École Polytechnique Fédérale de Lausanne

FastNRTF: Efficient Relighting of Complex Scenes using Neural Radiance Transfer Fields

by Elia Fantini

Master Project Report

Approved by the Examining Committee:

Sabine Süsstrunk
Head of Laboratory

Wang Dongqing
Project Supervisor

Ren Yufan
Project Supervisor

EPFL IC IINFCOM IVRL
BC 328 (Bâtiment BC)
Station 14
CH-1015 Lausanne

January 6, 2023

Abstract

In this project, we aim to reduce memory consumption and time requirements in the inverse rendering process for the reconstruction and relighting of 3D scenes from a set of images. Our approach utilizes the neural precomputed radiance transfer function (NRTF) framework proposed by [34] to handle complex global illumination effects while replacing the initialization process with a faster and lighter shape, lighting, and material estimation using Monte Carlo path tracing and denoising [20]. Previous methods that enable rendering under novel illumination conditions often rely on assumptions such as direct illumination and predetermined material models, which can lead to incomplete reconstructions of interreflections and shadow-free albedos. NRTF has proven effective in addressing this issue, but its implementation requires a computationally intensive process that necessitates a powerful GPU with a large amount of memory and a significant amount of time for training. To further improve efficiency, we claim that a large amount of synthetized light-stage images is unnecessary, reducing the generated data to approximately $1/70$ of the original volume, in addition to implementing other minor memory-light solutions. Our optimized approach results in a speedup of 10 to 40 times in training time and a minimum required VRAM of 6GB, while still producing high-quality relighting renderings.

1 Introduction

3D content creation is a time-consuming and technically demanding task that requires both artistic modeling skills and technical knowledge. Automating this process can help save production costs and increase the speed and diversity of content creation. Photogrammetry [45, 54] is one of the first techniques to convert multiple photos of an object into a 3D model. However, it involves multiple steps, including aligning cameras and finding correspondences, simplifying geometry, parameterizing textures, baking materials, and delighting, which can be challenging to optimize and prone to errors that propagate throughout the pipeline. As a result, artists often need to use a variety of software tools and make extensive manual adjustments to achieve the desired quality in the final 3D model. More controlled capture systems, including light stages with controlled lighting and cameras [18, 29, 69] still show clear limitations in terms of their usability and user-friendliness.

For this reason, differentiable rendering has recently taken place as a new method that increases automation in the process of reconstructing accurate 3D scenes from image observations. Firstly NeRF [36] used differentiable volume rendering to create high-quality view interpolation through neural, density-based light fields. A lot of the works following NeRF bake the scene's appearance in neural light fields, only allowing the generation of new images from novel points of view, excluding relighting or other forms of scene manipulation. On the other hand, the image formation process is influenced by many factors such as the scene geometry, the object materials, the lighting, and the properties of the recording camera. Some methods attempted to recover these scene's properties, but being an ill-posed and challenging inverse problem, most of them make several assumptions on the captured scene, such as considering direct lighting without shadows [44, 70, 4, 5, 55], combined with pre-filtered environment lighting [37, 67]. While these methods have achieved impressive results, their deviations from physically based shading models make it difficult to model complex global illumination effects and clearly distinguish between shape, material, and lighting.

Accordingly, we leverage Nvdiffrecmc [20], a recent approach that uses Monte Carlo path tracing for a more accurate light simulation that includes global illumination effects. In practice, however, the noise in multi-bounce Monte Carlo rendering makes gradient-based optimization challenging, so their solution is to use a denoiser and limit the light bounces to one. Hence, their approach still uses a predefined material model and does not properly model indirect illumination, but it extracts explicit triangle meshes with improved visual fidelity, shadow-free albedos, PBR materials, and environment lighting, in a format directly compatible with current DCC tools and game engines. Finally, the amount of training time necessary for an initial acceptable result is roughly an hour, which is competitive compared to other techniques and makes it preferable to achieve the first estimation for our pipeline.

To improve on those partial results, we use the framework proposed by [34] which consists of learning a precomputed radiance transfer (PRT) function parameterized by a neural network. PRT was first introduced as an efficient approximation of global illumination [48, 52, 53, 59, 63, 27], but never before it was estimated from just a set of images. Using a neural PRT has the advantage of not having a predefined approximation function (e.g. spherical harmonics), and a predefined material model so that the method can adapt to any complex light path such as indirect reflections and shadows.

Alternative techniques for applying global illumination to these applications typically utilize light-stage datasets, which consist of an object captured from various angles under various lighting conditions. However, it has been demonstrated that such a complex setup is not necessary. A crucial part of the PRT training is in fact the synthetization of light-stage data for the scene using a high-quality rendered [9], enabling for accurate separation of material and illumination properties of the scene. The real multi-view data is then utilized to capture the realistic details and address any general assumptions made by the renderer. Furthermore, unlike path tracing, the performance of this method at inference time does not depend on the complexity of light transport, and it is designed to generate noise-free rendering.

Unfortunately, the last-mentioned method does require a high amount of time and resources to train, getting far from the purpose of saving production costs and increasing the speed and diversity of content creation. To improve on this aspect, our project introduces different modifications to the original method. Firstly, the initial mesh, material, and lighting estimations are obtained using Nvdiffrecmc [20] instead of the combination of NeuS [60] and the differentiable renderer Mitsuba 2 [43]. Then, we argue that a large quantity of synthesized light-stage data is unnecessary for the inverse rendering process and can be significantly reduced. By decreasing the data volume ~ 70 times and randomizing the process, we are able to save a significant amount of computation time. Finally, the training pipeline is adapted and optimized to scale properly on consumer GPUs with a lower amount of memory, while shortening the required time to complete the training. A diagram of the entire process is provided in Figure 2.

Our results show that our approach significantly reduces the required time and memory while maintaining a similar quality to the state-of-the-art Neural Radiance Transfer Fields [34] under novel views and lighting conditions. The code is available at https://github.com/EliaFantini/FastNRTF.

2 Related Work

Inverse rendering is the process of determining the intrinsic properties (such as geometry, material, and/or lighting) that contribute to the appearance of an image. This problem is difficult because it is often ill-posed, meaning that there may be multiple possible solutions. To overcome this, priors and regularizations are added to the optimization process to make it more well-posed and increase the accuracy of the solution, while often reducing the generality of the approach as a counter-effect.

Classical multi-view 3D reconstruction methods can be divided into two categories: those that use inter-image correspondences [2, 1, 11, 49] to estimate depth maps and fuse them into point clouds or meshes, and those that use voxel grids to represent shapes and estimate occupancy and color for each voxel [50]. The former methods depend on accurate matching and may have difficulty correcting errors during post-processing, while the latter methods can be limited by their high memory requirements. For this reason, methods using a neural network to estimate a scene's properties have recently taken place, showing many advantages.

Neural reconstruction methods can also be classified into two groups: those that use implicit scene representations and those that use explicit ones. Implicit methods, such as NeRF [36] and its variants [35, 40, 46, 68, 61, 14, 38, 47, 66, 62], use volumetric representations and compute radiance by raymarching through a neurally encoded 5D light field. While these methods can produce impressive results for novel view synthesis, they may not produce high-quality geometry due to the ambiguity of volume rendering. Surface-based rendering methods [41, 65] instead, use implicit differentiation to optimize the underlying surface directly. NeuS [60] provides a conversion from a signed distance field (SDF) to density for volume rendering. These methods all rely on ray marching for rendering, which can be computationally expensive. For faster inference, implicit surfaces can be converted to meshes, but the process often introduces an additional error that is not accounted

for during optimization, degrading the final result. Explicit methods, on the other hand, estimate 3D meshes from images with the advantage that 3D meshes are much faster to render at inference time. Most approaches assume a given mesh topology [32, 8], but some recent methods also include topology optimization [31, 13, 51, 37]. In particular, DMTet [51] directly optimizes the surface mesh using a differentiable marching tetrahedral layer. However, it focuses on training with 3D supervision. Nvdiffrec [37] extended DMTet to 2D supervision, using differentiable rendering to jointly optimize topology, materials, and lighting. Finally, Nvdiffrecmc [20] improved on Nvdiffrec with a differentiable Monte Carlo renderer for direct illumination. Additionally, to reduce variance, they added a differentiable denoiser. Their method is the one we choose to obtain a first estimation of geometry, BRDF, and lighting.

