ScaNeRF: Scalable Bundle-Adjusting Neural Radiance Fields for Large-Scale Scene Rendering

XIUCHAO WU, State Key Lab of CAD&CG, Zhejiang University, China
JIAMIN XU, Hangzhou Dianzi University, China
XIN ZHANG, 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
YUJUN SHEN, Ant Group, China
JAMES TOMPKIN, Brown University, USA
WEIWEI XU*, State Key Lab of CAD&CG, Zhejiang University, China

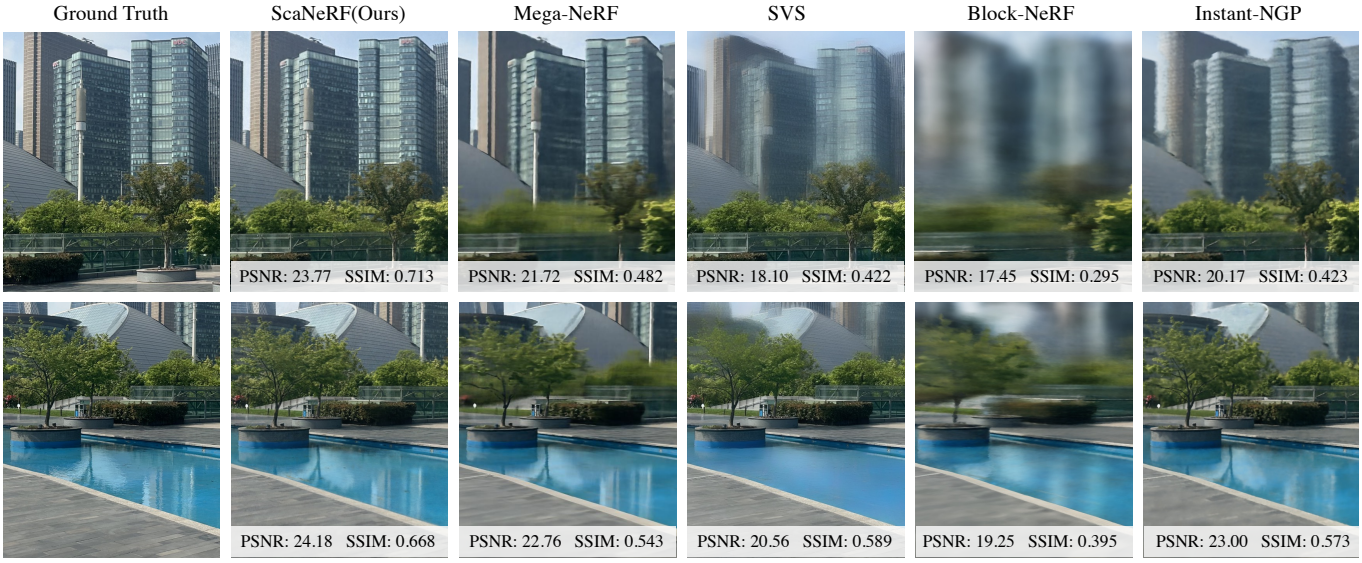


Fig. 1. **Real-world large-scale scene rendering is improved.** In the *Park* scene, both the reflecting pool and the distant city are more accurately reproduced than in current methods for deep texture IBR (SVS [Riegler and Koltun 2021]), single hash grid (Instant-NGP [Müller et al. 2022]), or large-scale scenes (Block-NeRF [Tancik et al. 2022], Mega-NeRF [Turki et al. 2022]). As these methods model static scenes, dynamic elements like trees remain a challenge.

*Corresponding author

Authors' addresses: Xiuchao Wu, wuxiuchao@zju.edu.cn, State Key Lab of CAD&CG, Zhejiang University, China; Jiamin Xu, superxjm@yeah.net, Hangzhou Dianzi University, China; Xin Zhang, 22121075@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; Yujun Shen, shenyujun0302@gmail.com, Ant Group, China; 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.

0730-0301/2023/12-ART260 $15.00
https://doi.org/10.1145/3618369

High-quality large-scale scene rendering requires a scalable representation and accurate camera poses. This research combines tile-based hybrid neural fields with parallel distributive optimization to improve bundle-adjusting neural radiance fields. The proposed method scales with a divide-and-conquer strategy. We partition scenes into tiles, each with a multi-resolution hash feature grid and shallow chained diffuse and specular multi-layer perceptrons (MLPs). Tiles unify foreground and background via a spatial contraction function that allows both distant objects in outdoor scenes and planar reflections as virtual images outside the tile. Decomposing appearance with the specular MLP allows a specular-aware warping loss to provide a second optimization path for camera poses. We apply the alternating direction method of multipliers (ADMM) to achieve consensus among camera poses while maintaining parallel tile optimization. Experimental results show that our method outperforms state-of-the-art neural scene rendering method quality by 5%–10% in PSNR, maintaining sharp distant objects and view-dependent reflections across six indoor and outdoor scenes.

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

Additional Key Words and Phrases: Neural fields, NeRF, alternating direction method of multipliers, large-scale scenes

ACM Reference Format:
Xiuchao Wu, Jiamin Xu, Xin Zhang, Hujun Bao, Qixing Huang, Yujun Shen, James Tompkin, and Weiwei Xu. 2023. ScaNeRF: Scalable Bundle-Adjusting Neural Radiance Fields for Large-Scale Scene Rendering. *ACM Trans. Graph.* 42, 6, Article 260 (December 2023), 18 pages. https://doi.org/10.1145/3618369

1 INTRODUCTION

Recent advancements in real-world scene reconstruction from photographs use neural radiance fields (NeRFs) to compactly encode 3D geometry and appearance into neural networks [Barron et al. 2021; Mildenhall et al. 2020; Verbin et al. 2022; Xie et al. 2022; Yu et al. 2021b; Zhang et al. 2020]. This is accomplished by optimizing network weights to minimize a photometric loss that penalizes the difference between a set of real-world images with known camera poses and the reproduction of those images through volumetric rendering. Camera poses are typically obtained from visual feature-point correspondences via structure from motion (SfM; [Schonberger and Frahm 2016]) or through simultaneous localization and mapping (SLAM; [Cadena et al. 2016; Zhu et al. 2022]). As the scene size increases, especially to include outdoor scenes, optimization requires a scalable representation to achieve high quality and practical compute times. These often use parallel processing and space tiling schemes [Tancik et al. 2022; Turki et al. 2022; Wu et al. 2022].

Within this setting, camera poses may be inaccurate since view-dependent effects, repeated structures, and occlusions between objects inevitably create imprecision and outliers in the required feature-point correspondences. Precise camera poses are required to reproduce fine details in indoor scenes, like the thin grout lines between repeating kitchen tiles, and in outdoor scenes, where imprecise poses will blur distant objects like the structured facades of buildings. To refine camera poses, recent research has jointly optimized poses and network weights based on a photometric loss—so-called *photometric bundle adjustment* [Delaunoy and Pollefeys 2014]—to improve view synthesis quality [Chng et al. 2022; Clark 2022a; Lin et al. 2021]. However, current bundle-adjusting NeRF methods cannot directly apply to large-scale scenes due to computational and memory costs because they assume access to the entire representation of space at once. Further, appearance is not always enough: classical bundle adjustment jointly optimizes 3D points and poses, but photometric bundle adjustment does not explicitly account for shape-radiance ambiguities [Zhang et al. 2020] and so can falsely explain appearance with pose errors.

We address the bundle-adjusting NeRF problem for large-scale 3D scenes by combining a tile-based hybrid neural field scene representation with parallel distributive training via the alternating direction method of multipliers (ADMM [Boyd et al. 2011]). The representation stores per-tile features in multi-resolution hash grids to accelerate optimization [Clark 2022a; Müller et al. 2022]. Rather than decode hash features into appearance only by a single MLP [Müller et al. 2022], they are decoded by two chained MLPs into diffuse appearance and specular appearance [Verbin et al. 2022], but critically we represent specular appearance at reflection distances as virtual images. This helps cope with sparse camera inputs while maintaining render quality. We also develop an IBR-inspired specular-aware warping loss to assist multi-view consistency using predicted appearance and surface depth (akin to 3D points). Estimating specular appearance lets us reduce the influence of view-dependent effects on this loss, mitigating the shape-radiance ambiguity. Tiles contain both bound foreground space and unbound background space accessed via a contraction function. Post-optimization rendering traces rays through multiple tile foregrounds to sample the farthest tile background regions for sharpness. Multi-tile foreground rendering uses point-based blending to reduce artifacts in tile overlap regions.

This approach has three benefits over existing work:

- The representation is unified for indoor and outdoor scenes. Representing tile background lets us capture both the far background in outdoor scenes and reflections on planar surfaces in indoor scenes, such as in a mirror, as virtual images outside the foreground of a tile. The specular MLP models glossy reflections that cannot be handled using virtual images.
- Distributed parallel optimization of camera poses and network weights via the ADMM scheme provides bounded memory cost per GPU and accelerated training to cope with large scenes.
- The PSNR of rendered images increases by 5%–10% on two indoor and four outdoor scenes, with sharp objects close-up and in the distance, high-quality planar reflections, and reduced tile boundary artifacts (Fig. 1).

2 RELATED WORK

2.1 Neural Radiance Fields

NeRF [Mildenhall et al. 2020] emerged as a powerful scene reconstructor and led to fast progress in view synthesis. Given camera rays as input, NeRF uses MLPs to predict continuous volume density and view-dependent color fields. Many follow-up methods improve final rendering speed by baking color and opacity fields into discretized voxel grids [Garbin et al. 2021; Hedman et al. 2021; Wizadwongsa et al. 2021; Yu et al. 2021a], or distilling the radiance field into a set of small MLPs [Reiser et al. 2021]. Other methods improve the optimization speed by removing redundant sample points [Hu et al. 2022; Liu et al. 2019b; Sun et al. 2022a], or using priors such as upon monocular depths or normals [Roessle et al. 2022; Wang et al. 2022; Yu et al. 2022], input from sparse point clouds [Deng et al. 2022], semantics [Jain et al. 2021], appearance priors [Niemeyer et al. 2022], and Manhattan-world assumptions [Guo et al. 2022]. Local feature grid representations, such as voxel grids [Fridovich-Keil et al. 2022; Karnewar et al. 2022; Sun et al. 2022b], tri-planes [Chan et al. 2022; Chen et al. 2022b; Reiser et al. 2023], and point clouds [Xu et al. 2022], can also accelerate optimization. Our method uses multi-resolution hash grids to accelerate optimization [Müller et al. 2022].

NeRF can also be applied to the rendering of large-scale scenes [Rematas et al. 2022; Xiangli et al. 2022]. To reduce the memory cost of feature-grid-based representation, Zhang et al. [2023] combine a hash grid with a multi-resolution feature plane, while Xu et al. [2023] use multi-resolution vertical feature planes for UAV data. Another approach is to partition scenes into blocks or tiles and optimize local radiance fields [Tancik et al. 2022; Wu et al. 2022]. Similar efforts on breaking the scene into components have also been embraced in the

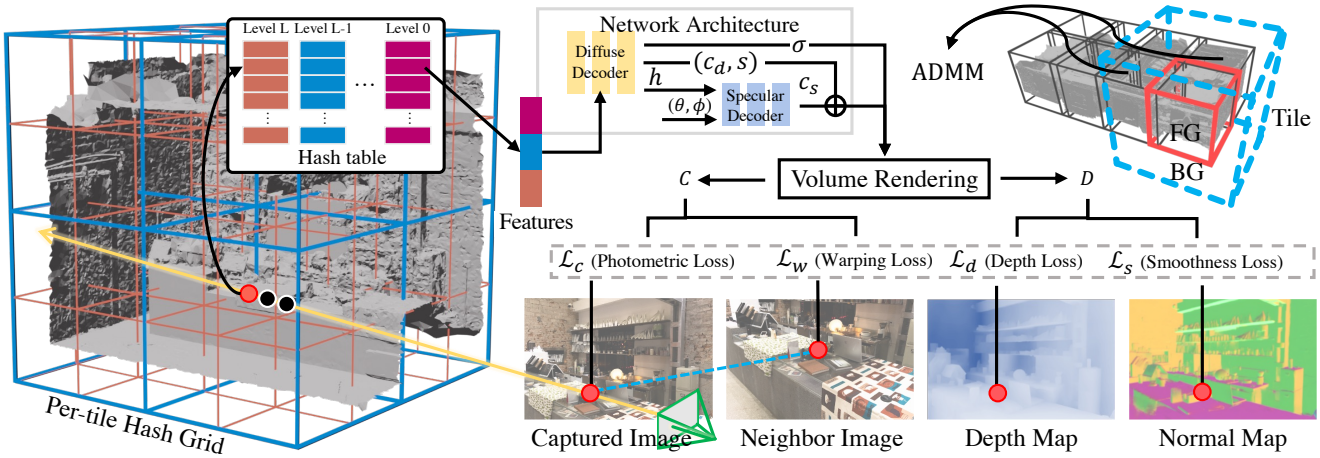


Fig. 2. **Our proposed scene representation.** *Top Right:* An example of scene partition. The dashed lines indicate an unbounded background region of the tile (FG: Foreground, BG; Background). The per-tile neural scene representation consists of a multi-resolution hash feature grid, a diffuse decoder, and a specular decoder. The tile-based local features, network weights, and camera poses are optimized in parallel and distributively using ADMM.

realm of precomputed radiance transfer [Sloan et al. 2003] and light transport acquisition [Nayar et al. 2004]. Mega-NeRF [Turki et al. 2022] and SUDS [Turki et al. 2023] extend scene partitions by parallel optimization from thousands of drone images or by reconstructing large-scale urban scenes. We also use tile partitions, and focus on their joint optimization with camera poses.

