HybridNeRF: Efficient Neural Rendering via Adaptive Volumetric Surfaces

Haithem Turki[1, 2] Vasu Agrawal[1] Samuel Rota Bulò[1] Lorenzo Porzi[1]
Peter Kontschieder[1] Deva Ramanan[2] Michael Zollhöfer[1] Christian Richardt[1]

[1]Meta Reality Labs [2]Carnegie Mellon University

Abstract

Neural radiance fields provide state-of-the-art view synthesis quality but tend to be slow to render. One reason is that they make use of volume rendering, thus requiring many samples (and model queries) per ray at render time. Although this representation is flexible and easy to optimize, most real-world objects can be modeled more efficiently with surfaces instead of volumes, requiring far fewer samples per ray. This observation has spurred considerable progress in surface representations, such as signed distance functions, but these may struggle to model semi-opaque and thin structures. We propose a method, HybridNeRF, that leverages the strengths of both representations by rendering most objects as surfaces while modeling the (typically) small fraction of challenging regions volumetrically. We evaluate HybridNeRF against the challenging Eyeful Tower dataset [38] along with other commonly used view synthesis datasets. When comparing to state-of-the-art baselines, including recent rasterization-based approaches, we improve error rates by 15–30% while achieving real-time framerates (at least 36 FPS) for virtual-reality resolutions (2K×2K). Project page: https://haithemturki.com/hybrid-nerf/.

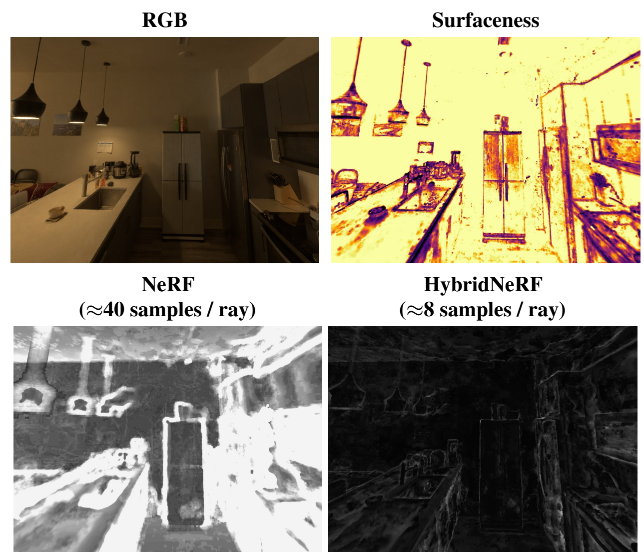

Figure 1. **HybridNeRF.** We train a hybrid surface–volume representation via *surfaceness* parameters that allow us to render most of the scene with few samples. We track Eikonal loss as we increase surfaceness to avoid degrading quality near fine and translucent structures (such as wires). On the bottom, we visualize the number of samples per ray (brighter is higher). Our model renders in high fidelity at 2K×2K resolution at real-time frame rates.

1. Introduction

Recent advances in volumetric rendering of neural radiance fields [22] (NeRFs) have led to significant progress towards photorealistic novel-view synthesis. However, while NeRFs provide state-of-the-art rendering quality, they remain slow to render.

Efficiency. We seek to construct a representation that enables high-quality efficient rendering, which is necessary for immersive applications, such as augmented reality or virtual teleconferencing.

While recent rasterization-based techniques, such as mesh baking [5, 40] or Gaussian splatting [14], are very efficient, they still struggle to capture transparent or fine structures, and view-dependent effects (like reflections or specularities), respectively. Instead, we focus on NeRF's standard ray casting paradigm, and propose techniques that enable a better speed–quality trade-off.

Rendering. We start with the observation that neural implicit surface representations, such as signed distance functions (SDFs), which were originally proposed to improve the geometry quality of NeRFs via regularization [35, 39], can *also* be used to dramatically increase efficiency by requiring fewer samples per ray. In the limit, only a single sample on the surface is required. In practice, renderers still need to identify the location of the target sample(s), which can be done by generating samples via an initial proposal network [2] or other techniques, such as sphere tracing [20].

Surfaceness. While surface-based neural fields are convenient for rendering, they often struggle to reconstruct scenes with thin structures or view-dependent effects, such as reflec-

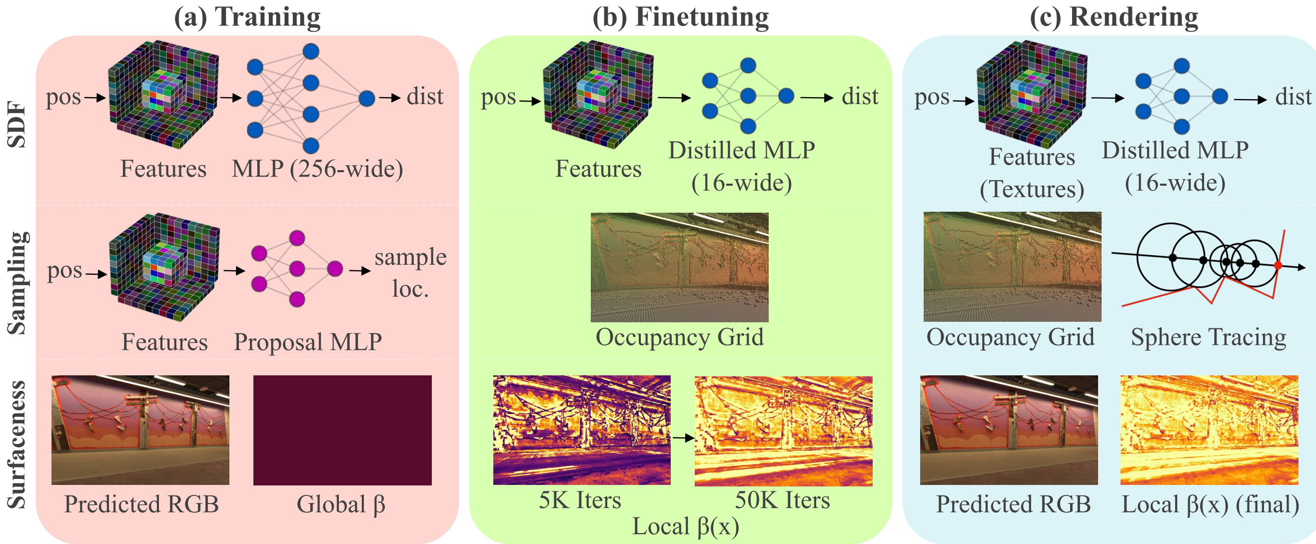

Figure 2. **Approach.** In the first phase of our pipeline (**a**), we train a VolSDF-like [39] model with distance-adjusted Eikonal loss to model backgrounds without a separate NeRF (Sec. 3.3). We then crucially transition from a uniform surfaceness parameter β to position-dependent $\beta(\boldsymbol{x})$ values to model most of the scene as thin surfaces (needing few samples) without degrading quality near fine and semi-opaque structures (**b**). Since our model behaves as a valid SDF in >95% of the scene, we use sphere tracing at render time (**c**) along with lower-level optimizations (hardware texture interpolation) to query each sample as efficiently as possible.