Related to the estimation of surface radiometric properties such as bidirectional reflectance distribution functions (BRDFs) and spatially varying BRDFs (SVBRDFs) from images, the previous method required special viewing configurations, lighting patterns, or complex capturing setups [29, 15, 16, 17, 6, 3]. More recent methods use neural networks to predict BRDFs from images [12, 19, 30, 42], and differentiable rendering methods [8] can learn to predict geometry, SVBRDFs, and sometimes lighting using photometric loss. There are also neural 3D reconstruction methods that decompose shapes, materials, and lighting into their intrinsic components from images. These methods represent illumination using mixtures of spherical Gaussians [4, 37, 67, 71], pre-filtered approximations [5, 37], or low-resolution environment maps [70]. In some approaches, optimization is split into two passes, where geometry is fixed before the shadow term is sampled. Other approaches represent indirect illumination using neural networks [71]. On the other hand, all of these methods rely on some restricting assumptions or predefined functions that can model only a limited amount of light effects, and most of the time indirect illumination is not included. For this reason, PRT is preferable.

Precomputed Radiance Transfer (PRT) [52] is a technique for efficiently rendering global illumination [48] in computer graphics by partially precomputing light transport for free-viewpoint synthesis and dynamic lighting. However, previous work has not focused on recovering the PRT solely from a set of real-world images of an object. In contrast, a recent paper [34] recovers the PRT from images by training a neural PRT network. PRT's versatility has sparked interest in using various types of angular basis functions, including spherical harmonics [52, 5], Haar wavelets [11], and spherical isotropic [65]. To those functions, it has been preferred the generality of encoding the full radiance transfer into a neural field [64, 56], because they do not have any prior and can potentially model any illumination effect. Traditionally, the process of transferring the appearance of an object from distant illumination to local lighting has focused on calculating the incoming radiance [24] at a surface point, without taking into account the effect of reflectance. This is done for efficiency and to reduce storage requirements. In contrast, their method adopts a method used in PRT-based relighting [39], which involves predicting the outgoing radiance directly.

One of the most effective ways to obtain a light-reflectance decomposition in the presence of global illumination is to use a controllable illumination system, such as a multi-view light stage [10], and capture the appearance of an object under a single light source at a time. While this method is not without challenges, techniques such as novel-view synthesis and relighting boil down to clever interpolation. Inspired by this, their method takes as input casually-captured multi-view data under a single unknown illumination, but also includes synthetic generated data where the object is lightened by a single light source at a time, used during the training process to assist with disentangling the reflectance and illumination components.

3 Method

Our method takes as input roughly 100 multi-view images of an object under a single unknown illumination condition and generates new images from new user-defined points of view and illumination conditions. An illustration of the entire process is provided in Figure 2.

3.1 Geometry, lighting, and material estimation

The process to do so is divided into different steps, with the first one being the optimization of a 3D mesh, a PBR texture, and a 2D environment map that estimate the geometry, the SVBRDF, and lighting, respectively.

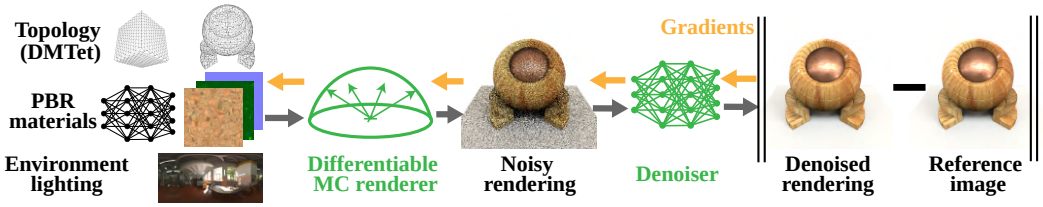

Figure 1: Representation of the optimization pipeline in the first step of the process, where geometry, lighting, and material are jointly estimated. Image is taken from Nvdiffrecmc's paper.

The original framework presented in [34] begins by estimating the geometry of the object using NeuS [60], which utilizes a signed distance field and converts it into a 3D mesh using Marching Cubes [33]. The resulting mesh is then loaded into the differentiable renderer Mitsuba 2 [43] and assigned a material, which is modeled as a convex combination of a rough conductor BSDF with roughness alpha and a diffuse BSDF with a 512x512 texture. The renderer uses path tracing to generate images of the object under illumination from a 16x32 environment map, and the difference between the generated and real images is used as a loss to optimize both the material and lighting via backpropagation.

However, this process has some limitations. The training of NeuS requires a significant amount of time (14 hours for 300k iterations on an RTX 2080ti) and the use of Marching Cubes can introduce additional errors into the final geometry estimation, not considered during the NeuS training. Additionally, the optimization of lighting and material is decoupled from the optimization of geometry and requires a large amount of GPU memory when using differentiable rendering with Mitsuba 2 and PyTorch.

In our project, we utilize Nvdiffrecmc [20] to jointly estimate the geometry, material, and lighting of an object. This method requires a longer training time to achieve the best results, but it is capable of producing high-quality results in a relatively short amount of time (less than 1 hour on an RTX 3070 with 8 GB of VRAM). The geometry is represented by a signed distance field defined on a 3D grid and then converted into a triangular surface mesh using Deep Marching Tetrahedra [51], which has been shown to produce higher quality reconstructions compared to Marching Cubes. Furthermore, the final mesh reconstruction is taken into account during the training process. The material model used by Nvdiffrecmc is based on the physically-based (PBR) model proposed by Disney [7], which is a widely-used and well-established model that can be easily imported into various software programs and engines (such as Blender [9]) without modification. The model consists of a diffuse term and an isotropic, specular GGX lobe [58], and also includes a tangent space normal map to capture high-frequency shading details. The lighting of the scene is estimated as a high dynamic range light probe, represented as an environment map with a resolution of 512x1024 pixels.

The process of Nvdiffrecmc is composed of multiple stages, as illustrated in Figure 1. To optimize the parameters ϕ (including shape, spatially varying materials, and light probe), a rendering process is applied to generate an image $I_\phi(c)$ for a given camera pose c. The resulting image includes noise due to the use of Monte Carlo integration during rendering, so a denoising process D_θ with parameters θ is applied to reduce the noise and produce a denoised image, $I_\phi^{\text{denoised}}(c) = D_\theta(I_\phi(c))$. This denoised image is then compared to a reference image $I_{\text{ref}}(c)$, captured from the same camera pose, and a photometric loss function L is used to measure the difference between the two images. L is defined as the sum of an image space loss, L_{image} ($L1$ norm on tone mapped colors), a mask loss, L_{mask} (squared $L2$), and a regularizer L_{reg} to improve geometry. L_{reg} is defined as the sum of the binary cross-entropy H of the sigmoid function σ and the sign function $\text{sgn}(x)$ applied to each unique edge, i, j, in the tetrahedral grid. This regularization term aims to reduce the number of sign flips and simplify the surface by penalizing internal geometry or floaters. The equation can be written as:

$$L_{\text{reg}} = \sum_{i,j \in \mathbb{S}_e} H\left(\sigma(s_i), \text{sgn}\left(s_j\right)\right) + H\left(\sigma(s_j), \text{sgn}\left(s_i\right)\right), \tag{1}$$

where $\mathbb{S}_e$ represents the set of unique edges for which $\text{sgn}(s_i) \neq \text{sgn}(s_j)$.

