Scalable Neural Indoor Scene Rendering

XIUCHAO WU*, State Key Lab of CAD&CG, Zhejiang University, China
JIAMIN XU*, State Key Lab of CAD&CG, Zhejiang University, China
ZIHAN ZHU, State Key Lab of CAD&CG, Zhejiang University, China
HUJUN BAO, State Key Lab of CAD&CG, Zhejiang University, China
QIXING HUANG, University of Texas at Austin, USA
JAMES TOMPKIN, Brown University, USA
WEIWEI XU†, State Key Lab of CAD&CG, Zhejiang University, China

Fig. 1. Neural reconstruction and rendering of large indoor spaces from unstructured photographs is a challenge, especially due to complex reflections on polished surfaces like wood and metal that contribute significantly to the realism of free-viewpoint renderings. Our proposed method enables interactive neural rendering of large indoor scenes with more realistic view-dependent effects than existing methods. Screenshots from interactive renderings of our reconstructed scenes at novel viewpoints: (a) *Living Room* scene from Xu et al. [2021]; note the ceiling spotlight reflection in the floor (top) and the thin light reflection on the ceiling (bottom). (b) *Coffee Shop*; note the reflected coffee machine and surrounding environment in the polished metal door (top). Similar highlights reflected in polished wood, glass, and plastic in (c) *Living Room* scene from INRIA [Philip et al. 2021], (d) *Bar*, and (e) *Sofa* scenes.

We propose a scalable neural scene reconstruction and rendering method to support distributed training and interactive rendering of large indoor scenes. Our representation is based on *tiles*. Tile appearances are trained in parallel through a background sampling strategy that augments each tile with distant scene information via a proxy global mesh. Each tile has two low-capacity MLPs: one for view-independent appearance (diffuse color and shading) and one for view-dependent appearance (specular highlights, reflections). We leverage the phenomena that complex view-dependent scene reflections can be attributed to virtual lights underneath surfaces at the total ray distance to the source. This lets us handle sparse samplings of the input scene where reflection highlights do not always appear consistently in input images. We show interactive free-viewpoint rendering results from five scenes, one of which covers an area of more than 100 m^2. Experimental results show that our method produces higher-quality renderings than a single large-capacity MLP and five recent neural proxy-geometry and voxel-based baseline methods. Project webpage: https://xchaowu.github.io/papers/scalable-nisr.

CCS Concepts: • **Computing methodologies** → **Image-based rendering**.

Additional Key Words and Phrases: Reflection, MLP, Neural Field

ACM Reference Format:
Xiuchao Wu, Jiamin Xu, Zihan Zhu, Hujun Bao, Qixing Huang, James Tompkin, and Weiwei Xu. 2022. Scalable Neural Indoor Scene Rendering. *ACM Trans. Graph.* 41, 4, Article 98 (July 2022), 16 pages. https://doi.org/10.1145/3528223.3530153

*Joint first authors
†Corresponding author

Authors' addresses: Xiuchao Wu, wuxiuchao@zju.edu.cn, State Key Lab of CAD&CG, Zhejiang University, China; Jiamin Xu, superxjm@yeah.net, State Key Lab of CAD&CG, Zhejiang University, China; Zihan Zhu, zihan.zhu@zju.edu.cn, State Key Lab of CAD&CG, Zhejiang University, China; Hujun Bao, bao@cad.zju.edu.cn, State Key Lab of CAD&CG, Zhejiang University, China; Qixing Huang, huangqx@cs.utexas.edu, University of Texas at Austin, USA; James Tompkin, james_tompkin@brown.edu, Brown University, USA; Weiwei Xu, xww@cad.zju.edu.cn, State Key Lab of CAD&CG, Zhejiang University, China.

0730-0301/2022/7-ART98 $15.00
https://doi.org/10.1145/3528223.3530153

1 INTRODUCTION

Reconstructing real-world 3D scenes and rendering them photo-realistically at interactive rates has long been desired in visual computing. Given captured images and reconstructed scene geometry, many image-based rendering (IBR) techniques can synthesize realistic renderings at novel views to enable free-viewpoint navigation [Hedman et al. 2018, 2016; Xu et al. 2021]. Past methods have required high input sample rates or high-quality geometry to avoid ghosting or tearing artifacts within rendered images.

Recently, neural methods have gained popularity in IBR due to their capability to synthesize realistic novel views with view-dependent effects. For instance, NeRF [Mildenhall et al. 2020] optimizes multi-layer perceptron neural networks (MLPs) to output volume radiance and density by rendering views via volume integration along rays. Applying NeRF to model large scenes requires large MLPs to maintain detailed appearance, and requires long rendering times as we must sample 3D points along cast rays and pass them to an MLP to obtain volume radiance and density. Further, given sparse inputs in which view-dependent highlights do not consistently appear, this model fails to reproduce reflections and highlights in reconstructions.

A common strategy to accelerate neural rendering is to trade memory for time. For example, one approach is to explore sparse voxel grids to cache or bake different outputs of the MLP. Examples include deep radiance maps [Garbin et al. 2021], spherical harmonic coefficients [Yu et al. 2021a], and the combination of RGB color, volume density, and features [Hedman et al. 2021]. While these approaches efficiently speed up rendering, memory costs are still high for large 3D scenes, which can be a bottleneck in training that limits scene detail or scale. In KiloNerf [Reiser et al. 2021], thousands of tiny MLPs represent the scene to avoid a high memory footprint. However, this method needs to train a single large-capacity MLP as a teacher to begin training tiny MLPs, which increases training time and hinders its application to large scenes. In summary, applying neural techniques to large scenes remains a technical challenge.

We propose a neural scene rendering method for static indoor scenes that is scalable in both training and rendering. Given an initial reconstructed proxy geometry (the *global mesh*), we partition the 3D space into tiles each with MLPs to avoid training a single large-capacity MLP. Based on this tile-based neural scene representation, we make two key improvements to achieve a scalable solution:

(1) We introduce a background sampling method to enable parallel training of tile-based MLPs. This uses the global mesh to release tile dependence during training by sampling background points at ray intersections with the proxy. To tolerate errors in the proxy geometry, we sample over intervals around the proxy surface. Background sampling allows a tile to be trained on a single GPU, and thus the allocation of tiles to different GPUs in a load-balanced manner with reduced communication overhead. A second training step and post-process CNN refines any tile differences, overall resulting in a more scalable solution.
(2) Each tile has a *surface MLP* to encode view-independent components, such as diffuse colors and shadows, and a *reflection MLP* to encode view-dependent specular reflections. Such a two-MLP representation, termed tMLPs, lets us leverage the phenomena that the view-dependent reflection of a planar surface can be attributed to a reflected view-independent virtual light source. This approach reduces the required number of training images to capture complex appearance highlights. Moreover, tMLPs enable different network capacity, storage, and rendering schemes for view-independent and view-dependent components. We use an octree to store colors and volume densities for the surface MLP, and neural weights with a fixed parameter budget for the reflection MLP. Together, these allow interactive rendering (20 fps) of complex appearance over large scenes.

Over five scenes, we demonstrate that our method can produce realistic interactive rendering results for free-viewpoint navigation of indoor scenes of more than 100 m^2. Our method can automatically handle opaque surfaces with reflections and transparent surfaces that have an opaque surface behind them within a tile, like glass-fronted cabinets. In comparisons to state-of-the-art neural rendering methods, our rendering results of indoor scenes with complex view-dependent effects show improved quality, especially in the reproduction of highlights and reflections.

2 RELATED WORK

Image-based rendering (IBR) has a long history in visual computing [Gortler et al. 1996; Levoy and Hanrahan 1996; Shum and Kang 2000; Zhang and Chen 2003]. Recently, IBR results show significant improvements thanks to neural rendering. First, we provide a brief overview of this current topic, then we focus on IBR works that are most relevant to our approach.

2.1 Neural Rendering

Neural rendering aims to incorporate neural networks into the pipeline for rendering high quality images [Tewari et al. 2020, 2021]. Applications include volumetric novel view synthesis [Lombardi et al. 2019; Pandey et al. 2019; Sitzmann et al. 2019a], local light field fusion [Mildenhall et al. 2019], neural textures [Thies et al. 2019], semantic photo manipulation [Bau et al. 2020; Karras et al. 2019; Park et al. 2019b], relighting [Meka et al. 2019; Philip et al. 2019; Zhou et al. 2019], and animatable photo-realistic avatars [Liu et al. 2021, 2019; Peng et al. 2021a,b]. One recent example is Mixtures of Volumetric Primitives [Lombardi et al. 2021]. This approach employs a set of volumetric primitives to represent a scene or object, creating high-quality real-time renderings of challenging materials such as hair and clothing. Our work also partitions the 3D space into regions; however, we focus on rendering indoor scenes with reflections.

2.2 IBR with Geometric Proxies

Geometric proxies are frequently exploited in IBR to reduce the number of captured images required and resolve the ambiguity of cross-image correspondence [Gortler et al. 1996]. Debevec et al. [1998] proposed novel view synthesis by view-dependent texture mapping with manually constructed geometric proxies. View-dependent texturing is widely used in IBR [Buehler et al. 2001; Chen and Williams 1993; Matusik et al. 2000, 2002; Miller and Rubin 1998; Wood et al. 2000]. However, these approaches often produce ghosting artifacts due to geometry reconstruction errors, especially near occlusion boundaries. Several approaches seek to reduce these artifacts using

optical flow correction [Casas et al. 2015; Du et al. 2019; Eisemann et al. 2008], per-view refinements that align geometric and image boundaries [Chaurasia et al. 2013; Hedman et al. 2018, 2016; Xu et al. 2021], or soft scene forms [Penner and Zhang 2017]. To further reduce ghosting and aliasing artifacts, DeepBlending [Hedman et al. 2018] proposed to train a CNN to predict adaptive per-pixel blending weights, and Xu et al. [2021] employed a post-processing network to perform temporal super-sampling.

Another line of work [Kopf et al. 2013; Rodriguez et al. 2020; Sinha et al. 2012; Xu et al. 2021] aims to use two-layer representations to separate the surface and reflection components of appearance for improved view-dependent effects. Along this line, multi-layer representations, such as Multi-plane Image (MPI) or Layered Depth Image (LDI) from input images [Flynn et al. 2019; Srinivasan et al. 2019; Szeliski and Golland 1999; Zhou et al. 2018], were proposed to handle occlusions and reflection effects simultaneously.

View interpolation can also exploit learned features. Riegler et al. [2020] map features extracted from input images to object surfaces and leverage a recurrent network to generate novel views. Rendering quality can improve through view-dependent on-surface feature aggregation [Riegler and Koltun 2021]. Philip et al. [2021] proposed a hybrid image- and physically-based rendering algorithm that enables both the navigation and relighting of indoor scenes.

This paper focuses on representations and algorithms that address challenges in scaling-up neural view-synthesis techniques to large indoor scenes while still representing reflections.

2.3 IBR with Neural Fields

This line of research advocates optimizing neural networks to estimate fields that represent the geometry and appearance of objects [Niemeyer et al. 2019; Park et al. 2019a; Sitzmann et al. 2019b; Xie et al. 2022]. Neural networks take point coordinates as input and output quantities like volume density, occupancy, signed distance, or radiance. For example, neural radiance fields (NeRF) [Mildenhall et al. 2020] is an MLP-based scene representation of volume density and view-dependent color per 3D location that can provide photo-realistic rendering. Many follow-up works improve robustness, quality [Barron et al. 2021; Martin-Brualla et al. 2021; Verbin et al. 2022; Zhang et al. 2020], and generalization [Chan et al. 2021; Rematas et al. 2021; Schwarz et al. 2020; Trevithick and Yang 2020; Yu et al. 2021b], optimize camera poses [Lin et al. 2021; Sucar et al. 2021; Wang et al. 2021; Yen-Chen et al. 2021; Zhu et al. 2022], and apply to dynamic scenes [Attal et al. 2021; Gafni et al. 2021; Park et al. 2020, 2021; Pumarola et al. 2021].

One major issue is that both training and rendering of neural fields require substantial compute, and different methods can accelerate both. Examples include avoiding samples in vacant areas [Liu et al. 2020a], dividing the 3D space into thousands of small regions and training a tiny MLP for each region [Reiser et al. 2021], caching or baking values learned by the MLP into a representation [Garbin et al. 2021; Hedman et al. 2021; Wizadwongsa et al. 2021; Yu et al. 2021a], optimizing a representation directly without an MLP [Sun et al. 2021; Yu et al. 2022]; and pre-computing values during volume rendering [Lindell et al. 2021; Sitzmann et al. 2021].