tions and translucency. This is one reason that surfaces are often transformed into volumetric models for rendering [39]. A crucial transformation parameter is a scalar temperature β that is used to convert a β-scaled signed distance value into a density. Higher temperatures tend to produce an ideal binary occupancy field that can improve rendering speed but can struggle for challenging regions as explained above. Lower temperatures allow the final occupancy field to remain flexible, whereby the β-scaled SDF essentially acts as a reparameterization of the underlying occupancy field. As such, we refer to β as the *surfaceness* of the underlying scene (see Fig. 1). Prior work treats β as a global parameter that is explicitly scheduled or learned via gradient descent [39]. We learn it in a spatially adaptive manner.

Contributions. Our primary contribution is a hybrid surface–volume representation that combines the best of both worlds. Our key insight is to replace the global parameter β with spatially-varying parameters $\beta(\boldsymbol{x})$ corresponding to the surfaceness of regions in the 3D scene. At convergence, we find that most of the scene ($> 95\%$) can be efficiently modeled as a surface. This allows us to render with far fewer samples than fully volumetric methods, while achieving higher fidelity than pure surface-based approaches. Additionally,

1. We propose a weighted Eikonal regularization that allows our method to render high-quality complex backgrounds without a separate background model.
2. We implement specific rendering optimizations, such as hardware texture interpolation and sphere tracing, to significantly accelerate rendering at high resolutions.
3. We present state-of-the-art reconstruction results on three different datasets, including the challenging Eyeful Tower dataset [38], while rendering almost 10× faster.

2. Related Work

Many works try to accelerate the rendering speed of neural radiance fields (NeRF). We discuss a representative selection of such approaches below.

Voxel baking. Some of the earliest NeRF acceleration methods store precomputed non-view dependent model outputs, such as spherical harmonics coefficients, into finite-resolution structures [6, 7, 9, 13, 43]. These outputs are combined with viewing direction to compute the final radiance at render time, bypassing the original model entirely. Although these methods render extremely quickly (some >200 FPS [9]), they are limited by the finite capacity of the caching structure and cannot capture fine details at room scale.

Feature grids. Recent methods use a hybrid approach that combines a learned feature grid with a much smaller MLP than the original NeRF [5, 8, 23]. Instant-NGP [23] (iNGP), arguably the most popular of these methods, encodes features into a multi-resolution hash table. Although these representations speed up rendering, they cannot reach the level needed for real-time HD rendering alone, as even iNGP reaches less than 10 FPS on real-world datasets at high resolution. MERF [29] comes closest through a baking pipeline that uses various sampling and memory layout optimizations that we also make use of in our implementation.

Surface–volume representations. Several methods [25, 35, 39] derive density values from the outputs of a signed distance function, which are then rendered volumetrically as in NeRF. These hybrid representations retain NeRF's ease of optimization while improving surface geometry. Follow-ups [10, 40] bake the resulting surface geometry into a mesh that is further optimized and simplified. Similar to early voxel-baking approaches, these methods render quickly (>70 FPS) but are limited by the capacity of the mesh and texture, and thus struggle to model thin structures, transparency, and view-dependent effects. We train a similar SDF representation in our method but continue using the base neural model at render time. Concurrent to our work, Adaptive shells [37] augments NeuS [35] with a spatially-varying kernel similar to our adaptive surfaceness described in Sec. 3.2.

Sample efficiency. Several approaches accelerate rendering by intelligently placing far fewer samples along each ray than the original hierarchical strategy proposed by NeRF [2, 11, 18, 24, 27]. These methods all train auxiliary networks that are cheaper to evaluate than the base model. However, as they are based on purely volumetric representations, they are limited in practice as to how few samples they can use per ray without degrading quality, and therefore exhibit a different quality–performance tradeoff curve than ours.

Gaussians. Recent methods take inspiration from NeRF's volume rendering formula but discard the neural network entirely and instead parameterize the scene through a set of 3D Gaussians [14–16, 34]. Of these, 3D Gaussian splatting [14] has emerged as the new state of the art, rendering at >100 FPS with higher fidelity than previous non-neural approaches. Although encouraging, it is sensitive to initialization (especially in far-field areas) and limited in its ability to reason about inconsistencies within the training dataset (such as transient shadows) and view dependent effects.

3. Method

Given a collection of RGB images and camera poses, our goal is to learn a 3D representation that generates novel views at VR resolution (at least 2K×2K pixels) in real-time (at least 36 FPS), while achieving a high degree of visual fidelity. As we target captures taken under real-world conditions, our representation must be able to account for inconsistencies across training images due to lighting changes and shadows (even in "static" scenes). We build upon NeRF's raycasting paradigm, which can generate highly photorealistic renderings, and improve upon its efficiency. As the world mostly consists of surfaces, we train a representation that can render surfaces with few samples and without degrading the rest of the scene. We outline our method in Fig. 2 and present our model architecture and the first training stage in Sec. 3.1, which is followed by finetuning of our model to accelerate rendering without compromising quality in Sec. 3.2. We discuss how to model unbounded scenes in Sec. 3.3 and present final render-time optimizations in Sec. 3.4.

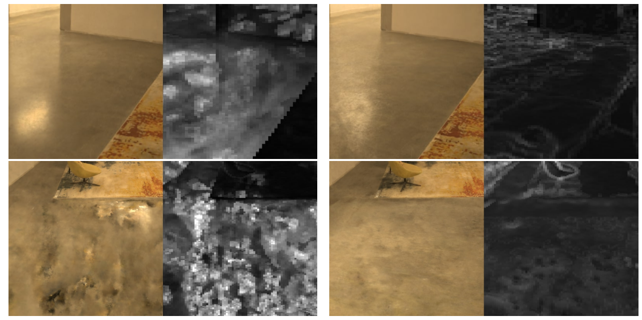

Figure 3. **Surfaces.** Since NeRF directly predicts density, it often 'cheats' by modeling specular surfaces, such as floors, as semi-transparent volumes that require many samples per ray (heatmaps shown on the **right**, with brighter values corresponding to more samples). Methods that derive density from signed distances, such as ours, improve surface geometry and appearance while using fewer samples per ray.

