BakedSDF: Meshing Neural SDFs for Real-Time View Synthesis

Lior Yariv[*] [†]
Weizmann Institute of Science
Israel
lior.yariv@weizmann.ac.il

Peter Hedman[*]
Google Research
United Kingdom
hedman@google.com

Christian Reiser[†]
Tübingen AI Center
Germany
christian.reiser@tue.mpg.de

Dor Verbin
Google Research
United States of America
dorverbin@google.com

Pratul P. Srinivasan
Google Research
United States of America
pratuls@google.com

Richard Szeliski
Google Research
United States of America
szeliski@google.com

Jonathan T. Barron
Google Research
United States of America
barron@google.com

Ben Mildenhall
Google Research
United States of America
bmild@google.com

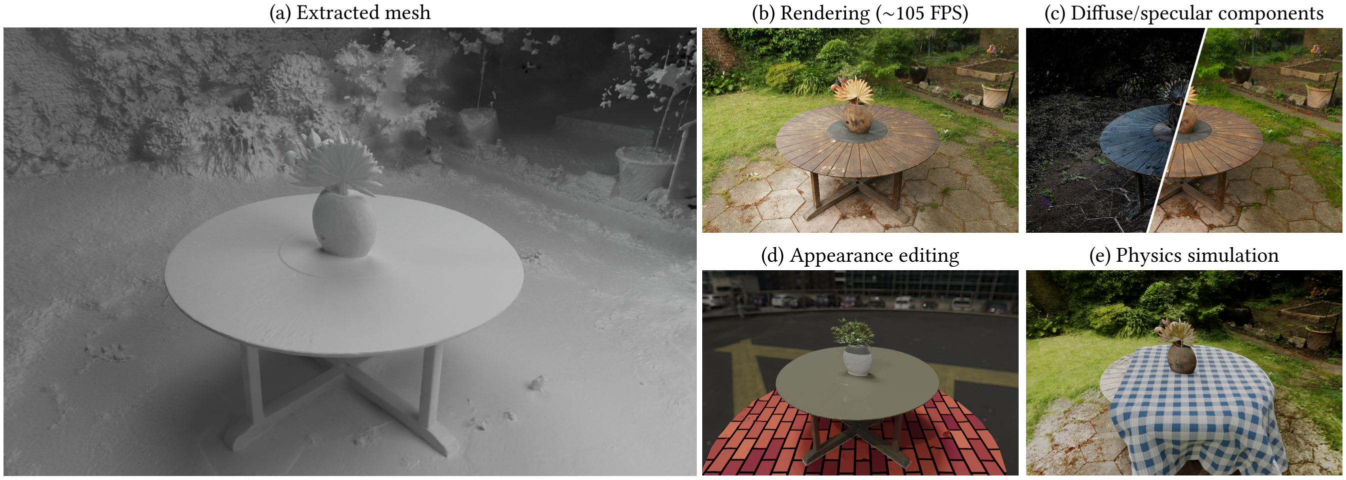

Figure 1: Our method, *BakedSDF*, optimizes a neural surface-volume representation of a complex real-world scenes and (a) "bakes" that representation into a high-resolution mesh. These meshes (b) can be rendered in real time on commodity hardware, and support other applications such as (c) separating material components, (d) appearance editing with accurate cast shadows, and (e) physics simulation for inserted objects. Interactive demo at https://bakedsdf.github.io/.

ABSTRACT

We present a method for reconstructing high-quality meshes of large unbounded real-world scenes suitable for photorealistic novel view synthesis. We first optimize a hybrid neural volume-surface scene representation designed to have well-behaved level sets that correspond to surfaces in the scene. We then bake this representation into a high-quality triangle mesh, which we equip with a simple and fast view-dependent appearance model based on spherical Gaussians. Finally, we optimize this baked representation to best reproduce the captured viewpoints, resulting in a model that can leverage accelerated polygon rasterization pipelines for real-time view synthesis on commodity hardware. Our approach outperforms previous scene representations for real-time rendering in terms of accuracy, speed, and power consumption, and produces high quality meshes that enable applications such as appearance editing and physical simulation.

[*]Both authors contributed equally to this research.
[†]Work done while interning at Google.

SIGGRAPH '23 Conference Proceedings, August 06–10, 2023, Los Angeles, CA, USA

ACM ISBN 979-8-4007-0159-7/23/08.
https://doi.org/10.1145/3588432.3591536

CCS CONCEPTS

• **Computing methodologies** → **Reconstruction**; *Neural networks*; *Volumetric models*.

KEYWORDS

Neural Radiance Fields, Signed Distance Function, Surface Reconstruction, Image Synthesis, Real-Time Rendering, Deep Learning.

ACM Reference Format:

Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P. Srinivasan, Richard Szeliski, Jonathan T. Barron, and Ben Mildenhall. 2023. BakedSDF: Meshing Neural SDFs for Real-Time View Synthesis. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings (SIGGRAPH '23 Conference Proceedings), August 06–10, 2023, Los Angeles, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3588432.3591536

1 INTRODUCTION

Current top-performing approaches for novel view synthesis — the task of using captured images to recover a 3D representation that can be rendered from unobserved viewpoints — are largely based on Neural Radiance Fields (NeRF) [Mildenhall et al. 2020]. By representing a scene as a continuous volumetric function parameterized by a multilayer perceptron (MLP), NeRF is able to produce photorealistic renderings that exhibit detailed geometry and view-dependent effects. Because the MLP underlying a NeRF is expensive to evaluate and must be queried hundreds of times *per pixel*, rendering a high resolution image from a NeRF is typically slow.

Recent work has improved NeRF rendering performance by trading compute-heavy MLPs for discretized volumetric representations such as voxel grids. However, these approaches require substantial GPU memory and custom volumetric raymarching code and are not amenable to real-time rendering on commodity hardware, since modern graphics hardware and software is oriented towards rendering polygonal surfaces rather than volumetric fields.