To help in this task, our method separates scene appearance into diffuse and specular components [Bi et al. 2020; Boss et al. 2021; Srinivasan et al. 2021]. Ref-NeRF [Verbin et al. 2022] trains a spatial MLP to predict diffuse colors and surface normals and then produces specular reflections via normal-reflected rays and a directional MLP. Inverse rendering representations with surface, normal, lighting, albedo, and bidirectional reflectance distribution functions (BRDFs) are also possible, but cannot yet apply to large-scale scenes [Hasselgren et al. 2022; Laine et al. 2020; Liu et al. 2019a; Munkberg et al. 2022; Yao et al. 2022; Zhang et al. 2021a,b]. We represent reflections using both diffuse virtual image elements via background contraction and via a specular MLP. The specular MLP also lets us de-emphasize difficult regions in our warping loss.

2.2 Bundle Adjustment

Bundle adjustment (BA) [Agarwal et al. 2010; Triggs et al. 2000] is a key component of structure from motion (SfM). Given an initial estimate, BA jointly optimize 3D scene structure and camera poses by minimizing the geometric re-projection error (geometric BA [Engel et al. 2017, 2014]) or photometric error (photometric BA [Delaunoy and Pollefeys 2014]). Geometric BA relies on keypoints or line correspondences, and can support the distributed and scalable 3D reconstruction of large-scale scenes [Eriksson et al. 2016; Jung and Weiss 2021; Peng et al. 2016; Zhu et al. 2021]. Zhang et al. [2017] proposed a distributed geometric-BA algorithm based on alternating direction method of multipliers (ADMM; [Boyd et al. 2011]) to achieve global camera consensus. However, geometric BA produces inaccurate camera poses when keypoints cannot be reliably detected. On the other hand, photometric BA exploits dense photometric consistency across images and is usually applied to the local refinement of camera poses. Both geometric and photometric BA can be naturally integrated into deep learning frameworks through feature-space alignment [Lindenberger et al. 2021; Sarlin et al. 2021; Tang and Tan 2018].

2.3 Bundle-Adjusting NeRF

NeRF quality relies on accurate camera poses, leading to research in joint optimization of NeRF models and camera poses. Such techniques are extensions of photometric BA due to the photometric loss used in optimization. These includes pose refinement [Lin et al. 2021], pose estimation with no initialization but for limited scenes like forward-facing settings [Bian et al. 2023; Wang et al. 2021], and for video sequences [Meuleman et al. 2023]. However, all photometric optimizations suffers from the shape-radiance ambiguity, which is exacerbated by inaccurate camera poses. Incorporating monocular depth priors may reduce ambiguity [Bian et al. 2023]. Yen-Chen et al. [2021] and Lin et al. [2023] estimate the pose of an RGB image using a NeRF as a target object. Most methods rely on refining a good camera initialization to mitigate ambiguity [Chng et al. 2022; Clark 2022b; Jeong et al. 2021; Lin et al. 2021], as we do. Our method adds scalable distributive optimization for large-scale scenes based on ADMM, helping to achieve higher-quality rendering of fine details.

3 METHOD OVERVIEW

For easier reading, we defer less important details and parameters to Appendix A and focus on higher-level aspects and their effects. We denote input variables, $\hat{\cdot}$, as distinct from optimized ones. Our goals are threefold. We wish to design a representation and optimization scheme that can:

(1) Scale to large indoor & outdoor scenes with complex reflections,
(2) Operate in parallel with limited inter-process data transfer, and
(3) Refine camera poses to preserve fine details up close and afar.

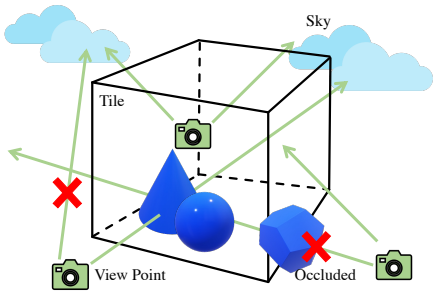


Fig. 3. **Assigning rays to tiles.** We select rays without red crosses to optimize the tile. Red cross rays either do not intersect the tile or are occluded by earlier proxy geometry.

As input, the proposed method takes a set of posed RGB images $\langle \hat{I}_i, \hat{T}_i \rangle \in \hat{\mathcal{T}}$ of a real-world static 3D scene. $\hat{I}_i$ denotes the captured i-th image, and $T_i = \{\mathbf{o}_i, \mathbf{R}_i\}$ denotes the corresponding camera extrinsic parameters of position $\mathbf{o}_i$ and orientation $\mathbf{R}_i$. The pixels of each posed camera defines a set of rays $\hat{\mathbf{r}} \in \hat{\mathcal{R}}$ each with a color $\hat{\mathbf{c}}$ that we can use as an optimization objective. As output, the method produces refined camera poses and a hybrid tile-based neural field scene representation: This contain the scene geometry and diffuse and specular appearance of both foreground regions and background regions beyond the tile (Fig. 2).

We optimize the representation's latent features, network weights, tile voxel occupancy, and camera poses to minimize the visual difference between the input RGB images and renderings of the scene representation (Sec. 5). To make this scalable, we use the alternating direction method of multipliers (ADMM) to optimize these features in tiles with consistent camera poses (Sec. 7). After completing optimization, we render novel views by blending samples across tiles in a background-aware way (Sec. 8).

4 PREPROCESSING

Defining and partitioning space. First, the method reconstructs initial camera parameters and proxy geometry G using existing non-neural multi-view stereo methods [CapturingReality 2016; Schonberger and Frahm 2016]. Camera intrinsic parameters are fixed and not optimized; only extrinsic parameters are optimized. The proxy geometry facilitates tile partitioning and provides rough visibility since the geometric information lets us judge whether a pixel in a image is visible or not in other viewpoints.

Next, to partition the scene space into tiles $k \in \mathcal{K}$ according to the proxy geometry, we uniformly divide the bounding box of the reconstructed proxy mesh G into axis-aligned cube tiles, numbering $L \times W \times H$. Tiles are of equal size with 20% overlap along each axis. Our scenes show indoor or low-height nearby outdoor scenes, so practically height $H = 1$.

Assigning rays to tiles. Per tile k, we assign subsets of rays $\hat{\mathcal{R}}_k$ based on the occlusion relationships between rays, tiles, and the proxy mesh. As illustrated in Figure 3, Ray $\hat{\mathbf{r}}$ will be added to $\hat{\mathcal{R}}_k$ if the following two conditions are satisfied:

(1) The first point of intersection between the ray origin and the proxy mesh lies within a tile. The ray could originate from a camera inside the tile, or from a camera outside the tile either within another tile or within space that no tile covers.
(2) The ray intersects any tile face and does not meet condition (1). This includes rays that originate from a camera within a tile, rays that pass through a tile and do not intersect the proxy geometry (such as in sky regions), and rays that first intersect the proxy geometry outside the tile.

One might think that the rays that meet condition 1 are approximately 'tile foreground rays' and the rays that meet condition 2 are approximately 'tile background rays', but practically points along both sets are used to train both foreground and background tile regions of the representation. Finally, for stable camera pose optimization, only cameras with at least 10% of their rays assigned to a tile are used to optimize that tile.

5 TILE-BASED HYBRID NEURAL FIELDS

5.1 Representation

Our scene representation is optimized by the differentiable volume rendering approach of Mildenhall et al. [2020] that requires density σ and view-dependent appearance $\mathbf{c}$ fields. Given a ray origin $\mathbf{o}$ and its direction $\boldsymbol{\omega}$, we sample N 3D points $\mathbf{x}$ along a ray, with $\mathbf{x}$ defined by the distance along the ray t: $\mathbf{r} = \mathbf{o} + t\boldsymbol{\omega}$. Integrating density and color at points along the ray and weighting them by the transmission density α at each point produces a final pixel color $\mathbf{c}$. The integral can be solved discretely with numerical quadrature:

$$\alpha_n = \exp\left(-\sum_{m<n} \sigma_m \delta_m\right)\left(1 - \exp\left(-\sigma_n \delta_n\right)\right), \tag{1}$$

$$\mathbf{c} = \sum_n \alpha_n \mathbf{c}_n, \tag{2}$$

where n is the index of the sampled point, and $\delta_n = t_{n+1} - t_n$ is the distance between adjacent samples. Ray depth follows similarly:

$$d = \sum_n \alpha_n t_n. \tag{3}$$

Diffuse and specular appearance. Rather than directly optimize MLPs to produce density and view-dependent appearance fields from a sampled point x and direction $\boldsymbol{\omega}$, and similar to Verbin et al. [2022], we use two chained MLPs per tile: a density and diffuse appearance MLP D_θ and a specular appearance MLP S_θ (Fig. 2). We compute the density σ_x, diffuse color $\mathbf{c}_d$, and specular color $\mathbf{c}_s$:

$$D_\theta(\mathbf{f}_x) = (\sigma_x, \mathbf{c}_d, s, \mathbf{h}_x), \tag{4}$$

$$S_\theta(\mathrm{sh}\left(\boldsymbol{\omega}\right), \mathbf{h}_x) = \mathbf{c}_s, \tag{5}$$

where $\mathbf{f}_x$ is a hashed feature of sample $\mathbf{x}$, s is the specular tint coefficient, $\mathbf{h}_x$ is an intermediate latent feature output from D_θ, and sh is the projection of $\boldsymbol{\omega}$ into the first 16 spherical harmonic coefficients. The final color $\mathbf{c}_x$ is:

$$\mathbf{c}_x = \mathbf{c}_d + s \cdot \mathbf{c}_s. \tag{6}$$

Unlike Verbin et al. [2022], we do not estimate a surface normal around which to reflect a specular ray because our input data for large scenes are sparse. We will discuss this choice shortly.

(a) Ours (b) Normal-based

Fig. 4. **Using reflected ray directions avoids artifacts.** Reconstructing normals from sparse inputs needed for large scenes is difficult. Instead, we represent reflections using a virtual scene geometry at reflected path length.

Tile-based voxel hashing and $\mathbf{f}_x$. For fast optimization, we make our MLPs shallow and pass features as inputs into the diffuse MLP. The features come from a multi-resolution voxel grid and hash tables [Müller et al. 2022]: $\{\Phi^l_{\theta_l}\}^L_{l=1}$, where Φ denotes the hash tables and θ_l denotes the features within each hash table. Each tile has a multi-resolution voxel grid with $L = 16$ levels. Given a 3D point sample $\mathbf{x}$, at each level, we hash its eight neighbor vertex integer coordinates, look up the corresponding feature in the hash table, and linearly interpolate the features with position weights of $\mathbf{x}$ from its surrounding neighbors to produce one output feature per level. We concatenate the output features from all levels to produce $\mathbf{f}_x$.

In addition, to quickly skip empty space and reduce the influence of floating density in undersampled regions, we use an occupancy field $\{0, 1\}$ constructed as a voxel grid in a coarse-to-fine manner, where low density voxels are set to 0 to indicate empty space at regular intervals during optimization.

Foreground and contracted background. Capturing outdoor scenes requires representing unbound space such as distant buildings or sky. As such, each tile contains both foreground and background regions. Representing a per-tile background instead of a per-scene background [Zhang et al. 2020] lets us pick the closest background to a novel view ray's exit from the tiled region to retain sharpness.

The distinction between foreground and background regions is defined by a ray contraction function [Reiser et al. 2023]: The inner foreground region of a tile contains a linear space mapping, and the outer background region of a tile contains a non-linear mapping. This allows both regions to be represented by the same voxel hash grid. To map unbound background into a bounded cubic region, we transform the position of 3D points using a contraction function with L_∞ norm (Appendix B). The same contraction function is also proposed by concurrent work in the Nerfacto model of Nerfstudio [Tancik et al. 2023].

Our approach exhibits several advantages over recent work. Compared to MERF [Reiser et al. 2023], our contraction function removes discontinuities at grid boundaries, leading to sharper results with less noise (Fig. 5). Compared to spherical contraction functions in Mip-NeRF 360 [Barron et al. 2021] and NeRF++ [Zhang et al. 2020], our contracted cubic bounded region matches the shape of the multi-resolution hash grid better and so reduces wasted space.

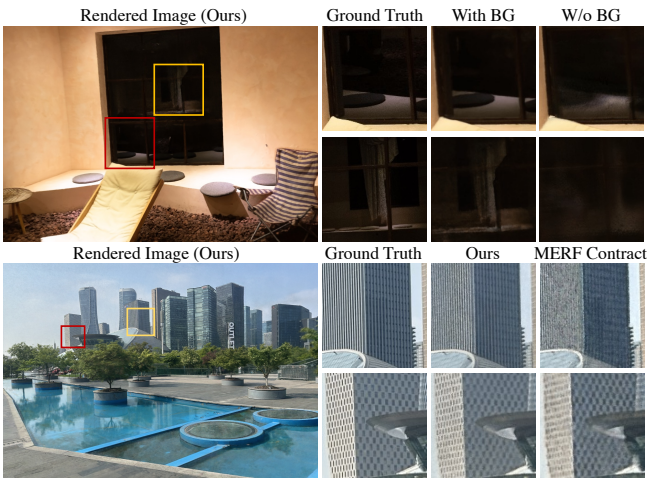


Fig. 5. **Background fields and our spatial contraction function.** *Top*: Reconstructing a virtual image behind window glass using a background field to accurately simulate reflections. *Bottom*: Our contraction mapping function is capable of producing sharper results with less noise compared to the mapping function in MERF [Reiser et al. 2023].

