2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)

2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) | 978-1-6654-6946-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/CVPR52688.2022.01258

Mega-NeRF: Scalable Construction of Large-Scale NeRFs for Virtual Fly-Throughs

Haithem Turki[1] Deva Ramanan[1,2] Mahadev Satyanarayanan[1]
[1]Carnegie Mellon University [3]Argo AI

Abstract

We use neural radiance fields (NeRFs) to build interactive 3D environments from large-scale visual captures spanning buildings or even multiple city blocks collected primarily from drones. In contrast to single object scenes (on which NeRFs are traditionally evaluated), our scale poses multiple challenges including (1) the need to model thousands of images with varying lighting conditions, each of which capture only a small subset of the scene, (2) prohibitively large model capacities that make it infeasible to train on a single GPU, and (3) significant challenges for fast rendering that would enable interactive fly-throughs. To address these challenges, we begin by analyzing visibility statistics for large-scale scenes, motivating a sparse network structure where parameters are specialized to different regions of the scene. We introduce a simple geometric clustering algorithm for data parallelism that partitions training images (or rather pixels) into different NeRF submodules that can be trained in parallel. We evaluate our approach on existing datasets (Quad 6k and UrbanScene3D) as well as against our own drone footage, improving training speed by 3x and PSNR by 12%. We also evaluate recent NeRF fast renderers on top of Mega-NeRF and introduce a novel method that exploits temporal coherence. Our technique achieves a 40x speedup over conventional NeRF rendering while remaining within 0.8 db in PSNR quality, exceeding the fidelity of existing fast renderers.

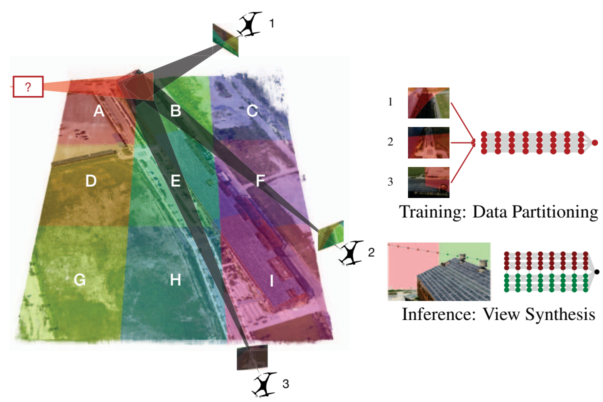


Figure 1. We scale neural reconstructions to massive urban scenes 1000x larger than prior work. To do so, Mega-NeRF decomposes a scene into a set of spatial cells (**left**), learning a separate NeRF submodule for each. We train each submodule with geometry-aware pixel-data partitioning, making use of *only* those pixels whose rays intersect that spatial cell (**top right**). For example, pixels from image 2 are added to the trainset of cells A, B, and F, reducing the size of each trainset by 10x. To generate new views for virtual fly-throughs, we make use of standard raycasting and point sampling, but query the encompassing submodule for each sampled point (**bottom right**). To ensure view generation is near-interactive, we make use of temporal coherence by caching occupancy and color values from nearby previous views (Fig. 4).

1. Introduction

Recent advances in neural rendering techniques have lead to significant progress towards photo-realistic novel view synthesis, a prerequisite towards many VR and AR applications. In particular, Neural Radiance Fields (NeRFs) [24] have attracted significant attention, spawning a wide range of follow-up works that improve upon various aspects of the original methodology.

Scale. Simply put, our work explores the scalability of NeRFs. The vast majority of existing methods explore single-object scenes, often captured indoors or from synthetic data. To our knowledge, Tanks and Temples [17] is the largest dataset used in NeRF evaluation, spanning 463 m^2 on average. In this work, we scale NeRFs to capture and interactively visualize urban-scale environments from drone footage that is orders of magnitude larger than any dataset to date, from 150,000 to over 1,300,000 m^2 per scene.

Search and Rescue. As a motivating use case, consider search-and-rescue, where drones provide an inexpensive means of quickly surveying an area and prioritizing limited first responder resources (e.g., for ground team deployment). Because battery life and bandwidth limits the ability to capture sufficiently detailed footage in real-time [6], collected footage is typically reconstructed into 2D "birds-eye-view" maps that support post-hoc analysis [42]. We imagine a future in which neural rendering lifts this analysis into 3D, enabling response teams to inspect the field as if they were flying a drone in real-time at a level of detail far beyond the