While current NeRF-like approaches are able to recover high-quality real-time-renderable meshes of individual objects with simple geometry [Boss et al. 2022], reconstructing detailed and well-behaved meshes from captures of real-world unbounded scenes (such as the "360 degree captures" of Barron et al. [2022]) has proven to be more difficult. Recently, MobileNeRF [Chen et al. 2022a] addressed this problem by training a NeRF whose volumetric content is restricted to lie on the faces of a polygon mesh, then baking that NeRF into a texture map. Though this approach yields reasonable image quality, MobileNeRF initializes the scene geometry as a collection of axis-aligned tiles that turns into a textured polygon "soup" after optimization. The resulting geometry is less suitable for common graphics applications such as texture editing, relighting, and physical simulation.

In this work, we demonstrate how to extract high-quality meshes from a NeRF-like neural volumetric representation. Our system, which we call BakedSDF, extends the hybrid volume-surface neural representation of VolSDF [Yariv et al. 2021] to represent unbounded real-world scenes. This representation is designed to have a well-behaved zero level set corresponding to surfaces in the scene, which lets us extract high-resolution triangle meshes using marching cubes.

Our key idea is to define the SDF in *contracted coordinate space* [Barron et al. 2022], as it has these advantages: It more strongly regularizes distant content, and it allows us to also extract the mesh in contracted space which distributes the triangle budget better (more in the center, fewer in the periphery).

We then equip this mesh with a fast and efficient view-dependent appearance model based on spherical Gaussians, which is fine-tuned to reproduce the input images of the scene. The output of our system can be rendered at real-time frame rates on commodity devices, and we show that our real-time rendering system outperforms prior work in terms of realism, speed, and power consumption. Additionally we show that (unlike comparable prior work) the mesh produced by our model is accurate and detailed, enabling standard graphics applications such as appearance editing and physics simulation.

To summarize, our key contributions are:

(1) High-quality neural surface reconstruction of unbounded real-world scenes,
(2) a framework for real-time rendering of these scenes in a browser, and
(3) we demonstrate that spherical Gaussians are a practical representation of view-dependence appearance for view-synthesis.

2 RELATED WORK

View synthesis, i.e., the task of rendering novel views of a scene given a set of captured images, is a longstanding problem in the fields of computer vision and graphics. In scenarios where the observed viewpoints are sampled densely, synthesizing new views can be done with light field rendering — straightforward interpolation into the set of observed rays [Gortler et al. 1996; Levoy and Hanrahan 1996]. However, in practical settings where observed viewpoints are captured more sparsely, reconstructing a 3D representation of the scene is crucial for rendering convincing novel views. Most classical approaches for view synthesis use triangle meshes (typically reconstructed using a pipeline consisting of multi-view stereo [Furukawa and Hernández 2015; Schönberger et al. 2016], Poisson surface reconstruction [Kazhdan et al. 2006; Kazhdan and Hoppe 2013], and marching cubes [Lorensen and Cline 1987]) as the underlying 3D scene representation, and render novel views by reprojecting observed images into each novel viewpoint and blending them together using either heuristically-defined [Buehler et al. 2001; Debevec et al. 1996; Wood et al. 2000] or learned [Hedman et al. 2018; Riegler and Koltun 2020, 2021] blending weights. Although mesh-based representations are well-suited for real-time rendering with accelerated graphics pipelines, the meshes produced by these approaches tend to have inaccurate geometry in regions with fine details or complex materials, which leads to errors in rendered novel views. Alternatively, point-based representations [Kopanas et al. 2021; Rückert et al. 2022] are better suited for modeling thin geometry, but cannot be rendered efficiently without visible cracks or unstable results when the camera moves.

Most recent approaches to view synthesis sidestep the difficulty of high-quality mesh reconstruction by using volumetric representations of geometry and appearance, such as voxel grids [Lombardi et al. 2019; Penner and Zhang 2017; Szeliski and Golland 1999; Vogiatzis et al. 2007] or multiplane images [Srinivasan et al. 2019; Wizadwongsa et al. 2021; Zhou et al. 2018]. These representations are well-suited to gradient-based optimization of a rendering loss, so they can be effectively optimized to reconstruct detailed geometry seen in the input images. The most successful of these volumetric approaches is Neural Radiance Fields (NeRF) [Mildenhall et al. 2020], which forms the basis for many state-of-the-art view synthesis methods (see Tewari et al. [2022] for a review). NeRF represents a scene as a continuous volumetric field of matter that emits and absorbs light, and renders an image using volumetric ray-tracing. NeRF uses an MLP to map from a spatial coordinate to a volumetric density and emitted radiance, and that MLP must be evaluated at a set of sampled coordinates along a ray to yield a final color.

Subsequent works have proposed modifying NeRF's representation of scene geometry and appearance for improved quality and editability. Ref-NeRF [Verbin et al. 2022] reparameterizes NeRF's view-dependent appearance to enable appearance editing and improve the reconstruction and rendering of specular materials. Other works [Boss et al. 2021; Kuang et al. 2022; Srinivasan et al. 2021; Zhang et al. 2021a,b] attempt to decompose a scene's view-dependent appearance into material and lighting properties. In addition to modifying NeRF's representation of appearance, papers including UNISURF [Oechsle et al. 2021], VolSDF [Yariv et al. 2021], NeuS [Wang et al. 2021], MetaNLR++ [Bergman et al. 2021], and NeuMesh [Bao and Yang et al. 2022] augment NeRF's fully-volumetric representation with hybrid volume-surface models, but do not target real-time rendering and show results only for objects and bounded scenes.

The MLP NeRF uses to represent a scene is usually large and expensive to evaluate, and this means that a NeRF is slow to train (hours or days per scene) and slow to render (seconds or minutes per megapixel). While rendering can be accelerated with a sampling network that reduces the MLP queries per ray [Neff et al. 2021], recent methods have improved both training and render time by replacing the large MLP with a voxel grid [Karnewar et al. 2022; Sun et al. 2022], a grid of small MLPs [Reiser et al. 2021], low-rank [Chen et al. 2022b] or sparse [Yu et al. 2022] grid representations, or a multiscale hash encoding with a small MLP [Müller et al. 2022].