Fig. 6. **Lighting decomposition occurs as a byproduct of optimizing our representation.** Foreground (FG) and background (BG) each include both diffuse and specular components; diffuse and specular each include both foreground and background components.

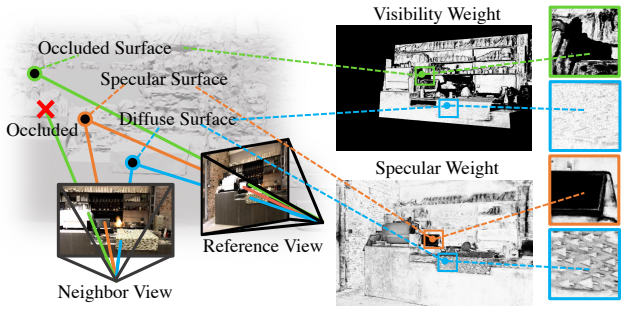


Fig. 7. **Specular-aware warping loss exploits lighting decomposition.** Considering visibility and likely specular reflections from the chained MLPs produces adaptive weights that are close to 0.

5.2 Discussion

Our specular MLP takes the view direction $\boldsymbol{\omega}$ as input, not the direction reflected by normal as in Verbin et al. [Verbin et al. 2022]. The normal calculated as the derivative of density wrt. 3D coordinates was too noisy when using sparsely captured images for large scale scenes (Fig. 4). In addition, if we only leverage the specular MLP to represent view-dependent reflections, reflected objects far from reflective surfaces were often too blurred compared to the ground truth or even missing entirely after optimization (Figs. 4 and 5), and had knock-on effects that degraded distributed camera pose refinement (Sec. 7).

Instead, our approach can represent reflections as virtual geometry and appearance 'behind' surfaces at the reflected ray path length [Sinha et al. 2012; Wu et al. 2022]. Recall also that both the foreground and background regions are decoded by the same diffuse and specular MLPs such that view-dependent appearance can exist in both. Coupling these two attributes with foreground and contracted background tiles lets us model both indoor and outdoor scenes from sparse inputs with the same representation (Fig. 5), and decompose diffuse and specular effects without added input (Fig. 6).

In outdoor scenes, far background objects are naturally reconstructed at their correct distance in the background region. This includes view-dependent effects such as reflections from glass-sided skyscrapers (Figs. 1 and 5). In indoor scenes, high-frequency reflections on planar surfaces (such as on window glass) are reconstructed as virtual images within diffuse background areas as their total reflected path length lies outside a tile (Fig. 5). Planar specular reconstruction 'behind' diffuse surfaces has a higher 3D consistency than reconstruction as a view-dependent appearance when given sparse inputs. Background modeling here can also extend to view-dependent effects in reflected path length appearance due to the specular MLP. Surfaces with glossy (or lower-frequency) reflections are more tolerant to reconstruction error, and are reconstructed within the tile foreground by the view-dependent specular MLP. This also provides some curved surface reflection reconstruction.

6 OPTIMIZATION LOSSES

We use four losses: a photometric loss $\mathcal{L}_c$, a specular-aware warping loss $\mathcal{L}_w$, and two regularizing losses for depth $\mathcal{L}_d$ and surface smoothness $\mathcal{L}_s$:

$$\mathcal{L} = \lambda_c \mathcal{L}_c + \lambda_w \mathcal{L}_w + \lambda_d \mathcal{L}_d + \lambda_s \mathcal{L}_s, \tag{7}$$

where the trade-off weights $\lambda_c, \lambda_d, \lambda_s$, and λ_w balance the loss terms.

We denote pixel coordinates as p, or $p^{i,j}$, with the ray corresponding to p as $\mathbf{r}(p)$.

Photometric loss $\mathcal{L}_c$. This supervises view synthesis by minimizing the difference between a rendered ray color $\mathbf{c}$ and its matched input image pixel color $\hat{\mathbf{c}}$ summed over (a batch of) tile rays $\hat{\mathcal{R}}^k$:

$$\mathcal{L}_c = \sum_{\hat{\mathbf{r}} \in \hat{\mathcal{R}}^k} \|\hat{\mathbf{c}}_\mathbf{r} - \mathbf{c}_\mathbf{r}\|_2^2. \tag{8}$$

Specular-aware warping loss $\mathcal{L}_w$. Inspired by classic image-based rendering (IBR), this loss encourages density to be localized and smooth by using rendered depth maps to reproject (or *warp*) predicted ray colors between neighboring camera poses (Fig. 8). Classically, this task is difficult because specular appearance cannot warp correctly, but our representation lets us downweight likely specular effects. As this loss involves reprojection operation, it provides a second optimization path to refine the camera poses that is based on scene geometry. This mitigates the shape-radiance ambiguity problem [Zhang et al. 2020].

To select views within neighborhood $\mathcal{N}_p$, we define an IBR cost e_{IBR} similar to past work [Buehler et al. 2001; Hedman et al. 2016]:

$$e_{\text{IBR}}\left(\mathbf{o}_i, \mathbf{o}_j, \mathbf{x}\right) = 0.9 e_{\text{Angle}} + 0.1 e_{\text{Dist}}, \tag{9}$$

$$e_{\text{Angle}} = 1.0 - \cos\left(\mathbf{o}_i \rightarrow \mathbf{x} \rightarrow \mathbf{o}_j\right), \tag{10}$$

$$e_{\text{Dist}} = \max\left(0, 1 - \frac{\|\mathbf{o}_i - \mathbf{x}\|_2}{\|\mathbf{o}_j - \mathbf{x}\|_2}\right), \tag{11}$$

where world point $\mathbf{x}$ is obtained by the inverse projection of p using its rendered depth. e_{Angle} employs the angle between the direction vectors from $\mathbf{x}$ towards the camera positions $\mathbf{o}_i$ at reference view and

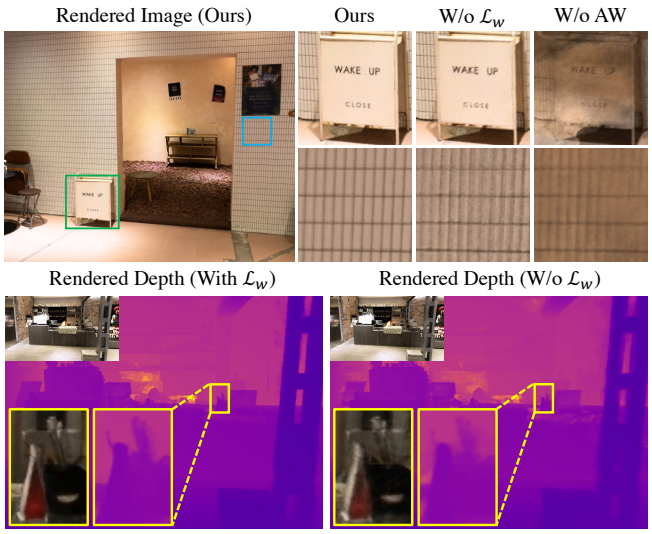


Fig. 8. **Adaptive-weight (AW) warping recovers sharper color and depth.** Yellow boxes correspond to color and depth for the marked region.

$\mathbf{o}_j$ at a neighboring view. e_{Dist} depends on the ratio of the distances between the current position $\mathbf{x}$ and $\mathbf{o}_i$ and $\mathbf{o}_j$.

Including neighboring rays adds computational cost in optimization, so we must pick a small set of important rays. Inspired by multi-view stereo methods, we choose neighborhood rays between 15° and 45°: too small a ray angle provides limited depth information given limited input image resolution, and too large a ray angle is more likely to suffer from occlusions and significant color variations—we will downweight occluded or specular rays, so their inclusion is not worthwhile. We choose neighborhood views with e_{Dist} distances between 0 and 0.2 such that we reduce blur by comparing only with neighboring cameras closer to a world point than the reference view (via max()) and such that the world space scale of a pixel does not vary significantly. Resolving these parameters means that views are selected when $0.030 < e_{\text{IBR}} < 0.176$.

Having defined a neighborhood $\mathcal{N}_p$, the specular-aware weight loss is a normalized weighting of input pixels:

$$\mathcal{L}_w = \sum_{p\in\mathcal{P}} \frac{\sum_{p'\in\mathcal{N}_p} w_{pp'} ||\hat{\mathbf{c}}_p - \hat{\mathbf{c}}_{p'}||_2^2}{\sum_{p'\in\mathcal{N}_p} w_{pp'}}, \tag{12}$$

$$\hat{\mathbf{c}}_{p'} = \text{warp}\left(\hat{\mathbf{c}}_p, d_p\right),$$

where the warp$(\cdot)$ operation reprojects pixel p in the reference view to p' in the neighboring view according to its rendered depth d_p and samples a color.

The subtlety comes in defining adaptive weight $w_{pp'}$ that must reduce the influence of specular reflections and occlusions (Fig. 7). This consists of three sub-terms:

$$w_{pp'} = w_p^s \cdot w_{p'}^s \cdot w_{pp'}^{vis}. \tag{13}$$

The specular weights w_p^s and $w_{p'}^s$ are computed per camera and reduces the influence of specular reflections using the estimated specular tint s and color $\mathbf{c}_s$:

$$w_p^s = \exp\left(-(s\cdot\mathbf{c}_s)/\,\sigma_d\right), \tag{14}$$

where σ_d is the variance parameter to control the sensitivity of specular weight to specular reflections.

The visibility weight $w_{pp'}^{vis}$ between pixels p and p' is:

$$w_{pp'}^{vis} = \exp\left(-\left\|d_p - d_{p'}\right\|_2/\sigma_v\right), \tag{15}$$

where d_p and $d_{p'}$ are the rendered depth for the pixel coordinate and its corresponding reprojected coordinate. σ_v variance controls the sensitivity of the visibility weight to depth difference.

Depth loss $\mathcal{L}_d$ and smoothness loss $\mathcal{L}_s$. These two terms are prior-based regularizers that improve rendering quality. We use predicted depth and normal maps from a pretrained monocular deep neural network from Omnidata [Eftekhar et al. 2021]. Our depth loss is the same as in MonoSDF [Yu et al. 2022]:

$$\mathcal{L}_d = \sum_{r\in\mathcal{R}^k} \left\|(\alpha\hat{d}_r + \beta) - d_r\right\|_2^2, \tag{16}$$

where α and β scale and shift to align predicted relative depth $\hat{d}$ to the scene scale.

The smoothness loss $\mathcal{L}_s$ measures the consistency between the reconstructed normal derived from depth and a monocularly-predicted surface normal. Since the predicted normals are smooth, this loss can suppress reconstructed geometry fluctuations (Fig. 9). First, we sample a 2×2 patch $[i, i+1]\times[j, j+1]$ for each pixel $p^{i,j}$. Then, we average the predicted normals of these four pixels to obtain $\hat{\mathbf{N}}_{p^{i,j}}$. The smoothness loss encourages that the vectors between adjacent pixels along the x and y axis are perpendicular to $\hat{\mathbf{N}}_{p^{i,j}}$:

$$\mathcal{L}_s = \sum_{p\in\mathcal{P}} \Big(\left\|\hat{\mathbf{N}}_{p^{i,j}}^\top \mathbf{V}_{p^{i+1,j}}^x\right\|_1 + \left\|\hat{\mathbf{N}}_{p^{i,j}}^\top \mathbf{V}_{p^{i,j+1}}^y\right\|_1 + \left\|\hat{\mathbf{N}}_{p^{i,j}}^\top \mathbf{V}_{p^{i+1,j+1}}^x\right\|_1 + \left\|\hat{\mathbf{N}}_{p^{i,j}}^\top \mathbf{V}_{p^{i+1,j+1}}^y\right\|_1 \Big). \tag{17}$$

Vectors $\mathbf{V}_{p^{i,j}}^x$ and $\mathbf{V}_{p^{i,j}}^y$ are computed as follows:

$$\mathbf{V}_{p^{i,j}}^x = \pi\left(p^{i,j}, d_{r(p^{i,j})}\right) - \pi\left(p^{i-1,j}, d_{r(p^{i-1,j})}\right),$$

$$\mathbf{V}_{p^{i,j}}^y = \pi\left(p^{i,j}, d_{r(p^{i,j})}\right) - \pi\left(p^{i,j-1}, d_{r(p^{i,j-1})}\right).$$

Here, $\pi(\cdot)$ denotes the 2D to 3D inverse projection operator, and $i, i-1$ and $j, j-1$ comparisons define vectors with x,y directions pointing rightward and upward respectively.

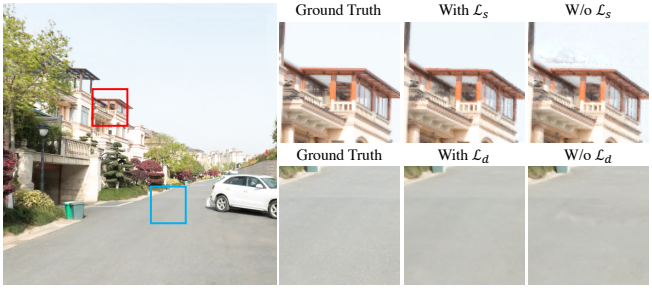


Fig. 9. **Effect of smoothness loss and depth loss**, which help to reduce floating artifacts and artifacts in areas lacking texture, respectively.