Several recent methods have proposed neural representations for the relighting and rendering of 3D scenes with complex appearance. These methods go beyond the absorption-emission volumetric rendering model [Max 1995]. NeRFReN [Guo et al. 2022] models complex reflections by splitting a scene into transmitted and reflected components with separate neural radiance fields. Ref-NeRF [Verbin et al. 2022] trains a spatial MLP to output diffuse colors and surface normals, then uses these normals with a reflected ray direction to reproduce high-frequency specular reflections via a directional MLP. Neural reflectance fields [Bi et al. 2020] store volume density, surface normals, and bi-directional reflectance distribution functions (BRDFs), allowing rendering under arbitrary lighting conditions. PhySG [Zhang et al. 2021a] and NeRD [Boss et al. 2021] use mixtures of spherical Gaussians to represent environment lighting and scene BRDFs. NeRV [Srinivasan et al. 2021] employs an MLP to directly output the visibility from a 2D light direction to a 3D point, enabling shadows and self-occlusion effects. Finally, NeRFactor [Zhang et al. 2021b] uses data-driven bidirectional reflectance distribution functions (BRDF) priors from real-world BRDF measurements [Matusik et al. 2003] and smoothness regularizations to recover neural fields of surface normals, light visibility, albedo, and BRDFs.

While these works can produce photo-realistic view-dependent renderings, and some allow interactive or real-time rendering, they typically do not apply to large scenes. We provide a scalable solution for indoor scene rendering by designing a tile-based MLP with surface and reflected appearance components that supports parallel training and adaptive storage allocation. Our solution achieves 20fps on a workstation PC when rendering large indoor scenes.

3 SCALABLE NEURAL SCENE REPRESENTATION

When using MLPs, the architecture and capacity determine the efficiency and quality of the rendering. Larger scenes require larger networks to achieve equivalent quality, and require more time in volume rendering to integrate points over larger distances. Our goal is to reduce network size without sacrificing quality such that neural rendering can scale to large scenes both during optimization and for interactive rendering.

We propose two approaches to reach this goal (Fig. 2): First, we create tiles $\mathcal{T}$ over the volumetric scene and optimize per-tile MLPs. This has two benefits:

(1) Tile MLPs are locally low capacity. This allows faster optimization than one large scene network, but still provides the ability to represent a complex scene collectively.
(2) Tiles can be trained in parallel using a new background sampling method. Further, along a specific ray, we can coordinate 3D point samplings across MLPs for efficiency.

Second, rather than encode radiance absolutely at each point and in each direction in the volume [Mildenhall et al. 2020], we use two MLPs—tMLPs—for view-independent reflection $\mathbf{c}^s$ (e.g., diffuse) and additive view-dependent reflection $\mathbf{c}^r$ (e.g., specular) such that final color $\mathbf{c} = \mathbf{c}^s + \mathbf{c}^r$. This has three benefits:

(3) Once view-independent MLPs have been optimized, we can sample them and *bake* them for classical fast rendering. For this, we use a voxel octree of depth 2, though voxel hashing approaches are also possible [Nießner et al. 2013].

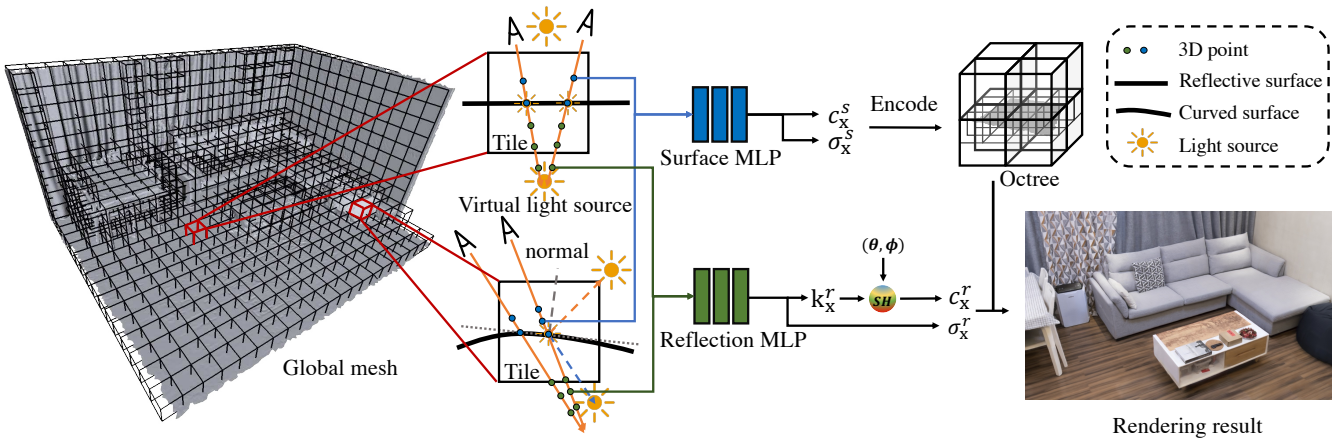


Fig. 2. *Scalable neural scene representation.* Tiles are allocated based on a global mesh proxy. Each tile has two MLPs: 1) The *surface MLP* that encodes density and view independent color, which is later stored in an octree for fast rendering. 2) The *reflection MLP* that encodes view-dependent effects like highlights using virtual points 'underneath' the surface at the ray distance of the reflected light. Color outputs from both paths are combined in the final rendering.

(4) Capturing specular reflections accurately requires high angular sampling in the input images, which is not practical for large inside-out real-world scenes. Separating reflections allows us to represent them as virtual points that lie underneath surfaces at the distance of the *reflected geometry*. For example, the glossy reflection of a ceiling light on a polished wooden floor is represented by virtual points beneath the wooden floor. This reduces the need for high angular input sampling on planar surfaces that are common in indoor environments. It is also beneficial on rough or slightly curved convex surfaces, as reflections can be clustered virtually underneath the surface geometry.
(5) Both previous benefits allow view-dependent MLPs for specular reflections to again be lower capacity, easing optimization and hastening rendering.

In the remainder of Section 3, we describe the input, representation, rendering, and reconstruction optimization losses without any practical implementation details. Then, in Section 4, we describe how tMLPs are practically constructed, and in Section 5 how they are practically rendered including a CNN-based post-processing.

3.1 Input and Preprocessing

Throughout the document, we use $\hat{\cdot}$ to denote input images, patches, and colors, as contrasted with their reconstructed counterparts. Given a set of images $\hat{I} \in \hat{\mathcal{I}}$ of an indoor scene, we reconstruct camera intrinsic and extrinsic matrices. This let us define a set of rays $\hat{\mathbf{r}} \in \hat{\mathcal{R}}$ each with an origin $\mathbf{o}$, a direction $\boldsymbol{\omega}$, and a color $\hat{\mathbf{c}}$ from an input pixel. We also assume a *global mesh* reconstruction of the scene generated by existing software [CapturingReality 2016; Schonberger and Frahm 2016] (See Section 7.1).

To help separate view dependent from view independent reflections, for each input ray we compute a view independent color $\bar{\mathbf{c}}$. As specular reflections such as highlights vary surface appearance when a point is viewed from different angles, with sufficient input views we can remove specular reflections: Views with large angular difference to a reference view likely do not contain highlights, so weighting the many appearances of a world point by their viewing directions will remove specular highlights.

Given a reference view ray $\hat{\mathbf{r}}_i$, we intersect it with the global mesh, then warp all images containing views of that point into the reference view to create rays $\mathbf{r}_j$. Then, view independent color $\bar{\mathbf{c}}$ is:

$$\bar{\mathbf{c}}_i = \sum_j^N \frac{w_j}{\sum_j^N w_j} \hat{\mathbf{c}}_j, \quad \text{where } w_j = 1 - \boldsymbol{\omega}_j \cdot \boldsymbol{\omega}_i, \tag{1}$$

where N is the number of views of the point, $\hat{\mathbf{c}}_j$ is the pixel color warped from the jth view via the global mesh, and weight w_j prefers rays with a large angle difference to ith ray. Computing Eq. 1 for all input pixels produces a second set of images where view-dependent effects like specular reflections have largely been removed.

While using the minimum or the second darkest color from all view directions is one strategy to obtain diffuse colors, minor inaccuracies in the global mesh and camera poses create slight edge misalignments between an image and the geometry and lead to noise from occlusion errors. In practice, we found our weighting scheme was more robust to noise from these errors.

3.2 Representation and Rendering

Representation. We define a 3D point as input to our neural fields as $\mathbf{x} = \{x, y, z\}$. For view-independent reflection, we define a *surface MLP* $f_\theta^s : \mathbb{R}^3 \rightarrow \mathbb{R}^4$ with parameters θ that outputs diffuse color $\mathbf{c}^s$ and volume density σ^s: $f_\theta^s(\mathbf{x}) = (\mathbf{c}^s, \sigma^s)$. These points lie in a space on or in front of the surface of the global mesh.

For the view-dependent reflection, we define a second *reflection MLP* $f_\theta^r : \mathbb{R}^5 \rightarrow \mathbb{R}^{28}$ with respect to a point $\mathbf{x}$ and a ray direction $\boldsymbol{\omega}$. These points lie in a space behind the surface of the global mesh and represent reflected light at a virtual location (Fig. 2). Along with density, rather than output view-dependent color $\mathbf{c}^r$, function f_θ^r outputs spherical harmonic (SH) coefficients $\mathbf{k}^r$ [Müller 1966] to represent view-dependent reflection in all directions at point $\mathbf{x}$:

$$f_\theta^r(\mathbf{x}) = (\mathbf{k}^r, \sigma^r), \quad \mathbf{k}^r = (k_l^m)_{l:0 \le l \le l_{max}}^{m:-l \le m \le l}, \tag{2}$$

Fig. 3. *Sparse sampling causes reflection reconstruction problems for simpler representations.* Over 27 input images of the framed painting in the *Bar* scene, only 8 contain the reflections of a lamp in the scene—a typical scenario for inside-out indoor scenes. A single MLP that uses SH coefficients at the surface with an equivalent number of network parameters fails to reconstruct the highlight, whereas our tMLPs approach of explicitly modeling reflections at virtual distances fairs better.

where l and m are the degree and order of the SH functions. The maximum degree of SH functions l_{max} in our system is set to 2 to balance between quality and speed. To compute the view-dependent color $\mathbf{c}^r$ with respect to ray direction $\boldsymbol{\omega}$:

$$\mathbf{c}^r = g(\mathbf{k}_r, \boldsymbol{\omega}) = S\left(\sum_{l=0}^{l_{max}} \sum_{m=-l}^{l} k_l^m Y_l^m(\boldsymbol{\omega})\right), \tag{3}$$

Where $Y_l^m : \mathbb{S}^2 \rightarrow \mathbb{R}$ are SH basis functions; $S(\cdot)$ is the sigmoid function as the interval of color values is normalized to be $[0..1]$. We use SHs for each RGB channel, so $\mathbf{k}^r$ is a 27-dimensional vector.

Rendering tMLPs by Volume Ray Casting. Given a camera pose, we define a point t units along a viewing ray: $r(t) = \mathbf{x} = \mathbf{o} + t\boldsymbol{\omega}$, where $\mathbf{o}$ is the ray origin (camera center) and $\boldsymbol{\omega}$ is the ray direction. Across tiles, we sample points along the ray and sum the color outputs $\mathbf{c}^s \in [0,1]^3$ and $\mathbf{c}^r \in [0,1]^3$ from both f_θ^s and f_θ^r:

$$\mathbf{c}^s = \sum_i^N T_i \cdot (1 - \exp(-\sigma_i^s \delta_i)) \cdot \mathbf{c}_{\mathbf{x}_i}^s, \tag{4}$$

$$\mathbf{c}^r = \sum_i^M T_i \cdot (1 - \exp(-\sigma_i^r \delta_i)) \cdot \mathbf{c}_{\mathbf{x}_i}^r, \tag{5}$$

$$\mathbf{c} = \max(\min(\mathbf{c}^s + \mathbf{c}^r, 1), 0), \tag{6}$$

where $T_i = \exp(-\sum_{k=1}^{i-1} \sigma_{\mathbf{x}_k} \delta_k)$ is the transmittance value, δ_i is the distance between adjacent samples (for σ^s, σ^r equivalently), and M, N are the number of sampled points along the ray.