978-1-6654-6946-3/22/$31.00 ©2022 IEEE
DOI 10.1109/CVPR52688.2022.01258

	Resolution	# Images	# Pixels/Rays	Scene Captured / Image
Synthetic NeRF - Chair	400 x 400	400	256,000,000	0.271
Synthetic NeRF - Drums	400 x 400	400	256,000,000	0.302
Synthetic NeRF - Ficus	400 x 400	400	256,000,000	0.582
Synthetic NeRF - Hotdog	400 x 400	400	256,000,000	0.375
Synthetic NeRF - Lego	400 x 400	400	256,000,000	0.205
Synthetic NeRF - Materials	400 x 400	400	256,000,000	0.379
Synthetic NeRF - Mic	400 x 400	400	256,000,000	0.518
Synthetic NeRF - Ship	400 x 400	400	256,000,000	0.483
T&T - Barn	1920 x 1080	384	796,262,400	0.135
T&T - Caterpillar	1920 x 1080	368	763,084,800	0.216
T&T - Family	1920 x 1080	152	315,187,200	0.284
T&T - Ignatius	1920 x 1080	263	545,356,800	0.476
T&T - Truck	1920 x 1080	250	518,400,000	0.225
Mill 19 - Building	4608 x 3456	1940	30,894,981,120	0.062
Mill 19 - Rubble	4608 x 3456	1678	26,722,566,144	0.050
Quad 6k	1708 x 1329	5147	11,574,265,679	0.010
UrbanScene3D - Residence	5472 x 3648	2582	51,541,512,192	0.059
UrbanScene3D - Sci-Art	4864 x 3648	3019	53,568,749,568	0.088
UrbanScene3D - Campus	5472 x 3648	5871	117,196,056,576	0.028

Table 1. Scene properties from the commonly used Synthetic NeRF and Tanks and Temples datasets (T&T) compared to our target datasets (**below**). Our targets contain an order-of-magnitude more pixels (and hence rays) than prior work. Moreoever, each image captures significantly less of the scene, motivating a modular approach where spatially-localized submodules are trained with a fraction of relevant image data. We provide more details and additional statistics in Sec. H of the supplement.

achievable with classic Structure-from-Motion (SfM).

Challenges. Within this setting, we encounter multiple challenges. Firstly, applications such as search-and-rescue are time-sensitive. According to the National Search and Rescue Plan [1], "the life expectancy of an injured survivor decreases as much as 80 percent during the first 24 hours, while the chances of survival of uninjured survivors rapidly diminishes after the first 3 days." The ability to train a usable model within a few hours would therefore be highly valuable. Secondly, as our datasets are orders of magnitude larger than previously evaluated datasets (Table 1), model capacity must be significantly increased in order to ensure high visual fidelity, further increasing training time. Finally, although interactive rendering is important for fly-through and exploration at the scale we capture, existing real-time NeRF renderers either rely on pretabulating outputs into a finite-resolution structure, which scales poorly and significantly degrades rendering performance, or require excessive preprocessing time.

Mega-NeRF. In order to address these issues, we propose Mega-NeRF, a framework for training large-scale 3D scenes that support interactive human-in-the-loop fly-throughs. We begin by analyzing visibility statistics for large-scale scenes, as shown in Table 1. Because only a small fraction of the training images are visible from any particular scene point, we introduce a sparse network structure where parameters are specialized to different regions of the scene. We introduce a simple geometric clustering algorithm that partitions training images (or rather pixels) into different NeRF submodules that can be trained in parallel. We further exploit spatial locality at render time to implement a just-in-time visualization technique that allows for interactive fly-throughs of the captured environment.

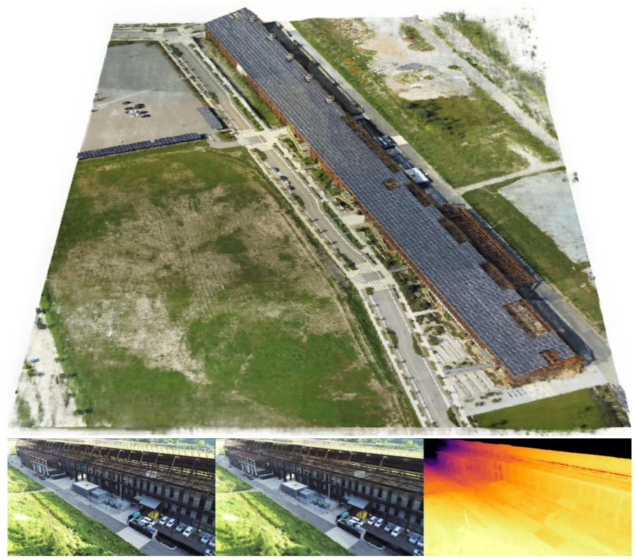


Figure 2. Visualization of Mill 19 by Mega-NeRF. The top panel shows a high-level 3D rendering of Mill 19 within our interactive visualizer. The bottom-left panel contains a ground truth image captured by our drone. The following two panels illustrate the model reconstruction along with the associated depth map.