3.1. Representation

Preliminaries. NeRF [22] represents a scene as a continuous volumetric radiance field that encodes the scene's geometry and view-dependent appearance within the weights of an MLP. NeRF renders pixels by sampling positions $\boldsymbol{x}_i$ along the corresponding camera ray, querying the MLP to obtain density and color values, $\sigma_i := \sigma(\boldsymbol{x}_i)$ and $\mathbf{c}_i := \boldsymbol{c}(\boldsymbol{x}_i, \boldsymbol{d}_r)$, respectively (with $\boldsymbol{d}_r$ as the ray direction). The density values σ_i are converted into opacity values $\alpha_i := 1 - \exp(-\sigma_i \delta_i)$, where δ_i is the distance between samples. The final ray color $\hat{\boldsymbol{c}}_r := \sum_{i=0}^{N-1} \boldsymbol{c}_i w_i$ is obtained as the combination of the color samples $\boldsymbol{c}_i$ with weights $w_i := \exp(-\sum_{j=0}^{i-1} \sigma_j \delta_j) \alpha_i$. The training process optimizes the model by sampling batches of image pixels and minimizing the L2 reconstruction loss. We refer to Mildenhall et al. [22] for details.

Modeling density. The original NeRF representation has the flexibility of representing semi-transparent surfaces, for the density field is not forced to saturate. However, the model often abuses this property by generating semi-transparent volumes to mimic reflections and other view-dependent effects (Fig. 3). This hampers our goal of minimizing the samples per ray needed for rendering.

To address this problem, surface–volume representations [25, 35, 39] learn well-defined surfaces by interpreting MLP outputs $f(\boldsymbol{x})$ as a signed distance field (SDF) to represent scene surfaces as the zero-level set of the function f.

As the norm of the gradient of an SDF should typically be 1, the MLP is regularized via the Eikonal loss:

$$\mathcal{L}_{\text{Eik}}(\mathbf{r}) := \sum_{i=0}^{N-1} \eta_i (\|\nabla f(\boldsymbol{x}_i)\| - 1)^2, \tag{1}$$

where η_i is a per-sample loss weight typically set to 1. The signed distances are converted into densities σ_{SDF} that are

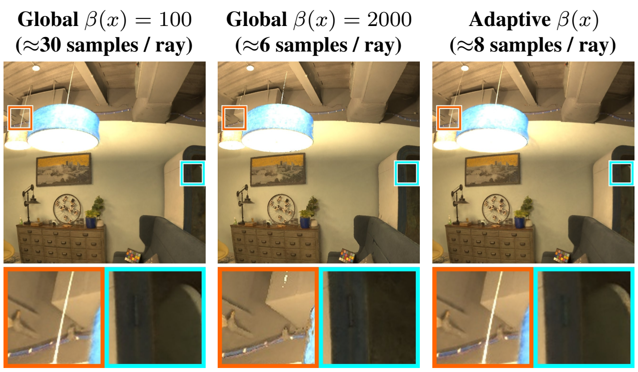

Figure 4. **Choice of β.** Increasing β reduces the number of samples needed to render per ray, but negatively impacts quality near fine objects (lamp wires) and transparent structures (glass door).

paired with color predictions, and rendered as in NeRF. Specifically, we follow VolSDF's approach [39] and define:

$$\sigma_{\text{SDF}}(\boldsymbol{x}) \coloneqq \beta(\boldsymbol{x})\Psi(f(\boldsymbol{x})\beta(\boldsymbol{x})), \tag{2}$$

where $\beta(\boldsymbol{x}) > 0$ determines the *surfaceness* of point $\boldsymbol{x}$, *i.e.* how concentrated the density should be around the zero-level set of f, and Ψ is the CDF of a standard Laplace distribution:

$$\Psi(s) = \begin{cases} \frac{1}{2}\exp(-s) & \text{if } s > 0 \\ 1 - \frac{1}{2}\exp(s) & \text{if } s \leq 0. \end{cases} \tag{3}$$

In prior works, the surfaceness $\beta(\boldsymbol{x})$ is independent of position $\boldsymbol{x}$. We instead consider a surfaceness *field* implemented as a 512^3 grid of values queried via nearest-neighbor interpolation. We first constrain the surfaceness parameters to be globally uniform, and allow them to diverge spatially during the finetuning stage (Sec. 3.2).

Model architecture. We use dense multi-resolution 3D feature grids in combination with multi-resolution triplanes [4, 8] to featurize 3D sample locations. We predict color $\boldsymbol{c}$ and signed distance f with separate grids, each followed by an MLP, and use a small proposal network similar to that used by Nerfacto [33] to improve sampling efficiency. For a given 3D point, we fetch $K = 4$ features per level from (1) the 3D feature grids at 3 resolution levels (128^3, 256^3 and 512^3) via trilinear interpolation, and (2) from triplanes at 7 levels (from 128^2 to $8{,}192^2$) via bilinear interpolation. We sum the features across levels (instead of concatenation [8, 23]), and concatenate the summed features from the 3D grid to those from the 3 triplanes to obtain a $4K = 16$-dimensional MLP input. We encode viewing direction through spherical harmonics (up to the 4th degree) as an auxiliary input to the color MLP. As our feature grid is multi-resolution, we handle aliasing as in VR-NeRF [38]. See Appendix B and Appendix C for more details.

Optimization. We sample random batches of training rays and optimize our color and distance fields by minimizing the photometric loss $\mathcal{L}_{\text{photo}}$ and Eikonal loss $\mathcal{L}_{\text{Eik}}$ along with interlevel loss $\mathcal{L}_{\text{prop}}$ [2] to train the proposal network:

$$\mathcal{L}(\boldsymbol{r}) \coloneqq \mathcal{L}_{\text{photo}}(\boldsymbol{r}) + \lambda_{\text{Eik}}\mathcal{L}_{\text{Eik}}(\boldsymbol{r}) + \mathcal{L}_{\text{prop}}(\boldsymbol{r}), \tag{4}$$

with $\lambda_{\text{Eik}} = 0.01$ in our experiments.