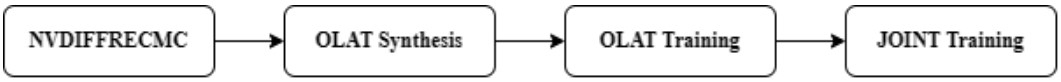

Figure 2: Representation of the whole training pipeline.

The empirical risk is then defined as :

$$\underset{\phi,\theta}{\operatorname{argmin}}\ \mathbb{E}_c\big[L\big(D_\theta(I_\phi(c)), I_{\text{ref}}(c)\big)\big] \tag{2}$$

and minimized by Adam [25] based on gradients w.r.t. the optimization parameters, $\partial L/\partial\phi$, and $\partial L/\partial\theta$, which are obtained through differentiable rendering.

The rendering process is summarized by the following equation [23]:

$$L(\omega_o) = \int_\Omega L_i(\omega_i) f(\omega_i, \omega_o)(\omega_i \cdot \mathbf{n}) d\omega_i. \tag{3}$$

The outgoing radiance $L(\omega_o)$ in a given direction ω_o can be modeled as an integral of the product of the incident radiance $L_i(\omega_i)$ from direction ω_i and the BSDF $f(\omega_i, \omega_o)$ over the hemisphere Ω around the surface normal $\mathbf{n}$. This rendering equation can be approximated using Monte Carlo integration, as shown in the following equation:

$$L(\omega_o) \approx \frac{1}{N} \sum_{i=1}^{N} \frac{L_i(\omega_i) f(\omega_i, \omega_o)(\omega_i \cdot \mathbf{n})}{p(\omega_i)}, \tag{4}$$

where the samples are drawn from the distribution $p(\omega_i)$ using multiple importance sampling (MIS) [57]. For further details, please refer to the Nvdiffrecmc [20] paper. This equation allows for the modeling of all-frequency lighting, including shadows. Since the gradients of diffuse scattering are small compared to the gradients of primary visibility, for performance reasons the shadow ray visibility gradients are separated when evaluating the hemisphere integral, and shape optimization is driven by the primary visibility gradients obtained from Nvdiffrast [26].

The incorporation of denoising techniques into the differentiable rendering process is crucial, as it enhances the visual fidelity of the rendered image and diminishes optimization errors and gradient noise during the computation of the image loss. It also allows for the sharing of gradients between neighboring pixels, improving the accuracy of the rendering. Denoisers are effective at improving rendering outcomes, especially when working with a small number of samples, and can improve the convergence and reconstruction of high-frequency environment lighting. Furthermore, using a small number of samples per pixel reduces memory and computation time for rendering, making Nvdiffrecmc a preferable choice for initial estimation compared to other methods.

3.2 OLAT Synthesis

The second step of the framework involves the generation of one-light-at-a-time (OLAT) renderings of the scene. These OLAT images are used to train the neural network to disentangle the material and illumination properties of the object. This is typically achieved using light-stage capture setups that involve multiple lighting conditions. However, these methods are inflexible, and it is possible to replace them with a virtual light stage that only requires captures of the real object under a single unknown illumination condition. To accomplish this, we use the 3D renderer Blender [9] with a range of OLAT illumination conditions, where only a single pixel on the environment map is active at a time. The 3D mesh estimated in the previous step of the pipeline is imported into Blender with its textures, and the cameras used to capture the real images are virtually recreated in the renderer, along with 200 additional randomly generated cameras in a sphere around the object with a similar radius to the original cameras.

In the original method, an OLAT image is generated for every permutation of a single white pixel in the environment map and camera. This means that for 100 original camera poses, 200 additional randomly generated ones, and 512 possible white pixels in the 16x32 environment map, a total of 153600 OLAT images must be generated. The OLAT representation would then form a complete basis for illumination, meaning that any environment map could be expressed as a combination of

OLAT environment maps, but this is computationally expensive and while the original method uses a CPU cluster, it may not be a feasible solution in many situations.

To address this issue, we significantly reduce computation time by introducing randomization to the process. Leveraging the ability of neural networks to generalize, we generate OLAT images using 2200 random permutations of white pixels in the environment map and cameras. Additionally, we reduce the number of pixels that can be chosen from the environment map by excluding the top and bottom rows, as these pixels are rendered with a lower light intensity in Blender due to being mapped to small curved triangles around the poles of the sphere. This can create outliers in the training data and slow down the training process. Finally, we remove the bottom half of the environment map for objects that are not viewable from the bottom in the original captures, such as the hotdog scene from NeRFactor [70]. OLAT images generated using these OLAT environment maps would be mostly black and thus useless for training.

3.3 OLAT Training

Once all the OLAT images have been generated, **the next step is the training of the neural radiance transfer field (NRTF)**. This involves replacing the rendering equation with a discretized version:

$$L_\theta(\mathbf{u}) = L_\theta\left(\mathbf{x}, \boldsymbol{\omega}_o\right) = \sum_{\hat{\boldsymbol{\omega}}_i} \hat{L}_\mathrm{e}(\hat{\boldsymbol{\omega}}_i) \cdot T_\theta\left(\mathcal{H}(\mathbf{x}), \mathbf{n}, \mathcal{F}(\hat{\boldsymbol{\omega}}_i), \mathcal{F}(\boldsymbol{\omega}_o)\right). \tag{5}$$

Where part of the multiplication is replaced with a collapsed radiance transfer function T_θ of parameters θ, which converts global, distant illumination from direction $\boldsymbol{\omega}_i$ into local reflected radiance at position $\mathbf{x}$ in the direction $\boldsymbol{\omega}_o$. In order to compute global illumination for a given pixel using an environment map and the scene-specific transfer function T_θ, we need to determine the primary intersection point $\mathbf{x}$, evaluate T_θ for all texels in the environment map, multiply the resulting values by the corresponding illumination, and sum all of the contributions. If T_θ is compact and easy to evaluate, it can be efficiently computed on a GPU using a map-reduce approach.

To achieve this, T_θ **is estimated by a multi-layer perceptron (MLP)** that takes as input a point on the surface and its normal, as well as the incoming and outgoing light directions, and predicts the radiance transfer of the scene. This radiance can then be multiplied by an arbitrary environment map to enable global illumination relighting. T_θ is defined as follows:

$$T_\theta\left(\mathcal{H}(\mathbf{x}), \mathbf{n}, \mathcal{F}(\boldsymbol{\omega}_i), \mathcal{F}(\boldsymbol{\omega}_o)\right) = \mathbf{c}_t, \tag{6}$$

In addition to the NRTF's MLP, the 3D position $\mathbf{x}$ is embedded by a neural multi-resolution hash encoding $\mathcal{H}(\cdot)$ [38], which speeds up the convergence of training and inference. The light directions $\boldsymbol{\omega}_i$ and $\boldsymbol{\omega}_o$ are instead embedded by a spherical harmonics encoding $\mathcal{F}(\cdot)$.

During this step, the NRTF MLP is initially trained using only the synthesized OLAT images. The loss to be minimized using an Adam optimizer is defined as follows:

$$\mathcal{L}_\mathrm{OLAT}(\theta) = \sum_{i=1}^{N_\mathrm{c}} \sum_{\mathbf{u} \in \mathcal{M}_\mathrm{i}} \left|\left| \frac{L_{\theta,i}(\mathbf{u}) - O_i(\mathbf{u})}{\mathrm{sg}\left(L_{\theta,i}(\mathbf{u})\right) + \epsilon} \right|\right|^2, \tag{7}$$

Where O_i is the ith OLAT image from Blender and $L_{\theta,i}$ is the corresponding estimate from NRTF's MLP using Eq. 5. The stop gradient is denoted by $\mathrm{sg}(\cdot)$ and ϵ is used to avoid division by zero and optimize for the network parameters θ. As shown in Noise2Noise [28], this loss is effective for high-dynamic range images in the presence of path-tracing noise.