While these representations reduce the computation required for both training and rendering (at the cost of increased storage), rendering can be further accelerated by precomputing and storing, i.e., "baking", a trained NeRF into a more efficient representation. SNeRG [Hedman et al. 2021], FastNeRF [Garbin et al. 2021], Plenoctrees [Yu et al. 2021], and Scalable Neural Indoor Scene Rendering [Wu et al. 2022] all bake trained NeRFs into sparse volumetric structures and use simplified models of view-dependent appearance to avoid evaluating an MLP at each sample along each ray. These methods have enabled real-time rendering of NeRFs on high-end hardware, but their use of volumetric raymarching precludes real-time performance on commodity hardware. Concurrent to our work, Reiser et al. [2023] developed Memory-Efficient Radiance Fields (MERF), a compressed representation volumetric for unbounded scenes that facilitates fast rendering on commodity hardware. When compared with our meshes, this volumetric representation achieves higher quality scores, but requires more memory, needs a complex renderer, and is not straightforward to use for downstream graphics applications such as physics simulation. Please refer to the MERF paper for a direct comparison with our method.

3 PRELIMINARIES

In this section, we describe the neural volumetric representation that NeRF [Mildenhall et al. 2020] uses for view synthesis as well as improvements introduced by mip-NeRF 360 [Barron et al. 2022] for representing unbounded "360 degree" scenes.

A NeRF is a 3D scene representation consisting of a learned function that maps a position $\mathbf{x}$ and outgoing ray direction $\mathbf{d}$ to a volumetric density τ and color $\mathbf{c}$. To render the color of a single pixel in a target camera view, we first compute the ray corresponding to that pixel $\mathbf{r} = \mathbf{o} + t\mathbf{d}$, and then evaluate the NeRF at a series of points $\{t_i\}$ along the ray. The resulting outputs $\tau_i, \mathbf{c}_i$ at each point are composited together into a single output color value $\mathbf{C}$:

$$\mathbf{C} = \sum_i \exp\left(-\sum_{j<i} \tau_j \delta_j\right)\left(1 - \exp\left(-\tau_i \delta_i\right)\right)\mathbf{c}_i\,, \quad \delta_i = t_i - t_{i-1}\,. \quad (1)$$

This definition of $\mathbf{C}$ is a quadrature-based approximation of the volume rendering equation [Max 1995].

NeRF parametrizes this learned function using an MLP whose weights are optimized to implicitly encode the geometry and color of the scene: A set of training input images and their camera poses are converted into a set of (ray, color) pairs, and gradient descent is used to optimize the MLP weights such that the rendering of each ray resembles its corresponding input color. Formally, NeRF minimizes a loss between the ground truth color $\mathbf{C}_{\text{gt}}$ and the color $\mathbf{C}$ produced in Equation 1, averaged over all training rays:

$$\mathcal{L}_{\text{data}} = \mathbb{E}\left[\left\|\mathbf{C} - \mathbf{C}_{\text{gt}}\right\|^2\right]\,. \quad (2)$$

If the input images provide sufficient coverage of the scene (in terms of multiview 3D constraints), this simple process yields a set of MLP weights that accurately describe the scene's 3D volumetric density and appearance.

Mip-NeRF 360 [Barron et al. 2022] extends the basic NeRF formulation to reconstruct and render real-world "360 degree" scenes where cameras can observe unbounded scene content in all directions. Two improvements introduced in mip-NeRF 360 are the use of a contraction function and a proposal MLP. The contraction function maps unbounded scene points in $\mathbb{R}^3$ to a bounded domain:

$$\operatorname{contract}(\mathbf{x}) = \begin{cases} \mathbf{x} & \|\mathbf{x}\| \leq 1 \\ \left(2 - \frac{1}{\|\mathbf{x}\|}\right)\frac{\mathbf{x}}{\|\mathbf{x}\|} & \|\mathbf{x}\| > 1 \end{cases}, \quad (3)$$

which produces contracted coordinates that are well-suited to be positionally encoded as inputs to the MLP. Additionally, mip-NeRF 360 showed that large unbounded scenes with detailed geometry require prohibitively large MLPs and many more samples along each ray than is tractable in the original NeRF framework. Mip-NeRF 360 therefore introduced a proposal MLP: a much smaller MLP that is trained to bound the density of the actual NeRF MLP. This proposal MLP is used in a hierarchical sampling procedure that efficiently generates a set of input samples for the NeRF MLP that are tightly focused around non-empty content in the scene.

4 METHOD

Our method is composed of three stages, which are visualized in Figure 2. First we optimize a surface-based representation of the

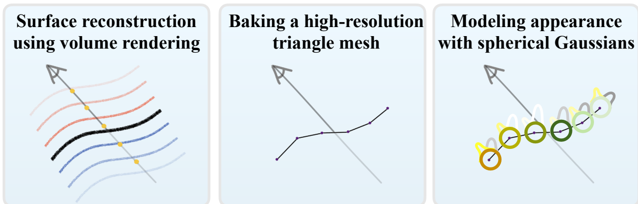

Figure 2: An illustration of the three stages of our method. We first reconstruct the scene using a surface-based volumetric representation (Section 4.1), then bake it into a high-quality mesh (Section 4.2), and finally optimize a view-dependent appearance model based on spherical Gaussians (Section 4.3).

geometry and appearance of a scene using NeRF-like volume rendering. Then, we "bake" that geometry into a mesh, which we show is accurate enough to support convincing appearance editing and physics simulation. Finally, we train a new appearance model that uses spherical Gaussians (SGs) embedded within each vertex of the mesh, which replaces the expensive NeRF-like appearance model from the first step. The resulting 3D representation that results from this approach can be rendered in real-time on commodity devices, as rendering simply requires rasterizing a mesh and querying a small number of spherical Gaussians.