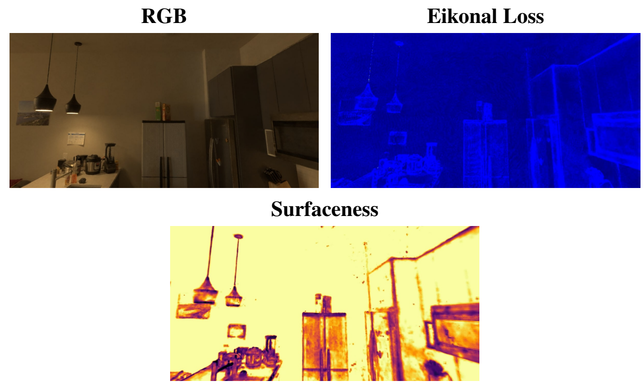

Figure 5. **Spatially adaptive surfaceness.** We make $\beta(\boldsymbol{x})$ spatially adaptive by means of a 512^3 voxel grid that we increase during the finetuning stage. We track Eikonal loss as we increase surfaceness as it is highest near object boundaries and semi-transparent surfaces (**top-right**, brighter = higher loss) that degrade when surfaceness is too high (Fig. 4). We stop increasing surfaceness in regions that cross a given threshold.

3.2. Finetuning

Adaptive surfaceness. The first stage of our pipeline uses a global surfaceness value $\beta(\boldsymbol{x}) = \bar{\beta}$ for all $\boldsymbol{x}$, as in existing approaches [35, 39]. As $\bar{\beta}$ increases, the density σ_{SDF} in free-space areas converges to zero (Eq. 3), reducing the required number of samples per ray. However, uniformly increasing this scene-wide parameter degrades the rendering quality near fine-grained and transparent structures (see Fig. 4).

We overcome this limitation by making $\beta(\boldsymbol{x})$ spatially adaptive via a 512^3 voxel grid. One possible approach is to directly optimize $\beta(\boldsymbol{x})$ via gradient descent, but we find that this overly relaxes the constraint on SDF correctness such that $f(\boldsymbol{x})$ predicts arbitrary density values as in the original NeRF. We instead rely on the Eikonal loss as a natural indicator of where the model cannot accurately reconstruct the scene via an SDF (and where we should therefore use a "softer" formulation). We collect per-sample triplets $(\boldsymbol{x}, \eta, w)$ rendered during the finetuning process, accumulate them over multiple training iterations (5,000), and partition them across the voxels of the surfaceness grid. Let $\Lambda_{\boldsymbol{v}}$ be the subset associated with voxel $\boldsymbol{v}$ corresponding to $\beta_{\boldsymbol{v}}$. We increase $\beta_{\boldsymbol{v}}$ by a fixed increment (100) if:

$$\frac{\sum_{(\boldsymbol{x},\eta,w)\in\Lambda_{\boldsymbol{v}}} w\eta(\|\nabla f(\boldsymbol{x})\| - 1)^2}{\sum_{(\ldots,w)\in\Lambda_{\boldsymbol{v}}} w} < \bar{\gamma}, \tag{5}$$

where $\bar{\gamma} \coloneqq 0.25$ is a predefined threshold. Fig. 5 illustrates our approach.

Proposal network baking. Although the proposal network allows us to quickly learn the scene geometry during the first stage of training, it is too expensive to evaluate in real time. We follow MERF's protocol [29] to bake the proposal network into a 1024^3 binary occupancy grid. We render all training rays and mark a voxel as occupied if there exists at least one sampled point $\boldsymbol{x}_i$ such that $\max(w_i, \sigma_i) > 0.005$. We finetune our model using the occupancy grid to prevent any loss in quality.

MLP distillation. We find it important to use a large 256 channel-wide MLP to represent the signed distance f during the first training phase in order to learn accurate scene geometry. However, we later distill f into a smaller 16-wide network (f_{small}). We do so by sampling random rays from our training set for 5,000 iterations and minimizing the difference between $f(\boldsymbol{x}_i)$ and $f_{\text{small}}(\boldsymbol{x}_i)$ at every sampled point:

$$\mathcal{L}_{\text{dist}}(\boldsymbol{r}) := \sum_{i=0}^{N-1} |f(\boldsymbol{x}_i) - f_{\text{small}}(\boldsymbol{x}_i)|, \tag{6}$$

with a stop gradient applied to the outputs of f. We then discard the original SDF f and switch to using the distilled counterpart f_{small} for the rest of the finetuning stage.

3.3. Backgrounds

Many scenes we wish to reconstruct contain complex backgrounds that surface–volume methods struggle to replicate [19, 35, 39]. BakedSDF [40] defines a contraction space [2] in which the Eikonal loss of Eq. 1 is applied. However, we found this to negatively impact foreground quality. Other approaches use separate NeRF background models [45], which effectively doubles inference and memory costs, and makes them ill-suited for real-time rendering.

Relation between volumetric and surface-based NeRFs. We discuss how to make a single MLP behave as an approximate SDF in the foreground and a volumetric model in the background. Both types of NeRF derive density σ by applying a non-linearity to the output of an MLP. Our insight is that although the original NeRF uses ReLU, any non-linear mapping to $\mathbb{R}^+$ may be used in practice, including our scaled CDF Ψ (β omitted without loss of generality). Since Ψ is invertible (as it is a CDF), $\sigma(\boldsymbol{x})$ and $\Psi(f(\boldsymbol{x}))$ are functionally equivalent as there exists an f such that $\Psi(f(\boldsymbol{x})) = \sigma(\boldsymbol{x})$ for any given point $\boldsymbol{x}$. Put otherwise, it is the Eikonal regularization that causes the divergence in behavior between both methods — in its absence, an "SDF" MLP is free to behave exactly as the density MLP in the original NeRF!

Distance-adjusted loss. We use a *distance-adjusted* Eikonal loss during training by using per-sample loss weights $\eta_i = \frac{1}{d_i^2}$ (where d_i is the metric distance along the ray of sample $\boldsymbol{x}_i$) instead of commonly-used uniform weights ($\eta_i = 1$) to downweight the loss applied to far-field regions. Intuitively, this encourages our method to behave as a valid SDF in the foreground (with well-defined surfaces) and more like NeRF in the background (to enable accurate reconstruction) without the need for separate foreground and background models. Fig. 6 and Tab. 7 illustrate the different approaches.