Discussion. Point lights represented at reflected distances from planar surfaces need not be view dependent, making the SH representation seem redundant. However, while salient reflections in indoor scenes frequently appear on planar surfaces like floors, ceilings, and tables, many reflective surfaces are not perfectly flat. For reflections of light sources that themselves are view dependent or for slightly non-planar or curved surfaces with reflected highlights, position-dependent reflections vary with the surface normal. Using SH coefficients lets us represent some of this variation (Fig. 4).

Fig. 4. *SH reflection representation helps produce more accurate highlights. Top:* Training a model for virtual reflected lights underneath the global mesh surface that exhibit view-independent reflectance (equivalent to 0th-order SH) fails to reproduce any specular highlights on curved surfaces, while our approach fares better. *Bottom:* On planar reflective surfaces like a reflective floor, training with 0th-order SH can in principle capture view-independent reflectance, but the quality degrades.

While predicting SH coordinates at each location is similar to PlenOctree [Yu et al. 2021a], our method stores reflection MLP weights instead of a SH coefficient octree to reduce memory use.

3.3 Reconstruction Losses

To optimize the representation, we minimize a color loss $\mathcal{L}_c$ between the rendered ray color $\mathbf{c}_i$ and the captured image color $\hat{\mathbf{c}}_i$:

$$\mathcal{L}_c = \frac{1}{N} \sum_{i \in N} ||\hat{\mathbf{c}}_i - \mathbf{c}_i||_2^2 \tag{7}$$

where i is the ray index and N is the number of rays in patch $p \in \mathcal{P}$. We organize neighboring rays during training into 91×91 patches because this lets us penalize a structural similarity (SSIM) loss $\mathcal{L}_{\text{SSIM}}$ [Wang et al. 2004] and a VGG perceptual loss $\mathcal{L}_{\text{VGG}}$ [Johnson et al. 2016] against captured image patches $\hat{p}$:

$$\mathcal{L}_{\text{SSIM}} = 1 - \text{SSIM}(\hat{p}, p), \tag{8}$$

$$\mathcal{L}_{\text{VGG}} = \|v(\hat{p}) - v(p)\|^2, \tag{9}$$

where v computes activations using the first three layers of a VGG network [Simonyan and Zisserman 2014].

To obtain view-independent colors from captured images, we formulate the surface color loss term $\mathcal{L}_s$ between the color output by the surface MLP $\mathbf{c}^s$ and the preprocessed weighted average input ray color $\bar{\mathbf{c}}$ that has reflection effects removed:

$$\mathcal{L}_s = \frac{1}{N} \sum_{i \in N} ||\bar{\mathbf{c}}_i - \mathbf{c}_i^s||_2^2. \tag{10}$$

Finally, noise in reconstructed camera poses can lead to random color and densities at some 3D points. This phenomenon is magnified for reflections since our virtual light sources are at reflected

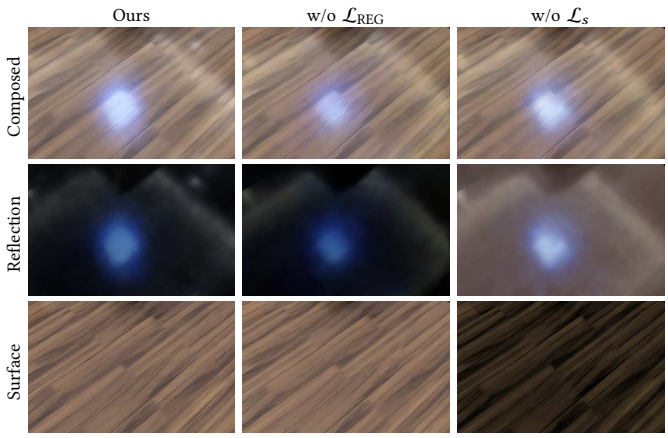


Fig. 5. *Penalizing additional SH reguarlizer $\mathcal{L}_{REG}$ and surface color loss $\mathcal{L}_s$ improves separation of view independent and dependent reflection.* Results show the reflection of a ceiling spotlight on a polished wooden floor in *Living Room from Xu et al.* [2021].

distances from the view point. To alleviate this issue, we add a regularization loss $\mathcal{L}_{\text{REG}}$ to encourage non-0-th order SH coefficients output by reflection MLP to only be used when necessary:

$$\mathcal{L}_{\text{REG}} = \frac{1}{N_r}\sum_{j}^{N_r}(\mathbf{k}^r_{\mathbf{x}_j})^2, \quad \mathbf{k}^r_{\mathbf{x}} = k_l^m(x)_{l:1\le l\le l_{max}}^{m:-l\le m\le l}, \tag{11}$$

where N_r indicates the number of sampled 3D points along the ray, and subscript j indexes the points. This slightly penalizes the view dependence of the reflections, without penalizing the overall intensity (regularizing 0-th order SHs would darken reflections).

The total loss used in training is:

$$\mathcal{L} = \lambda_c\mathcal{L}_c + \lambda_{\text{SSIM}}\mathcal{L}_{\text{SSIM}} + \lambda_{\text{VGG}}\mathcal{L}_{\text{VGG}} + \lambda_s\mathcal{L}_s + \lambda_{\text{REG}}\mathcal{L}_{\text{REG}} \tag{12}$$

where $\lambda_c, \lambda_{\text{SSIM}}, \lambda_{\text{VGG}}, \lambda_s$ and λ_{REG} are determined empirically.

4 TILE-BASED SCENE CONSTRUCTION

While the representation and reconstruction losses are relatively straightforward, scaling volume rendering through tiles is practically challenging as integrating radiance along a ray requires points from across tiles. This section describes how we tackle three technical issues in tile-based MLP scene construction: removing dependence between tiles with *background sampling* to allow parallel tile training, accommodating sparse input images and sparse light sources with *reflection tile groups*, and removing tile boundary artifacts with *two-pass training*. We also discuss pre-processing and post-processing to help overcome these challenges.

4.1 Pre-processing

Volume Tiling. We divide the global mesh's bounds into $30 \times 30 \times 30$ cm tiles, based on a scale estimated from a known scene object. Then, we cull tiles that do not intersect any global mesh per-triangle bounding box. We expand bounding boxes by ≈ 1 cm to avoid culling errors due to minor reconstructed geometry errors or possible missing geometries. Finally, we store the indices of global mesh triangles inside a tile to speed up later computations.

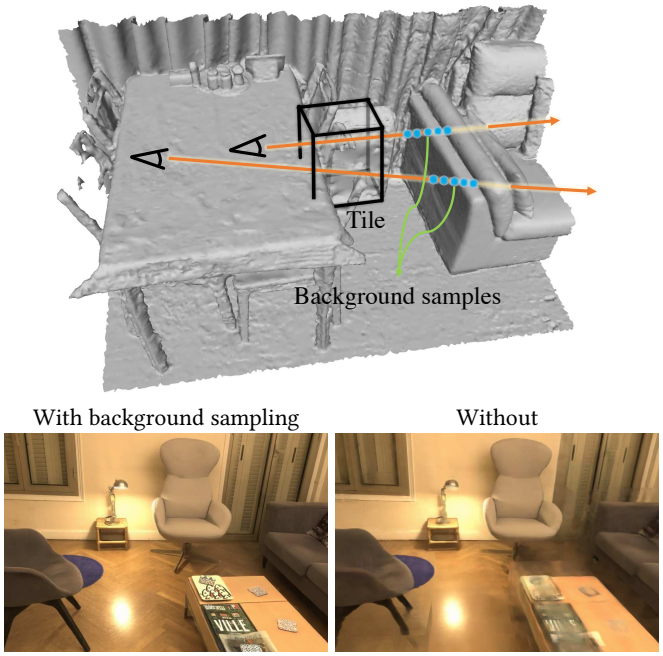


Fig. 6. *Background sampling enables parallel tMLP training. Top:* If a ray does not terminate within a tile, we additionally sample 32 points along the ray where it intersects the global mesh to reproduce world points that affect a ray's integral. This removes the dependency between tiles at training time. *Bottom:* Background sampling also improves reconstruction accuracy because it allows an individual tile to disambiguate the location of radiance within the volume. Without background sampling, we see blurring from phantom volume density; with, we see improved surface texture on the wooden floor and table books (*Living Room from Inria* [Philip et al. 2021]).

Assigning Input Rays to Tiles. For each input ray r, we obtain a set of candidate tiles $\mathcal{T}^c$ using fast voxel traversal [Amanatides et al. 1987]. Within each candidate tile, we intersect the ray with all global mesh triangles of the tile and record the intersection point p that is closest to the camera, if any. If a tile contains p or is closer to the camera than p, then r is assigned as a training ray for that tile. Tiles farther than p are not trained by r. As stated earlier, this assumes that scene objects are opaque or that transparent objects like glass lie within a tile in front of an opaque surface within the global mesh, such as for a window onto another building (*Living Room* from Xu et al. [2021]; kitchen area) or a glass display cabinet (*Coffee Shop*; please see Appendix A). For objects that do not appear in the global mesh but that require volumetric reconstruction, we mark their tiles for inclusion or add proxy geometry (Appendix B).

4.2 Background Sampling for Parallel Tile Training

Each ray cast might pass through multiple tiles before termination, creating dependencies between tMLPs that require more memory and compute or inefficient access patterns during training. To avoid

tile dependence and allow tMLPs to be trained in parallel and across multiple GPUs, we deliberately sample and reproduce 3D points beyond a tile. We call this *background sampling* (Fig. 6).

Background sampling handles the case when a ray intersecting a tile terminates outside the tile; that is, the ray intersects the global mesh at point $\mathbf{x}$ outside the tile. We sample 32 additional points $\mathbf{x}_b \in \mathcal{B}$ along the ray, centered at $\mathbf{x}$ and over a half-tile range to tolerate any global mesh reconstruction inaccuracy. The sampled points $\mathcal{B}$ are sent to the surface MLP to output color and density values; thus, tiles also reconstruct their background for all assigned non-terminating rays passing through them. This encourages the surface MLP to output color $\mathbf{c}^s = 0$ and density $\sigma^s = 0$ for the empty space within a tile.

The output surface color $\mathbf{c}^s$ with additional background sampling step is the sum of the within-tile color $\mathbf{c}_t^s$ and background color $\mathbf{c}_b^s$:

$$\mathbf{c}^s = \mathbf{c}_t^s + \mathbf{c}_b^s, \tag{13}$$

$$\mathbf{c}_t^s = \sum_i^N T_i \cdot (1 - \exp(-\sigma_{\mathbf{x}_i}^t \delta_i)) \cdot \mathbf{c}_{\mathbf{x}_i}^t, \tag{14}$$

$$\mathbf{c}_b^s = T_t \cdot \sum_i^{|\mathcal{B}|} T_i \cdot (1 - \exp(-\sigma_{\mathbf{x}_i}^b \delta_i)) \cdot \mathbf{c}_{\mathbf{x}_i}^b, \tag{15}$$

where T_t is the remaining transparency value after sampling the within-tile 3D points. Note that reflection color $\mathbf{c}^r$ is also added to produce the final ray color (Eq. 6). Thus, background sampling also influences the training of the reflection MLPs.

Background sampling is not necessary for rays that terminate inside a tile. However, due to possible erroneous floating or phantom geometry in the global mesh, we still perform background sampling for these rays as it helps to ignore erroneous geometry. For cases where a ray does not intersect the global mesh at all, such as with missing geometry, background sampling is not possible and $\mathbf{c}_b^s = 0$.

4.3 Reflection Tile Groups for Sparse Images and Lights

In indoor scenes, light sources that create reflections are likely to be sparse; we also wish scene image sampling to be sparse too to reduce capture workload. This leads to situations where rays from input images may miss highlights entirely (Fig. 7) and fail to associate them to tiles, causing missing highlights in renderings. We overcome this by sharing reflection MLP weights between groups of neighbouring tiles. For instance, planar regions are likely to share reflection highlights caused by the same virtual light sources—e.g., all tiles that cover a large polished table have highlights from the same ceiling light. Grouping also reduces the number of reflection MLPs during training and rendering, increasing speed.