Prior art. Our approach of using "multiple" NeRF submodules is closely inspired by the recent work of DeRF [28] and KiloNeRF [29], which use similar insights to accelerate *inference* (or rendering) of an existing, pre-trained NeRF. However, even obtaining a pre-trained NeRF for our scene scales is essentially impossible with current training pipelines. We demonstrate that modularity is vital for *training*, particularly when combined with an intelligent strategy for "sharding" training data into the appropriate modules via geometric clustering.

Contributions. We propose a reformulation of the NeRF architecture that sparsifies layer connections in a spatially-aware manner, facilitating efficiency improvements at training and rendering time. We then adapt the training process to exploit spatial locality and train the model subweights in a fully parallelizable manner, leading to a 3x improvement in training speed while exceeding the reconstruction quality of existing approaches. In conjunction, we evaluate existing fast rendering approaches against our trained Mega-NeRF model and present a novel method that exploits temporal coherence. Our technique requires minimal preprocessing, avoids the finite resolution shortfalls of other renderers, and maintains a high level of visual fidelity. We also present a new large-scale dataset containing thousands of HD images gathered from drone footage over 100,000 m^2 of terrain near an industrial complex.

2. Related work

Fast rendering. Conventional NeRF rendering falls well below interactive thresholds. Plenoctree [45], SNeRG [13], and FastNeRF [12] speed up the process by storing precomputed non-view dependent model outputs into a separate data structure such as a sparse voxel octree. These renderers then bypass the original model entirely at render time by computing the final view-dependent radiance through a separate smaller multi-layer perceptron (MLP) or through spherical basis computation. Although they achieve interactivity, they suffer from the finite capacity of the caching structure and poorly capture low-level details at scale.

DeRF [28] decomposes the scene into multiple cells via spatial Voronoi partitioning. Each cell is independently rendered using a smaller MLP, accelerating rendering by 3x over NeRF. KiloNeRF [29] divides the scene into thousands of even smaller networks. Although similar in spirit to Mega-NeRF, these methods use spatial partitioning to speed up *inference* while we use it to enable *data parallelism* for scalable training. Both DeRF and KiloNERF are initialized with a single large network trained on all data which is then distilled into smaller networks for fast inference, increasing processing time by over 2x for KiloNeRF. Training on all available data is prohibitive at our scale. Instead, our crucial insight is to geometrically partition training pixels into small data shards relevant for each submodule, which is essential for efficient training and high accuracy.

DONeRF [25] accelerates rendering by significantly reducing the number of samples queried per ray. To maintain quality, these samples are placed more closely around the first surface the ray intersects, similar to our guided sampling approach described in Sec. 3.3. In contrast to our method, DONeRF uses a separate depth oracle network trained against ground truth depth data.

Unbounded scenes. Although most NeRF-related work targets indoor areas, NeRF++ [48] handles unbounded environments by partitioning the space into a unit sphere foreground region that encloses all camera poses and a background region that covers the inverted sphere complement. A separate MLP model represents each area and performs ray casting independently before a final composition. Mega-NeRF employs a similar foreground/background partitioning although we further constrain our foreground and sampling bounds as described in Sec. 3.1.

NeRF in the Wild [21] augments NeRF's model with an additional transient head and learned per-image embeddings to better explain lighting differences and transient occlusions across images. Although it does not explicitly target unbounded scenes, it achieves impressive results against outdoor sequences in the Phototourism [15] dataset. We adopt similar appearance embeddings for Mega-NeRF and quantify its impact in Sec. 4.2.

Concurrent to us, Urban Radiance Fields [30] (URF), CityNeRF [43], and BlockNeRF [34] target urban-scale environments. URF makes use of lidar inputs, while CityNeRF makes use of multi-scale data modeling. Both methods can be seen as complementary to our approach, implying combining them with Mega-NeRF is promising. Most related to us is BlockNeRF [34], which decomposes a scene into spatial cells of fixed city blocks. Mega-NeRF makes use of geometry visibility reasoning to decompose the set of training pixels, allowing for pixels captured from far-away cameras to still influence a spatial cell (Fig. 1).

Training speed. Several works speed up model training by incorporating priors learned from similar datasets. PixelNeRF [46], IBRNet [40], and GRF [38] condition NeRF on predicted image features while Tancik et al. [35] use meta-learning to find good initial weight parameters that converge quickly. We view these efforts as complementary to ours.