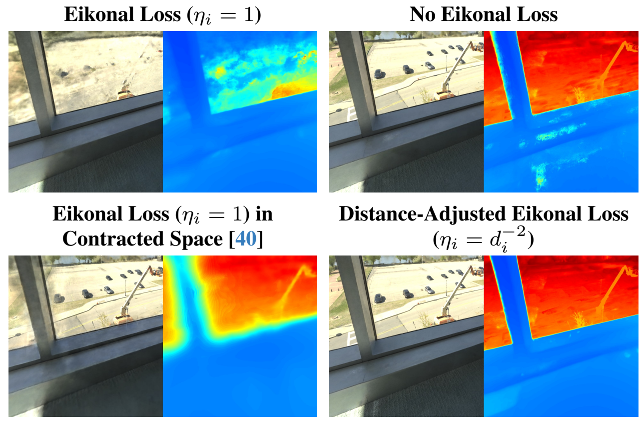

Figure 6. **Backgrounds.** Using standard Eikonal loss affects background reconstruction (**top-left**) while applying it in contracted space [40] affects the foreground (**bottom-left**). Omitting Eikonal loss entirely causes surface–volume methods to revert to NeRF's behavior, which improves background quality but degrades foreground surface reconstruction (**top-right**). By using distance-adjusted sample weights $\eta_i = d_i^{-2}$, we improve background reconstruction without impacting foreground quality (**bottom-right**).

3.4. Real-Time Rendering

Texture storage. Our architecture enables us to use lower-level optimizations. Methods such as iNGP [23] use concatenated multi-resolution features stored in hash tables. Since we use explicit 3D grids and triplanes, we can store our features as textures at render time, taking advantage of increased memory locality and texture interpolation hardware. As we sum our multi-resolution features during training, we optimize the number of texture fetches by storing pre-summed features g' at resolution level L (where we store $g'(\boldsymbol{v}) = \sum_{l=0}^{L} g(\boldsymbol{v}, l)$ for each texel in L). For a given sample $\boldsymbol{x}$ at render time, we obtain its anti-aliased feature by interpolating between the two levels implied by its pixel area $p(\boldsymbol{x})$, reducing the number of texture fetches to 8 queries per MLP evaluation from the original $3+3\times7=24$ (assuming three 3D grids and seven triplane levels), a $3\times$ reduction.

Sphere tracing. Volumetric methods that use occupancy grids [e.g. 23, 29] sample within occupied voxels using a given step size. This hyperparameter must be carefully tuned to strike the proper balance between quality (not skipping thin surfaces) and performance (not excessively sampling empty space). Modeling an SDF allows us to sample more efficiently by advancing toward the predicted surface using *sphere tracing* [31]. At each sample point $\boldsymbol{x}_i$ and predicted surface distance $s = f(\boldsymbol{x}_i)$, we advance by $0.9s$ (chosen empirically to account for our model behaving as an approxi-

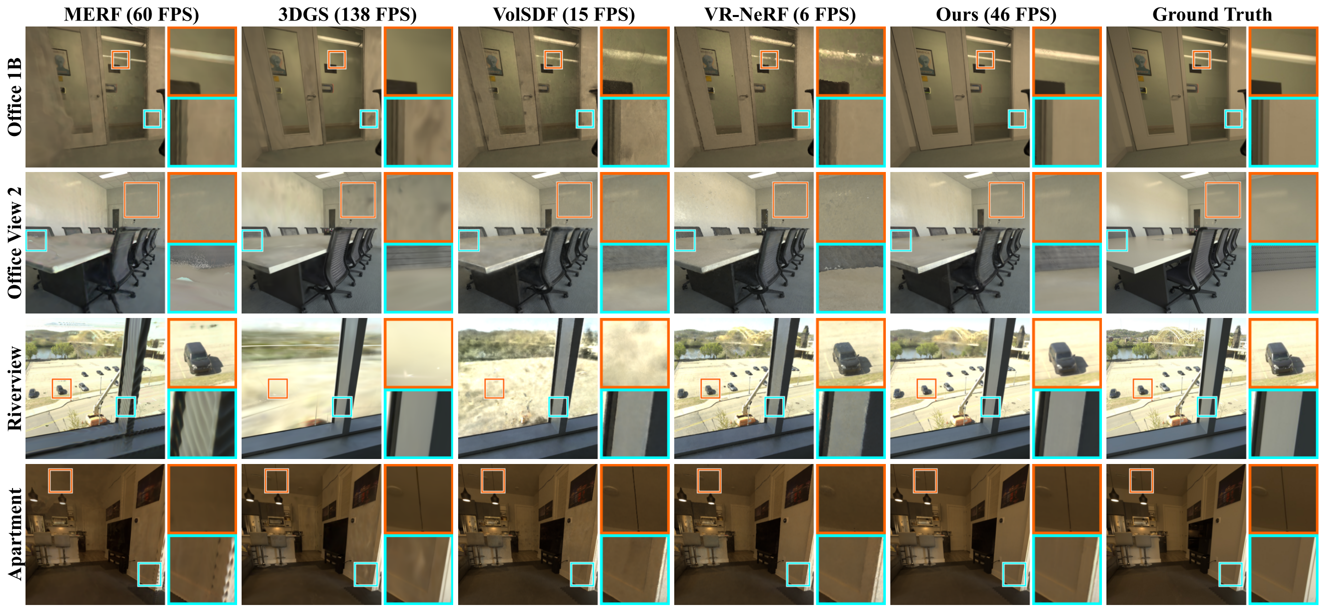

Figure 7. **Eyeful Tower [38].** HybridNeRF is the only method to accurately model reflections and shadows (**first two rows**), far-field content (**third row**) and fine structures (**bottom row**) at real-time frame rates at 2K×2K resolution.

Table 1. **Eyeful Tower [38] results.** We omit 3DGS results for fisheye scenes as their implementation does not handle fisheye projection. Along with 3DGS and MERF, ours is the only to reach the 36 FPS target for VR along with a >1.5 dB PSNR improvement in quality.