3.4 JOINT Training

In the previous step of training, the MLP heavily relies on the lighting-reflectance ambiguity and the assumption of a global specularity parameter. However, in order to improve the accuracy of results, **the training is now fine-tuned using both real and synthetic images**. During this final step of the pipeline, images are fragmented into small batches of random pixels to limit memory usage.

The loss function for this stage is defined as follows:

$$\mathcal{L}(\theta, \tilde{L}_\mathrm{e}) = \mathcal{L}_\mathrm{OLAT}(\theta, \tilde{L}_\mathrm{e}) + \lambda_\mathrm{PRT}\mathcal{L}_\mathrm{PRT}(\theta, \tilde{L}_\mathrm{e}) + \lambda_\mathrm{Env}\mathcal{L}_\mathrm{Env}(\tilde{L}_\mathrm{e}). \tag{8}$$

Where $\mathcal{L}_{\text{PRT}}$ is the $L2$ loss between the generated images and the real captures, $\mathcal{L}_{\text{OLAT}}$ is the same as in the previous step, and $\mathcal{L}_{\text{Env}}$ is the $L2$ loss between the initial envmap estimated by Nvdiffrecmc and the current finetuned envmap. Since the envmap estimated by Nvdiffrecmc may be incorrect, this loss term allows for fine-tuning of the envmap.

3.5 Implementation design

One goal of inverse rendering is to create more autonomous methods for reconstructing 3D objects. To this end, **our code is designed to reduce the number of user interactions required during the training process**.

In the original method for inverse rendering, the process for reconstructing a 3D mesh involves running NeuS's training, which can be challenging for some users due to the complexity of the setup process. Additionally, another training must be run using Mitsuba 2 to estimate materials and lighting. The generation of one-light-at-a-time (OLAT) images using Blender requires multiple camera imports and exports, the generation of random cameras, and the synthesis of OLAT images, a total of 7 files that must be run manually and with various Blender settings to be adjusted. The training of the neural radiance transfer field (NRTF) requires image-to-tensor conversion, buffer generation, OLAT training, joint training, and final relight, requiring the manual execution of 5 python files.

In contrast, our modified method only requires the execution of one file for the Nvdiffrecmc reconstruction, another python file run from Blender's scripting mode to automatically adjust default settings, properly import cameras, the reconstructed mesh and materials, set the correct material model's properties, generate random cameras, and synthesize OLAT images. Finally, another python file is run to render the buffer, convert images to tensors, rotate the estimated environment map, run both OLAT and JOINT training steps, and relight the scenes under different novel views and novel environment maps, calculating an average of PSNR, SSIM, and LPIPS metrics. This process requires only 3 manual operations from the user, making it more user-friendly and efficient.

4 Experiments and results

4.1 Lighting, geometry, material estimation

The goal of this project is to reduce the amount of time and memory required to perform the various steps in the inverse rendering pipeline. The hardware used for this purpose is an NVIDIA RTX 3070 with 8 GB of VRAM and an NVIDIA GTX TITAN X with 12 GB of VRAM, with the aim of making each step possible even on a GPU with as little as 8 GB of VRAM.

Originally, the code for Nvdiffrecmc [20] was not publicly available, so a different pipeline was used: **Nvdiffrec [37] was used to estimate the initial 3D mesh, lighting, and materials of the object**. However, Nvdiffrec does not accurately model shadows and indirect illumination due to its use of a rasterizer instead of a path tracer, resulting in reconstructed models without shadow-free albedos. To address this issue, the generated mesh was loaded into the differentiable path tracer Mitsuba 3 [21]. In comparison to the original method [34], the initial reconstruction using Nvdiffrec takes only about an hour, compared to roughly 14 hours using NeuS [60].

Additionally, **we used Mitsuba 3 instead of Mitsuba 2**, which does not support the material model used by Nvdiffrec. This allowed us to directly improve on the output estimated by Nvdiffrec, resulting in faster convergence. Furthermore, the differentiable rendering in Mitsuba 2 is handled by PyTorch, which requires a large amount of memory and cannot run on a GPU with only 12 GB of VRAM. In contrast, our method leveraged Dr.Jit [22], a compiler for physically-based differentiable rendering, for a faster and lighter training process that can run on an RTX 3070 with 8 GB of VRAM.

Later, when the code for Nvdiffrecmc became available, it replaced Nvdiffrec into the pipeline and the step using Mitsuba 3 was removed. A comparison of the reconstructions obtained using Nvdiffrec and Nvdiffrecmc is shown in Figure 3.

Our experiments are conducted using the NeRFactor [70] dataset, comprising of four scenes: Lego, Drums, Hotdog, and Ficus. The results presented in Nvdiffrecmc's paper are obtained by running the training process for 10 hours on an RTX A6000 with 48GB of VRAM. However, **we are able to achieve less accurate but still good estimations in just one hour** using an RTX 3070 on all four scenes, with a peak memory usage of 5.46GB in the Lego scene. We modify the batch size to reduce

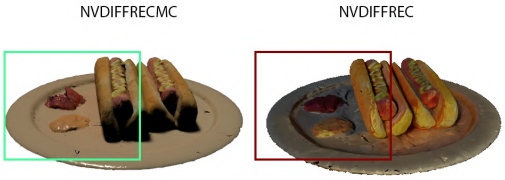

Figure 3: Nvdiffrecmc results on the left, Nvdiffrec results on the right. Despite the light illuminating the object in the Nvdiffrec image is coming from the left, shadows are still visible on the left of the Hotdogs as they are baked into the albedo. The mesh also shows artifacts. Both of these problem are solved in the reconstruction made by Nvdiffrecmc.

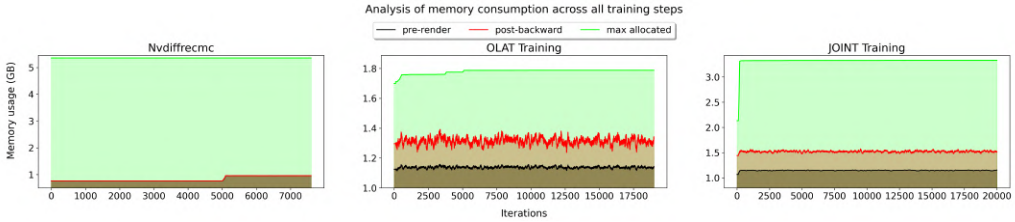

Figure 4: Analysis of memory usage across all three training steps recorded before the rendering process, after the backward step, and as maximum allocated memory. In the Nvdiffrecmc phase, we observe an increase in memory usage during the second phase of training, which is more demanding in terms of memory. Additionally, we observe that memory usage is generally stable, although there are occasional spikes during the early iterations that cause the maximum allocated memory to be higher than the average memory usage per iteration.

memory usage, although this leads to slightly less accurate meshes in complex scenes such as Drums and Ficus.

Our **memory analysis** reveals that the second phase of Nvdiffrecmc's training, in which mesh optimization is locked while material and lighting are fine-tuned, requires significantly more memory. As a result, we modify the original code to use a batch of 6 in the first phase, which is crucial for mesh estimation, while keeping the batch size at 1 in the second phase. A detailed plot of the used memory is shown in Figure 4.

4.2 OLAT Synthesis

In order to reduce the time required to generate images for training, we implement a randomization process for the generation of OLAT renderings. Instead of synthesizing all possible combinations of cameras and OLAT environment maps, as is done in the original method, we randomly select a limited number of such combinations.