Our approach has three steps:

(1) We apply variational shape approximation (VSA) [Cohen-Steiner et al. 2004] to cluster connected co-planar triangles in the global mesh. For this, we use the L2 distortion error between each cluster and a planar proxy to segment the global mesh into different clusters of co-planar triangles (Fig. 8).

(2) Any tiles that intersect any triangle in a co-planar cluster from VSA form a set of tiles $\mathcal{T}^s$.

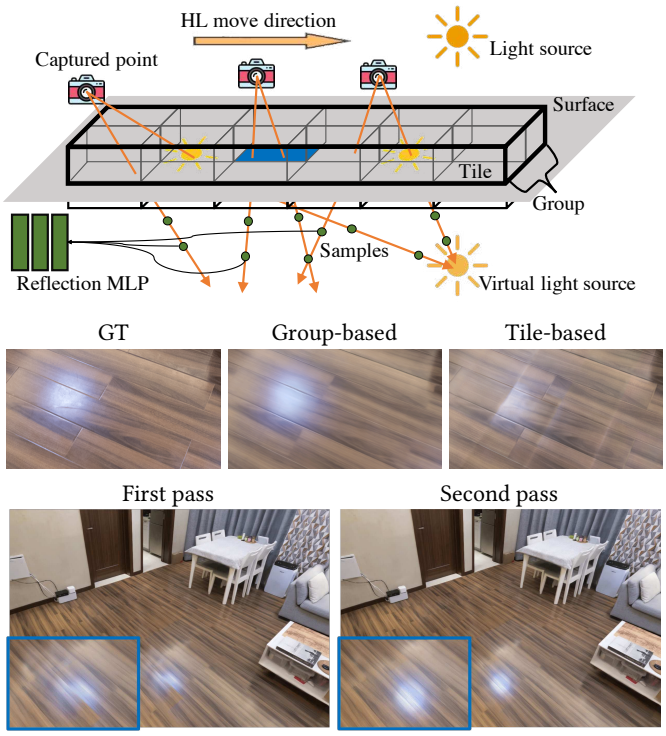


Fig. 7. *Reflection groups and two-pass training. Top*: Reproducing highlight reflections is difficult when captured images are sparse, as they may not associate a highlight with a tile. Suppose the gray surface is a polished table reflecting an overhead light. The blue tile in the middle of the table is only captured by rays that do not see the reflection, even though the reflection should appear on the surface represented by the blue tile when viewed from a slightly different viewpoint. To overcome this, we use weight sharing across tiles to group reflection MLPs. *Middle*: Training reflection MLPs per tile leads to incomplete specular reflection renderings. This is significantly improved by grouping tiles. *Bottom:* Rendering results are produced through a two-pass training strategy that also removes boundary artifacts between tiles. Novel rendered view from the *Living Room from Xu et al.* [2021] scene.

(3) Co-planar regions can be large, such as the floor or ceiling, which would restrict the quality of our reflections if it were covered by only one reflection MLP. Instead, we partition sets of co-planar tiles spatially by applying the k-means algorithm to the center positions of each set of tiles $\mathcal{T}^s$. This produces smaller final reflection group tiles $\mathcal{T}^g$ that can be represented by lower-capacity MLPs. We set $k = |\mathcal{T}^s|/40$.

This process recreates reflections more accurately than per-tile reflection MLPs and reduces training and rendering time (Fig. 7).

4.4 Two-pass Training to Remove Boundary Artifacts

Background sampling allows tile surface MLPs to be trained in parallel independently; however, this can lead to discontinuity artifacts in rendered images along tile boundaries. Further, tiles are also grouped to share reflection MLPs, which requires training on the same GPU to reduce memory access costs.

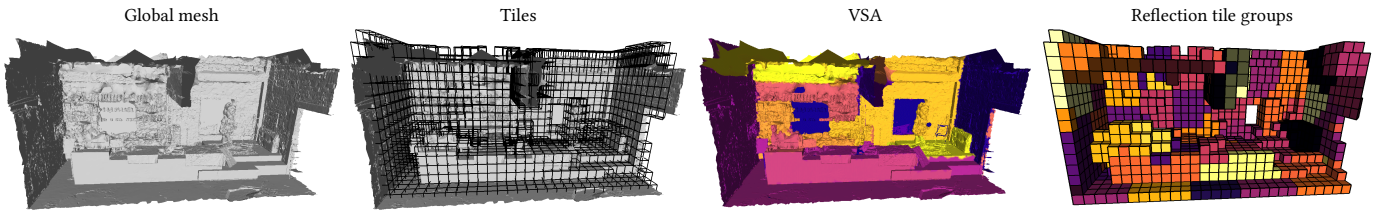


Fig. 8. *Reflection grouping results after using variational shape approximation (VSA) [Cohen-Steiner et al. 2004].* We show the global mesh, the tiles allocation, the VSA clusters, and final reflection tile grouping results.

To overcome these two issues, our training routine has two passes. First, each tile's tMLPs are trained in parallel among GPUs by using the loss in Equation 12, with the surface MLP trained first for 10 epochs as a bootstrap. We observe that the view-independent surface color is mostly captured after this pass. Second, we perform cross-group training to reduce discontinuity artifacts at tile boundaries. We train tile reflection MLPs using neighboring tile surface MLPs to encourage consistent reflections among tiles. We find neighboring tiles for tile groups $\mathcal{T}^g$ via morphological dilation. Then, we fix all surface MLPs parameters, and train the reflection MLP of $\mathcal{T}^g$ using our standard training losses. This strategy produces smoother transitions of reflections across tile boundaries (Fig. 7).

4.5 Ray Sampling, Octrees, and Voxel Pruning

Surface MLPs. To reduce the computational cost of rendering high-resolution images using neural fields, we use an octree to store the output of the surface MLP. For this, during training, each tile is subdivided into $64 \times 64 \times 64$ voxels that store color and volume density values at their center (Fig. 9). We train the surface MLP for a tile via the voxel data structure: we sample 3D points along a ray within the tile, where each point is formed by trilinear interpolation of nearby voxels with values predicted by the surface MLP.

We fix the number of sampled 3D points to 64 during training. However, to improve sampling locality and accelerate training, we perform voxel pruning after every 10 training epochs according to the predicted volume density [Liu et al. 2020a]: We remove voxels with low volume density σ or low accumulated transparency T. This is because small σ may indicate empty space, and small T along each ray in the training data may indicate an invisible voxel. After pruning, we resample the 64 points along the remaining line segments of the ray that still intersect voxels. Over training, this concentrates sample points at high density areas within the tile.

To prune, first we group tile voxels into blocks: Voxels with index (i, j, k) form a block if the values $(\lfloor i/G \rfloor, \lfloor j/G \rfloor, \lfloor k/G \rfloor$ are the same. Hence, a block contains G^3 voxels. We conservatively prune the block if every voxel inside a block has a density less than a threshold:

$$\min_{j=1,\ldots,G^3} \sigma_j < -\log(\gamma) \tag{16}$$

where σ_j are voxel volume densities in the block.

Some voxels are occluded from all input views and so cannot be pruned using the previous method. For example, voxels within a tile but behind an opaque surface are occluded, but may still have large density from MLP initialization. As such, after volume-density-based pruning, we scatter transparency values computed for sampled 3D points along each training ray to voxel centers using trilinear interpolation. If the maximum transparency value at a voxel center is less than a threshold τ, we prune the voxel. This effectively prunes occluded voxels because they do not accumulate significant transparency as rays are sampled.

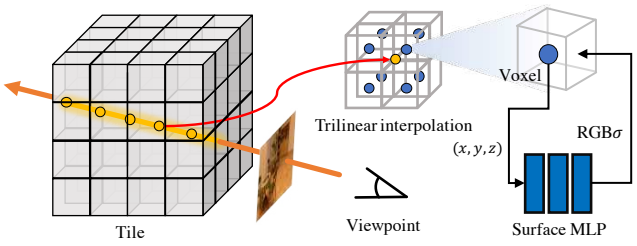


Fig. 9. *Voxel parameterization within each tile.* Each 3D sample point receives its color and volume density value by feeding the nearby voxel centers into the surface MLP and performing trilinear interpolation on the MLP outputs.

Reflection MLPs. Reflection samples in the volume are located 'underneath' surfaces in the global mesh. Although the global mesh provides a proxy for the true surface location within each tile, inaccuracies such as floating or missing geometries might influence the sampled 3D points. Thus, we start reflection MLP point sampling when a ray first intersects an unpruned tile voxel. At each world point, the reflection might contain the entire scene. For instance, in Fig. 1b (top), the whole *Coffee Shop* is reflected in a polished metal door. As such, we set the ray length for reflection sampling to be the diagonal length of the global mesh bounding box, and we uniformly sample 64 points using $1/t$ as the sampling space.

Storage. After the training is finished, we construct an octree of depth two for each tile by setting the un-pruned voxels as leaf nodes. We chose a depth of two to balance memory size and leaf-node access cost. We do not store reflection MLP SH coefficients within the octree as this would significantly increase our memory footprint [Yu et al. 2021a]; instead, the weights of reflection MLPs are stored independently and queried through their tile association.

In summary, we sample 3D points for surface and reflection MLPs respectively during training. Combining these 3D points with background sampling points lets each tMLPs be trained in parallel.

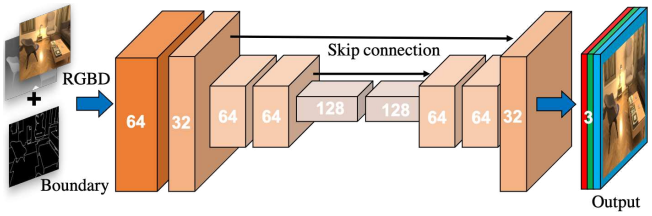


Fig. 10. *Post-process CNN architecture.* It uses 3×3 kernels for all layers.

5 RENDERING

Our tMLPs representation lets us store all scene data in GPU memory to ease a CUDA rendering implementation. To render an image at a viewpoint, we trace rays through image plane pixels in parallel and determine their associated tiles using fast voxel traversal [Amanatides et al. 1987]. Then, we render all rays in parallel using tile octrees to obtain a surface color image I^s. For reflections, we determine each tile's reflection group MLP, and then similarly compute the reflection color for each ray in parallel to obtain the reflection image I^r. Then, we add I^s and I^r. As rendering I^r is considerably more expensive than rendering I^s, this separation lets us balance computation time and quality by rendering I^r at a reduced resolution and then upsampling it.

Render I^s with Octrees. Each CUDA thread is responsible for a single rendered ray. For each tile intersected, we recursively search its octree for non-empty children to determine the intersected leaf nodes. Then, we ray march points along the ray within leaf voxels with step length $0.2 \times l$, where l is the side-length of the voxel. The pixel color of each ray is computed using Eq. 4.

Render I^r with Reflection MLPs. Each CUDA block is responsible for a different reflection group MLP, and each CUDA thread is responsible for a set of ordered 3D points sampled along a ray that intersects a tile associated with the reflection group. We store reflection MLP weights in shared GPU memory (48-99 KB on NVIDIA 3090) using the smaller float16 format: As shared GPU memory is 100× faster than global GPU memory, this makes real-time reflection rendering possible. We implement a GPU kernel to obtain the reflection color of each ray using Eq. 5.

RGBD + Boundary CNN Post-processing. Rendering shallow octrees and reduced-resolution reflections can result in slightly blurred images. To overcome this, we use an inexpensive feed-forward CNN as a post-process to improve detail and reduce artifacts such as residual tile inconsistencies. The CNN takes as input an RGB image and a depth image from the novel viewpoint. During rendering of I^s, we create the depth image by replacing RGB colors in Eq. 4 with depth values. We additionally input a tile boundary image to provide more signal to the CNN about potential inconsistencies. This is created by finding edges in images of the tile reflection group index.

The network is trained using L2, SSIM, and VGG losses:

$$\mathcal{L}_{\text{CNN}} = \mathcal{L}_{\text{RGB}} + \lambda_{\text{BOUND}}\mathcal{L}_{\text{BOUND}} + \mathcal{L}_{\text{SSIM}} + \lambda_{\text{VGG}}\mathcal{L}_{\text{VGG}} \quad (17)$$

where $\mathcal{L}_{\text{RGBD}}$, $\mathcal{L}_{\text{BOUND}}$ are mean square error for RGB and boundary images, and $\mathcal{L}_{\text{SSIM}}$, $\mathcal{L}_{\text{VGG}}$ are as defined previously.

