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:

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
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 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 produce 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 control over environment lighting in the rendered material output.

Neural rendering for material visualization as proposed by Zsoltai-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.

3 A NEURAL RENDERING MODEL

The incoming radiance at each pixel \( x \) of a ray-traced 2D image can be modeled as:

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

where \( p_x \) is the 3D point corresponding to the 2D pixel \( x \), \( \omega_i \) is the incoming light direction at \( x \), \( \omega_o \) is the direction towards pixel \( x \) from point \( p_x \), \( L(p_x, \omega_i) \) is the radiance of incoming light at point \( p_x \) from direction \( \omega_i \), \( \Omega \) 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 \( \omega_s \) 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 \( \omega_s \) and turbidity \( c \), the solution of Eq. (1) is estimated by a convolutional neural network \( \phi \) 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
We compare our performance with two contemporary paradigms: (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].

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.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Quantitative</th>
<th>Run-time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSNR(dB)</td>
<td>SSIM</td>
</tr>
<tr>
<td>Zsolnai-Fehér et al.</td>
<td>36.105</td>
<td>0.992</td>
</tr>
<tr>
<td>Params: 5,374,75,643</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chaitanya et al.</td>
<td>30.437</td>
<td>0.965</td>
</tr>
<tr>
<td>Params: 15,05,453</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ours</td>
<td>37.656</td>
<td>0.985</td>
</tr>
<tr>
<td>Params: 117,52,404</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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