To determine the appropriate number of images to generate, we use statistical analysis. In the NeRFactor dataset, there are 100 cameras, and we randomly generate 100 additional cameras (as opposed to 200 in the original method). The probability that one OLAT environment map will be chosen from a total of 512 in N generated images follows the Binomial distribution with parameters $n = N$ and $p = 1/512$ ($X \sim Bin(n, p)$). If N=2000, the probability P of being chosen at least once is $P(X \geq 1) = 0.98855$, and the probability of being chosen at least five times is $P(X \geq 5) = 0.4612$. For scenes like Lego and Hotdog, where only lights coming from the top half of the environment map are able to illuminate the object, these probabilities become $P(X \geq 1) = 0.99987$ and $P(X \geq 5) = 0.9429$.

We determine that these probabilities are satisfactory and **choose to generate 2000 OLAT images for our experiments**, in addition to 200 for validation purposes. We then test using fewer images

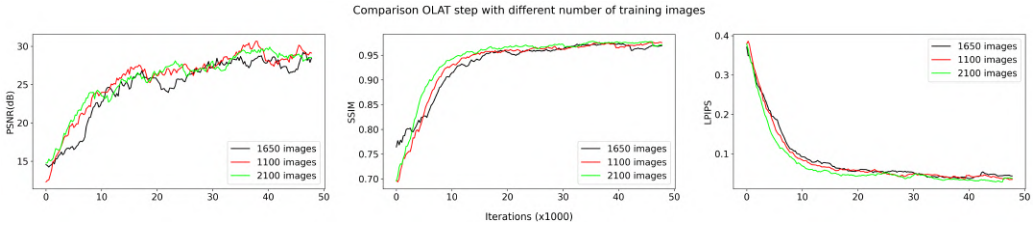

Figure 5: The plots show PSNR, SSIM and LPIPS metrics on three different trainings using different numbers of training images. No significant difference is observable.

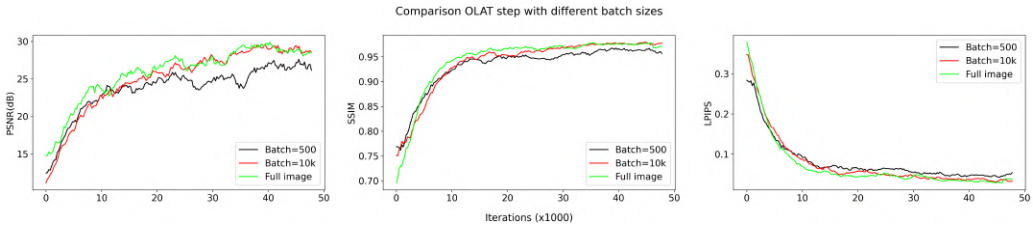

Figure 6: The plots show PSNR, SSIM and LPIPS metrics on three different trainings using different batch sizes. ”Full image” means that the whole image was rendered instead of ”batch size” random pixels. No significant difference is observable.

and observed no difference in the convergence speed of all metrics (see Figure 5), suggesting that even a lower number of training images would produce good final results.

4.3 OLAT Training

Various methods are implemented to optimize the memory consumption in the OLAT step of the training. The original code requires more than 12GB of memory to run on a TITAN X and the largest MLP that can fit in memory has 5 layers of 128 neurons each, so we delete unused tensors, avoid pre-loading the entire dataset into memory, store the Hash Embedder weights as float32 instead of float64, and reduce the number of images processed per iteration from 20 to 1. These efforts allow us to reduce the maximum allocated memory to a maximum of 5.8GB in the Lego scene using the original MLP size (7 layers of 512 neurons each), making it runnable on an RTX 3070.

In addition, **we implement the ability to process each image by only using a random subset of pixels**, further reducing memory usage and allowing the code to run on even smaller GPUs. A comparison of convergence and memory usage with different batch sizes is shown in Figure 6, demonstrating similar behavior in the validation metrics LPIPS, SSIM, and PSNR.

These results, along with the previous findings on the effect of different numbers of OLAT training images, indicate that the PRT network is highly adept at learning shadows from a small number of images and that using a smaller batch of pixels instead of the entire image does not significantly slow down the training process, but it does reduce the time required to synthesize images with Blender and the required memory to 4.4GB with a batch size of 500.

4.4 JOINT Training

Similarly to the previous step, the joint training part can not run on 12GB of VRAM originally. To reduce memory requirements, **we introduce the same changes applied to the previous step, preloading only the dataset containing the real captures. We also use PyTorch’s mixed precision training**, which automatically scales tensors to float16 to save memory. As a result, we are able to reach a peak memory usage of 7.2GB with a batch size of 500 pixels, and of 6.4GB with batch size 250. Evaluation metrics including PSNR, SSIM, and LPIPS show almost no difference between the two batch sizes.

However, **our initial relighting results after the final training step were incorrect**, as depicted in Figure 7. This was due to Nvdiffrecmc producing an environment map that is rotated by -90°

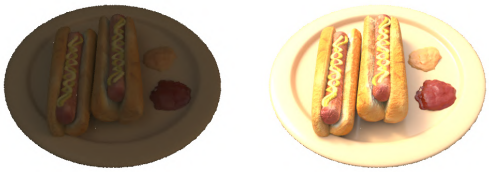

Figure 7: The figure displays the results of our initial relighting experiment through a comparison of the relighted image with its reference. It is clear that the illumination is different in the two images and that the shadows' directions are also incorrect.

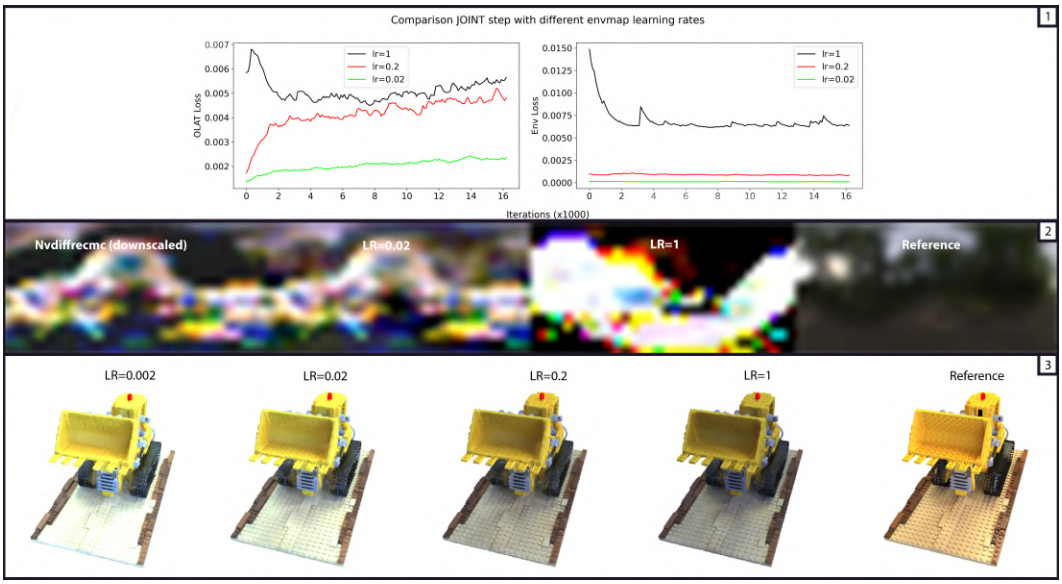

Figure 8: In this figure, we represent the effect of different learning rates for the environment map estimation. Plot (1) indicates that higher learning rates lead to larger deviations from the original estimation, increasing OLAT loss as the OLAT training images were generated based on the assumption that the correct environment map was the one estimated by Nvdiffrecmc. The resulting environment maps (2) show that a learning rate of 0.02 improves Nvdiffrecmc's estimation in the upper part, with brighter pixels that are more similar to the reference. However, excessively high values of the learning rate diverge from the optimal solution. A lower value of 0.002 leaves Nvdiffrecmc's result almost unchanged, leading to an overexposed rendering in the resulting relight images (3), consistent with the darker pixels in the environment map. While higher learning rates can alleviate this issue, colors remain different from the reference.

compared to the groundtruth envmap. By rotating the map in the opposite direction, we are able to correct the issue of incorrect shadows. Additionally, while the original method uses .exr format images with high dynamic range as input, NerFactor provides 16bit png images in the sRGB color space. To address this discrepancy, we tone map to the images in the RGB color space and encode the OLAT images generated with Blender as 16bit png files. This solution resolves the inconsistency in color between OLAT training samples and real captures.