Graphics. We note longstanding efforts within the graphics community covering interactive walkthroughs. Similar to our spatial partioning, Teller and Séquin [36] subdivide a scene into cells to filter out irrelevant geometry and speed up rendering. Funkhouser and Séquin [9] separately describe an adaptive display algorithm that iteratively adjusts image quality to achieve interactive frame rates within complex virtual environments. Our renderer takes inspiration from this gradual refinement approach.

Large-scale SfM. We take inspiration from previous large-scale reconstruction efforts based on classical Structure-from-Motion (SfM), in particular Agarwal et al's seminal "Building Rome in a Day," [3] which describes city-scale 3D reconstruction from internet-gathered data.

3. Approach

We first describe our model architecture in Sec. 3.1, then our training process in 3.2, and finally propose a novel renderer that exploits temporal coherence in 3.3.

3.1. Model Architecture

Background. We begin with a brief description of Neural Radiance Fields (NeRFs) [24]. NeRFs represent a scene within a continuous volumetric radiance field that captures both geometry and view-dependent appearance. NeRF encodes the scenes within the weights of a multilayer perceptron (MLP). At render time, NeRF projects a camera ray $\mathbf{r}$ for each image pixel and samples along the ray. For a given point sample p_i, NeRF queries the MLP at position $\mathbf{x}_i = (x, y, z)$ and ray viewing direction $\mathbf{d} = (d_1, d_2, d_3)$ to obtain opacity and color values σ_i and $\mathbf{c}_i = (r, g, b)$. It then composites a color prediction $\hat{C}(\mathbf{r})$ for the ray using numerical quadrature $\sum_{i=0}^{N-1} T_i(1 - \exp(-\sigma_i \delta_i))\, \mathbf{c}_i$, where $T_i = \exp(-\sum_{j=0}^{i-1} \sigma_j \delta_j)$ and δ_i is the distance between samples p_i and p_{i+1}. The training process optimizes the model by sampling batches R of image pixels and minimizing the loss function $\sum_{\mathbf{r}\in\mathcal{R}} \left\|C(\mathbf{r}) - \hat{C}(\mathbf{r})\right\|^2$. NeRF samples camera rays through a two-stage hierarchical sampling process and uses positional encoding to better capture high-frequency details. We refer the reader to the NeRF paper [24] for additional information.

Spatial partitioning. Mega-NeRF decomposes a scene into cells with centroids $\mathbf{n}_{\in\mathcal{N}} = (n_x, n_y, n_z)$ and initializes a corresponding set of model weights $f^{\mathbf{n}}$. Each weight submodule is a sequence of fully connected layers similar to the NeRF architecture. Similar to NeRF in the Wild [21], we associate an additional appearance embedding vector $l^{(a)}$ for each input image a used to compute radiance. This allows Mega-NeRF additional flexibility in explaining lighting differences across images which we found to be significant at the scale of the scenes that we cover. At query time, Mega-NeRF produces an opacity σ and color $\mathbf{c} = (r, g, b)$ for a given position $\mathbf{x}$, direction $\mathbf{d}$, and appearance embedding $l^{(a)}$ using the model weights $f^{\mathbf{n}}$ closest to the query point:

$$f^{\mathbf{n}}(\mathbf{x}) = \sigma \tag{1}$$

$$f^{\mathbf{n}}(\mathbf{x}, \mathbf{d}, l^{(a)}) = \mathbf{c} \tag{2}$$

$$\text{where } \mathbf{n} = \underset{n\in\mathcal{N}}{\text{argmin}} \left\|n - \mathbf{x}\right\|^2 \tag{3}$$

Centroid selection. Although we explored several methods, including k-means clustering and uncertainty-based partitioning as in [44], we ultimately found that tessellating the scene into a top-down 2D grid worked well in practice. This method is simple to implement, requires minimal preprocessing, and enables efficient assignment of point queries to centroids at inference time. As the variance in altitude between camera poses in our scenes is small relative to the differences in latitude and longitude, we fix the height of the centroids to the same value.

Foreground and background decomposition. Similar to NeRF++ [48], we further subdivide the scene into a foreground volume enclosing all camera poses and a background covering the complementary area. Both volumes are modeled with separate Mega-NeRFs. We use the same 4D outer volume parameterization and raycasting formulation as NeRF++ but improve upon its unit sphere partitioning by instead using an ellipsoid that more tightly encloses the camera poses and relevant foreground detail. We also take advantage of camera altitude measurements to further refine the sampling bounds of the scene by terminating rays near ground level. Mega-NeRF thus avoids needlessly querying underground regions and samples more efficiently. Fig. 3 illustrates the differences between both approaches.