	Pinhole			Fisheye			Overall			
	↑PSNR	↑SSIM	↓LPIPS	↑PSNR	↑SSIM	↓LPIPS	↑PSNR	↑SSIM	↓LPIPS	↑FPS
iNGP* [23]	27.35	0.826	0.361	33.32	0.938	0.155	30.06	0.877	0.267	4.55
VolSDF* [39]	27.10	0.856	0.310	34.09	0.951	0.116	30.28	0.899	0.222	15.29
MERF (pre-baking) [29]	26.44	0.831	0.506	31.18	0.922	0.549	28.59	0.872	0.526	18.11
MERF (baked) [29]	25.99	0.830	0.525	31.09	0.921	0.546	28.31	0.871	0.535	60.18
3D Gaussian splatting [14]	27.42	0.877	0.291	—	—	—	—	—	—	**138.22**
VR-NeRF [38]	28.08	0.834	0.326	34.53	0.951	0.130	31.01	0.888	0.237	6.05
Zip-NeRF [3]	**29.71**	0.868	0.305	34.19	**0.958**	**0.109**	**31.75**	0.909	0.216	<0.1
HybridNeRF	29.07	**0.880**	**0.268**	**34.57**	0.952	0.115	31.57	**0.913**	**0.198**	45.78

* Our implementation. VolSDF: with iNGP acceleration.

mate SDF) until hitting the surface (predicted as $s \leq 2{\times}10^{-4}$). We only perform sphere tracing where our model behaves as a valid SDF (determined by $\beta(\boldsymbol{x}_i) > 350$ in our experiments), and fall back to a predefined step size of 1 cm otherwise.

4. Experiments

As our goal is high-fidelity view synthesis at VR resolution ($\approx$ 4 megapixels), we primarily evaluate HybridNeRF against the Eyeful Tower dataset [38], which contains high-fidelity scenes designed for walkable VR (Sec. 4.2). We compare our work to a broader range of methods on additional datasets in Sec. 4.3. We ablate our design in Sec. 4.4.

4.1. Implementation

We train our models in the PyTorch framework [26] and implement our renderer in C++/CUDA. We parameterize unbounded scenes with MERF's piecewise-linear contraction [29] so that our renderer can query the occupancy grid via ray-AABB intersection. We train on each scene for 200,000 iterations (100,000 in each training stage) with 12,800 rays per batch using Adam [17] and a learning rate of 2.5×10^{-3}.

4.2. VR Rendering

Eyeful Tower dataset. The dataset consists of room-scale captures, each containing high-resolution HDR images at 2K resolution, captured using a multi-view camera rig. Although care is taken to obtain the best quality images possible, inconsistencies still appear between images due to lighting changes and shadows from humans and the capture rig itself. We model as much of the dynamic range as possible by mapping colors in the PQ color space [32], as proposed in VR-NeRF [38], during training and tonemap to sRGB space

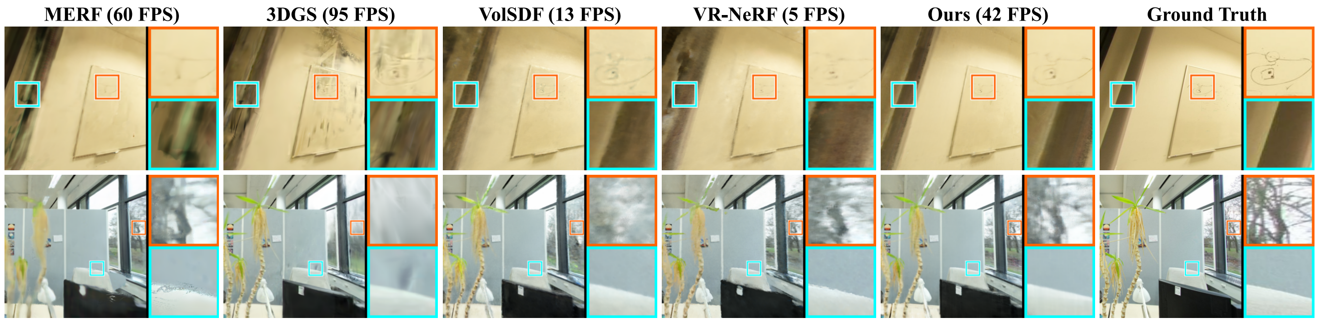

Figure 8. **ScanNet++ [41].** 3D Gaussian splatting [14] struggles with specular surfaces such as whiteboards (**above**) and far-field content (**below**). Our method performs best qualitatively while maintaining a real-time frame rate.

Table 2. **MipNeRF 360 [2].** Real-time methods are highlighted (**best**, second-best, *third-best*). Baseline numbers as published [6, 29, 40]. MobileNeRF [6] was not evaluated on indoor scenes. Our method performs similar to state-of-the-art real-time and offline methods.

	Outdoor			Indoor			Overall		
	↑PSNR	↑SSIM	↓LPIPS	↑PSNR	↑SSIM	↓LPIPS	↑PSNR	↑SSIM	↓LPIPS
NeRF [22]	21.46	0.458	0.515	26.84	0.790	0.370	23.85	0.606	0.451
NeRF++ [45]	22.76	0.548	0.427	28.05	0.836	0.309	25.11	0.676	0.375
SVS [30]	23.01	0.662	0.253	28.22	0.907	0.160	25.33	0.771	0.212
Mip-NeRF 360 [2]	*24.47*	0.691	0.283	31.72	0.917	0.180	27.69	0.791	0.237
iNGP [23]	22.90	0.566	0.371	29.15	0.880	0.216	25.68	0.706	0.302
Zip-NeRF [3]	**25.46**	**0.747**	**0.170**	**32.29**	**0.931**	*0.106*	**28.49**	**0.829**	**0.142**
Deep Blending [12]	21.54	0.524	0.364	26.40	0.844	0.261	23.70	0.666	0.318
MobileNeRF [6]	21.95	0.470	0.470	—	—	—	—	—	—
BakedSDF [40]	22.47	0.585	0.349	29.15	0.880	0.216	25.68	0.706	0.302
MERF [29]	23.19	0.616	0.343	27.80	0.855	0.271	25.24	0.722	0.311
3D Gaussian splatting [14]	24.13	*0.707*	0.211	30.94	0.927	**0.081**	27.16	*0.805*	0.153
HybridNeRF	24.73	0.716	*0.224*	*31.01*	*0.920*	0.095	*27.52*	0.806	*0.167*

during evaluation to compare against non-HDR baselines.

Baselines. We compare HybridNeRF to baselines across the fidelity/speed spectrum. We benchmark several volumetric methods, including (1) iNGP [23], (2) VR-NeRF [38], which extends iNGP's [23] primitives to better handle HDR reconstruction, (3) Zip-NeRF [3], an anti-aliasing method that generates high-quality renderings at the cost of speed, and (4) MERF [29], a highly optimized method that uses sampling and memory layout optimizations to accelerate rendering. We also compare to VolSDF [39] as a hybrid surface–volume method similar to the first stage of our method. As the original VolSDF implementation uses large MLPs that are unsuitable for real-time rendering, we use an optimized version built on top of iNGP's acceleration primitives as a fairer comparison. We also benchmark 3D Gaussian splatting [14] as a non-neural approach that represents the current state of the art with across rendering quality and speed.