However, **Nvdiffrecmc struggles to properly reconstruct lighting in the Lego scene**, resulting in worse relighting results compared to other scenes. To address this issue, we experiment with different learning rates for the envmap optimizer during the joint step, and find that higher values slightly improve the final result by more significantly modifying the initial envmap estimation. Further details can be found in Table 4.4 and Figure 8.

Envmap's Learning Rate	Tone Loss ↓	PSNR (dB) ↑	SSIM ↑	LPIPS ↓	PSNR Envmap ↑
Default [34]: 0.002	0.01159	15.04136	0.85856	0.27117	6.73
0.02	0.01371	16.92060	0.87473	0.22315	**8.38**
0.2	**0.02054**	**17.70357**	0.88372	0.19866	7.21
1	0.03404	15.27022	**0.86298**	0.14775	0.87

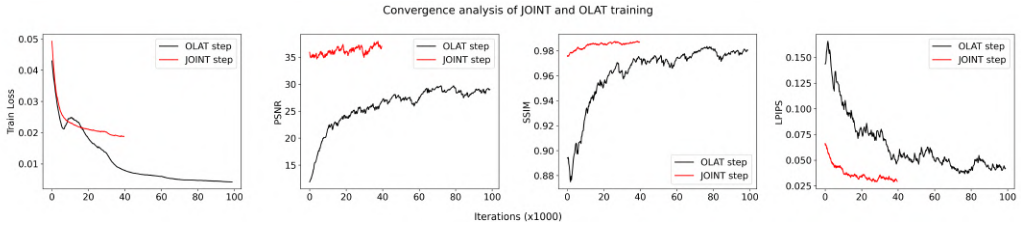

Figure 9: The convergence of Train Loss, PSNR, SSIM, and LPIPS metrics are plotted for the OLAT and JOINT training steps. The JOINT line terminates sooner due to the longer duration of each iteration which makes it necessary to run it for a smaller number of them. The plots demonstrate that the majority of improvements are achieved in the initial iterations, while subsequent iterations yield only slight improvements to the final result.

To analyze the convergence of the training process, we conduct several experiments. First, we attempt a long training of 100k iterations for the OLAT step (12 hours 15 minutes) and 50k for the JOINT step (13 hours 40 minutes). However, we found that training converges quickly (as shown in Figure 9). Therefore, we reduce the number of iterations to make the total training time more feasible, and evaluate the relight results by averaging metrics calculated on images generated using 8 random cameras and 3 random illumination conditions from the NeRFactor dataset. The results are shown in Table 4.4 and in Figure 11. Despite the relatively small differences between the initial lighting estimation of Nvdiffrecmc and the final result as measured by some metrics, the visual quality of the final result is significantly improved through the incorporation of small details and complex indirect illumination effects, particularly in scenes with a high concentration of translucent and semi-transparent objects such as the Drums scene. The NRTF model is able to make adjustments to the initial lighting estimation, correct mesh artifacts, and learn inter-reflections and complex illumination effects that were previously not captured. Table 4.4 shows results for the novel-view synthesis without relighting task.

To further reduce the training time, **we attempt to eliminate the OLAT step** by only training the PRT network with the JOINT step, which results in a significant reduction in final quality. Finally, **we experiment with using an MLP with 128 neurons per layer** instead of 512. While a smaller network results in faster convergence and inference (from an average of 12.46s per image generated to just 1.75s), it also leads to a small decrease in final performance. Example of relight from both experiments are shown in Figure 10.

Scene	Tone Loss	PSNR (dB) ↑	SSIM ↑	LPIPS ↓
Drums	0.05159	14.13055	0.91713	0.35030
Nvdiffrecmc - Drums	N/A	11.57039	0.89379	0.07537
Lego	0.02054	17.70357	0.88372	0.19866
Nvdiffrecmc - Lego	N/A	11.88978	0.83522	0.13175
Hotdog	0.00451	23.31084	0.93701	0.13735
Nvdiffrecmc - Hotdog	N/A	17.43790	0.91191	0.09938
Ficus	0.09918	23.83234	0.96024	0.02103
Nvdiffrecmc - Ficus	N/A	14.70438	0.93736	0.03444
Hotdog (w/o OLAT step)	0.00681	15.21425	0.86621	0.16953
Hotdog(smaller MLP)	0.03199	19.99928	0.92072	0.21560
Nvdiffrecmc average (all four scenes)	N/A	13.90	0.8945	0.0852
Our average (all four scenes)	0.04395	19.74	**0.9245**	0.1768
Original method (Drums and Hotdog)	N/A	**20.18**	0.7556	**0.0786**

Figure 10: The left figure shows a relighted version of the Hotdog scene using a smaller MLP with 128 neurons per layer, along with its reference. The right figure displays a relight of the Hotdog scene using only the JOINT optimization step, in comparison to its reference. While the relight produced using the smaller MLP yields acceptable results, the relight obtained through the use of only the JOINT optimization step exhibits significantly poor quality. Inference time using a smaller MLP goes down from 12.46s per image generated to 1.75s.

	Tone Loss	PSNR (dB) ↑	SSIM ↑	LPIPS ↓
Original method (Drums and Hotdog)	N/A	20.82	0.7577	0.0659
Our average (all four scenes)	0.03320	**29.86**	**0.9779**	**0.0372**

4.5 Time and memory

We demonstrate that it is possible to significantly reduce the training time of the inverse rendering process by synthesizing fewer OLAT images and running fewer iterations while still achieving high-quality results. Our experiments show that Nvdiffrecmc takes approximately 1 hour to run, and another hour is needed to generate 2200 OLAT images. The following OLAT and JOINT training steps may vary in duration depending on the complexity of the scene, as complex inter-reflections and small details take longer to learn. However, we found that most global illumination effects and details are learned in the early iterations, allowing for good results to be achieved in less time. For example, our Hotdog results were obtained with 85k iterations for the OLAT step, taking approximately 10 hours, and 9k iterations for the JOINT step, taking 3 hours. In total, this required 15 hours on a GTX TITAN X. In comparison, the original method required 14 hours for NeuS's mesh estimation, 30 minutes for Mitsuba lighting and material optimization, 67 hours of OLAT generation with Blender without using a cluster of servers, 8 hours for the OLAT step, and 16 hours for the final JOINT step, for a total of over 100 hours.

Additionally, **running the training on consumer GPUs with lower amounts of available memory is possible** by using smaller neural networks and lower batch sizes while maintaining convergence times and final performance. In particular, we found that 6GB of VRAM is sufficient to run the entire training process.

Finally, by **applying the insights gained from our experiments**, we were able to run a training on the Ficus scene using an MLP with 128 neurons per layer, an envmap learning rate equal to 0.02, and only 1100 OLAT images, generated in approximately 30 minutes. The subsequent 20k OLAT iterations and 10k JOINT iterations required 2 hours to run, for a total of 3 hours and 30 minutes for the entire pipeline and a peak of 3.33GB of maximum allocated GPU memory. Relighting over 8 random views with 3 random test environment maps resulted in an average PSNR of 23.83 dB, SSIM of 0.96024, and LPIPS of 0.02103, surpassing the results obtained with a larger MLP and more training time.

