our composite loss function by training a standalone network using only the L_1 loss. Figure 5 demonstrates that L_1 loss alone is unable to capture fine details, especially that of the normal map. We also justify the benefit of using skip connections in our network. Figure 5 shows this comparison. Without skip connections, several high-frequency details are lost (Fig. 5, last two rows).

Table 1 and Fig. 5 show that quantitative metrics like PSNR and SSIM do not faithfully reflect the visual quality of results. Although the average PSNR value of [Zsolnai-Fehér et al. 2018] is comparable to ours, their result contains artifacts in near perfectly dark or white regions. Another point to note is the resultant PSNR values produced by the denoiser and by our network. The quantitative values are very close, even though the latter's visual quality is superior (Fig. 5, last row).

We conduct an extensive user study consisting of 70 users to qualitatively validate the impact of flexible lighting for the task of material selection. From a rendered scene, we pick one dominant material from the scene and render four materials on the shaderball, with one of them being the correct material and others encoding slight variations. We ask the users to pick the correct materials in two independent experiments conducted with and without the freedom to view the shaderball under flexible lighting. We find that only 17.9% of users are able to identify the correct material under fixed lighting conditions. This number increases to 49.3% once the flexible lighting is made available. This clearly demonstrates the benefit of using flexible lighting in our renderings for material visualization.

6 CONCLUSION

In summary, we present a convolutional neural network for accurate rendering and visualization of both uniform and spatially-varying materials. We enable control over the environment lighting through our architecture, and verify its benefit through a qualitative user study. Comparison with denoising and neural rendering methods shows superior quantitative and qualitative results. We provide a large-scale dataset of uniform and spatially-varying material parameter-render pairs. In the future, we are interested in generalizing such networks to arbitrary geometry.

REFERENCES

Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. 2017. Interactive Reconstruction of Monte Carlo Image Sequences Using a Recurrent Denoising Autoencoder. *ACM Trans. Graph.* 36, 4 (July 2017).

Valentin Deschaintre, Miika Aittala, Frédo Durand, George Drettakis, and Adrien Bousseau. 2018. Single-Image SVBRDF Capture with a Rendering-Aware Deep Network. *ACM Trans. Graph.* 37, 128 (Aug 2018).

Lukas Hosek and Alexander Wilkie. 2012. An Analytic Model for Full Spectral Sky-Dome Radiance. *ACM Trans. Graph.* 31, 4 (July 2012).

Aakash KT, Parikshit Sakurikar, Saurabh Saini, and P. J. Narayanan. 2019. A Flexible Neural Renderer for Material Visualization. (Aug 2019). arXiv:1908.09530

Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. 2018. Noise2Noise: Learning Image Restoration without Clean Data. *CoRR* abs/1803.04189 (2018).

Peiran Ren, Yue Dong, Stephen Lin, Xin Tong, and Baining Guo. 2015. Image Based Relighting Using Neural Networks. *ACM Trans. Graph.* 34, 4 (July 2015).

Zexiang Xu, Kalyan Sunkavalli, Sunil Hadap, and Ravi Ramamoorthi. 2018. Deep image-based relighting from optimal sparse samples. *ACM Trans. Graph.* 37, 4 (2018).

Károly Zsolnai-Fehér, Peter Wonka, and Michael Wimmer. 2018. Gaussian material synthesis. *ACM Trans. Graph.* 37, 4 (2018).