Table 1. *Scene Information.* #Img denotes the total number of captured images of the scene. Total/Octree Storage (GB) denotes the storage needed in total and the storage for octree. Living Room[1] denotes *Living Room from Inria* [Philip et al. 2021], and Living Room[2] denotes *Living Room from Xu et al.* [2021]. N/A means area information was not provided [Philip et al. 2021].

Scene	Area (m^2)	#Img	Total/Octree Storage (GB)	Training (day) tMLPs + CNN
Sofa	N/A	283	2.912 / 2.909	0.7 + 0.2
Living Room[1]	N/A	322	3.698 / 3.597	1.0 + 0.2
Living Room[2]	8.2 × 6.3	2289	5.074 / 5.068	3.0 + 0.5
Bar	8.0 × 10.0	1323	13.062 / 13.050	6.1 + 0.5
Coffee Shop	8.3 × 16.7	1288	14.136 / 14.128	6.5 + 0.5

6 PARAMETERS AND TRAINING ROUTINES

tMLP Architecture. For the surface MLP, we use 8 fully-connected layers of size 128. As in Midenhall et al. [2020], we use 10 frequency bands for the positional encoding. For the reflection MLP, we use 8 fully-connected layers of size 64, which can be stored in the shared memory of kernel blocks on modern GPUs for efficient rendering.

tMLP Training. We use the ADAM optimizer [Kingma and Ba 2015] with weight decay $1e-5$. The learning rates for the surface and reflection MLP are initially set to be $1e-3$ and $1e-4$ respectively and decay by a factor of 0.95 every 5 epochs. All scenes are trained for 50 epochs with a batch size of 91×91 patches of rays. For pixels with the highest 10% $\mathcal{L}_c$ values, we increase their $\mathcal{L}_c$ by 100% to accelerate training. By default, $\lambda_c = 1.0, \lambda_{\text{SSIM}} = 1.0, \lambda_{\text{VGG}} = 0.1, \lambda_s = 1.0$, and $\lambda_{\text{REG}} = 1.0$. For mirrors (Appendix A), we set $\lambda_s = 0.1$.

At the beginning of training, the volume density predicted by the initialized surface MLPs is still converging. Hence, we train tMLPs with surface MLP only for ten epochs, and then continue with training both surface and reflection MLPs.

Voxel Pruning. We initialize $\gamma = 0.8$ and $\tau = 0.004$. We decrease γ by 0.1 and increase τ by 0.002 every 10 epochs within the first 30 epochs. Block size $G = 2$ in the first 20 epochs, reducing to 1 afterwards due to the more reliable tMLPs estimate.

Post-process CNN. The CNN architecture is a U-Net [Ronneberger et al. 2015] (Fig. 10) that is trained separately for each scene. We use the ADAM optimizer with weight decay $1e-5$. Training images are 128×128 random crops from the captured images, in batches of 32. The learning rate is set to $5e-4$ at the beginning and decayed by a factor of 0.95 every 20 epochs. We train for 1500 epochs, and set $\lambda_{\text{VGG}} = 0.1$ and $\lambda_{\text{BOUND}} = 2$.

7 EXPERIMENTS

7.1 Setup

7.1.1 Scenes. We test our method on five indoor scenes (Tab. 1). *Sofa* and *Living Room from Inria* are from Philip et al. [2021], and we use a second *Living Room from Xu et al.* [2021]. We captured two larger indoor scenes *Bar* and *Coffee Shop* using a `Canon EOS 60D` digital single-lens reflex camera. For missing areas larger than a tile, we manually created simple proxies to fill the space (Appendix B).

Fig. 11. Comparisons with geometry proxy methods **DeepBlending** [Hedman et al. 2018], **FVS** [Riegler and Koltun 2020] and **SVS** [Riegler and Koltun 2021] on large scale indoor scene. Our method can faithfully reconstruct the accurate ceiling reflection of the outside beyond the window (top row), the highlights reflected by the ground floor (middle row), and sharper details of the image synthesized at a virtual viewpoint (bottom row), while deepblending, FVS and SVS methods either fail to reconstruct the specular reflections (top two rows) or produce blurred results (bottom row).

Table 2. *Quantitative comparisons.* While the numbers are often close, please see our complementary qualitative results that show significant differences between these methods and improved quality of our method.

Scene	Metric	Deep Blending	FVS	SVS	Ours
Sofa	PSNR↑	26.87	22.79	27.78	29.44
	SSIM↑	0.849	0.821	0.873	0.870
Living Room from Inria [2021]	PSNR↑	26.93	22.30	27.62	29.26
	SSIM↑	0.897	0.859	0.916	0.888
Living Room from Xu et al. [2021]	PSNR↑	27.87	24.97	29.91	29.56
	SSIM↑	0.861	0.818	0.882	0.839
Bar	PSNR↑	26.14	22.12	27.15	26.69
	SSIM↑	0.754	0.667	0.791	0.739
Coffee Shop	PSNR↑	21.78	20.03	24.84	24.13
	SSIM↑	0.633	0.586	0.751	0.676

Camera Poses and Global Mesh Reconstruction. For our captured data *Bar* and *Coffee Shop*, we use software RealityCapture [CapturingReality 2016], which can handle large textureless regions like walls via Delaunay tetrahedralization [Jancosek and Pajdla 2011; Labatut et al. 2007]. The global mesh of *Living Room from Xu et al.* [2021] was also reconstructed using RealityCapture using a Kinect Azure RGBD camera.

Software and Hardware. We have implemented our method using PyTorch 1.9.0 [Paszke et al. 2019] for training. For each scene, the tMLPs are optimized on a GPU server with eight Tesla V100 GPUs. The post-process CNN is optimized on one NVIDIA GeForce RTX 3090 GPU. We report storage memory and training times of tMLPs and CNN for each scene in Table 1. Per-frame render time for 1280×720 pixels is 50ms on average after accelerating the post-process CNN using TensorRT [Nvidia 2018] with 16-bit precision. The rendering configuration is a desktop PC with an NVIDIA GeForce RTX 3090 GPU and an Intel Xeon E5-2678 v3 2.50GHz CPU.

7.2 Rendering Results and Comparisons

Note: Please refer to our supplemental video to see improved temporal coherence and rendering of specular reflections over baselines.

We qualitatively compare our method to view synthesis methods **DeepBlending** [Hedman et al. 2018], Free View Synthesis (**FVS**) [Riegler and Koltun 2020], and Stable View Synthesis (**SVS**) [Riegler and Koltun 2021] in Figure 11. Our method produces photo-realistic renderings with higher-quality view-dependent effects like highlights and reflections. Table 2 holds quantitative measures, showing that our method produces competitive PSNR and SSIM scores.

We also compare our method to **NeRF** [Mildenhall et al. 2020], **PlenOctrees** [Yu et al. 2021a] and **NSVF** [Liu et al. 2020a] in Figure 12 and Table 3. We train all methods with all images from *Living Room from Inria* [Philip et al. 2021] for the same amount of time. We found **NSVF** has slow convergence rates and struggles to prune empty space in large-scale scenes. However, if reconstructed geometry were incorporated to facilitate empty-space pruning, then convergence and rendering quality may improve. **NeRF** with its

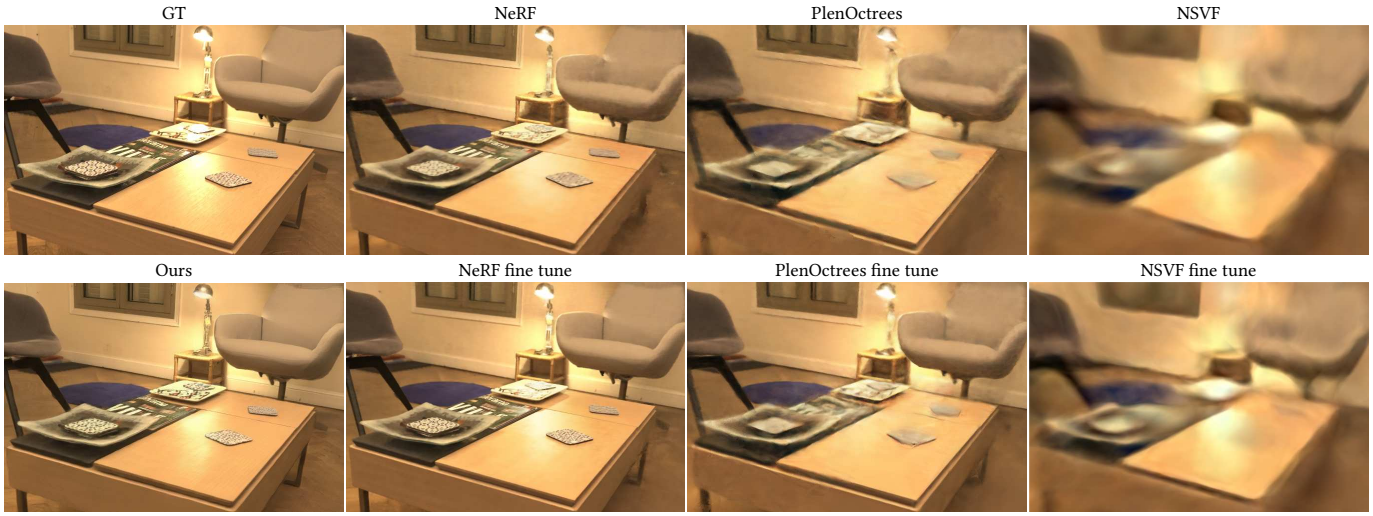
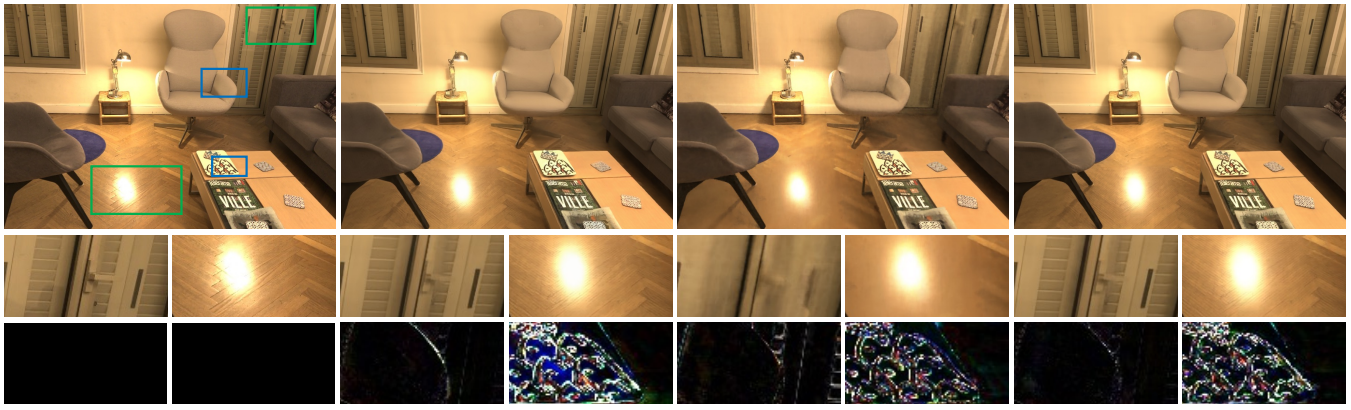


Fig. 12. Comparisons with **NeRF** [Mildenhall et al. 2020], **PlenOctrees** [Yu et al. 2021a] and **NSVF** [Liu et al. 2020b]. We fine tune both methods using just images facing the coffee table; our result has both high quality and interactive rendering.



(a) GT (b) Ours, PSNR = 29.71, SSIM = 0.896 (c) NeRF, PSNR = 30.97, SSIM = 0.866 (d) w/larger CNN, PSNR = **31.01**, SSIM = **0.901**

Fig. 13. *NeRF's higher PSNR is caused by boundary misalignment.* Areas around reflections are blurred for standard NeRF (c), while our renderings (b) look closer to the ground truth (a). Error maps (third row) show that our slightly lower PSNR is caused by boundary alignment errors. (d) Using a post-processing CNN with 2× width in each layer reduces quantitative alignment errors and achieves higher PSNR and SSIM score than NeRF. However, a larger CNN slows down rendering (6ms increase per frame).

Table 3. *Quantitative comparisons.* On the Living Room from Inria [Philip et al. 2021] scene, our approach and a large MLP via NeRF produce similar PSNR and SSIM numbers even though quality differs; please see Figure 13.