7 DISTRIBUTIVE OPTIMIZATION WITH ADMM

While the MLP parameters, hash tables, and occupancy grids of a tile are independent, the cameras that supervise tile reconstruction often see multiple tiles within their fields of view. We denote a camera as *overlapped* if its camera rays encode scene appearance in different tiles. Optimizing the pose parameters of overlapped cameras independently at each tile will produce inconsistent poses with inconsistent density and appearance at tile boundaries. Inspired by Zhang et al. [2017], we model the distributive optimization of our camera poses and tile-based neural scene representation as a global consensus problem that can be solved with the alternating direction method of multipliers (ADMM) [Boyd et al. 2011]. ADMM allows data parallel optimization for multi-GPU computers and for networked computers, especially as only the pose variables of overlapped cameras need to be transferred between tiles to achieve global camera consensus.

Definition. The global consensus problem is:

$$\min \mathcal{L}(\theta, \mathcal{T}) = \sum_k \mathcal{L}_k\left(\theta^k, \mathcal{T}^k\right), \tag{18}$$

$$s.t.\ \mathbf{T}_i^{k_i} = \mathbf{Z}_i, k_i \in \{k_1, ..., k_{N_i}\}.$$

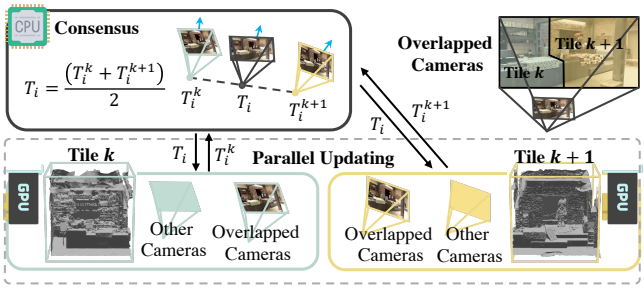


Fig. 10. **ADMM overview.** Parallel optimization of tiles and camera poses is interjected with consensus steps that broadcast and synchronize poses.

Here, θ^k is the MLP parameters and multi-resolution features independently optimized at each tile, and $\mathcal{T}^k$ is the set of camera poses for tile k. $\mathbf{T}_i^{k_i}$ is the duplicated pose variables of the overlapped camera T_i, where k_i is the index on the set of overlapped tiles and N_i is the number of tiles that share T_i. Variable $\mathbf{Z_i}$ is introduced to guarantee the consistency of the pose parameters.

Algorithm. ADMM alternates between two steps until convergence (Fig. 10): 1) *Parallel update step*, where per-tile network weights, hash table features, and relevant camera poses are optimized. For this step, the pose parameters of overlapped cameras are duplicated into independent variables. 2) *Consensus step*, where the overlapped camera parameters are broadcast and synchronized to preserve global consensus.

We apply the augmented Lagrange method to minimize Eq. (18) [Boyd et al. 2011]. Suppose, at iteration t, the global variables $\left(\theta^k, \mathcal{T}^k\right)^t$ of each tile k, $(\mathbf{Z}_i)^t$ and Lagrange multipliers $(\tilde{T}_i^k)^t$ are known.

1) Parallel update step. We minimizes each $\mathcal{L}_k$ in parallel on multiple GPUs to obtain the updated $(\theta_k, \mathcal{T}_k)^{t+1}$:

$$\left(\theta^k, \mathcal{T}^k\right)^{t+1} = \arg\min\left(\mathcal{L}_k\left(\theta^k, \mathcal{T}^k\right) + h\left(\mathcal{T}^k\right)\right),$$
$$h\left(\mathcal{T}^k\right) = \frac{\rho}{2}\sum_{T_i^k \in \mathcal{T}^k}\left\|\left(T_i^k\right)^t + \left(\tilde{T}_i^k\right)^t - (\mathbf{Z}_i)^t\right\|_2^2, \tag{19}$$

where ρ is set to 200 in our experiments, and $(\tilde{T}_i^k)^0$ is set as zero. $h\left(\mathcal{T}^k\right)$ is the augmented Lagrangian term.

2) Consensus step. We aggregate information from tiles to obtain updated $(\mathbf{Z}_i)^{t+1}$ as follows:

$$(\mathbf{Z}_i)^{t+1} = \sum_{k\in\{k_1,\ldots,k_{N_i}\}}\left(T_i^k\right)^t / N_i,$$
$$\left(\tilde{T}_i^k\right)^{t+1} = \left(\tilde{T}_i^k\right)^t + \left(T_i^k\right)^t - (\mathbf{Z}_i)^t, \tag{20}$$

where N_i is the number of tiles that share T_i. To reduce synchronization cost between GPUs, we perform the consensus step in Eq. (20) after every 100 parallel update steps (optimization iterations).

Stopping criterion. This is defined via the primal residual $\mathbf{r}^t$ that measures the global camera consensus and the dual residual $\mathbf{s}^t$ that measures the relative camera movement between successive iterations [Boyd et al. 2011]:

$$\left\|\mathbf{r}^t\right\|_2^2 = \sum_k\left\|\left(T_i^k\right)^t - (\mathbf{Z}_i)^t\right\|_2^2, \tag{21}$$
$$\left\|\mathbf{s}^t\right\|_2^2 = \sum_i\left\|(\mathbf{Z}_i)^{t+1} - (\mathbf{Z}_i)^t\right\|_2^2. \tag{22}$$

We stop optimization based on the average of $\mathbf{r}^t$ and $\mathbf{s}^t$. Upon stopping, our refined poses and tiles produce sharper scenes (Fig. 11).

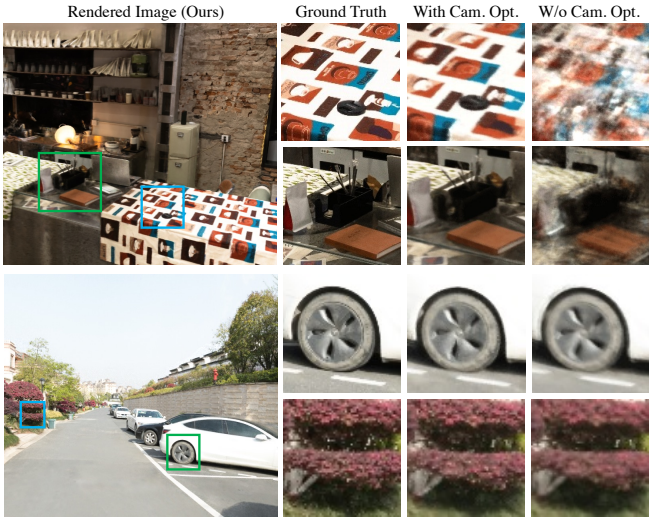


Fig. 11. **ADMM effect.** Results with and without camera pose optimization. The floating artifacts in the result of W/o camera pose optimization are due to the inaccurate camera poses obtained via SFM.

7.1 Discussion

While ADMM requires the objective function to be convex, ADMM also converges for non-convex objective functions if the gradient is Lipschitz-continuous and the parameter ρ is greater than the Lipschitz constant of the gradient function [Kaplan and Tichatschke 1998; Zhang et al. 2017]. We observe that the gradient of our loss wrt. the network weights and hash table features is bounded in our experiments and thus can be viewed as Lipschitz-continuous.

In our design, consider that there are two paths to back-propagate the loss gradient to the camera poses:

Via 3D points. The loss gradient wrt. the network weights and hash table features will be back-propagated to the camera pose via sampled 3D points. The gradient of a 3D point $\mathbf{x}$ wrt. the camera position $\mathbf{o}_i$ can be calculated using $\mathbf{x} = \mathbf{o}_i + t\mathbf{R}_i\mathbf{d}$, where $\mathbf{d}$ is a fixed (not optimized) direction corresponding to a pixel in the *local* coordinate system of the camera i. The gradient of $\mathbf{x}$ with respect to $\mathbf{o}_i$ is Lipschitz-continuous. For $\mathbf{R}_i$, in our implementation we use axis-angle to represent rotation matrices. The gradient of $\mathbf{R}_i$ with respect to its axis-angle representation is Lipschitz-continuous [Zhang et al. 2017]: Since $\mathbf{R}_i\mathbf{d}$ is a linear operation, the gradient of $\mathbf{x}$ wrt. $\mathbf{R}_i$ is Lipschitz-continuous.

Via the specular-aware warping loss The gradient of the warp loss can be back-propagated to camera poses since it transforms a local 3D point in a reference view into the global coordinate system and then projects it to a neighboring view by the camera pose parameters. The local to global transformation is also a linear operation, and the gradient of projection operation wrt. a camera pose is also Lipschitz-continuous if the z component of the local coordinate of the transformed point, i.e., the depth value, in the neighboring view is greater than 0 [Zhang et al. 2017]. This can be guaranteed in 3D point sampling.

Thus, ADMM will converge with a proper value of ρ.

8 POST-OPTIMIZATION MULTI-TILE RENDERING

After optimizing camera poses and tiles, at final render time we must integrate the per-tile neural radiance fields to create novel views. We assume that world point quality will be best within tile foregrounds; this implies a sampling strategy for the photographer.

Multi-tile rendering proceeds in four steps:

(1) Determine the set of tiles intersected by r, sample points along r in each tile, and include them given each tile's occupancy field;
(2) Compute the color and density for each point in the foreground fields. As tile foregrounds overlap by 20%, we blend point color and density values from the set of foregrounds to achieve seamless rendering;
(3) For points beyond the farthest intersected tile's foreground, compute the color and density for each point in the background field of the farthest tile;
(4) Compute the final pixel color via volume rendering.

Step 1: 3D point sampling. We intersect $\mathbf{r}$ with $\mathcal{K}$ and sort the tiles nearest to farthest. Within each tile's intersected ray segment, we sample 128 points inside non-empty voxels in the occupancy field. Then, we compute transmittance T_n along the ray:

$$T_n = \prod_{m=0}^{n-1} \exp(-\sigma_m \delta_m). \tag{23}$$

When T_n falls below 1e-5, we terminate 3D point sampling.

For the background of the farthest tile, we again sample 128 background points linearly before contraction. This usually produces sharper results as rays assigned to the farthest tile are closest to the true background world points.

Step 2: Foreground volume rendering with point-based blending. If a sample point $\mathbf{x}_n$ lies in a tile overlap region, we apply point-based linear blending:

$$\mathbf{c}_f = \sum_{n=0}^{N} T_n^f \frac{\sum_{k\in\mathcal{S}(\mathbf{x}_n)} w_n^k \cdot (1-\exp(-\sigma_n^k\delta_n))\cdot \mathbf{c}_n^k}{\sum_{k\in\mathcal{S}(\mathbf{x}_n)} w_n^k}, \tag{24}$$

$$T_n^f = \prod_{m=0}^{n-1} \frac{\sum_{k\in\mathcal{S}(\mathbf{x}_m)} w_m^k \cdot \exp(-\sigma_m^k\delta_m)}{\sum_{k\in\mathcal{S}(\mathbf{x}_m)} w_m^k},$$

where $\mathcal{S}(\mathbf{x}_n)$ is the set of tiles that covers the point $\mathbf{x}_n$. For $\mathbf{x}_n$, we blend $(1-\exp(-\sigma_n^k\delta_n))\cdot\mathbf{c}_n^k$ and local transmittance $\exp(-\sigma_n^k\delta_n$ over a segment δ_n. We choose $(1-\exp(-\sigma_n^k\delta_n))\cdot\mathbf{c}_n^k$ since it automatically suppresses point colors associated with low density.

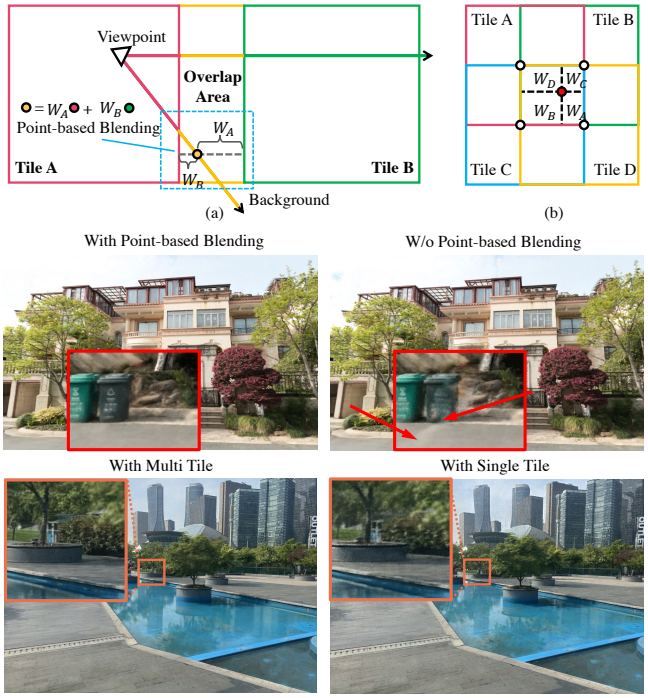


Fig. 12. **Multi-tile rendering with point-based blending** reduces blur and artifacts at tile boundaries.

Blending weight w_n^k is computed using the nearest 2, 4, or 8 tile voxels (e.g., two/linear in Fig. 12a, and four/bilinear in Fig. 12b).

Step 3: Background volume rendering with deferred point-based blending. Rays that exit overlapped foreground regions require background blending. We blend backgrounds to produce $\mathbf{c}_b$:

$$\mathbf{c}_b = \frac{\sum_{k\in\mathcal{S}(\mathbf{x}_N)} w_k \cdot \left(\sum_{n=N}^{N+128} \alpha_n^k \cdot \mathbf{c}_n^k\right)}{\sum_{k\in\mathcal{S}(\mathbf{x}_N)} w_n^k}. \tag{25}$$

In this equation, $\mathcal{S}(\mathbf{x}_N)$ is the set of tiles that covers the exit point $\mathbf{x}_N$, and $\sum_{n=N}^{N+128} \alpha_n^k \cdot \mathbf{c}_n^k$ is the integrated background segment color of tile k. Unlike in Eq. (24), colors and densities are integrated along the segment before blending—a deferred point-based blending—since the tile boundary used to compute weights does not exist in the background region. Therefore, we choose compute w_k for the exit 3D point located exactly on the tile boundary (Fig. 12).