As previously stated, metrics alone do not provide a comprehensive measure of visual fidelity. Nonetheless, our results indicate that it is possible to significantly reduce the required training time, from ~ 10 up to ~ 40 times, without significantly decreasing quality. This represents a favorable trade-off between time and quality.

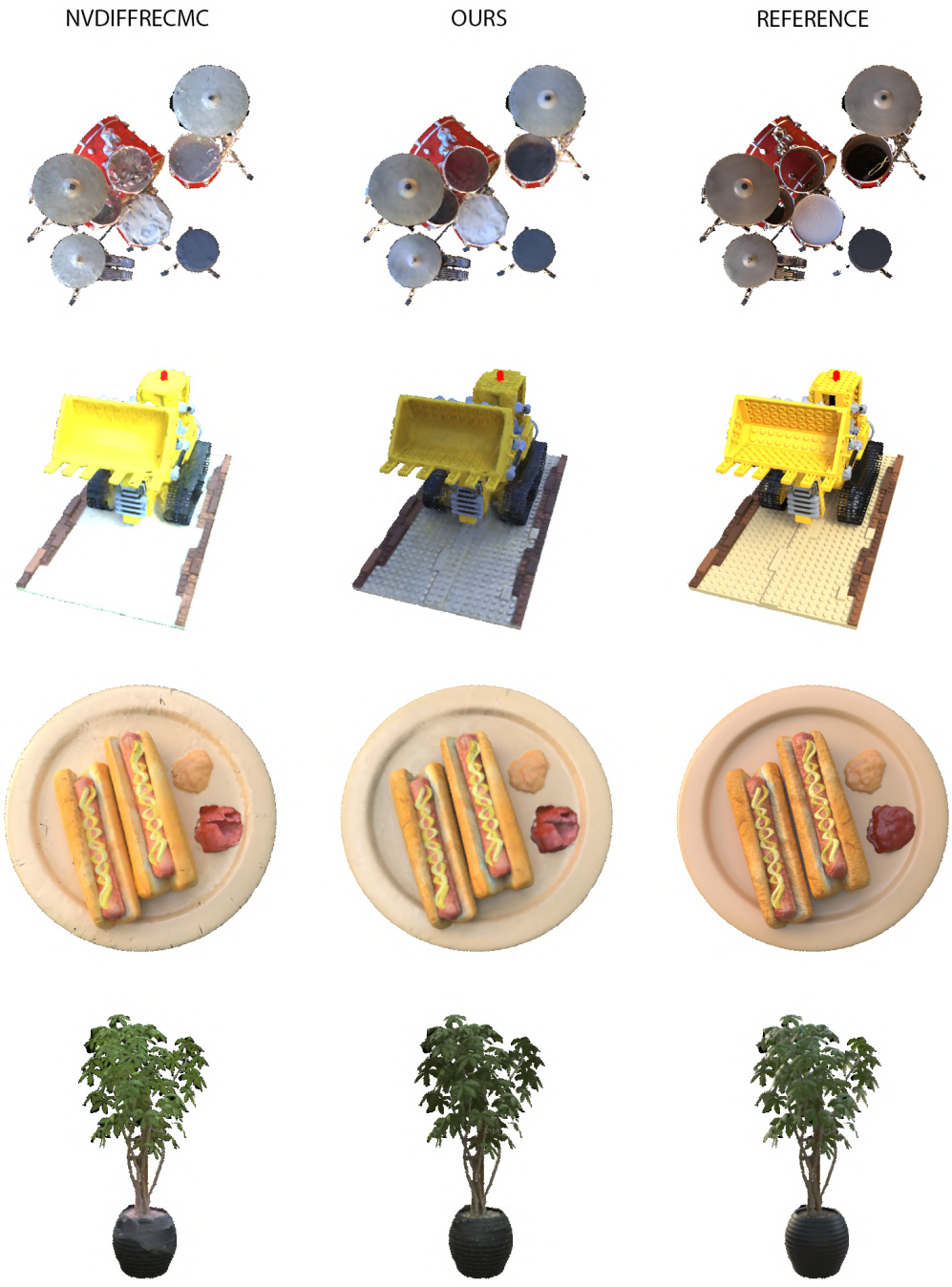

Figure 11: Relighting examples from the four scenes of the NeRFactor dataset. Results from Nvdiffrecmc are compared to our results and to the reference.

5 Conclusions

In this project, we have proposed modifications to the original neural radiance transfer field training method to optimize the process for global illumination scene relighting and view synthesis from multi-view images of an object. By utilizing Nvdiffrecmc for initial geometry, material, and lighting estimation and adding randomization to the OLAT generation process, we have achieved a significant improvement in training time, ranging from a speedup factor of ~ 10 up to ~ 40, as well as a reduction in required GPU memory down to as low as 6GB. Our results show that these modifications result in visual fidelity that is largely comparable to the original method, with only minor losses.

However, we have also identified that the initial lighting estimation by Nvdiffrecmc is not always perfectly reconstructed, and fine-tuning it in later stages is insufficient to fully resolve this issue. In addition, while the use of a smaller MLP improves inference speed, it is not yet sufficient for real-time performance. Therefore, future work could focus on further optimizing runtime and improving the accuracy of lighting estimation.

References

[1] Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376, 2010.

[2] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M. Seitz, and Rick Szeliski. Building rome in a day. *Communications of the ACM*, 54:105–112, 2011.

[3] Sai Bi, Zexiang Xu, Pratul Srinivasan, Ben Mildenhall, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. Neural reflectance fields for appearance acquisition, 2020.

[4] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T. Barron, Ce Liu, and Hendrik P. A. Lensch. Nerd: Neural reflectance decomposition from image collections, 2020.

[5] Mark Boss, Varun Jampani, Raphael Braun, Ce Liu, Jonathan T. Barron, and Hendrik P. A. Lensch. Neural-pil: Neural pre-integrated lighting for reflectance decomposition, 2021.

[6] Mark Boss, Varun Jampani, Kihwan Kim, Hendrik P.A. Lensch, and Jan Kautz. Two-shot spatially-varying BRDF and shape estimation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2020.

[7] Brent Burley. Physically Based Shading at Disney. In *SIGGRAPH Courses: Practical Physically Based Shading in Film and Game Production*, 2012.

[8] Wenzheng Chen, Jun Gao, Huan Ling, Edward J. Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer, 2019.

[9] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[10] Paul Debevec, Tim Hawkins, Chris Tchou, Haarm-Pieter Duiker, Westley Sarokin, and Mark Sagar. Acquiring the reflectance field of a human face. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, page 145–156, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[11] S. Galliani, Katrin Lasinger, Konrad Schindler, and Konrad Schindler. Gipuma: Massively parallel multi-view stereo reconstruction. 2016.

[12] DUAN GAO, Xiao Li, Yue Dong, Pieter Peers, Kun Xu, and Xin Tong. Deep inverse rendering for high-resolution svbrdf estimation from an arbitrary number of images. *ACM Trans. Graph.*, 38(4), jul 2019.

[13] Jun Gao, Chen Wenzheng, Tommy Xiang, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Learning deformable tetrahedral meshes for 3d reconstruction. 11 2020.

[14] Stephan J. Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps, 2021.

[15] Andrew Gardner, Chris Tchou, Tim Hawkins, and Paul Debevec. Linear light source reflectometry. *ACM Trans. Graph.*, 22(3):749–758, jul 2003.

[16] Abhijeet Ghosh, Tongbo Chen, Pieter Peers, Cyrus A. Wilson, and Paul Debevec. Estimating specular roughness and anisotropy from second order spherical gradient illumination. *Computer Graphics Forum*, 28(4):1161–1170, 2009.