Metric	Ours	NeRF	PlenOctrees	NSVF
PSNR↑	29.26	30.99	26.13	22.06
SSIM↑	0.888	0.877	0.806	0.733

MLP representation that directly optimizes view-dependent color and volume density converges slower than our method, resulting in less sharp images and missing reflection highlights. In contrast, our approach produces photo-realistic renderings.

As an additional test, we fine tune **NeRF**, **PlenOctrees**, and **NSVF** with images facing a smaller coffee table region for the same amount of time again that was spent training this area. For fair comparison, the fine-tuning images are not used to compute PSNR and SSIM in Table 3. Our method is on-par with **NeRF** in that both produce realistic view dependent effects on the small region, but our rendering speed is interactive. While **PlenOctrees** also enables interactive rendering, its renderings contain blurry artifacts. One noteworthy item is that NeRF achieves higher PSNR than our method (Tab. 3) even though our results are qualitatively better. This is explained in Figure 13, where slight edge misalignments cause high error. The cause is the post-processing CNN; adding more trainable weights to the network resolves this issue.

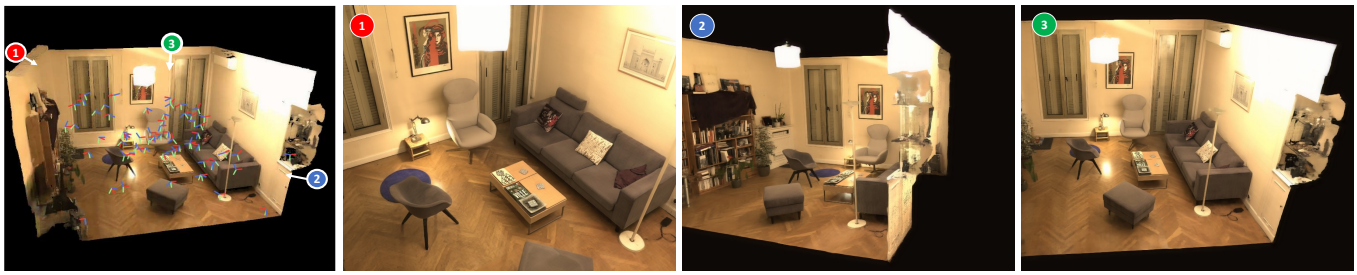


Fig. 14. *Renderings from viewpoints far from captured images.* On *Living Room from Inria* [Philip et al. 2021]. *Left:* Captured viewpoints visualized as basis vectors centered and rotated to match the camera, and interpolated viewpoints marked with numbers 1, 2, and 3. *Number 1:* Viewpoint from the ceiling. *Number 2:* Viewpoint from outside the wall. *Number 3:* Viewpoint above the the ceiling and outside the wall.

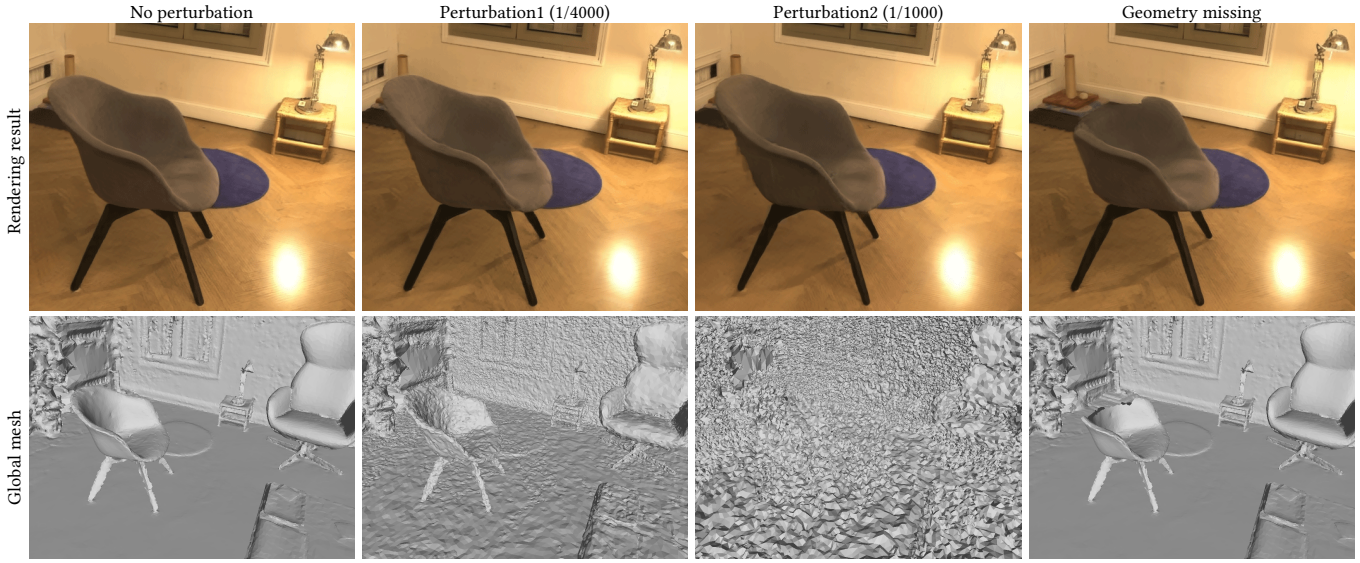


Fig. 15. *Dependence on accurate global mesh.* The results show that our method can tolerate a certain degree of geometric error in the global mesh.

7.3 Evaluations

tMLP Representation. To evaluate the benefits of our two-MLP representation, we compare the rendering quality of tMLPs with that of a single MLP. To model view-dependent effects, the single MLP outputs SH coefficients for RGB colors (Fig. 3). We optimize the single MLP and tMLPs under the same training setting, and for fairness we use an MLP with more free parameters than we use for tiles (10 linear layers each of size 128). Even with larger capacity, the single MLP fails to reconstruct the reflection, whereas our tMLPs representation fairs better in reproducing the sharp reflection.

View-dependent Reflections. Figure 4 shows that a representation of view-dependent effects as virtual lights at reflection distances using only 0-th order SH coefficients—effectively a diffuse reflection that still varies with viewpoint—fails to reproduce specular reflections on curved surfaces. In contrast, the 2nd-order SH used in our method begins to allow the salient specular reflections to emerge on the curved glass display. Even on planar surfaces, the higher-order SH coefficients provide a benefit in reproducing reflections more accurately (Fig. 4, bottom, with quantitative metrics in Tab. 5).

Rendering at Viewpoints Far from Captured Poses. When rendering at viewpoints extrapolated beyond the convex hull of captured camera poses, such as outside the wall or on the ceiling, our method still has good rendering quality (Fig. 14).

Global Mesh Accuracy Dependence. We test our algorithm using a global mesh that is perturbed under different noise levels, using *Living Room from Inria* [Philip et al. 2021]. We add Gaussian noise to each vertex along its normal, with mean and standard deviation set to $1/1000$, $1/4000$ of the shortest length of the global mesh bounding box respectively. As our method only uses the global mesh as an intersection proxy, it is robust to significant noise that causes the living room to be hard to recognize (Fig. 15 and Tab. 5). We also test the case of geometry missing from the global mesh by removing a chair back that would otherwise be occupied by 7 tiles. The rendering result is comparable to that of the full global mesh except that the back of chair disappears, as expected (Fig. 15).

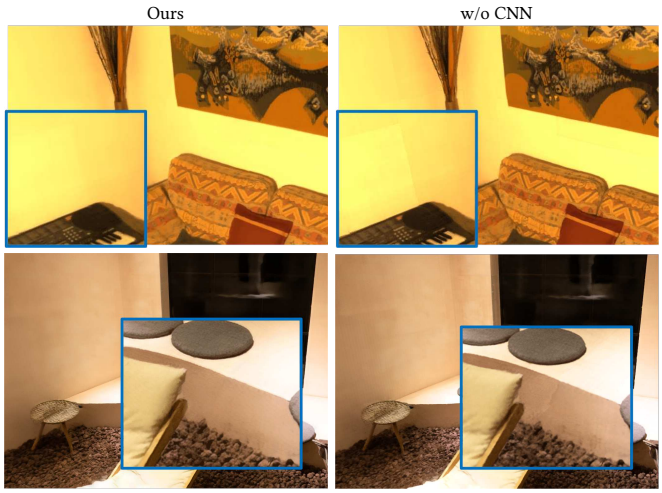


Fig. 16. *Post-process CNN reduces miscoloring and tile boundary line artifacts.* On the *Sofa* (top) and *Coffee Shop* (bottom) scenes.

Table 4. Loss ablations on *Living Room from Inria* [Philip et al. 2021].

Metric	Ours	w/o $\mathcal{L}_s$	w/o $\mathcal{L}_{\text{VGG}}$	w/o $\mathcal{L}_{\text{SSIM}}$	w/o $\mathcal{L}_{\text{REG}}$
PSNR↑	28.44	28.20	28.03	27.38	28.14
SSIM↑	0.870	0.865	0.862	0.848	0.863

Table 5. Ablations on *Living Room from Inria* [Philip et al. 2021]. BG: Background sampling. P1 / P2: Global mesh vertex perturbation sampled from a Gaussian with mean and stdev. of P1 $1/4000$ and P2 $1/1000$ of the shortest length of the total mesh bounding box.

Metric	Ours	w/o BG	w/o CNN	P1	P2	0th-order SH
PSNR↑	28.44	22.55	28.05	27.73	26.74	27.05
SSIM↑	0.870	0.769	0.859	0.859	0.847	0.866

CNN Post-process. The CNN post-process refines the rendering, e.g., making the tile boundaries more consistent (Fig. 16). We report quantitative results for this improvement using PSNR/SSIM metrics in Table 5, computed over a randomly-selected set of 10% of the captured images held-out from the training set.

Ablation Studies. We validate our loss terms and our background sampling strategy by eliminating each from training individually (Tab. 4 and 5). The data term $\mathcal{L}_c$ in Equation 7 always remains to obtain meaningful results. Qualitatively, the perception loss $\mathcal{L}_{\text{VGG}}$ and the SSIM loss $\mathcal{L}_{\text{SSIM}}$ help to obtain sharper rendering results with less blurry artifacts near image boundaries (Fig. 17). The regularization loss $\mathcal{L}_{\text{REG}}$ reduces blurring in recovered planar reflections. The surface loss $\mathcal{L}_s$ encourages more accurate decomposition (Fig. 5). Finally, without background sampling, the background color for each tile must be optimized through 3D points inside the tile; this influences the output of the surface MLP and the reflection MLP, causing blurring from phantom volume density (Fig. 6).

8 CONCLUSION AND DISCUSSION

For neural rendering to be broadly useful, it must scale to larger scenes and provide interactive rendering. To scale, our method represents the scene via tiles over a volume. To parallelize tile training, we proposed a background sampling algorithm to remove dependence between tiles. Tiles include two MLPs: a surface MLP to encode view-independent reflection and a reflection MLP to encode view-dependent reflection via virtual lights, which significantly reduces the angular sampling rate required to reconstruct high-frequency reflections. This separation also allows us to achieve rendering at 20 fps on average by baking view independence into a fixed octree.

Tile Size. In terms of trade-offs, tile size determines scene detail and both training and rendering times. Too large and the MLPs struggles to represent all detail; too small and training/render memory and time becomes a bottleneck. Further, as tile size increases, so too does its empty space. This causes more rays to pass through without intersecting any of the tile's global mesh geometry, again increasing integration time as the tile is sampled along the ray.

Limitations. We wish to capture images sparsely for large scenes to reduce capture effort. However, sparse samples might lead to inadequate training data for some tiles, resulting in phantom voxels or blurred renderings. For instance, this occurs in *Coffee Shop* at the top of a girder (Fig. 18). Moreover, not all reflections are always reconstructed—sparse unstructured input makes it very difficult to ensure highlights are always reconstructed, as highlights might appear in only a handful of images in the dataset (Figs. 3 and 4). The quality of our reflections can also be a little blurry. Finally, for curved surfaces with high curvature, the virtual points beneath the surface vary according to the surface normal, which increases the required amount of captured images to reconstruct specular reflections faithfully. Under this situation, it is still hard to leverage tMLPs to encode all specular reflections (Fig. 4).