Metrics. We report quantitative results based on PSNR, SSIM [36], and the AlexNet implementation of LPIPS [46] and measure frame rates rendered at 2K$\times$2K resolution on a single NVIDIA RTX 4090 GPU.

Results. We summarize our results in Tab. 1 along with qualitative results in Fig. 7. VR-NeRF [38], iNGP [23], and Zip-NeRF [3] render well below real-time frame rates. Our VolSDF implementation, which uses the same primitives as iNGP, is 3$\times$ faster merely from the benefits of using a surface representation (and fewer samples per ray). MERF [29], as a volume representation, relies instead on precomputation to accelerate rendering by explicitly storing diffuse color and density outputs during its baking stage and using only a small MLP to model view-dependent effects. Although it reaches a high frame rate, it provides the least visually appealing results amongst our baselines. 3D Gaussian splatting [14] renders the fastest, but struggles with shadows and lighting changes across the training views and models them as unsightly floaters. Our method is the only to achieve both high quality and real-time frame rates.

Table 3. **Diagnostics.** A global learned β ($\approx$ 200) produces the highest-quality renderings, but is slow to render as much of the scene is modeled volumetrically. Increasing β improves rendering speed but results in worse accuracy. Our full method (with spatially-varying $\beta(\boldsymbol{x})$) gets the best of both worlds. Other innovations such as distance-adjusted Eikonal loss are crucial for ensuring high accuracy for scenes with complex backgrounds. Finally, distillation and hardware acceleration come at a minor quality cost while doubling rendering speed.

Methods	$\beta(\boldsymbol{x})$	Dist.	Distill	Textures	↑PSNR	↑SSIM	↓LPIPS	↑FPS
w/ Global β (learned)	✗	✓	✓	✓	**31.76**	**0.923**	**0.188**	28.79
w/ Global β = 2000	✗	✓	✓	✓	27.16	0.835	0.345	**47.47**
w/o distance-adjusted Eik.	✓	✗	✓	✓	29.97	0.856	0.260	45.42
w/o MLP Distillation	✓	✓	✗	✓	<u>31.65</u>	0.915	<u>0.193</u>	35.25
w/o CUDA Textures	✓	✓	✓	✗	31.62	<u>0.921</u>	0.195	28.48
Full Method	✓	✓	✓	✓	31.57	0.913	0.198	<u>45.78</u>

Table 4. **ScanNet++ [41] results.** Similar to Tab. 1, our method is the only to hit VR FPS rates along with 3DGS and MERF. Our quality is near-identical to Zip-NeRF while rendering >400× faster.

Method	↑PSNR	↑SSIM	↓LPIPS	↑FPS
iNGP* [23]	23.69	0.815	0.308	5.39
VolSDF* [39]	24.26	0.834	0.246	13.18
MERF (pre-baking) [29]	23.44	0.821	0.306	12.08
MERF (baked) [29]	23.19	0.820	0.308	<u>60.21</u>
3D Gaussian splatting [14]	23.76	0.830	0.248	**94.95**
VR-NeRF [38]	24.00	0.814	0.301	5.38
Zip-NeRF [3]	**24.79**	**0.863**	**0.216**	<0.1
HybridNeRF	<u>24.64</u>	<u>0.835</u>	<u>0.236</u>	41.90

* Our implementation. VolSDF: with iNGP acceleration.

4.3. Additional Comparisons

Datasets. We evaluate HybridNeRF on MipNeRF-360 [2] as a highly-referenced dataset evaluated by many prior methods, and ScanNet++ [41] as a newer benchmark built from high-resolution captures of indoor scenes that are relevant to our goal of enabling immersive AR/VR applications. We test on all scenes in the former and a subset of the latter.

Baselines. We compare HybridNeRF to a wide set of baselines on Mip-NeRF 360. We use the same set of baselines as in Sec. 4.2 for ScanNet++.

Results. We list results in Tab. 2 and Tab. 4. Our method performs comparably to the best on Mip-NeRF 360 across both real-time [14] and offline [2] methods. Although ScanNet++ [41] contains fewer lighting inconsistencies across training images than the Eyeful Tower dataset [38], 3D Gaussian splatting still struggles to reconstruct specular surfaces (whiteboards, reflective walls) and backgrounds (Tab. 4). Our method performs the best amongst real-time methods and comparably to Zip-NeRF [3], while rendering >400× faster.

4.4. Diagnostics

Methods. We ablate our design decisions by individually omitting the major components of our method, most notably: our distance-adjusted Eikonal loss, our adaptive surfaceness $\beta(\boldsymbol{x})$, MLP distillation, and hardware-accelerated textures (vs. iNGP [23] hash tables commonly used by other fast NeRF methods).

Results. We present results against the Eyeful Tower [38] in Tab. 3. Spatially adaptive surfaceness is crucial as using a global parameter degrades either speed (when β is optimized for quality) or rendering quality (when set for speed). Applying uniform Eikonal loss instead of our distance-adjusted variant degrades quality in unbounded scenes. Omitting the distillation process has a minor impact on quality relative to rendering speed. We note a similar finding when using iNGP [23] primitives instead of CUDA textures, which suggests that introducing hardware acceleration into these widely used primitives is a potential avenue for future research.

5. Limitations

Memory. Storing features in dense 3D grids and triplanes consumes significantly more memory than with hash tables [23]. Training is especially memory-intensive as intermediate activations must be stored for backpropagation along with per-parameter optimizer statistics. Storing features in a hash table during the training phase before "baking" them into explicit textures as in MERF [29] would ameliorate training-time consumption but not at inference time.

Training time. Although our training time is much faster than the original NeRF, it is about 2× slower than iNGP due to the additional backprogation needed for Eikonal regularization (in line with other "fast" surface approaches such as NeuS-facto [44]), and slower than 3D Gaussian splatting.

6. Conclusion

We present a hybrid surface–volume representation that combines the best of surface and volume-based rendering into a single model. We achieve state-of-the-art quality across several datasets while maintaining real-time frame rates at VR resolutions. Although we push the performance frontier of raymarching approaches, a significant speed gap remains next to splatting-based approaches [14]. Combining the advantages of our surface–volume representation with these methods would be a valuable next step.