Step 4: Final color. With $\mathbf{c}_f$ and $\mathbf{c}_b$, we obtain final ray color c:

$$\mathbf{c} = \mathbf{c}_f + T_N^f \mathbf{c}_b, \tag{26}$$

where T_N^f is the foreground accumulated transmittance. Multi-tile rendering gives sharper results than single-tile rendering, and point-based blending reduces visual artifacts between adjacent tiles (Fig. 12).

9 EXPERIMENTS

We evaluate the technical components in our pipeline via ablation studies, and provide system-level comparisons of our method to

Table 1. **Scene information.** #Img and #Tile denote the total number of captured images and tiles of a 3D scene. Area denotes the total area in square meters occupied by all tiles. Consensus ratio is defined as (the total number of overlapped cameras) / (total number of cameras).

Type	Scene	Area (m^2)	#Img	#Tile	Consensus ratio	Training (hours)
Indoor	*Bar*	80.00	1323	6	0.83	22.50
	Coffee Shop	138.61	1288	7	0.86	16.39
Outdoor	*Street*	800.00	1382	8	0.91	15.15
	Shady Path	675.00	551	3	0.51	15.26
	Community	916.56	396	4	0.92	16.16
	Park	258.00	223	2	0.70	10.43
	Rubble	200000.00	1678	8	0.64	35.00
	Polytech	15000.00	770	6	0.91	11.50

Table 2. **Tile size abaltion.** "Larger" denotes merging each two neighboring tiles. "Smaller" denotes splitting each tile into two sub-tiles.

Scene	Metric	Current	2× larger	2× smaller	4× smaller
Community	PSNR↑	24.27	24.02	23.54	23.02
	SSIM↑	0.738	0.720	0.721	0.708

state-of-the-art neural view synthesis and bundle-adjusting NeRF methods. Please also see our accompanying video.

Scenes. We conduct experiments on six scenes with different sizes, types, and material reflections (Tab. 1). Specifically, we use the *Coffee Shop* and *Bar* indoor scenes from Wu et al. [2022], and four outdoor scenes: *Street*, *Shady Path*, *Community*, and *Park*. To show that our method does not rely on specific photographic equipment, we use an iPhone 14 Pro Max to capture the *Park* scene and a Canon EOS 60D DSLR camera to capture the other outdoor scenes.

Metrics. We report results with Peak Signal-to-Noise Ratio (PSNR) and Structure Similarity Index Measure (SSIM) [Wang et al. 2004].

9.1 Tile-based Representation

9.1.1 Robustness to proxy mesh accuracy. To assess the robustness of our algorithm to the accuracy of the proxy mesh G, we perturb the mesh with varying noise levels. We added Gaussian noise along the normal of each vertex, with a mean equal to 0 and standard deviations of 1/40 and 1/20. Since we only use the proxy mesh as a guide to determine tile location and select optimizing rays, the method can handle significant noise levels without compromising performance (Fig. 13).

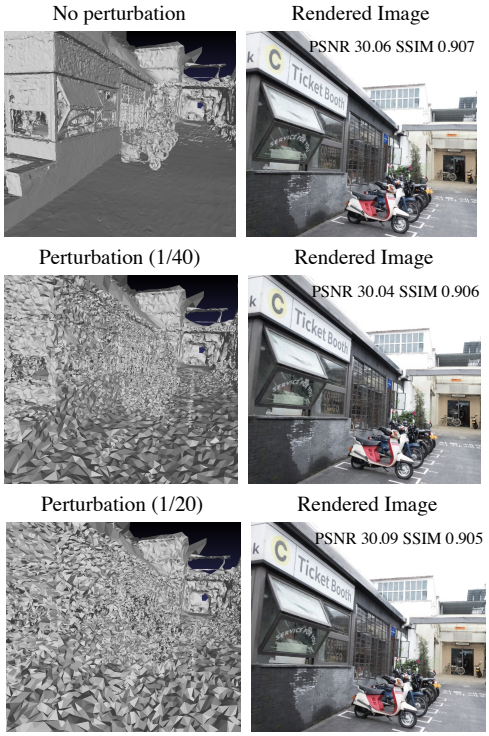


Fig. 13. **Robustness to proxy mesh accuracy.** The magnitude of noise applied to the reconstructed proxy mesh is increased from top to bottom. Our method is not sensitive to random geometric error in the proxy mesh.

9.1.2 Tile size. Tile size selection follows the principle of maximizing the size of each tile while ensuring that it can be trained on a GPU. Since the observation distances are different for indoor and outdoor scenes, we set the ground area of tiles to around 20 m^2 for indoor scenes and 100–200 m^2 for outdoor scenes (Tab. 1).

To validate our current tile size, we vary tiles sizes by merging and splitting current tiles (Tab. 2) This provides evidence for our current approach as both PSNR and SSIM decrease. The current tile size strikes a balance between 1) preserving details, which favors smaller tile sizes, and 2) global consistency (along with optimization efficiency), which favors larger tile sizes.

9.1.3 ADMM. Our joint optimization of network weights and camera poses can achieve sharper rendering results than without ADMM-based camera pose optimization (Figs. 11 and 14b). When camera refinement is not applied, both indoor and outdoor scenes exhibit a decrease in PSNR and SSIM scores (Tab. 3). We evaluate the ADMM hyperparameters ρ and SI (synchronization interval) on *Coffee Shop* (Fig. 14). This scene uses 1288 images where 86% of cameras overlap. Increasing ρ can improve consistency between overlapped cameras. More frequent synchronization by reducing SI benefits inter-tile camera consistency but causes more CPU-GPU data interaction, resulting in a slower overall optimization speed. Finally, increasing the synchronization interval from 1 to 1000 improves training time. To balance rendering quality and training speed, we set $SI = 100$ for our experiments.

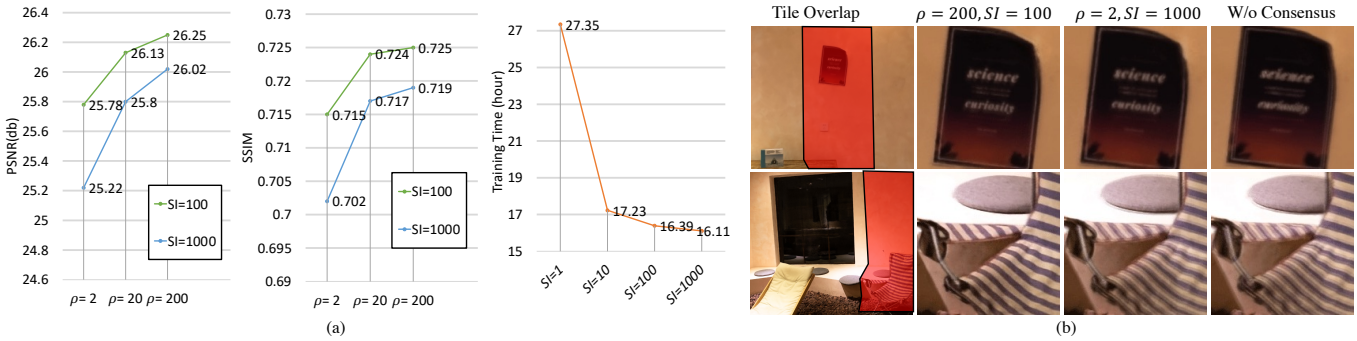


Fig. 14. **ADMM parameters influence overall optimization speed and achieved quality.** Quantitative and qualitative comparisons with different ADMM parameter settings show that lower synchronization intervals (SI) and higher ρ lead to better rendering quality, but lower SI adds computational cost.

Table 3. **Ablation studies.** Best results are highlighted as 1st, 2nd and 3rd. AW: Adaptive Weight. PB: Point-based Blending.

Scene	Metric	Full	W/o Cam. Opt.	W/o Consensus	W/o $\mathcal{L}_w$	W/o $\mathcal{L}_d$	W/o $\mathcal{L}_s$	W/o AW	W/o Specular	W/o PB
(Indoor)	PSNR↑	26.25	25.38	24.85	25.76	26.24	26.31	18.16	26.08	25.94
Coffee Shop	SSIM↑	0.725	0.696	0.689	0.709	0.725	0.725	0.557	0.710	0.722
(Outdoor)	PSNR↑	24.27	24.13	23.47	24.16	24.17	24.00	16.59	24.04	23.15
Community	SSIM↑	0.738	0.733	0.695	0.732	0.733	0.726	0.540	0.729	0.729

9.1.4 Ablation studies. Our method section shows qualitative comparisons to help build intuition; here, we show quantitative analyses of our loss terms by excluding each of them individually from optimization (Tab. 3). The photometric loss $\mathcal{L}_c$ is always included.

Removing camera pose refinement entirely—"W/o Cam. Opt."—reduces quality, as does removing the consensus step from ADMM ("W/o Consensus") to make it redundant. These two studies verify the importance of global camera consensus.

The prior losses help to reduce fluctuations in the geometric reconstruction from our relatively sparse captured data. In *Coffee Shop*, removing the normal smoothness term slightly *increases* PSNR. For scene areas with reflected content, such as high-frequency reflections in glass windows, there can be a mismatch in this loss: as it is supervised on mostly-synthetic data and with semantics, the screen-space normal predicted by the Omnidata network tends to describe the glass surface and not the reflected geometry. Penalizing the loss given this mismatch can cause more blurry reflections.

The warping loss improves PSNR/SSIM, and the adaptive weighting is a critical component as PSNR/SSIM drop severely when it is removed. Without adaptive weighting, the error in the predicted depth and specular components of a pixel will lead to incorrect optimization gradients, especially at the beginning of optimization. The column "W/o Specular" shows that disabling the specular weight within adaptive weighting decreases performance: achieving a diffuse/specular decomposition can suppress the influence of view-dependent effects upon the optimization.

Continuing with reflections, we show what visual content the specular MLP encodes by disabling background encoding of reflections during optimization (Fig. 15). To achieve this, we stop sampling points along a ray once those points leave the foreground box, e.g., in *Coffee Shop*, the foreground box ends just behind the large metal door. With this technique, we can see that the specular MLP can still reconstruct some reflections from the metal door within the sparse capture setting for large-scale scenes on its own. But, combining it with the background reflection encoding is necessary.

We also evaluate the influence of point-based blending. In "W/o PB", we pick the tile with the largest weight to infer color and density for each overlapped sample point. This blending scheme improves PSNR/SSIM scores even though it only focuses on tile boundaries because most rays in large scenes intersect multiple tiles.

9.2 Method Comparisons

9.2.1 Comparisons with scene rendering methods. We compare our method to other state-of-the-art novel view synthesis methods on indoor and outdoor scenes both quantitatively (Tab. 4) and qualitatively (Fig. 17). All comparisons use original author's implementations and hyperparameters, if available.

For indoor scenes, we compare our method with **DeepBlending** [Hedman et al. 2018], **Stable View Synthesis (SVS)** [Riegler and Koltun 2021] , **Instant-NGP** [Müller et al. 2022] and **Scalable Neural Indoor Scene Rendering (SNISR)** [Wu et al. 2022]. Deep Blending and SVS both use proxy geometry. Instant-NGP and our method use hash grids; both were trained for 40,000 iterations. SNISR, which uses voxels only, is stated by Wu et al. as requiring twice as many iterations.

For the comparisons on outdoor scenes, we compare with **Mega-NeRF** [Turki et al. 2022], **Block-NeRF** [Tancik et al. 2022], **SVS** [Riegler and Koltun 2021] and **Instant-NGP** [Müller et al. 2022]. The source code for Block-NeRF is not publicly available; thus, we attempt to re-implement it. Their data is spherical camera views from Google Street View. Since our outdoor scenes use different data, we replace their spheres with our tiles to reuse our scene-partition

Fig. 15. **The ablation study of specular MLP.** The demonstrated decomposition result is achieved by disabling background encoding of reflections during optimization.

Fig. 16. **Comparisons** with Nerfacto [Tancik et al. 2023] (top) and Mega-NeRF [Turki et al. 2022] on data *rubble* (bottom).

Table 4. **Quantitative comparisons with neural scene rendering methods.** Best results are highlighted as 1st, 2nd and 3rd.