4.1 Modeling density with an SDF

Our representation combines the benefits of mip-NeRF 360 for representing unbounded scenes with the well-behaved surface properties of VolSDF's hybrid volume-surface representation [Yariv et al. 2021]. VolSDF models volumetric density of the scene as a parametric function of an MLP-parameterized signed distance function (SDF) f that returns the signed distance $f(\mathbf{x})$ from each point $\mathbf{x} \in \mathbb{R}^3$ to the surface. Because our focus is reconstructing unbounded real-world scenes, we parameterize f in *contracted* space (Equation 3) rather than world-space. The underlying surface of the scene is the zero-level set of f, i.e., the set of points at distance zero from the surface:

$$\mathcal{S} = \{\mathbf{x} : f(\mathbf{x}) = 0\} \ . \tag{4}$$

Following VolSDF, we define the volume density τ as:

$$\tau(\mathbf{x}) = \alpha \Psi_\beta\left(f(\mathbf{x})\right) , \tag{5}$$

where Ψ_β is the cumulative distribution function of a zero-mean Laplace distribution with scale parameter $\beta > 0$. Note that as β approaches 0, the volumetric density approaches a function that returns α inside any object and 0 in free space. To encourage f to approximate a valid signed distance function (i.e. one where $f(\mathbf{x})$ returns the signed Euclidean distance to the level set of f for all $\mathbf{x}$), we penalize the deviation of f from satisfying the Eikonal equation [Gropp et al. 2020]:

$$\mathcal{L}_{\text{SDF}} = \mathbb{E}_{\mathbf{x}}\left[(\|\nabla f(\mathbf{x})\| - 1)^2\right] . \tag{6}$$

Note that as f is defined in contracted space, this constraint also operates on contracted space.

Recently, Ref-NeRF [Verbin et al. 2022] improved view-dependent appearance by parameterizing it as a function of the view direction reflected about the surface normal. Our use of an SDF-parameterized density allows this to be easily adopted as SDFs have well-defined surface normals: $\mathbf{n}(\mathbf{x}) = \nabla f(\mathbf{x})/\|\nabla f(\mathbf{x})\|$. Therefore, when training this stage of our model we adopt Ref-NeRF's appearance model and compute color using separate diffuse and specular components, where the specular component is parameterized by the concatenation of the view direction reflected about the normal direction, the dot product between the normal and view direction, and a 256 element bottleneck vector output by the MLP that parametrizes f.

We use a variant of mip-NeRF 360 as our model (see Appendix A in supplementary material for specific training details). Similarly to VolSDF [Yariv et al. 2021], we parameterize the density scale factor as $\alpha = \beta^{-1}$ in Equation 5. However, we find that scheduling β rather than leaving it as a free optimizable parameter results in more stable training. We therefore anneal β according to $\beta_t = \beta_0\left(1 + \frac{\beta_0 - \beta_1}{\beta_1} t^{0.8}\right)^{-1}$, where t goes from 0 to 1 during training, $\beta_0 = 0.1$, and β_1 for the three hierarchical sampling stages is 0.015, 0.003, and 0.001 respectively. Because the Eikonal regularization needed for an SDF parameterization of density already removes floaters and results in well-behaved normals, we do not find it necessary to use the orientation loss or predicted normals from Ref-NeRF, or the distortion loss from mip-NeRF 360.

4.2 Baking a high-resolution mesh

After optimizing our neural volumetric representation, we create a triangle mesh from the recovered MLP-parameterized SDF by querying it on a regular 3D grid and then running Marching Cubes [Lorensen and Cline 1987]. Note that VolSDF models boundaries using a density fall-off that extends beyond the SDF zero crossing (parameterized by β). We account for this spread when extracting the mesh and choose 0.001 as the iso-value for surface crossings, as otherwise we find the scene geometry to be slightly eroded.

Visibility and free-space culling. When running Marching Cubes, the MLP-parameterized SDF may contain spurious surface crossings in regions that are occluded from the observed viewpoints as well as regions that the proposal MLP marks as "free space". The SDF MLP's values in both of these types of regions are not supervised during training, so we must cull any surface crossings that would show up as spurious content in the reconstructed mesh. To address this, we inspect the 3D samples taken along the rays in our training data. We compute the volumetric rendering weight for each sample, i.e., how much it contributes to the training pixel color. We then splat any sample with a sufficiently large rendering weight (> 0.005) into the 3D grid and mark the corresponding cell as a candidate for surface extraction.

Mesh extraction. We sample our SDF grid at evenly spaced coordinates in the contracted space, which yields unevenly spaced non-axis-aligned coordinates in world space. This has the desirable property of creating smaller triangles (in world space) for foreground content close to the origin and larger triangles for distant content. Effectively, we leverage the contraction operator as a level-of-detail strategy: as our desired rendered views are close to the scene origin, and because the shape of the contraction is designed to undo the effects of perspective projection, all triangles will have approximately equal areas when projected onto the image plane.

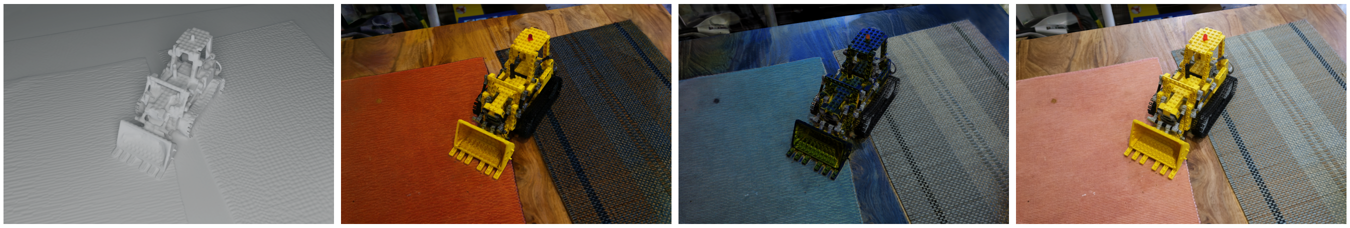

Figure 3: Our method produces an accurate mesh and decomposes appearance into diffuse and specular color.