[17] D. Guarnera, G.C. Guarnera, A. Ghosh, C. Denk, and M. Glencross. Brdf representation and acquisition. *Computer Graphics Forum*, 35(2):625–650, 2016.

[18] Kaiwen Guo, Peter Lincoln, Philip L. Davidson, Jay Busch, Xueming Yu, Matt Whalen, Geoff Harvey, Sergio Orts-Escolano, Rohit Pandey, Jason Dourgarian, Danhang Tang, Anastasia Tkach, Adarsh Kowdle, Emily Cooper, Mingsong Dou, Sean Ryan Fanello, Graham Fyffe, Christoph Rhemann, Jonathan Taylor, Paul E. Debevec, and Shahram Izadi. The relightables: volumetric performance capture of humans with realistic relighting. *ACM Trans. Graph.*, 38(6):217:1–217:19, 2019.

[19] Yu Guo, Cameron Smith, Miloš Hašan, Kalyan Sunkavalli, and Shuang Zhao. Materialgan: Reflectance capture using a generative svbrdf model. *ACM Trans. Graph.*, 39(6), nov 2020.

[20] Jon Hasselgren, Nikolai Hofmann, and Jacob Munkberg. Shape, light, and material decomposition from images using monte carlo rendering and denoising, 2022.

[21] Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, Merlin Nimier-David, Delio Vicini, Tizian Zeltner, Baptiste Nicolet, Miguel Crespo, Vincent Leroy, and Ziyi Zhang. Mitsuba 3 renderer, 2022. https://mitsuba-renderer.org.

[22] Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, and Delio Vicini. Dr.jit: A just-in-time compiler for differentiable rendering. *Transactions on Graphics (Proceedings of SIGGRAPH)*, 41(4), July 2022.

[23] James T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150, aug 1986.

[24] Jan Kautz, Peter-Pike J. Sloan, and Jaakko Lehtinen. Precomputed radiance transfer: theory and practice. *ACM SIGGRAPH 2005 Courses*, 2005.

[25] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[26] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering, 2020.

[27] Jaakko Lehtinen. A framework for precomputed and captured light transport. *ACM Trans. Graph.*, 26(4):13–es, oct 2007.

[28] Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. Noise2noise: Learning image restoration without clean data, 2018.

[29] Hendrik P.A. Lensch, Jochen Lang, Asla M. Sá, and Hans-Peter Seidel. Planned sampling of spatially varying brdfs. *Computer Graphics Forum*, 22(3):473–482, 2003.

[30] Zhengqin Li, Mohammad Shafiei, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. Inverse rendering for complex indoor scenes: Shape, spatially-varying lighting and svbrdf from a single image, 2019.

[31] Yiyi Liao, Simon Donné, and Andreas Geiger. Deep marching cubes: Learning explicit surface representations. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2916–2925, 2018.

[32] Shichen Liu, Weikai Chen, Tianye Li, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. pages 7707–7716, 10 2019.

[33] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, page 163–169, New York, NY, USA, 1987. Association for Computing Machinery.

[34] Linjie Lyu, Ayush Tewari, Thomas Leimkuehler, Marc Habermann, and Christian Theobalt. Neural radiance transfer fields for relightable novel-view synthesis with global illumination, 2022.

[35] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections, 2020.

[36] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020.

[37] Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Mueller, and Sanja Fidler. Extracting Triangular 3D Models, Materials, and Lighting From Images. *arXiv:2111.12503*, 2021.

[38] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4):1–15, jul 2022.

[39] Ren Ng, Ravi Ramamoorthi, and Pat Hanrahan. All-frequency shadows using non-linear wavelet lighting approximation. *ACM SIGGRAPH 2003 Papers*, 2003.

[40] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[41] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision, 2019.

[42] Merlin Nimier-David, Zhao Dong, Wenzel Jakob, and Anton Kaplanyan. Material and Lighting Reconstruction for Complex Indoor Scenes with Texture-space Differentiable Rendering. In Adrien Bousseau and Morgan McGuire, editors, *Eurographics Symposium on Rendering - DL-only Track*. The Eurographics Association, 2021.

[43] Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. Mitsuba 2: A retargetable forward and inverse renderer. *Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 38(6), Dec. 2019.

[44] Julien Philip, Michaël Gharbi, Tinghui Zhou, Alexei Efros, and George Drettakis. Multi-view relighting using a geometry-aware network. *ACM Transactions on Graphics (SIGGRAPH Conference Proceedings)*, 38(4), July 2019.

[45] Marc Pollefeys and Luc Van Gool. From images to 3d models. *Commun. ACM*, 45:50–55, 07 2002.

[46] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[47] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps, 2021.

[48] Tobias Ritschel, Carsten Dachsbacher, Thorsten Grosch, and Jan Kautz. The state of the art in interactive global illumination. *Computer Graphics Forum*, 31(1):160–188, 2012.

[49] Johannes L. Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision*, 2016.

[50] Steven Seitz and Charles Dyer. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 25(3), November 1999.

[51] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[52] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. Graph.*, 21(3):527–536, jul 2002.

[53] Peter-Pike Sloan, Ben Luna, and John Snyder. Local, deformable precomputed radiance transfer. *ACM Trans. Graph.*, 24(3):1216–1224, jul 2005.

[54] Noah Snavely, Steven Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. acm trans graph 25(3):835-846. *ACM Trans. Graph.*, 25:835–846, 07 2006.

[55] Pratul P. Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T. Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis, 2020.

[56] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, Yifan Wang, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, Tomas Simon, Christian Theobalt, Matthias Niessner, Jonathan T. Barron, Gordon Wetzstein, Michael Zollhoefer, and Vladislav Golyanik. Advances in neural rendering, 2021.

[57] Eric Veach and Leonidas J. Guibas. Optimally combining sampling techniques for monte carlo rendering. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, page 419–428, New York, NY, USA, 1995. Association for Computing Machinery.

[58] Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, EGSR'07, page 195–206, Goslar, DEU, 2007. Eurographics Association.

[59] Jingwen Wang and Ravi Ramamoorthi. Analytic spherical harmonic coefficients for polygonal area lights. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2018)*, 37(4), 2018.

[60] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction, 2021.

[61] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. Nerf–: Neural radiance fields without known camera parameters, 2021.

[62] Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. Nex: Real-time view synthesis with neural basis expansion, 2021.

[63] Lifan Wu, Guangyan Cai, Shuang Zhao, and Ravi Ramamoorthi. Analytic spherical harmonic gradients for real-time rendering with many polygonal area lights. *ACM Trans. Graph.*, 39(4), aug 2020.

[64] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond, 2021.

[65] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Ronen Basri, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance, 2020.

[66] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields, 2021.

[67] Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. Physg: Inverse rendering with spherical gaussians for physics-based material editing and relighting, 2021.

[68] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields, 2020.

[69] Xiuming Zhang, Sean Fanello, Yun-Ta Tsai, Tiancheng Sun, Tianfan Xue, Rohit Pandey, Sergio Orts-Escolano, Philip Davidson, Christoph Rhemann, Paul Debevec, Jonathan T. Barron, Ravi Ramamoorthi, and William T. Freeman. Neural light transport for relighting and view synthesis. *ACM Transactions on Graphics*, 40(1):1–17, jan 2021.

[70] Xiuming Zhang, Pratul P. Srinivasan, Boyang Deng, Paul Debevec, William T. Freeman, and Jonathan T. Barron. NeRFactor. *ACM Transactions on Graphics*, 40(6):1–18, dec 2021.

[71] Yuanqing Zhang, Jiaming Sun, Xingyi He, Huan Fu, Rongfei Jia, and Xiaowei Zhou. Modeling indirect illumination for inverse rendering, 2022.