Indoor Scene	Metric	Deep Blending	SVS	Instant-NGP	SNISR	Ours
Bar	PSNR↑	26.14	27.15	27.44	26.69	28.77
	SSIM↑	0.754	0.791	0.740	0.739	0.775
Coffee Shop	PSNR↑	21.78	24.84	23.77	24.13	26.25
	SSIM↑	0.633	0.751	0.630	0.676	0.725
Average	PSNR↑	23.96	26.00	25.61	25.41	27.51
	SSIM↑	0.694	0.771	0.685	0.708	0.750

Outdoor Scene	Metric	Mega NeRF	Block NeRF	SVS	Instant-NGP	Ours
Street	PSNR↑	22.04	20.94	25.91	22.16	25.85
	SSIM↑	0.707	0.641	0.879	0.686	0.825
Shady Path	PSNR↑	19.04	18.22	18.83	19.11	20.43
	SSIM↑	0.356	0.317	0.517	0.372	0.554
Community	PSNR↑	22.34	20.98	22.79	22.32	24.27
	SSIM↑	0.576	0.538	0.751	0.609	0.738
Park	PSNR↑	22.76	20.33	20.55	20.72	23.70
	SSIM↑	0.629	0.559	0.671	0.579	0.701
Average	PSNR↑	21.55	20.12	22.02	21.08	23.56
	SSIM↑	0.567	0.514	0.704	0.562	0.705

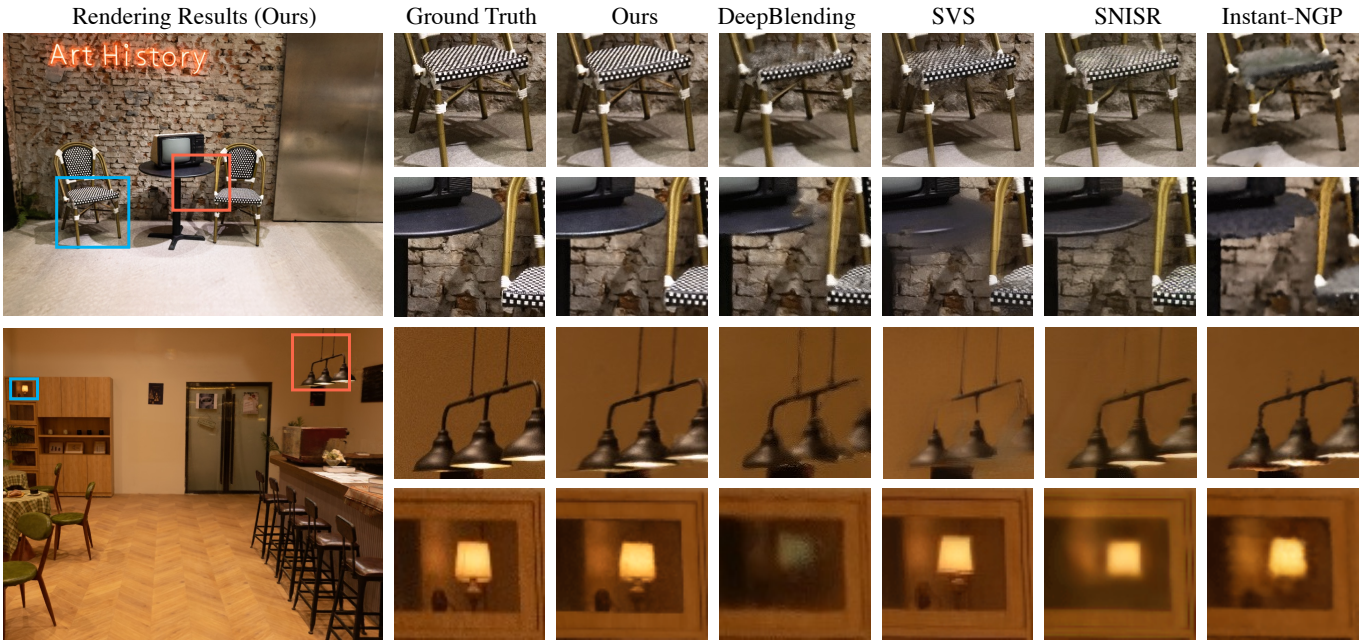
data structure, and then apply their image-space blending operation to a virtual viewpoint inside the overlapped regions of tiles. As our dataset was captured with little illumination variation, we also did not implement exposure value and appearance embedding optimization. We train Mega-NeRF, Block-NeRF, and Instant-NGP using our data with the same number of iterations as ours (40,000).

Our method achieves the highest PSNR scores on average (Tab. 4). Our rendered reflections are temporally smooth, unlike SVS. For outdoor scenes, our method produces sharper details in the trees and background buildings (Fig. 17b). Our Block-NeRF implementation tends to produce blurry rendering results for distant objects because Block-NeRF does not have contraction mapping and samples distant objects with low density. Similarly, Mega-NeRF also blurs distant objects despite using an unbound scene parameterization similar to NeRF++ [Zhang et al. 2020]. This is likely due to the use of a set of MLPs to represent the scene without a local feature grid, which can lead to slower optimization convergence.

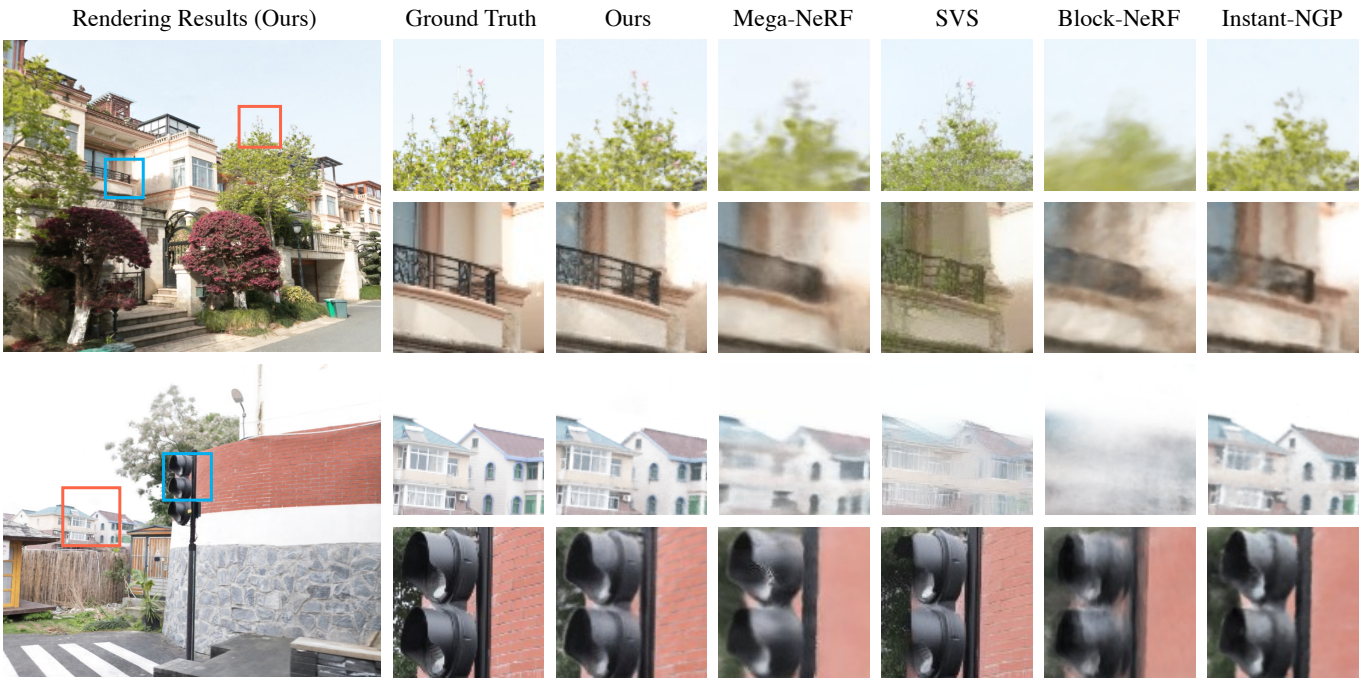
As our approach uses the same contraction function as the contemporaneous Nerfacto model in Nerfstudio [Tancik et al. 2023], we compare to Nerfacto [Tancik et al. 2023] on *Park* (Fig. 16). Our rendering results preserve more details: The quantitative results indicate that our approach achieves a PSNR of 23.07 and SSIM of 0.701, surpassing Nerfacto's PSNR of 20.07 and SSIM of 0.576.

We also evaluate our method on *Rubble* (1K resolution) from Mega-NeRF [Turki et al. 2022]. Since there is no proxy mesh provided for this aerial imagery, we use the near/far plane provided with this data to allocate eight tiles in a 2×4 horizontal layout. During training, we gradually update the training rays for each tile: we render depth maps for cameras inside a tile, then share depth with other tiles to provide visibility information for their training ray selection. As in Mega-NeRF, we train the model for 500K iterations and sample 512/256 ray points in foreground/background regions. In qualitative comparisons (Fig. 16), our method generates renderings with more details than Mega-NeRF. However, our method achieves PSNR/SSIM of 22.93/0.673 while Mega-Nerf achieves 24.71/0.586. This discrepancy is explained when considering that Mega-NeRF better handles lighting differences in input images with an appearance embedding vector. Such embedding vectors could also be applied to our method to improve PSNR/SSIM.

9.2.2 Comparisons with bundle-adjusting NeRF methods. We compare to **BARF** [Lin et al. 2021], **Nope-NeRF** [Bian et al. 2023], **GARF** [Chng et al. 2022], and **NeRFmm** [Wang et al. 2021]. We use the open-source implementations of BARF, Nope-NeRF, and NeRFmm. For GARF, we replace the activation function in BARF with the Gaussian activation function. We set the camera pose initialization, batch size, and total training iterations of all methods to be the same as ours. Our method achieves the highest PSNR/SSIM score on *Coffee Shop* (Tab. 5) and produces visually sharper results (Fig. 18). Nope-NeRF produces significantly poorer results than other methods. This may be because Nope-NeRF adds relative pose constraints to consecutive pairs of images, which is designed for sequential images and may not be suitable for our out-of-order images. For BARF and GARF, the default weighting scheme and controllable parameter in BARF's positional encoding strategy and the default variance for GARF's Gaussian activation function may be more suitable for forward-facing scenes rather than large-scale scenes like ours. To verify these assumptions, we optimize a part of *Coffee*

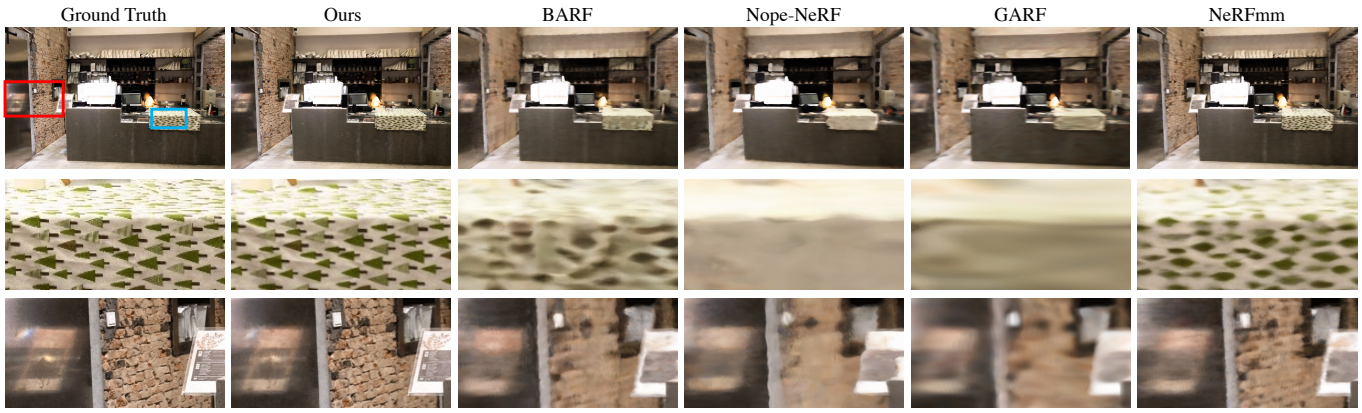


(a) Indoor Scene Comparisons

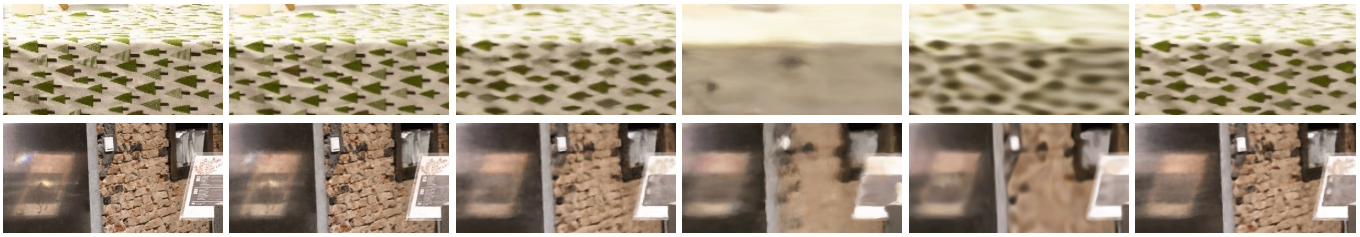


(b) Outdoor Scene Comparisons

Fig. 17. **Comparisons with SOTA neural scene rendering methods show improved quality of our method.** (a) Comparisons with **DeepBlending** [Hedman et al. 2018], **SVS** [Riegler and Koltun 2021], **SNISR** [Wu et al. 2022] and **Instant-NGP** [Müller et al. 2022] on large indoor scenes. Our method can produce higher-quality renderings on high-frequency textured areas like chairs, on thin objects, and for reflected light. (b) Comparisons with **Mega-NeRF** [Turki et al. 2022], **SVS** [Riegler and Koltun 2021], **Block-NeRF** [Tancik et al. 2022], and **Instant-NGP** [Müller et al. 2022] on large outdoor scenes. Our method can reconstruct foreground elements and background buildings with sharper details.



(a) Whole Scene Test



(b) A Cut-out Forward-facing Scene Test

Fig. 18. **Bundle-adjusting NeRF comparisons.** With respect to **BARF** [Lin et al. 2021], **Nope-NeRF** [Bian et al. 2023], **GARF** [Chng et al. 2022], and **NeRFmm** [Wang et al. 2021], our approach fares favorably. The renderings of BARF look better than GARF, even though GARF's PSNR/SSIM is higher due to the smooth nature of both GARF's Gaussian activation functions and the averaging nature of the metrics themselves. The rendering results after refining camera poses for the whole scene or a cut-out forward-facing scene inside a tile are shown in (a) and (b), respectively.