Region growing. After extracting the triangle mesh, we use a region growing procedure to fill small holes that might exist in regions that were either unobserved by input viewpoints or missed by the proposal MLP during the baking procedure. We iteratively mark voxels in a neighborhood around the current mesh and extract any surface crossings that exist in these newly active voxels. This *region-growing* strategy effectively remedies situations where a surface exists in the SDF MLP but was not extracted by marching cubes due to insufficient training view coverage or errors in the proposal MLP. We then transform the mesh into world space so it is ready for rasterization by a conventional rendering engine that operates in Euclidean space.

Implementation. We use a 2048^3 grid for both visibility and free-space culling and marching cubes. Initially, we run marching cubes only on voxels that were not culled, i.e. visible and non-empty. We then complete the mesh with 32 region-growing iterations, where we re-run marching cubes in a 8^3 voxel neighborhood around the vertices in the current mesh. Finally, we post-process the mesh using vertex order optimization [Sander et al. 2007], which speeds up rendering performance on modern hardware by allowing vertex shader outputs to be cached and reused between neighboring triangles. In Appendix B we detail additional steps for mesh extraction which do not strictly improve reconstruction accuracy, but enable a more pleasing interactive viewing experience.

4.3 Modeling view-dependent appearance

The baking procedure described above extracts high-quality triangle mesh geometry from our MLP-based scene representation. To model the scene's appearance, including view-dependent effects such as specularities, we equip each mesh vertex with a diffuse color $\mathbf{c}_d$ and a set of spherical Gaussian lobes. As far-away regions are only observed from a limited set of view directions, we do not need to model view dependence with the same fidelity everywhere in the scene. In our experiments, we use three spherical Gaussian lobes in the central regions ($\|\mathbf{x}\| \leq 1$) and one lobe in the periphery. Figure 3 demonstrates our appearance decomposition.

This appearance representation satisfies our efficiency goal for both compute and memory and can thus be rendered in real-time. Each spherical Gaussian lobe has seven parameters: a 3D unit vector μ for the lobe mean, a 3D vector $\mathbf{c}$ for the lobe color, and a scalar λ for the width of the lobe. These lobes are parameterized by the view direction vector $\mathbf{d}$, so the rendered color $\mathbf{C}$ for a ray intersecting any given vertex can be computed as:

$$\mathbf{C} = \mathbf{c}_d + \sum_{i=1}^{N} \mathbf{c}_i \exp\left(\lambda_i \left(\mu_i \cdot \mathbf{d} - 1\right)\right) . \tag{7}$$

To optimize this representation, we first rasterize the mesh into all training views and store the vertex indices and barycentric coordinates associated with each pixel. After this preprocessing, we can easily render a pixel by applying barycentric interpolation to the learned per-vertex parameters and then running our view-dependent appearance model (simulating the operation of a fragment shader). We can therefore optimize the per-vertex parameters by minimizing a per-pixel color loss as in Equation 2. As detailed in Appendix B, we also optimize for a background clear color to provide a more pleasing experience with the interactive viewer. To prevent that optimization from being biased by pixels that are not well-modeled by mesh geometry (e.g. pixels at soft object boundaries and semi-transparent objects), instead of the L2 loss that was minimized by VolSDF we use a robust loss $\rho(\cdot, \alpha, c)$ with hyperparameters $\alpha = 0$, $c = 1/5$ during training, which allows optimization to be more robust to outliers [Barron 2019]. We also model quantization with a straight-through estimator [Bengio et al. 2013], ensuring that the optimized values for view-dependent appearance are well represented by 8 bits of precision.

We find that directly optimizing this per-vertex representation saturates GPU memory, which prevents us from scaling up to high-resolution meshes. We instead optimize a compressed neural hash-grid model based on Instant NGP [Müller et al. 2022] (see Appendix A in supplemental material). During optimization, we query this model at each 3D vertex location within a training batch to produce our diffuse colors and spherical Gaussian parameters.

After optimization is complete, we bake out the compressed scene representation contained in the hash grids by querying the NGP model at each vertex location for the appearance-related parameters. Finally, we export the resulting mesh and per-vertex appearance parameters using the gLTF format [ISO/IEC 12113:2022 2022] and compress it with gzip, a format natively supported by web protocols.

5 EXPERIMENTS

We evaluate our method's performance both in terms of the accuracy of its output renderings and in terms of its speed, energy, and memory requirements. For accuracy, we test two versions of our model: the intermediate volume rendering results described in Section 4.1, which we refer to as our "offline" model, and the baked real-time model described in Sections 4.2 and 4.3, which

Figure 4: Test-set renderings (with insets) for our model and the two state-of-the-art real-time baselines we evaluate against, using scenes from the mip-NeRF 360 dataset. Deep Blending [Hedman et al. 2018] produces posterized renderings when the proxy geometry used as input is incorrect (such as in the background of the `bicycle` scene) and renderings from MobileNeRF [Chen et al. 2022a] tend to exhibit aliasing artifacts or oversmoothing.

we call the "real-time" model. As baselines we use prior offline models [Barron et al. 2022; Mildenhall et al. 2020; Müller et al. 2022; Riegler and Koltun 2021; Zhang et al. 2020] designed for fidelity, as well as with prior real-time methods [Chen et al. 2022a; Hedman et al. 2018] designed for performance. We additionally compare our method's recovered meshes with those extracted by COLMAP [Schönberger et al. 2016], mip-NeRF 360 [Barron et al. 2022], and MobileNeRF [Chen et al. 2022a]. All FPS (frames-per-second) measurements are for rendering at 1920 × 1080 resolution.

5.1 Real-time rendering of unbounded scenes

We evaluate our method on the dataset of real-world scenes from mip-NeRF 360 [Barron et al. 2022], which contains complicated indoor and outdoor scenes captured from all viewing angles. In Table 1 we present a quantitative evaluation of both the offline and real-time versions of our model against our baselines. Though our offline model is outperformed by some prior works (as we might expect, given that our focus is performance) our real-time method outperforms the two recent state-of-the-art real-time baselines we evaluate again across all three error metrics used by this benchmark. In Figure 4 we show a qualitative comparison of renderings from our model and these two state-of-the-art real-time baselines, and we observe that our approach exhibits more detail and fewer artifacts than prior work.