References

[1] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, 2021. 11

[2] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, 2022. 1, 3, 4, 5, 7, 8

[3] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-NeRF: Anti-aliased grid-based neural radiance fields. In *ICCV*, 2023. 6, 7, 8

[4] Ang Cao and Justin Johnson. HexPlane: A fast representation for dynamic scenes. In *CVPR*, 2023. 4

[5] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. TensoRF: Tensorial radiance fields. In *ECCV*, 2022. 1, 2

[6] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. MobileNeRF: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *CVPR*, 2023. 2, 7

[7] Daniel Duckworth, Peter Hedman, Christian Reiser, Peter Zhizhin, Jean-François Thibert, Mario Lučić, Richard Szeliski, and Jonathan T. Barron. SMERF: Streamable memory efficient radiance fields for real-time large-scene exploration. arXiv:2312.07541, 2023. 2, 11

[8] Sara Fridovich-Keil, Giacomo Meanti, Frederik Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *CVPR*, 2023. 2, 4

[9] Stephan J. Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. FastNeRF: High-fidelity neural rendering at 200FPS. In *ICCV*, 2021. 2

[10] Yuan-Chen Guo, Yan-Pei Cao, Chen Wang, Yu He, Ying Shan, Xiaohu Qie, and Song-Hai Zhang. VMesh: Hybrid volume-mesh representation for efficient view synthesis. In *SIGGRAPH Asia*, 2023. 3

[11] Kunal Gupta, Miloš Hašan, Zexiang Xu, Fujun Luan, Kalyan Sunkavalli, Xin Sun, Manmohan Chandraker, and Sai Bi. MCNeRF: Monte Carlo rendering and denoising for real-time NeRFs. In *SIGGRAPH Asia*, 2023. 3

[12] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Trans. Graph.*, 37(6), 2018. 7

[13] Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. *ICCV*, 2021. 2

[14] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139:1–14, 2023. 1, 3, 6, 7, 8, 12

[15] Leonid Keselman and Martial Hebert. Approximate differentiable rendering with algebraic surfaces. In *ECCV*, 2022.

[16] Leonid Keselman and Martial Hebert. Flexible techniques for differentiable rendering with 3D Gaussians. arXiv:2308.14737, 2023. 3

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 6

[18] Andreas Kurz, Thomas Neff, Zhaoyang Lv, Michael Zollhöfer, and Markus Steinberger. AdaNeRF: Adaptive sampling for real-time rendering of neural radiance fields. In *ECCV*, 2022. 3

[19] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H. Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. In *CVPR*, 2023. 5

[20] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. DIST: Rendering deep implicit signed distance function with differentiable sphere tracing. In *CVPR*, 2020. 1

[21] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Trans. Graph.*, 38(4):29:1–14, 2019. 12

[22] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1, 3, 7

[23] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–15, 2022. 2, 4, 5, 6, 7, 8

[24] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H. Mueller, Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, and Markus Steinberger. DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. *Computer Graphics Forum*, 2021. 3

[25] Michael Oechsle, Songyou Peng, and Andreas Geiger. UNISURF: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *ICCV*, 2021. 2, 3

[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8024–8035, 2019. 6

[27] Martin Piala and Ronald Clark. TermiNeRF: Ray termination prediction for efficient neural rendering. In *3DV*, 2021. 3

[28] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. KiloNeRF: Speeding up neural radiance fields with thousands of tiny MLPs. In *ICCV*, 2021. 11

[29] Christian Reiser, Richard Szeliski, Dor Verbin, Pratul P. Srinivasan, Ben Mildenhall, Andreas Geiger, Jonathan T. Barron, and Peter Hedman. MERF: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *ACM Trans. Graph.*, 42(4):89:1–12, 2023. 2, 5, 6, 7, 8, 11

[30] Gernot Riegler and Vladlen Koltun. Stable view synthesis. In *CVPR*, 2021. 7

[31] Radu Alexandru Rosu and Sven Behnke. PermutoSDF: Fast multi-view reconstruction with implicit surfaces using permutohedral lattices. In *CVPR*, 2023. 5

[32] SMPTE. High dynamic range electro-optical transfer function of mastering reference displays. SMPTE Standard ST 2084:2014, Society of Motion Picture and Television Engineers, 2014. 6

[33] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David Mcallister, Justin Kerr, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *SIGGRAPH*, pages 72:1–12, 2023. 4

[34] Angtian Wang, Peng Wang, Jian Sun, Adam Kortylewski, and Alan Yuille. VoGE: A differentiable volume renderer using Gaussian ellipsoids for analysis-by-synthesis. In *ICLR*, 2023. 3

[35] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. NeuS: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *NeurIPS*, 2021. 1, 2, 3, 4, 5

[36] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004. 7

[37] Zian Wang, Tianchang Shen, Merlin Nimier-David, Nicholas Sharp, Jun Gao, Alexander Keller, Sanja Fidler, Thomas Müller, and Zan Gojcic. Adaptive shells for efficient neural radiance field rendering. *ACM Trans. Graph.*, 42(6):259:1–15, 2023. 3

[38] Linning Xu, Vasu Agrawal, William Laney, Tony Garcia, Aayush Bansal, Changil Kim, Samuel Rota Bulò, Lorenzo Porzi, Peter Kontschieder, Aljaž Božič, Dahua Lin, Michael Zollhöfer, and Christian Richardt. VR-NeRF: High-fidelity virtualized walkable spaces. In *SIGGRAPH Asia*, 2023. 1, 2, 4, 6, 7, 8, 11, 12

[39] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *NeurIPS*, 2021. 1, 2, 3, 4, 5, 6, 7, 8

[40] Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P. Srinivasan, Richard Szeliski, Jonathan T. Barron, and Ben Mildenhall. BakedSDF: Meshing neural SDFs for real-time view synthesis. In *SIGGRAPH*, 2023. 1, 3, 5, 7

[41] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. ScanNet++: A high-fidelity dataset of 3D indoor scenes. In *ICCV*, 2023. 7, 8, 12

[42] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. ScanNet++ Toolkit. `https://github.com/scannetpp/scannetpp`, 2023. Accessed: 2023-11-01. 12

[43] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for real-time rendering of neural radiance fields. In *ICCV*, 2021. 2

[44] Zehao Yu, Anpei Chen, Bozidar Antic, Songyou Peng, Apratim Bhattacharyya, Michael Niemeyer, Siyu Tang, Torsten Sattler, and Andreas Geiger. SDFStudio: A unified framework for surface reconstruction, 2022. 8

[45] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. NeRF++: Analyzing and improving neural radiance fields. arXiv:2010.07492, 2020. 5, 7

[46] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 7