Our global mesh reconstructions occasionally suffer floating geometries in the recovered result. This can cause errors in tiles, especially in empty spaces that are only included along ray paths by floating geometry. We found that errors are usually small, and that discarding rays that intersect floating geometry for those tiles did not degrade the rendering of objects behind so long as the appearance is well defined by rays from other viewpoints.

Future Work. Our contributions improve scalability through tiling a scene; however, other approaches in this fast-moving field will likely complement our approach. For instance, integrating adaptive octree resolutions across tiles may improve quality, and hashing techniques can save time and memory [Müller et al. 2022].

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their professional and constructive comments, and Yiqing Liang for proofreading. Weiwei Xu is partially supported by NSFC grant No. 61732016. James Tompkin is partially supported by the US NSF CNS-2038897 and an Amazon Research Award. Qixing Huang is partially supported by US NSF CAREER IIS-2047677. This paper is supported by Information Technology Center and State Key Lab of CAD&CG, Zhejiang University.

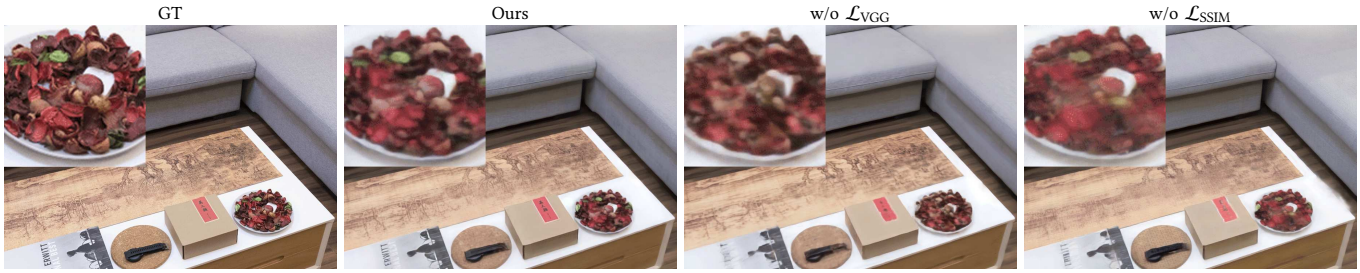


Fig. 17. *Adding structure similarity and VGG losses improves quality.* Qualitative ablation study on $\mathcal{L}_{\text{VGG}}$ and $\mathcal{L}_{\text{SSIM}}$, showing rendered results of the tea table in *Living Room from Xu et al.* [2021].

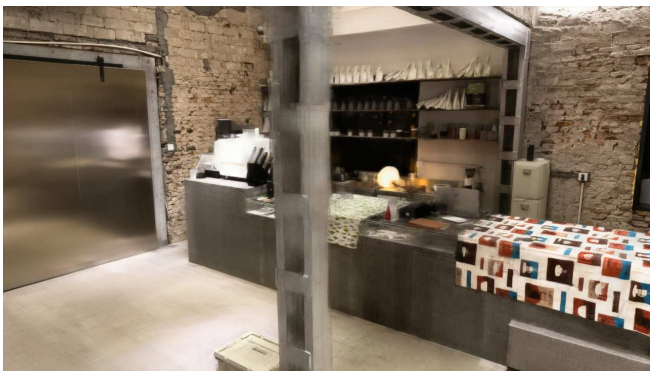


Fig. 18. *Rendering artifacts due to sparse image sampling.* Since *Coffee Shop* is large, we did not capture enough images to train the tMLPs for the tiles on the top of the girder. This results in inaccurate or phantom voxels and blurred rendering results.

REFERENCES

J. Amanatides, A. Woo, et al. 1987. A fast voxel traversal algorithm for ray tracing.. In *Eurographics*, Vol. 87. 3–10.

B. Attal, E. Laidlaw, A. Gokaslan, C. Kim, C. Richardt, J. Tompkin, and M. O'Toole. 2021. TöRF: Time-of-Flight Radiance Fields for Dynamic Scene View Synthesis. arXiv:2109.15271 [cs.CV]

J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan. 2021. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. arXiv:2103.13415 [cs.CV]

D. Bau, H. Strobelt, W. Peebles, J. Wulff, B. Zhou, J. Zhu, and A. Torralba. 2020. Semantic photo manipulation with a generative image prior. *arXiv preprint arXiv:2005.07727* (2020).

S. Bi, Z. Xu, P. Srinivasan, B. Mildenhall, K. Sulkavalli, M. Hašan, Y. Hold-Geoffroy, D. Kriegman, and R. Ramamoorthi. 2020. Neural Reflectance Fields for Appearance Acquisition. *https://arxiv.org/abs/2008.03824* (2020).

M. Boss, R. Braun, V. Jampani, J. T. Barron, C. Liu, and H. Lensch. 2021. NeRD: Neural Reflectance Decomposition from Image Collections. In *ICCV*.

C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. 2001. Unstructured lumigraph rendering. In *SIGGRAPH, ACM*. 425–432.

CapturingReality. 2016. Reality capture, http://capturingreality.com.

D. Casas, C. Richardt, J. Collomosse, C. Theobalt, and A. Hilton. 2015. 4D Model Flow: Precomputed Appearance Alignment for Real-time 4D Video Interpolation. *Computer Graphics Forum Journal of the European Association for Computer Graphics* (2015).

E. Chan, M. Monteiro, P. Kellnhofer, J. Wu, and G. Wetzstein. 2021. pi-GAN: Periodic Implicit Generative Adversarial Networks for 3D-Aware Image Synthesis. In *CVPR*.

G. Chaurasia, S. Duchene, O. Sorkine-Hornung, and G. Drettakis. 2013. Depth synthesis and local warps for plausible image-based navigation. 32, 3 (2013), 1–12.

S. E. Chen and L. Williams. 1993. View Interpolation for Image Synthesis. In *SIGGRAPH, ACM*. 279–288.

D. Cohen-Steiner, P. Alliez, and M. Desbrun. 2004. Variational shape approximation. In *SIGGRAPH, ACM*. 905–914.

P. Debevec, Y. Yu, and G. Borshukov. 1998. Efficient view-dependent image-based rendering with projective texture-mapping. In *Eurographics*. Springer, 105–116.

R. Du, M. Chuang, W. Chang, H. Hoppe, and A. Varshney. 2019. Montage4D: Real-Time Seamless Fusion and Stylization of Multiview Video Textures. *Journal of Computer Graphics Techniques* 8, 1 (2019), 1–34. https://doi.org/10.13140/RG.2.2.24965.09443

M. Eisemann, B. De Decker, M. Magnor, P. Bekaert, E. De Aguiar, N. Ahmed, C. Theobalt, and A. Sellent. 2008. Floating Textures. (2008).

J. Flynn, M. Broxton, P. Debevec, M. DuVall, G. Fyffe, R. Overbeck, N. Snavely, and R. Tucker. 2019. Deepview: View synthesis with learned gradient descent. In *CVPR*. 2367–2376.

G. Gafni, J. Thies, M. Zollhöfer, and M. Nießner. 2021. Dynamic Neural Radiance Fields for Monocular 4D Facial Avatar Reconstruction. In *CVPR*.

S. J. Garbin, M. Kowalski, M. Johnson, J. Shotton, and J. Valentin. 2021. FastNeRF: High-Fidelity Neural Rendering at 200FPS. In *ICCV*.

S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. 1996. The lumigraph. In *SIGGRAPH, ACM*. 43–54.

Y. Guo, D. Kang, L. Bao, Y. He, and S. Zhang. 2022. NeRFReN: Neural Radiance Fields with Reflections. In *CVPR*.

P. Hedman, J. Philip, T. Price, J. M. Frahm, G. Drettakis, and G. Brostow. 2018. Deep blending for free-viewpoint image-based rendering. 37, 6 (2018), 1–15.

P. Hedman, T. Ritschel, G. Drettakis, and G. Brostow. 2016. Scalable inside-out image-based rendering. 35, 6 (2016), 1–11.

P. Hedman, P. P. Srinivasan, B. Mildenhall, J. T. Barron, and P. Debevec. 2021. Baking Neural Radiance Fields for Real-Time View Synthesis. In *ICCV*.

M. Jancosek and T. Pajdla. 2011. Multi-view reconstruction preserving weakly-supported surfaces. In *CVPR*. IEEE, 3121–3128.

J. Johnson, A. Alahi, and L. Fei-Fei. 2016. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In *ECCV (Lecture Notes in Computer Science, Vol. 9906)*, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (Eds.). Springer, 694–711. https://doi.org/10.1007/978-3-319-46475-6_43

T. Karras, S. Laine, and T. Aila. 2019. A style-based generator architecture for generative adversarial networks. In *CVPR*. 4401–4410.

D. Kingma and J. Ba. 2015. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2015).

J. Kopf, F. Langguth, D. Scharstein, R. Szeliski, and M. Goesele. 2013. Image-based rendering in the gradient domain. 32, 6 (2013), 1–9.

P. Labatut, J. Pons, and R. Keriven. 2007. Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. In *ICCV*. IEEE, 1–8.

M. Levoy and P. Hanrahan. 1996. Light field rendering. In *SIGGRAPH, ACM*. 31–42.

C. Lin, W. Ma, A. Torralba, and S. Lucey. 2021. BARF: Bundle-Adjusting Neural Radiance Fields. In *ICCV*.

D. B. Lindell, J. N. P. Martel, and G. Wetzstein. 2021. AutoInt: Automatic Integration for Fast Neural Volume Rendering. In *CVPR*.

L. Liu, J. Gu, K. Lin, T. Chua, and C. Theobalt. 2020a. Neural sparse voxel fields. In *NeurIPS*, Vol. 33.

L. Liu, J. Gu, K. Z. Lin, T. S. Chua, and C. Theobalt. 2020b. Neural Sparse Voxel Fields. *NeurIPS* (2020).

L. Liu, M. Habermann, V. Rudnev, K. Sarkar, J. Gu, and C. Theobalt. 2021. Neural Actor: Neural Free-view Synthesis of Human Actors with Pose Control. *ACM SIGGRAPH Asia* (2021).

L. Liu, W. Xu, M. Zollhoefer, H. Kim, F. Bernard, M. Habermann, W. Wang, and C. Theobalt. 2019. Neural rendering and reenactment of human actor videos. 38, 5 (2019), 1–14.

S. Lombardi, T. Simon, J. Saragih, G. Schwartz, A. Lehrmann, and Y. Sheikh. 2019. Neural Volumes: Learning Dynamic Renderable Volumes from Images. (2019).

S. Lombardi, T. Simon, G. Schwartz, M. Zollhoefer, Y. Sheikh, and J. Saragih. 2021. Mixture of Volumetric Primitives for Efficient Neural Rendering. arXiv:2103.01954 [cs.GR]

R. Martin-Brualla, N. Radwan, M. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth. 2021. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR*.

W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. 2000. Image-Based Visual Hulls. In *SIGGRAPH, ACM*. 6 pages.

W. Matusik, H. Pfister, M. Brand, and L. McMillan. 2003. A Data-Driven Reflectance Model. *ACM Transactions on Graphics* 22, 3 (2003), 759–769.

W. Matusik, H. Pfister, A. Ngan, P. Beardsley, R. Ziegler, and L. Mcmillan. 2002. Image-Based 3D Photography Using Opacity Hulls. 21, 3 (2002), 427–437.

N. Max. 1995. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics* 1, 2 (1995), 99–108. https://doi.org/10.1109/2945.468400

A. Meka, C. Haene, R. Pandey, M. Zollhöfer, S. Fanello, G. Fyffe, A. Kowdle, X. Yu, J. Busch, J. Dourgarian, et al. 2019. Deep reflectance fields: high-quality facial reflectance field inference from color gradient illumination. 38, 4 (2019), 1–12.

B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar. 2019. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. 38, 4 (2019), 1–14.

B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and N. Ren. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*.

D. Miller and Rubin. 1998. Lazy decompression of surface light fields for precomputed global illumination. *Springer Vienna* (1998).

C. Müller. 1966. *Spherical harmonics / Claus Müller.* Springer-Verlag Berlin ; New York. 45 p. : pages.

T. Müller, A. Evans, C. Schied, and A. Keller. 2022. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *arXiv:2201.05989* (2022).

M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger. 2019. Differentiable Volumetric Rendering: Learning Implicit 3D Representations without 3D Supervision. In *CVPR*.

M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. 2013. Real-time 3D reconstruction at scale using voxel hashing. 32, 6 (2013), 1–11.