In Table 2 we evaluate our method's rendering performance by comparing against Instant-NGP (the fastest "offline" model we evaluate against) and MobileNeRF (the real-time model that produces the highest quality renderings after our own). We measure performance of all methods at 1920 × 1080. Both MobileNeRF and our method are running in-browser on a 16" Macbook Pro with a Radeon 5500M GPU while Instant NGP is running on a workstation equipped with a power NVIDIA RTX 3090 GPU. Though our approach requires more on-disk storage than MobileNeRF (1.27×) and Instant NGP (4.07×), we see that our model is significantly more efficient than both baselines — our model yields FPS/Watt metrics that are 1.44× and 77× greater respectively, in addition to producing higher quality renderings.

Our appreciably improved performance relative to MobileNeRF may seem unusual at first glance, as both our approach and MobileNeRF both yield optimized meshes that can be easily and quickly rasterized. This discrepancy is likely due to MobileNeRF's reliance on alpha masking (which results in a significant amount of compute-intensive overdraw) and MobileNeRF's use of an MLP to model

Table 1: Quantitative results of our model on the "outdoor" and "indoor" scenes from mip-NeRF 360 [Barron et al. 2022], with evaluation split for "offline" and "real-time" algorithms. Red, orange, and yellow indicate the first, second, and third best performing algorithms for each metric. Metrics not provided by a baseline are denoted with "–".

		Outdoor Scenes			Indoor Scenes		
		PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
offline	NeRF [Mildenhall et al. 2020]	21.46	0.458	0.515	26.84	0.790	0.370
offline	NeRF++ [Zhang et al. 2020]	22.76	0.548	0.427	28.05	0.836	0.309
offline	Stable View Synthesis [Riegler and Koltun 2021]	23.01	0.662	0.253	28.22	0.907	0.160
offline	Mip-NeRF 360 [Barron et al. 2022]	24.47	0.691	0.283	31.72	0.917	0.180
offline	Instant-NGP [Müller et al. 2022]	22.90	0.566	0.371	29.15	0.880	0.216
offline	Ours (offline)	23.40	0.619	0.379	30.21	0.888	0.243
real-time	Deep Blending [Hedman et al. 2018]	21.54	0.524	0.364	26.40	0.844	0.261
real-time	Mobile-NeRF [Chen et al. 2022a]	21.95	0.470	0.470	–	–	–
real-time	Ours (real-time)	22.47	0.585	0.349	27.06	0.836	0.258

Table 2: The performance (Watts consumed, frames per second, and their ratio) and storage requirements for our real-time method and two baselines. FPS is measured when rendering at 1920×1080 resolution.

	W ↓	FPS ↑	FPS/W ↑	MB (disk) ↓
Instant-NGP [Müller et al. 2022]	350	3.78	0.011	**106.8**
Mobile-NeRF [Chen et al. 2022a]	**85**	50.06	0.589	341.9
Ours	**85**	**72.21**	**0.850**	434.5

view-dependent radiance (which requires more compute to evaluate than our spherical Gaussian approach).

Compared to Deep Blending [Hedman et al. 2018], we see from Table 1 that our method achieves higher quality. However, it is also worth noting that our representation is also much *simpler*: while our meshes can be rendered in a browser, Deep Blending relies on carefully tuned CUDA rendering and must store both color and geometry for all training images in the scene. As a result, total storage cost for Deep Blending in the outdoor scenes is 2.66× higher (1154.78 MB on average) than for our corresponding meshes.

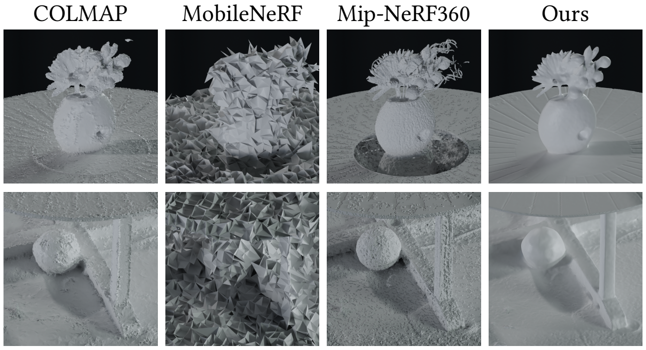

Figure 5: Comparing the meshes produced by our technique with baselines that yield meshes. Our meshes are higher in quality compared to those of COLMAP, MobileNeRF, and Mip-NeRF 360. COLMAP's mesh contains noise, floaters, and irregular object boundaries, MobileNeRF's mesh is a "polygon soup" that may not accurately represent scene geometry, and iso-surfaces from Mip-NeRF 360's density field tend to be noisy and represent reflections with inaccurate geometry.

5.2 Mesh extraction

In Figure 5 we present a qualitative comparison of our mesh with those obtained using COLMAP [Schönberger et al. 2016], MobileNeRF [Chen et al. 2022a] and an iso-surface of Mip-NeRF 360 [Barron et al. 2022]. We evaluate against COLMAP not only because it represents a mature structure-from-motion software package, but also because the geometry produced by COLMAP is used as input by Stable View Synthesis and Deep Blending. COLMAP uses volumetric graph cuts on a tetrahedralization of the scene [Jancosek and Pajdla 2011; Labatut et al. 2007] to obtain a binary segmentation of the scene and then forms a triangle mesh as the surface between these regions. Because this binary segmentation does not allow for any averaging of the surface, small noise in the initial reconstruction tends to result in noisy reconstructed meshes, which results in a "bumpy" appearance. MobileNeRF represents the scene as a disconnected collection of triangles, as its sole focus is view synthesis. As a result, its optimized and pruned "triangle soup" is highly noisy and may not be ideal for downstream tasks such as appearance editing.