Table 5. **Quantitative comparisons with bundle-adjusting NeRF methods.** Our method achieves higher photometric scores.

Scene	Metric	BARF	Nope-NeRF	GARF	NeRFmm	Ours
Coffee Shop	PSNR↑	20.45	19.92	20.89	21.06	26.25
	SSIM↑	0.538	0.542	0.542	0.548	0.725

Shop covered by a single tile and with only forward-facing input cameras. Except for Nope-NeRF, the comparison methods achieved better results. Our method achieves equivalent performance in both settings; that this performance is better than competing methods shows the promise of our camera pose optimization design.

10 DISCUSSION

Appearance decomposition. Since our design can represent reflections either in the background as virtual images [Sinha et al. 2012; Wu et al. 2022] or as view-dependent specular at a 3D point, there is ambiguity regarding how a reflection should be represented. In our experiments, we observed that this ambiguity is automatically resolved in most cases during optimization: the representation prioritizes modeling planar and sharp reflections using the background radiance field (Fig. 6, first row). This may be because the diffuse components in the reflections are view-independent and consistent across multiple views, creating an easier optimization task to explain them as virtual imagery behind reflective surfaces by using contracted 3D points in the unbound background region. For reflections that cannot be well modeled in this way, our method models them using the foreground specular MLP, as shown in the reflections on the chair and coffee machine in the bottom two rows of Figure 6. This representation is similar to surface light fields [Wood et al. 2000] that model reflections as spatially-varying view-dependent signals that encode the ray at each surface point. The final class of reflections are those that fall into the background specular MLP. While rarer as creating view dependence in far objects requires inducing large parallax upon the background region (a kind of paradox), the background specular MLP can help to represent near-planar or glossy background reflections.

10.1 Limitations

Our method cannot accurately reconstruct both reflected and transmitted parts of appearance from window glass as our representation only allows reflection at each 3D point through the specular MLP and background encoding. Modeling curved reflections from sparse input is also challenging and our approach performs worse with curved reflectors. In *Street*, the planar window reflections are accurately reconstructed but the reflections on the curved car windshield are severely blurred (Fig. 19). Other works have integrated semantic information, such as the segmentation of windshields from priors, to improve the rendering quality of specific objects [Rodriguez et al. 2020]. Further, during rendering, we only use the background from

the last tile along a ray. In principle, some elements encoded in the background of nearer tiles may be incorrectly ignored. In practice, we have not observed artifacts from this limitation in our scenes.

Losses like the warping and normal smoothness loss add neighborhood terms to encourage geometry to be smooth; the warping loss further encourage geometry to be opaque. This can be beneficial or at odds with a scene's true geometry, especially at boundaries in outdoor scenes with sky background where sharp edges are required. Further, many outdoor scenes are dynamic—even with no fauna, the flora that blow in the wind still require spatio-temporal alignment. We do not model this, nor the previously-mentioned dynamic lighting effects like moving shadows or from cameras without locked exposure.

Our current GPU implementation of the multi-tile rendering algorithm is not optimized and takes 4–6 seconds to render a single 1280×720 image on a single Nvidia V100 GPU card. This is not suitable for interactive applications. Rendering speed can be further optimized by baking the radiance fields into octrees or meshes [Chen et al. 2022a; Hedman et al. 2021; Yu et al. 2021a] or leveraging shared GPU memory to store shallow MLPs [Wu et al. 2022]. In addition, the camera pose optimization in our method needs to be initialized through structure from motion, and investigating how to obtain camera poses for large scenes without initialization could improve end-to-end reconstruction times.

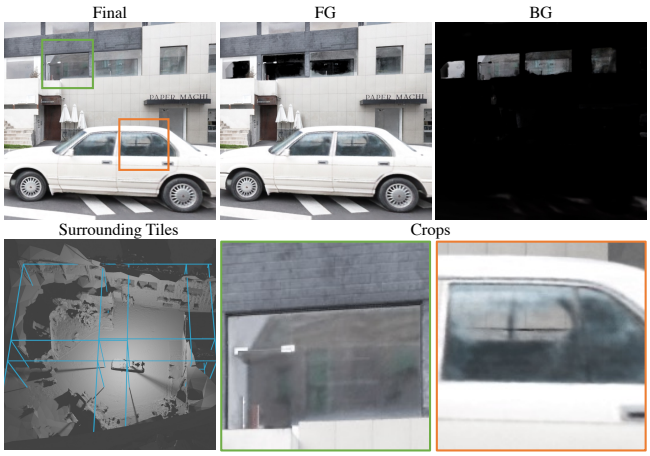


Fig. 19. **Limitations**: Rendering artifacts on a curved car windshield in *Street*. While the reflections on the planar window on the wall are accurately reconstructed, the reflections on the curved window shield are severely blurred. *Top Left:* The rendered image. *Bottom Left:* The geometric configuration of tiles surrounding the car. *FG/BG:* The FG/BG decomposition of this image. *Crops:* Zoomed-out views of the outlined pixels.

11 CONCLUSION

We have developed a distributive training method for constructing scalable bundle-adjusting neural radiance fields that opens up their application to large scale scenes while maintaining higher quality rendering than existing approaches. First, after partitioning a scene into overlapped tiles, a tile-based hybrid neural scene representation is proposed to combine a multi-resolution hash feature grid and shallow MLPs to represent the local radiance field at each tile. A key feature of our method is that we leverage unbound background regions to encode both distant objects and the virtual images of sharp reflections that are outside the tile bounding box. Second, an ADMM-based optimization algorithm is developed to achieve scene-level camera consensus after parallel per-tile joint optimization of scene representation and camera poses associated to the tile. With optimized scene representation and camera poses, our multi-tile rendering algorithm can produce high-quality scene rendering results by prioritizing the foreground radiance fields and removing discontinuities at tile boundaries using point-based blending. Experimental results show that our method outperforms state-of-the-art neural scene rendering methods and bundle-adjusting NeRF approaches in terms of the quality of the rendered images.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their professional and constructive comments. Weiwei Xu is partially supported by NSFC grant No. 61732016. Jiamin Xu is partially supported by NSFC grant No. 62302134. James Tompkin is partially supported by the US NSF CNS-2038897 and by CAREER IIS-2144956. Qixing Huang is partially supported by US NSF CAREER IIS-2047677. This paper is supported by Ant Group and Information Technology Center and State Key Lab of CAD&CG, Zhejiang University.

REFERENCES

Sameer Agarwal, Noah Snavely, Steven M Seitz, and Richard Szeliski. 2010. Bundle adjustment in the large. In *Eur. Conf. Comput. Vis.* 29–42.

Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. 2021. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Int. Conf. Comput. Vis.* 5855–5864.

Sai Bi, Zexiang Xu, Pratul Srinivasan, Ben Mildenhall, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. 2020. Neural reflectance fields for appearance acquisition. *arXiv preprint arXiv:2008.03824* (2020).

Wenjing Bian, Zirui Wang, Kejie Li, Jia-Wang Bian, and Victor Adrian Prisacariu. 2023. NoPe-NeRF: Optimising neural radiance field with no pose prior. In *IEEE Conf. Comput. Vis. Pattern Recog.* 4160–4169.

Mark Boss, Raphael Braun, Varun Jampani, Jonathan T Barron, Ce Liu, and Hendrik Lensch. 2021. Nerd: Neural reflectance decomposition from image collections. In *Int. Conf. Comput. Vis.* 12684–12694.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine learning* 3, 1 (2011), 1–122.

Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. 2001. Unstructured lumigraph rendering. In *Proc. of SIGGRAPH.* 425–432.

Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J. Leonard. 2016. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics* 32, 6 (2016), 1309–1332.

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

Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. 2022. Efficient geometry-aware 3D generative adversarial networks. In *IEEE Conf. Comput. Vis. Pattern Recog.* 16123–16133.

Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. 2022b. Tensorf: Tensorial radiance fields. In *Eur. Conf. Comput. Vis.* 333–350.

Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. 2022a. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. *arXiv preprint arXiv:2208.00277* (2022).

Shin-Fang Chng, Sameera Ramasinghe, Jamie Sherrah, and Simon Lucey. 2022. Gaussian activated neural radiance fields for high fidelity reconstruction and pose estimation. In *Eur. Conf. Comput. Vis.* 264–280.

Ronald Clark. 2022a. Volumetric bundle adjustment for online photorealistic scene capture. In *IEEE Conf. Comput. Vis. Pattern Recog.* 6124–6132.

Ronald Clark. 2022b. Volumetric bundle adjustment for online photorealistic scene capture. In *IEEE Conf. Comput. Vis. Pattern Recog.* 6124–6132.
Amaël Delaunoy and Marc Pollefeys. 2014. Photometric bundle adjustment for dense multi-view 3d modeling. In *IEEE Conf. Comput. Vis. Pattern Recog.* 1486–1493.
Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. 2022. Depth-supervised nerf: Fewer views and faster training for free. In *IEEE Conf. Comput. Vis. Pattern Recog.* 12882–12891.
Ainaz Eftekhar, Alexander Sax, Jitendra Malik, and Amir Zamir. 2021. Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. In *Int. Conf. Comput. Vis.* 10766–10776.
Jakob Engel, Vladlen Koltun, and Daniel Cremers. 2017. Direct sparse odometry. *IEEE Trans. Pattern Anal. Mach. Intell.* 40, 3 (2017), 611–625.
Jakob Engel, Thomas Schöps, and Daniel Cremers. 2014. LSD-SLAM: Large-scale direct monocular SLAM. In *Eur. Conf. Comput. Vis.* 834–849.
Anders Eriksson, John Bastian, Tat-Jun Chin, and Mats Isaksson. 2016. A consensus-based framework for distributed bundle adjustment. In *IEEE Conf. Comput. Vis. Pattern Recog.* 1754–1762.
Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. 2022. Plenoxels: Radiance fields without neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog.* 5501–5510.
Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. 2021. Fastnerf: High-fidelity neural rendering at 200fps. In *Int. Conf. Comput. Vis.* 14346–14355.
Haoyu Guo, Sida Peng, Haotong Lin, Qianqian Wang, Guofeng Zhang, Hujun Bao, and Xiaowei Zhou. 2022. Neural 3d scene reconstruction with the manhattan-world assumption. In *IEEE Conf. Comput. Vis. Pattern Recog.* 5511–5520.
Jon Hasselgren, Nikolai Hofmann, and Jacob Munkberg. 2022. Shape, light & material decomposition from images using monte carlo rendering and denoising. *arXiv preprint arXiv:2206.03380* (2022).
Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. 2018. Deep blending for free-viewpoint image-based rendering. *ACM Trans. Graph.* 37, 6 (2018), 1–15.
Peter Hedman, Tobias Ritschel, George Drettakis, and Gabriel Brostow. 2016. Scalable inside-out image-based rendering. *ACM Trans. Graph.* 35, 6 (2016), 1–11.
Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. 2021. Baking neural radiance fields for real-time view synthesis. In *Int. Conf. Comput. Vis.* 5875–5884.
Tao Hu, Shu Liu, Yilun Chen, Tiancheng Shen, and Jiaya Jia. 2022. Efficientnerf efficient neural radiance fields. In *IEEE Conf. Comput. Vis. Pattern Recog.* 12902–12911.
Ajay Jain, Matthew Tancik, and Pieter Abbeel. 2021. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *Int. Conf. Comput. Vis.* 5885–5894.
Yoonwoo Jeong, Seokjun Ahn, Christopher Choy, Anima Anandkumar, Minsu Cho, and Jaesik Park. 2021. Self-calibrating neural radiance fields. In *Int. Conf. Comput. Vis.* 5846–5854.
Roland Jung and Stephan Weiss. 2021. Scalable recursive distributed collaborative state estimation for aided inertial navigation. In *ICRA*. 1896–1902.
Alexander Kaplan and Rainer Tichatschke. 1998. Proximal point methods and nonconvex optimization. *Journal of global Optimization* 13 (1998), 389–406.
Animesh Karnewar, Tobias Ritschel, Oliver Wang, and Niloy Mitra. 2022. Relu fields: The little non-linearity that could. In *SIGGRAPH Conference*. 1–9.
Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. 2020. Modular primitives for high-performance differentiable rendering. *ACM Trans. Graph.* 39, 6 (2020), 1–14.
Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. 2021. Barf: Bundle-adjusting neural radiance fields. In *Int. Conf. Comput. Vis.* 5741–5751.
Yunzhi Lin, Thomas Müller, Jonathan Tremblay, Bowen Wen, Stephen Tyree, Alex Evans, Patricio A. Vela, and Stan Birchfield. 2023. Parallel Inversion of Neural Radiance Fields for Robust Pose Estimation. In *ICRA*.
Philipp Lindenberger, Paul-Edouard Sarlin, Viktor Larsson, and Marc Pollefeys. 2021. Pixel-perfect structure-from-motion with featuremetric refinement. In *Int. Conf. Comput. Vis.* 5987–5997.
Lingjie Liu, Weipeng Xu, Michael Zollhoefer, Hyeongwoo Kim, Florian Bernard, Marc Habermann, Wenping Wang, and Christian Theobalt. 2019b. Neural rendering and reenactment of human actor videos. *ACM Trans. Graph.* 38, 5 (2019), 1–14.
Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. 2019a. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Int. Conf. Comput. Vis.* 7708–7717.
Andreas Meuleman, Yu-Lun Liu, Chen Gao, Jia-Bin Huang, Changil Kim, Min H Kim, and Johannes Kopf. 2023. Progressively optimized local radiance fields for robust view synthesis. In *IEEE Conf. Comput. Vis. Pattern Recog.* 16539–16548.
Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2020. Nerf: Representing scenes as neural radiance fields for view synthesis. In *Eur. Conf. Comput. Vis.*
Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.* 41, 4 (2022), 1–15.
Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. 2022. Extracting triangular 3d models, materials, and lighting from images. In *IEEE Conf. Comput. Vis. Pattern Recog.* 8280–8290.
Shree K Nayar, Peter N Belhumeur, and Terry E Boult. 2004. Lighting sensitive display. *ACM Trans. Graph.* 23, 4 (2004), 963–979.
Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. 2022. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *IEEE Conf. Comput. Vis. Pattern Recog.* 5480–5490.
Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Adv. Neural Inform. Process. Syst.*, Vol. 32.
Zhimin Peng, Yangyang Xu, Ming Yan, and Wotao Yin. 2016. Arock: an algorithmic framework for asynchronous parallel coordinate updates. *SIAM Journal on Scientific Computing* 38, 5 (2016), A2851–A2879.
Sameera Ramasinghe and Simon Lucey. 2022. Beyond periodicity: towards a unifying framework for activations in coordinate-MLPs. In *Eur. Conf. Comput. Vis.* 142–158.
Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. 2021. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Int. Conf. Comput. Vis.* 14335–14345.
Christian Reiser, Richard Szeliski, Dor Verbin, Pratul P Srinivasan, Ben Mildenhall, Andreas Geiger, Jonathan T Barron, and Peter Hedman. 2023. Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *arXiv preprint arXiv:2302.12249* (2023).
Konstantinos Rematas, Andrew Liu, Pratul P. Srinivasan, Jonathan T. Barron, Andrea Tagliasacchi, Tom Funkhouser, and Vittorio Ferrari. 2022. Urban radiance fields. *IEEE Conf. Comput. Vis. Pattern Recog.* (2022).
Gernot Riegler and Vladlen Koltun. 2021. Stable view synthesis. In *IEEE Conf. Comput. Vis. Pattern Recog.* 12216–12225.
Simon Rodriguez, Siddhant Prakash, Peter Hedman, and George Drettakis. 2020. Image-based rendering of cars using semantic labels and approximate reflection flow. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 3 (2020).
Barbara Roessle, Jonathan T Barron, Ben Mildenhall, Pratul P Srinivasan, and Matthias Nießner. 2022. Dense depth priors for neural radiance fields from sparse input views. In *IEEE Conf. Comput. Vis. Pattern Recog.* 12892–12901.
Paul-Edouard Sarlin, Ajaykumar Unagar, Mans Larsson, Hugo Germain, Carl Toft, Viktor Larsson, Marc Pollefeys, Vincent Lepetit, Lars Hammarstrand, Fredrik Kahl, et al. 2021. Back to the feature: Learning robust camera localization from pixels to pose. In *IEEE Conf. Comput. Vis. Pattern Recog.* 3247–3257.
Johannes L Schonberger and Jan-Michael Frahm. 2016. Structure-from-motion revisited. In *IEEE Conf. Comput. Vis. Pattern Recog.* 4104–4113.
Sudipta N Sinha, Johannes Kopf, Michael Goesele, Daniel Scharstein, and Richard Szeliski. 2012. Image-based rendering for scenes with reflections. *ACM Trans. Graph.* 31, 4 (2012), 1–10.
Peter-Pike Sloan, Jesse Hall, John Hart, and John Snyder. 2003. Clustered principal components for precomputed radiance transfer. *ACM Trans. Graph.* 22, 3 (2003), 382–391.
Pratul P Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T Barron. 2021. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *IEEE Conf. Comput. Vis. Pattern Recog.* 7495–7504.
Cheng Sun, Min Sun, and Hwann-Tzong Chen. 2022b. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *IEEE Conf. Comput. Vis. Pattern Recog.* 5459–5469.
Jiaming Sun, Xi Chen, Qianqian Wang, Zhengqi Li, Hadar Averbuch-Elor, Xiaowei Zhou, and Noah Snavely. 2022a. Neural 3D reconstruction in the wild. In *SIGGRAPH Conference*. 1–9.
Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. 2022. Block-nerf: Scalable large scene neural view synthesis. In *IEEE Conf. Comput. Vis. Pattern Recog.* 8248–8258.
Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. 2023. Nerfstudio: A Modular Framework for Neural Radiance Field Development. In *SIGGRAPH Conference Proceedings*.
Chengzhou Tang and Ping Tan. 2018. Ba-net: Dense bundle adjustment network. In *Int. Conf. Learn. Represent.*
Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. 2000. Bundle adjustment—a modern synthesis. In *Vision Algorithms: Theory and Practice: International Workshop on Vision Algorithms*. 298–372.
Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. 2022. Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs. In *IEEE Conf. Comput. Vis. Pattern Recog.* 12922–12931.