3.2. Training

Spatial Data Parallelism. As each Mega-NeRF submodule is a self-contained MLP, we can train each in parallel with no inter-module communication. Crucially, as each image captures only a small part of the scene (Table 1), we limit the size of each submodule's trainset to only those potentially relevant pixels. Specifically, we sample points along the camera ray corresponding to each pixel for each training image, and add that pixel to the trainset for only those spatial cells it intersects (Fig. 1). In our experiments, this visibility partitioning reduces the size of each submodule's trainset by 10x compared to the initial aggregate trainset. This data reduction should be even more extreme for larger-scale scenes; when training a NeRF for North Pittsburgh, one need not add pixels of South Pittsburgh. We include a small overlap factor between cells (15% in our experiments) to further minimize visual artifacts near boundaries.

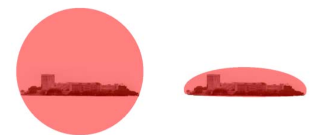


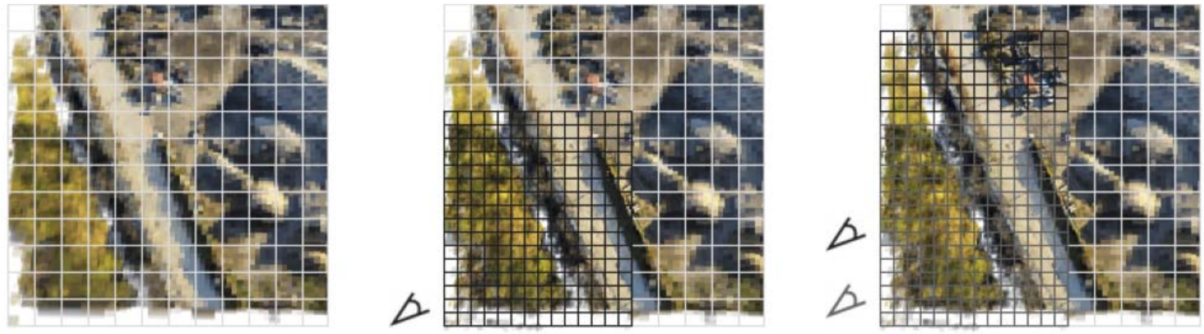
Figure 3. **Ray Bounds.** NeRF++ (**left**) samples within a unit sphere centered within and enclosing all camera poses to render its foreground component and uses a different methodology for the outer volume complement to efficiently render the background. Mega-NeRF (**right**) uses a similar background parameterization but models the foreground as an ellipsoid to achieve tighter bounds on the region of interest. It also uses camera altitude measurements to constrain ray sampling and not query underground regions.

Spatial Data Pruning. Note that the initial assignment of pixels to spatial cells is based on camera positions, irrespective of scene geometry (because that is not known at initialization). Once NeRF gains a coarse understanding of the scene, one could further prune away irrelevant pixels/rays that don't contribute to a particular NeRF due to an intervening occluder. For example, in Fig. 1, early NeRF optimization might infer a wall in cell F, implying that pixels from image 2 can then be pruned from cell A and B. Our initial exploration found that this additional visibility pruning further reduced trainset sizes by 2x. We provide details in Sec. A of the supplement.

3.3. Interactive Rendering

We propose a novel interactive rendering method in addition to an empirical evaluation of existing fast renderers on top of Mega-NeRF in Sec. 4.3. In order to satisfy our search-and-rescue usecase, we attempt to: (a) preserve visual fidelity, (b) minimize any additional processing time beyond training the base model, and (c) accelerate rendering, which takes over 2 minutes for a 720p frame with normal ray sampling, to something more manageable.

Caching. Most existing fast NeRF renderers make use of cached precomputation to speed up rendering, which may not be effective at our scene scale. For example, Plenoctree [45] precomputes a cache of opacity and spherical har-



(a) Fixed Octree (b) Dynamically Expanded Octree (c) Reused Octree (next frame)

Figure 4. **Mega-NeRF-Dynamic.** Current renderers (such as Plenoctree [45]) cache precomputed model outputs into a fixed octree, limiting the resolution of rendered images (**a**). Mega-NeRF-Dynamic *dynamically* expands the octree based on the current position of the fly-through (**b**). Because of the temporal coherence of camera views, the next-frame rendering (**c**) can reuse of much of expanded octree.