As recently shown [Oechsle et al. 2021; Wang et al. 2021; Yariv et al. 2021], extracting an iso-surface directly from the density field predicted by NeRF can sometimes fail to faithfully capture the geometry of the scene. In Figure 5 we show this effect using Mip-NeRF 360 and extract the iso-surface where its density field exceeds a value of 50. Note how the surface of the table is no longer flat, as the reflection of the vase is modeled using mirror-world geometry. In contrast, our method produces a smooth and high-fidelity mesh, which is better suited for appearance and illumination editing, as demonstrated in Figure 1.

5.3 Appearance model ablation

In Table 3 we present the results of an ablation study of our spherical Gaussian appearance model. We see that reducing the number of SGs to 2, 1, and 0 (i.e., a diffuse model) causes accuracy to degrade monotonically. However, when using 3 SGs in the periphery our model tends to overfit to the training views, causing a slight drop in quality compared to our proposed model with just a single peripheral SG. Furthermore, compared to 3 SGs everywhere, using a single SG in the periphery reduces the average size vertex by 1.52× (from 36 to 23.76 bytes), which significantly reduces the memory

Table 3: An ablation study of our view-dependent appearance model on all scenes from the mip-NeRF 360 dataset.

	PSNR ↑	SSIM ↑	LPIPS ↓	MB (GPU) ↓
Diffuse (0 Spherical Gaussians)	22.32	0.636	0.352	436.1
1 Spherical Gaussian	24.02	0.680	0.322	549.1
2 Spherical Gaussian	24.39	0.693	0.312	662.2
3 SGs in the periphery	24.34	0.688	0.317	775.3
View-dependent MLP [2021]	24.30	0.687	0.318	516.8
L2 loss	24.52	0.690	0.316	572.6
Ours	24.51	0.697	0.309	572.6

bandwidth consumption (a major performance bottleneck for rendering). Perhaps surprisingly, replacing our SG appearance model with the small view-dependent MLP used by both SNeRG [Hedman et al. 2021] and MobileNeRF [Chen et al. 2022a] significantly reduces rendering quality and yields error metrics that are roughly comparable to the "1 Spherical Gaussian" ablation. This is especially counter-intuitive given the significant cost of evaluating a small MLP (∼ 2070 FLOPS per pixel) compared to a single spherical Gaussian (21 FLOPS per pixel). Additionally, we ablate the robust loss used to train our appearance representation with a simple L2 loss, which unsurprisingly boosts PSNR (which is inversely proportional to MSE) at the expense of the other metrics.

Ground truth Our rendering Our mesh

Figure 6: Our framework is based on the neural SDF representation, which struggles to represent semi-transparent objects or thin structures. These limitations can further affect our rendering reconstruction performance.

5.4 Limitations

Although our model achieves state-of-the-art speed and accuracy for the established task of real-time rendering of unbounded scenes, there are several limitations that represent opportunities for future improvement: We represent the scene using a fully opaque mesh representation, and as such our model may struggle to represent semi-transparent content (glass, fog, etc.). And as is common for mesh-based approaches, our model sometimes fails to accurately represent areas with small or detailed geometry (dense foliage, thin structures, etc.). Figure 6 depicts additional extracted meshes visualization which demonstrates our surface reconstruction limitations and their effect on the rendering reconstruction. These concerns could perhaps be addressed by augmenting the mesh with opacity values, but allowing for continuous opacity would require a complex polygon sorting procedure that is difficult to integrate into a real-time rasterization pipeline. One additional limitation of our technique is that our model's output meshes occupy a significant amount of on-disk space (∼ 430 megabytes per scene), which may prove challenging to store or stream for some applications. This could be ameliorated through mesh simplification followed by UV atlasing. However, we found that existing tools for simplification and atlasing, which are mostly designed for artist-made 3D assets, did not work well for our meshes extracted by marching cubes.

6 CONCLUSION

We have presented a system that produces a high-quality mesh for real-time rendering of large unbounded real world scenes. Our technique first optimizes a hybrid neural volume-surface representation of the scene that is designed for accurate surface reconstruction. From this hybrid representation, we extract a triangle mesh whose vertices contain an efficient representation of view-dependent appearance, then optimize this meshed representation to best reproduce the captured input images. This results in a mesh that yields state-of-the-art results for real-time view synthesis in terms of both speed and in accuracy, and is of a high enough quality to enable downstream applications.

ACKNOWLEDGMENTS

We would like to thank Forrester Cole and Srinivas Kaza for their implementation of the JAX rasterizer, Simon Rodriguez as an invaluable source of knowledge for real-time graphics programming, and Marcos Seefelder for brainstorming the real-time renderer. We further thank Thomas Müller for his valuable advice on tuning Instant-NGP for the Mip-NeRF 360 dataset, and Zhiqin Chen for generously sharing with us the MobileNeRF evaluations. Lastly, we thank Keunhong Park for thoughtful review of our manuscript.

REFERENCES

Bao and Yang, Zeng Junyi, Bao Hujun, Zhang Yinda, Cui Zhaopeng, and Zhang Guofeng. 2022. NeuMesh: Learning Disentangled Neural Mesh-based Implicit Field for Geometry and Texture Editing. In *European Conference on Computer Vision (ECCV)*.

Jonathan T. Barron. 2019. A General and Adaptive Robust Loss Function. *CVPR* (2019).

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. 2022. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. *CVPR* (2022).

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432* (2013).

Alexander W. Bergman, Petr Kellnhofer, and Gordon Wetzstein. 2021. Fast Training of Neural Lumigraph Representations using Meta Learning. In *NeurIPS*.

Mark Boss, Raphael Braun, Varun Jampani, Jonathan T. Barron, Ce Liu, and Hendrik P. A. Lensch. 2021. NeRD: Neural Reflectance Decomposition from Image Collections. *ICCV* (2021).

Mark Boss, Andreas Engelhardt, Abhishek Kar, Yuanzhen Li, Deqing Sun, Jonathan T. Barron, Hendrik P.A. Lensch, and Varun Jampani. 2022. SAMURAI: Shape And Material from Unconstrained Real-world Arbitrary Image collections. *NeurIPS* (2022).

Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. 2001. Unstructured Lumigraph Rendering. *SIGGRAPH* (2001).

Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. 2022b. TensoRF: Tensorial Radiance Fields. *ECCV* (2022).

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:2208.00277* (2022).

Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. 1996. Modeling and Rendering Architecture from Photographs: A hybrid geometry- and image-based approach. *SIGGRAPH* (1996).

Yasutaka Furukawa and Carlos Hernández. 2015. Multi-View Stereo: A Tutorial. *Foundations and Trends in Computer Graphics and Vision* (2015).
Stephan J. Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. 2021. FastNeRF: High-Fidelity Neural Rendering at 200FPS. *ICCV* (2021).
Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. 1996. The lumigraph. *SIGGRAPH* (1996).
Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. 2020. Implicit Geometric Regularization for Learning Shapes. *Proceedings of Machine Learning and Systems* (2020).
Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. 2018. Deep blending for free-viewpoint image-based rendering. *SIGGRAPH Asia* (2018).
Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. 2021. Baking Neural Radiance Fields for Real-Time View Synthesis. *ICCV* (2021).
ISO/IEC 12113:2022 2022. *Information technology — Runtime 3D asset delivery format — Khronos glTF 2.0.* Standard. International Organization for Standardization.
Michal Jancosek and Tomás Pajdla. 2011. Multi-view reconstruction preserving weakly-supported surfaces. (2011).
Animesh Karnewar, Tobias Ritschel, Oliver Wang, and Niloy Mitra. 2022. ReLU fields: The little non-linearity that could. *SIGGRAPH* (2022).
Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. 2006. Poisson Surface Reconstruction. *Symposium on Geometry Processing* (2006).
Michael Kazhdan and Hugues Hoppe. 2013. Screened Poisson Surface Reconstruction. *ACM TOG* (2013).
Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. 2021. Point-Based Neural Rendering with Per-View Optimization. *Computer Graphics Forum* (2021).
Zhengfei Kuang, Kyle Olszewski, Menglei Chai, Zeng Huang, Panos Achlioptas, and Sergey Tulyakov. 2022. NeROIC: Neural Rendering of Objects from Online Image Collections. *SIGGRAPH* (2022).
Patrick Labatut, Jean-Philippe Pons, and Renaud Keriven. 2007. Efficient multi-view reconstruction of large-scale scenes using interest points, Delaunay triangulation and graph cuts. *ICCV* (2007).
Marc Levoy and Pat Hanrahan. 1996. Light field rendering. *SIGGRAPH* (1996).
Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. 2019. Neural Volumes: Learning Dynamic Renderable Volumes from Images. *SIGGRAPH* (2019).
W. E. Lorensen and H. E. Cline. 1987. Marching cubes: A high resolution 3D surface construction algorithm. *SIGGRAPH* (1987).
Nelson Max. 1995. Optical models for direct volume rendering. *IEEE TVCG* (1995).
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. *ECCV* (2020).
Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *SIGGRAPH* (2022).
Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H. Mueller, Chakravarty R. Alla Chaitanya, Anton Kaplanyan, and Markus Steinberger. 2021. DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. *Computer Graphics Forum* (2021).
Michael Oechsle, Songyou Peng, and Andreas Geiger. 2021. UNISURF: Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View Reconstruction. *ICCV* (2021).
Eric Penner and Li Zhang. 2017. Soft 3D Reconstruction for View Synthesis. *SIGGRAPH Asia* (2017).
Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. 2021. KiloNeRF: Speeding up neural radiance fields with thousands of tiny MLPs. *ICCV* (2021).
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. *SIGGRAPH* (2023).
Gernot Riegler and Vladlen Koltun. 2020. Free View Synthesis. *ECCV* (2020).
Gernot Riegler and Vladlen Koltun. 2021. Stable view synthesis. *CVPR* (2021).
Darius Rückert, Linus Franke, and Marc Stamminger. 2022. ADOP: Approximate differentiable one-pixel point rendering. *SIGGRAPH* (2022).
Pedro V. Sander, Diego Nehab, and Joshua Barczak. 2007. Fast Triangle Reordering for Vertex Locality and Reduced Overdraw. *SIGGRAPH* (2007).
Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. 2016. Pixelwise View Selection for Unstructured Multi-View Stereo. *ECCV* (2016).
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. *CVPR* (2021).
Pratul P. Srinivasan, Richard Tucker Joand nathan T. Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. 2019. Pushing the Boundaries of View Extrapolation with Multiplane Images. *CVPR* (2019).
Cheng Sun, Min Sun, and Hwann-Tzong Chen. 2022. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. *CVPR* (2022).
Richard Szeliski and Polina Golland. 1999. Stereo Matching with Transparency and Matting. *IJCV* (1999).
Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, W Yifan, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, et al. 2022. Advances in neural rendering. *Computer Graphics Forum* (2022).
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. *CVPR* (2022).
G. Vogiatzis, C. Hernández, P. Torr, and R. Cipolla. 2007. Multi-View Stereo via Volumetric Graph-Cuts and Occlusion Robust Photo-Consistency. *IEEE TPAMI* (2007).
Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. 2021. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. *NeurIPS* (2021).
Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. 2021. NeX: Real-time View Synthesis with Neural Basis Expansion. *CVPR* (2021).
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. *SIGGRAPH* (2000).
Xiuchao Wu, Jiamin Xu, Zihan Zhu, Hujun Bao, Qixing Huang, James Tompkin, and Weiwei Xu. 2022. Scalable Neural Indoor Scene Rendering. *ACM TOG* (2022).
Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. 2021. Volume rendering of neural implicit surfaces. *NeurIPS* (2021).
Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. 2022. Plenoxels: Radiance fields without neural networks. *CVPR* (2022).
Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. 2021. PlenOctrees for real-time rendering of neural radiance fields. *ICCV* (2021).
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. *CVPR* (2021).
Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. 2020. NeRF++: Analyzing and Improving Neural Radiance Fields. *arXiv:2010.07492* (2020).
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. *SIGGRAPH Asia* (2021).
Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. 2018. Stereo Magnification: Learning View Synthesis Using Multiplane Images. *SIGGRAPH* (2018).