Haithem Turki, Jason Y Zhang, Francesco Ferroni, and Deva Ramanan. 2023. SUDS: Scalable Urban Dynamic Scenes. *arXiv preprint arXiv:2303.14536* (2023).
Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. 2022. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In *IEEE Conf. Comput. Vis. Pattern Recog.* 5481–5490.
Jiepeng Wang, Peng Wang, Xiaoxiao Long, Christian Theobalt, Taku Komura, Lingjie Liu, and Wenping Wang. 2022. Neuris: Neural reconstruction of indoor scenes using normal priors. In *Eur. Conf. Comput. Vis.* 139–155.
Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE TIP* 13, 4 (2004), 600–612.
Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. 2021. NeRF−: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064* (2021).
Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. 2021. Nex: Real-time view synthesis with neural basis expansion. In *IEEE Conf. Comput. Vis. Pattern Recog.* 8534–8543.
Daniel N Wood, Daniel I Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H Salesin, and Werner Stuetzle. 2000. Surface light fields for 3D photography. In *Proc. of SIGGRAPH.* 287–296.
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 (2022), 1–16.
Yuanbo Xiangli, Linning Xu, Xingang Pan, Nanxuan Zhao, Anyi Rao, Christian Theobalt, Bo Dai, and Dahua Lin. 2022. Bungeenerf: Progressive neural radiance field for extreme multi-scale scene rendering. In *Eur. Conf. Comput. Vis.* 106–122.
Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. 2022. Neural fields in visual computing and beyond. *Comput. Graph. Forum* 41, 2 (2022), 641–676.
Linning Xu, Yuanbo Xiangli, Sida Peng, Xingang Pan, Nanxuan Zhao, Christian Theobalt, Bo Dai, and Dahua Lin. 2023. Grid-guided Neural Radiance Fields for Large Urban Scenes. *arXiv preprint arXiv:2303.14001* (2023).
Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. 2022. Point-nerf: Point-based neural radiance fields. In *IEEE Conf. Comput. Vis. Pattern Recog.* 5438–5448.
Yao Yao, Jingyang Zhang, Jingbo Liu, Yihang Qu, Tian Fang, David McKinnon, Yanghai Tsin, and Long Quan. 2022. Neilf: Neural incident light field for physically-based material estimation. In *Eur. Conf. Comput. Vis.* 700–716.
Lin Yen-Chen, Pete Florence, Jonathan T Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. 2021. inerf: Inverting neural radiance fields for pose estimation. In *IROS.* 1323–1330.
Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. 2021a. Plenoctrees for real-time rendering of neural radiance fields. In *Int. Conf. Comput. Vis.* 5752–5761.
Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. 2021b. pixelnerf: Neural radiance fields from one or few images. In *IEEE Conf. Comput. Vis. Pattern Recog.* 4578–4587.
Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. 2022. Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. *arXiv preprint arXiv:2206.00665* (2022).
Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. 2021a. Physg: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In *IEEE Conf. Comput. Vis. Pattern Recog.* 5453–5462.
Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. 2020. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492* (2020).
Runze Zhang, Siyu Zhu, Tian Fang, and Long Quan. 2017. Distributed very large scale bundle adjustment by global camera consensus. In *Int. Conf. Comput. Vis.* 29–38.
Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. 2021b. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Trans. Graph.* 40, 6 (2021), 1–18.
Yuqi Zhang, Guanying Chen, and Shuguang Cui. 2023. Efficient large-scale scene representation with a hybrid of high-resolution grid and plane features. *arXiv preprint arXiv:2303.03003* (2023).
Pengxiang Zhu, Patrick Geneva, Wei Ren, and Guoquan Huang. 2021. Distributed visual-inertial cooperative localization. In *IROS.* 8714–8721.
Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R Oswald, and Marc Pollefeys. 2022. Nice-slam: Neural implicit scalable encoding for slam. In *IEEE Conf. Comput. Vis. Pattern Recog.* 12786–12796.

A IMPLEMENTATION DETAILS

Tile hash grid resolution. Given the contracted bounded region $4 \times 4 \times 4$, the resolution of the hash-grid node at level l can be determined as as $R_l := \lfloor R_{\min} b^l \rfloor$, and $b = \exp\left(\frac{\ln R_{\max} - \ln R_{\min}}{L-1}\right)$ as in [Müller et al. 2022], where $R_{\max}$ is the finest resolution, which set to 4096 for indoor scenes and 8192 for outdoor scenes, and $R_{\min}$ is the coarsest resolution, set to 32.

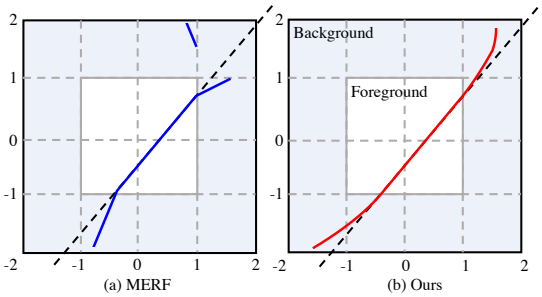


Fig. 20. **2D visualization of contraction.** (a) The contract function of MERF [Reiser et al. 2023]. (b) Our contract function. The black dashed line denotes the straight line in Euclidean Space. In MERF's contraction, the straight line is mapped to piece-wise lines (in blue) while in our contraction, the straight line is mapped to a curve (in red).

Tile occupancy voxel grid. For the occupancy voxel grid, each tile's voxel grid is initialized by proxy mesh with resolution 16 and subdivided every 2000 iterations. The subdivision stops when the resolution of voxel grid reaches 512. We pruning the voxel when $1 - \exp(-\sigma) < \lambda$ by setting its mask value to 0 after every 1000 iterations, where the σ is computed at each voxel center. The pruning threshold λ is initialized with 0.1 and increased by 0.1 every 2000 iterations until it reaches 0.4.

Post-optimization multi-tile rendering GPU implementation. Given the sorted intersection tiles with each ray, we first sample points in the first tile for each ray in parallel on GPU, and loop through next tile for each ray in parallel until no tile is left for sampling. On GPU, we also compute the accumulated transmittance value to check the early terminating condition for each ray. If early termination is triggered, the remaining tiles for a ray are skipped in the parallel sampling. Afterward, numerical quadrature in volume rendering is executed for each ray in parallel on GPU.

Diffuse and specular decoder architectures. Both our diffuse decoder and specular decoder are MLP networks with 2 hidden fully-connected layers. Each hidden layer has 64 neurons with Gaussian activation functions (mean: 0, std: 0.1) [Ramasinghe and Lucey 2022]. The 64-channel feature output by the last hidden layer of the diffuse decoder is split into two parts. Its first 32 channels are fed to a 32×3 layer to predict diffuse color, a 32×3 layer to predict specular tint, and a 32×1 layer to compute the volume density. The rest of 32 channel features, i.e. $\mathbf{h}_x$, are fed into the specular decoder. The activation function is set to sigmoid for diffuse color and specular tint, and SoftPlus for density as in Lin *et al.* [Lin et al. 2021]. Similarly, the final layer of the specular decoder is a 64×3 fully connected layer with sigmoid activation to compute the specular component of the 3D point color.

Training details. Our model is trained on a GPU server with eight Tesla V100 GPUs using PyTorch 1.9.0 [Paszke et al. 2019] with CUDA

11.1. We use the Adam optimizer [Kingma and Ba 2014] in the ADMM parallel updating step to minimize Eq. (19) at each iteration. The learning rate is set to 0.001 for the features and decoders, and it is decayed exponentially to 0.0001. For camera pose variables, the learning rate is initialized to 0.0001 and decayed to 0.00001 exponentially. At each iteration, the data batch for each tile contains 2^{14} rays uniformly sampled across all its associated cameras. For the training loss, we set $\lambda_c = 1.0$, $\lambda_d = 0.01$, $\lambda_s = 0.001$, and $\lambda_w = 1.0$. The warping loss is added to the training loss after the first 10,000 iterations. The σ_v and σ_d in adaptive weight are set to 0.1 and 0.05, respectively. In our experiments, we stop the ADMM training procedure after 40,000 iterations, and the L2 norms of the primal and dual residuals typically converge to around 1e-5.

B CONTRACTION FUNCTION

As illustracted in Fig. 20, we map unbound background into a bounded cubic region using the following contraction function:

$$\text{contract}(\mathbf{x})_j = \begin{cases} x_j, & \text{if } ||\mathbf{x}||_\infty \leq 1, \\ (2 - \frac{1}{||\mathbf{x}||_\infty})\frac{x_j}{||\mathbf{x}||_\infty}, & \text{if } ||\mathbf{x}||_\infty > 1, \end{cases} \tag{27}$$

where $||\cdot||_\infty$ is the L_∞ norm and the tile foreground region is normalized to $[-1, 1]$. Any 3D point sampled within a tile is first normalized and then contracted before querying the hash grid.