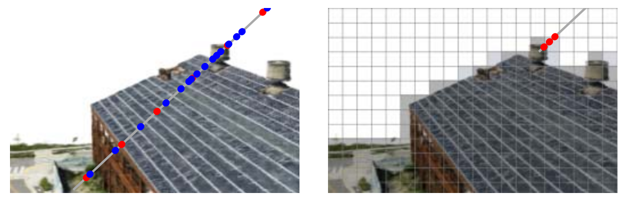
monic coefficients into a sparse voxel octree. Generating the entire 8-level octree for our scenes took an hour of computation and anywhere from 1 to 12 GB of memory depending on the radiance format. Adding a single additional level increased the processing time to 10 hours and the octree size to 55GB, beyond the capacity of all but the largest GPUs.

Temporal coherence. We explore an orthogonal direction that exploits the temporal coherence of interactive fly-throughs; once the information needed to render a given view is computed, we reuse much of it for the *next* view. Similar to Plenoctree, we begin by precomputing a coarse cache of opacity and color. In contrast to Plenoctree, we *dynamically* subdivide the tree throughout the interactive visualization. Fig. 4 illustrates our approach. As the camera traverses the scene, our renderer uses the cached outputs to quickly produce an initial view and then performs additional rounds of model sampling to further refine the image, storing these new values into the cache. As each subsequent frame has significant overlap with its predecessor, it benefits from the previous refinement and needs to only perform a small amount of incremental work to maintain quality. We provide further details in Sec. C of the supplement.

Guided sampling. We perform a final round of guided ray sampling after refining the octree to further improve rendering quality. We render rays in a single pass in contrast to NeRF's traditional two-stage hierachical sampling by using the weights stored in the octree structure. As our refined octree gives us a high-quality estimate of the scene geometry, we need to place only a small number of samples near surfaces of interest. Fig. 5 illustrates the difference between both approaches. Similar to other fast renderers, we further accelerate the process by accumulating transmittance along the ray and ending sampling after a certain threshold.

4. Experiments

Our evaluation of Mega-NeRF is motivated by the following two questions. First, given a finite training budget, how accurately can Mega-NeRF capture a scene? Furthermore, after training, is it possible to render accurately at scale while minimizing latency?



Standard Hierachical Sampling Guided Sampling

Figure 5. **Guided Sampling.** Standard NeRF **(left)** first samples coarsely at uniform intervals along the ray and subsequently performs another round of sampling guided by the coarse weights. Mega-NeRF-Dynamic **(right)** uses its caching structure to skip empty spaces and take a small number of samples near surfaces.

Qualitative results. We present two sets of qualitative results. Fig. 6 compares Mega-NeRF's reconstruction quality to existing view synthesis methods. In all cases Mega-NeRF captures a high level of detail while avoiding the numerous artifacts present in the other approaches. Fig. 7 then illustrates the quality of existing fast renderers and our method on top of the same base Mega-NeRF model. Our approach generates the highest quality reconstructions in almost all cases, avoiding the pixelization of voxel-based renderers and the blurriness of KiloNeRF.

4.1. Evaluation protocols

Datasets. We evaluate Mega-NeRF against multiple varied datasets. Our Mill 19 dataset consists of two scenes we recorded firsthand near a former industrial complex. Mill 19 - Building consists of footage captured in a grid pattern across a large $500 \times 250\ m^2$ area around an industrial building. Mill 19 - Rubble covers a nearby construction area full of debris in which we placed human mannequins masquerading as survivors. We also measure Mega-NeRF against two publicly available collections - the Quad 6k dataset [4], a large-scale Structure-from-Motion