Nvidia. 2017-2018. Nvidia Corporation. TensorRT. https://developer.nvidia.com/tensorrt.

R. Pandey, A. Tkach, S. Yang, P. Pidlypenskyi, J. Taylor, R. Martin-Brualla, A. Tagliasacchi, G. Papandreou, P. Davidson, C. Keskin, et al. 2019. Volumetric capture of humans with a single rgbd camera via semi-parametric learning. In *CVPR*. 9709–9718.

J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. 2019a. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *CVPR*. 165–174.

K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. Goldman, S. Seitz, and R. Martin-Brualla. 2020. Deformable Neural Radiance Fields. In *ICCV*.

K. Park, U. Sinha, P. Hedman, J. T. Barron, S. Bouaziz, D. B. Goldman, R. Martin-Brualla, and S. M. Seitz. 2021. HyperNeRF: A Higher-Dimensional Representation for Topologically Varying Neural Radiance Fields. 40, 6, Article 238 (2021).

T. Park, M. Liu, T. Wang, and J. Zhu. 2019b. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*. 2337–2346.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). 8024–8035.

S. Peng, J. Dong, Q. Wang, S. Zhang, Q. Shuai, H. Bao, and X. Zhou. 2021a. Animatable Neural Radiance Fields for Human Body Modeling. In *ICCV*.

S. Peng, Y. Zhang, Y. Xu, Q. Wang, Q. Shuai, H. Bao, and X. Zhou. 2021b. Neural Body: Implicit Neural Representations with Structured Latent Codes for Novel View Synthesis of Dynamic Humans. In *CVPR*.

E. Penner and L. Zhang. 2017. Soft 3D reconstruction for view synthesis. 36, 6 (2017), 1–11.

J. Philip, M. Gharbi, T. Zhou, A. A. Efros, and G. Drettakis. 2019. Multi-view relighting using a geometry-aware network. 38, 4 (2019), 1–14.

J. Philip, S. Morgenthaler, M. Gharbi, and G. Drettakis. 2021. Free-viewpoint Indoor Neural Relighting from Multi-view Stereo. *ACM Transactions on Graphics* (2021).

A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer. 2021. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *CVPR*.

C. Reiser, S. Peng, Y. Liao, and A. Geiger. 2021. KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs. arXiv:2103.13744 [cs.CV]

K. Rematas, R. Martin-Brualla, and V. Ferrari. 2021. ShaRF: Shape-conditioned Radiance Fields from a Single View. *arXiv preprint arXiv:2102.08860* (2021).

G. Riegler and V. Koltun. 2020. Free View Synthesis. In *ECCV*.

G. Riegler and V. Koltun. 2021. Stable View Synthesis. In *CVPR*.

S. Rodriguez, S. Prakash, P. Hedman, and G. Drettakis. 2020. Image-Based Rendering of Cars using Semantic Labels and Approximate Reflection Flow. *Proc. ACM Comput. Graph. Interact.* 3 (2020).

O. Ronneberger, P. Fischer, and T. Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*.

J. L. Schonberger and J. M. Frahm. 2016. Structure-from-Motion Revisited. In *CVPR*. 4104–4113.

K. Schwarz, Y. Liao, M. Niemeyer, and A. Geiger. 2020. Graf: Generative radiance fields for 3D-aware image synthesis. In *NeurIPS*, Vol. 33.

H. Y. Shum and S. B. Kang. 2000. *A Review of Image-based Rendering Techniques*. . Microsoft.

K. Simonyan and A. Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*.

S. N. Sinha, J. Kopf, M. Goesele, D. Scharstein, and R. Szeliski. 2012. Image-based rendering for scenes with reflections. 31, 4 (2012), 1–10.

V. Sitzmann, S. Rezchikov, B. Freeman, J. Tenenbaum, and F. Durand. 2021. Light field networks: Neural scene representations with single-evaluation rendering. *Advances in Neural Information Processing Systems* 34 (2021).

V. Sitzmann, J. Thies, F. Heide, M. Nießner, G. Wetzstein, and M. Zollhofer. 2019a. Deepvoxels: Learning persistent 3d feature embeddings. In *CVPR*. 2437–2446.

V. Sitzmann, M. Zollhöfer, and G. Wetzstein. 2019b. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *NeurIPS*. 1121–1132.

P. Srinivasan, B. Deng, X. Zhang, M. Tancik, B. Mildenhall, and J. T. Barron. 2021. NeRV: Neural Reflectance and Visibility Fields for Relighting and View Synthesis. In *CVPR*.

P. P. Srinivasan, R. Tucker, J. T. Barron, R. Ramamoorthi, R. Ng, and N. Snavely. 2019. Pushing the boundaries of view extrapolation with multiplane images. In *CVPR*. 175–184.

E. Sucar, S. Liu, J. Ortiz, and A. Davison. 2021. iMAP: Implicit Mapping and Positioning in Real-Time. In *ICCV*.

C. Sun, M. Sun, and H. Chen. 2021. Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction. *arXiv preprint arXiv:2111.11215* (2021).

R. Szeliski and P. Golland. 1999. Stereo matching with transparency and matting. *International Journal of Computer Vision* 32, 1 (1999), 45–61.

A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, K. Sunkavalli, R. Martin-Brualla, T. Simon, J. Saragih, M. Nießner, et al. 2020. State of the art on neural rendering. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 701–727.

A. Tewari, J. Thies, B. Mildenhall, P. Srinivasan, E. Tretschk, Y. Wang, C. Lassner, V. Sitzmann, R. Martin-Brualla, S. Lombardi, T. Simon, C. Theobalt, M. Niessner, J. T. Barron, G. Wetzstein, M. Zollhoefer, and V. Golyanik. 2021. Advances in Neural Rendering. arXiv:2111.05849 [cs.GR]

J. Thies, M. Zollhöfer, and M. Nießner. 2019. Deferred neural rendering: Image synthesis using neural textures. 38, 4 (2019), 1–12.

A. Trevithick and B. Yang. 2020. GRF: Learning a General Radiance Field for 3D Scene Representation and Rendering. In *ICCV*.

D. Verbin, P. Hedman, B. Mildenhall, T. Zickler, J. T. Barron, and P. P. Srinivasan. 2022. Ref-NeRF: Structured View-Dependent Appearance for Neural Radiance Fields. In *CVPR*.

Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. 2004. Image Quality Assessment: From Error Visibility to Structural Similarity. *Trans. Img. Proc.* 13, 4 (2004), 600–612. https://doi.org/10.1109/TIP.2003.819861

Z. Wang, S. Wu, W. Xie, M. Chen, and V. A. Prisacariu. 2021. NeRF–: Neural Radiance Fields Without Known Camera Parameters. *arXiv preprint arXiv:2102.07064* (2021).

S. Wizadwongsa, P. Phongthawee, J. Yenphraphai, and S. Suwajanakorn. 2021. Nex: Real-time view synthesis with neural basis expansion. In *CVPR*. 8534–8543.

D. N. Wood, D. I. Azuma, K. Aldinger, B. Curless, T. Duchamp, D. H. Salesin, and W. Stuetzle. 2000. Surface light fields for 3D photography. In *SIGGRAPH, ACM*. 287–296.

Y. Xie, T. Takikawa, S. Saito, Or Litany, S. Yan, N. Khan, F. Tombari, J. Tompkin, V. Sitzmann, and S. Sridhar. 2022. Neural Fields in Visual Computing and Beyond. *Computer Graphics Forum* (2022). https://doi.org/10.1111/cgf.14505

J. Xu, X. Wu, Z. Zhu, Q. Huang, Y. Yang, H. Bao, and W. Xu. 2021. Scalable Image-Based Indoor Scene Rendering with Reflections. 40, 4, Article 60 (2021), 14 pages. https://doi.org/10.1145/3450626.3459849

L. Yen-Chen, P. Florence, J. T. Barron, A. Rodriguez, P. Isola, and T. Lin. 2021. iNeRF: Inverting Neural Radiance Fields for Pose Estimation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

A. Yu, S. Fridovich-Keil, M.Tancik, Q. Chen, B. Recht, and A. Kanazawa. 2022. Plenoxels: Radiance Fields without Neural Networks. In *CVPR*.

A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa. 2021a. Plenoctrees for real-time rendering of neural radiance fields. In *ICCV*. 5752–5761.

A. Yu, V. Ye, M. Tancik, and A. Kanazawa. 2021b. pixelNeRF: Neural Radiance Fields from One or Few Images. In *CVPR*.

C. Zhang and T. Chen. 2003. A survey on image-based rendering. *Signal Processing Image Communication* 19 (2003), 1–28.

K. Zhang, F. Luan, Q. Wang, K. Bala, and N. Snavely. 2021a. PhySG: Inverse Rendering with Spherical Gaussians for Physics-based Material Editing and Relighting. In *CVPR*. 5453–5462.

K. Zhang, G. Riegler, N. Snavely, and V. Koltun. 2020. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492* (2020).
X. Zhang, P. P. Srinivasan, B. Deng, P. Debevec, W. T. Freeman, and J. T. Barron. 2021b. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. 40, 6 (2021), 1–18.
H. Zhou, S. Hadap, K. Sunkavalli, and D. W. Jacobs. 2019. Deep single-image portrait relighting. In *ICCV*. 7194–7202.
T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely. 2018. Stereo magnification: learning view synthesis using multiplane images. 37, 4 (2018), 1–12.
Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys. 2022. NICE-SLAM: Neural Implicit Scalable Encoding for SLAM. In *CVPR*.

Fig. 19. *Left:* Tiles enclosing transparent glasses displayed in black lines. *Right:* Rendering result for these tiles.

A TRANSPARENT GLASS AND MIRRORS

These two types of objects are special cases in our pipeline, since their geometries cannot be reconstructed using MVS software.

Without manual effort, our method can handle transparent glass in front of opaque surfaces within the same tile (Fig. 19). For example, a wine bottle display cabinet could have transparent glasses on its doors, so long as the depth of the cabinet, including the glasses and reconstructed surfaces of wine bottles, were enclosed by one tile. Here, rendered image quality is good because the glass does not appear in the global mesh and so it is reconstructed as a volumetric object. Note that specular reflections in the transparent glass are captured if the distance between the reflected object and the glass plane is larger than the distance between the glass and its opaque surface behind—this is typical for a display cabinet. In this case, the reflection MLP encodes the specular reflections.

If the thickness of the transparent object is larger than the tile thickness or there is no opaque surface behind the glass within the same tile, then our method will fail to render the transparent object.

For mirrors, like in *Living Room from Xu et al.* [2021], we manually mark them in the global mesh. Then, we simply ignore the surface MLPs for rays that intersect mirrors and instead encode the entire appearance using the reflection MLPs. Since small viewpoint changes lead to dramatic appearance changes for mirrors, we group all tiles that intersect a mirror.

One question with mirrors that may arise is how to ignore capturing the reflected image of the photographer capturing the scene. While Xu et al. [Xu et al. 2021] simply ray trace the rest of the reconstructed scene to avoid the photographer, our case is different. One reason that the photographer is not in the rendering might be because they are included in few images and because the photographer moves between images (unlike lights that cause reflections). This effectively makes them an outlier in the reconstruction.

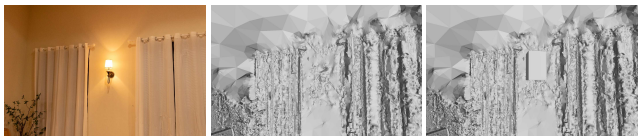


Fig. 20. *Left:* Captured input image of a light. *Middle:* Original geometry of the light obtained from the RealityCapture software. *Right:* Added proxy for the light that is later turned into a volumetric reconstruction.

B GEOMETRY PROXIES

In the *Bar* and *Coffee Shop* scenes that we captured ourselves, we found that the geometries for textureless thin objects or for light sources causing image overexposure might fail to be reconstructed. Even though our method can handle missing geometries enclosed by a tile, large missing areas might lead to no allocated tiles. For these occasional cases, we manually place cuboid and plane proxies into the global mesh before tiling (Fig. 20). For the *Bar* scene, we create 14 proxies: eight chairs, three lights, and three planes to cover large holes in walls. For the *Coffee Shop* scene, we place four planes to cover wall and ground holes, and place five planes for glass windows. If desired, the effort to create proxies for thin objects can be reduced by using a depth camera, though adding proxies for transparent objects would still be required.