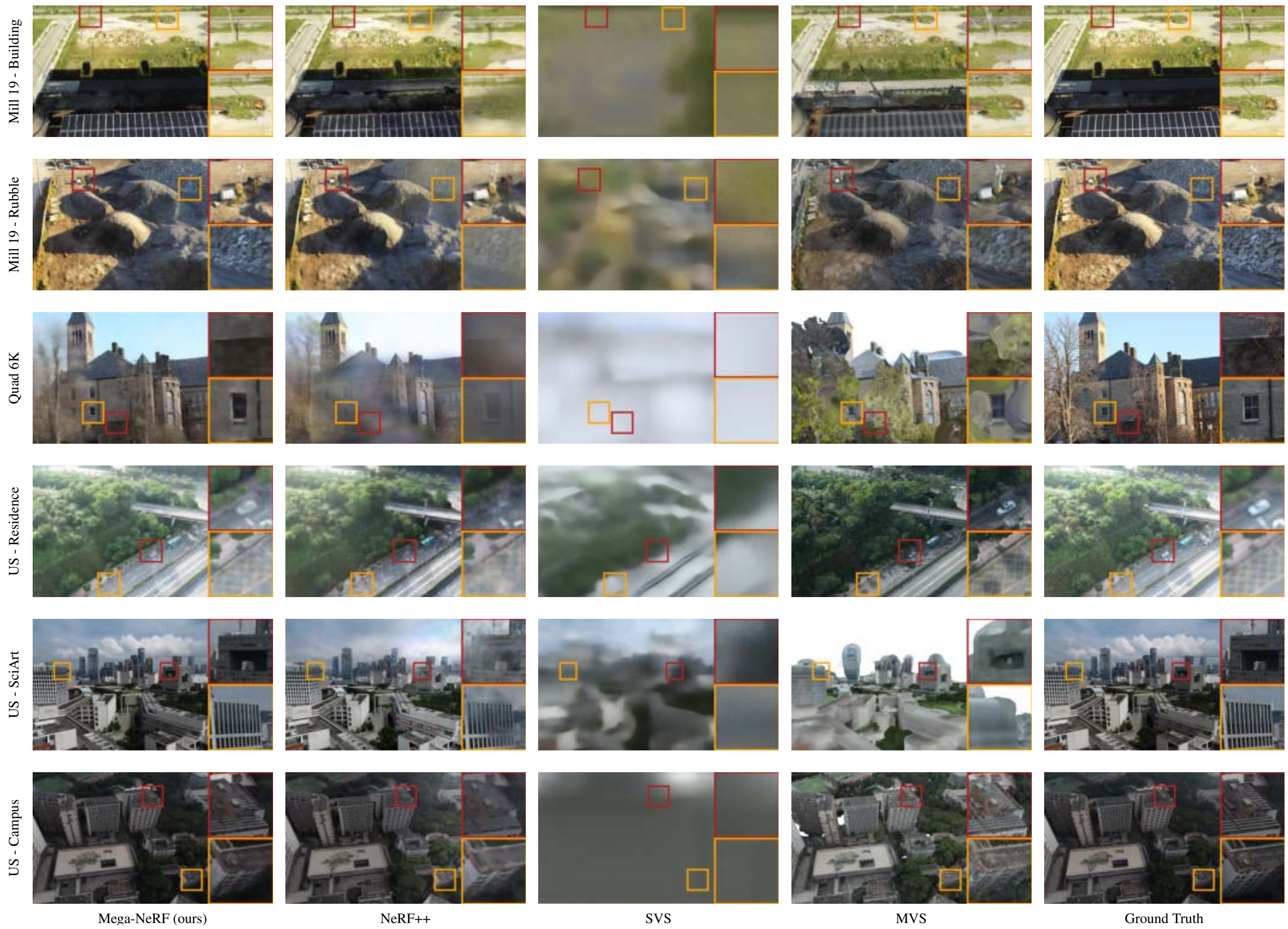


Figure 6. **Scalable training.** Mega-NeRF generates the best reconstructions while avoiding the artifacts present in the other approaches.

dataset collected within the Cornell Universty Arts Quad, and several scenes from UrbanScene3D [20] which contain high-resolution drone imagery of large-scale urban environments. We refine the initial GPS-derived camera poses in the Mill 19 and UrbanScene3D datasets and the estimates provided in the Quad 6k dataset using PixSFM [19]. We use a pretrained semantic segmentation model [7] to produce masks of common movable objects in the Quad 6k dataset and ignore masked pixels during training.

Training. We evaluate Mega-NeRF with 8 submodules each consisting of 8 layers of 256 hidden units and a final fully connected ReLU layer of 128 channels. We use hierarchical sampling during training with 256 coarse and 512 fine samples per ray in the foreground regions and 128/256 samples per ray in the background. In contrast to NeRF, we use the same MLP to query both coarse and fine samples which reduces our model size and allows us to reuse the coarse network outputs during the second rendering stage, saving 25% model queries per ray. We adopt mixed-precision training to further accelerate the process. We sample 1024 rays per batch and use the Adam optimizer [16] with an initial learning rate of 5×10^{-4} decaying exponentially to 5×10^{-5}. We employ the procedure described in [21] to finetune Mega-NeRF's appearance embeddings.

4.2. Scalable training

Baselines. We evaluate Mega-NeRF against the original NeRF [24] architecture and NeRF++ [48]. We also evaluate our approach against Stable View Synthesis [31], an implementation of DeepView [8], and dense reconstructions from COLMAP [33], a traditional Multi-View Stereo approach, as non-neural radiance field-based alternatives.

We use the same Pytorch-based framework and data loading infrastructure across all of NeRF variants to disentangle training speed from implementation specifics. We also use mixed precision training and the same number of samples per ray across all variants. We provide each implementation with the same amount of model capacity as Mega-NeRF by setting the MLP width to 2048 units. We provide additional details in Sec. D of the supplement.

Metrics. We report quantitative results based on PSNR, SSIM [41], and the VGG implementation of LPIPS [49].

	Mill 19 - Building				Mill 19 - Rubble				Quad 6k			
	↑PSNR	↑SSIM	↓LPIPS	↓Time (h)	↑PSNR	↑SSIM	↓LPIPS	↓Time(h)	↑PSNR	↑SSIM	↓LPIPS	↓Time(h)
NeRF	19.54	0.525	0.512	59:51	21.14	0.522	0.546	60:21	16.75	0.559	0.616	62:48
NeRF++	19.48	0.520	0.514	89:02	20.90	0.519	0.548	90:42	16.73	0.560	0.611	90:34
SVS	12.59	0.299	0.778	38:17	13.97	0.323	0.788	37:33	11.45	0.504	0.637	29:48
DeepView	13.28	0.295	0.751	31:20	14.47	0.310	0.734	32:11	11.34	0.471	0.708	19:51
MVS	16.45	0.451	0.545	32:29	18.59	0.478	0.532	31:42	11.81	0.425	**0.594**	**18:55**
Mega-NeRF	**20.93**	**0.547**	**0.504**	**29:49**	**24.06**	**0.553**	**0.516**	**30:48**	**18.13**	**0.568**	0.602	39:43
	UrbanScene3D - Residence				UrbanScene3D - Sci-Art				UrbanScene3D - Campus			
	↑PSNR	↑SSIM	↓LPIPS	↓Time (h)	↑PSNR	↑SSIM	↓LPIPS	↓Time(h)	↑PSNR	↑SSIM	↓LPIPS	↓Time(h)
NeRF	19.01	0.593	0.488	62:40	20.70	0.727	0.418	60:15	21.83	0.521	0.630	61:56
NeRF++	18.99	0.586	0.493	90:48	20.83	0.755	0.393	95:00	21.81	0.520	0.630	93:50
SVS	16.55	0.388	0.704	77:15	15.05	0.493	0.716	59:58	13.45	0.356	0.773	105:01
DeepView	13.07	0.313	0.767	30:30	12.22	0.454	0.831	31:29	13.77	0.351	0.764	33:08
MVS	17.18	0.532	**0.429**	69:07	14.38	0.499	0.672	73:24	16.51	0.382	**0.581**	96:01
Mega-NeRF	**22.08**	**0.628**	0.489	**27:20**	**25.60**	**0.770**	**0.390**	**27:39**	**23.42**	**0.537**	0.618	**29:03**

Table 2. **Scalable training.** We compare Mega-NeRF to NeRF, NeRF++, Stable View Synthesis (SVS), DeepView, and Multi-View Stereo (MVS) after running each method to completion. Mega-NeRF consistently outperforms the baselines even after allowing other approaches to train well beyond 24 hours.

We also report training times as measured on a single machine with 8 V100 GPUs.

Results. We run all methods to completion, training all NeRF-based methods for 500,000 iterations. We show results in Table 2 along with the time taken to finish training. Mega-NeRF outperforms the baselines even after training the other approaches for longer periods.

Diagnostics. We compare Mega-NeRF to several ablations. Mega-NeRF-no-embed removes the appearance embeddings from the model structure. Mega-NeRF-embed-only conversely adds Mega-NeRF's appearance embeddings to the base NeRF architecture. Mega-NeRF-no-bounds uses NeRF++'s unit sphere background/foreground partitioning instead of our formulation described in 3.1. Mega-NeRF-dense uses fully connected layers instead of spatially-aware sparse connections. Mega-NeRF-joint uses the same model structure as Mega-NeRF but trains all submodules jointly using the full dataset instead of using submodule-specific data partitions. We limit training to 24 hours for expediency.

We present our results in Table 4. Both the appearance embeddings and the foreground/background decomposition have a significant impact on model performance. Mega-NeRF also outperforms both Mega-NeRF-dense and Mega-NeRF-joint, although Mega-NeRF-dense comes close in several scenes. We however note that model sparsity accelerates rendering by 10x relative to fully-connected MLPs and is thus essential for acceptable performance.

4.3. Interactive exploration

Baselines. We evaluate two existing fast renderers, Plenoctree and KiloNeRF, in addition to our dynamic renderer. We base all renderers against the same Mega-NeRF model trained in 4.2 with the exception of the Plenoctree method which is trained on a variant using spherical harmonics. We accordingly label our rendering variants as Mega-NeRF-Plenoctree, Mega-NeRF-KiloNeRF, and Mega-NeRF-Dynamic respectively. We measure traditional NeRF rendering as an additional baseline, which we refer to as Mega-NeRF-Full, and Plenoxels [32] which generates a sparse voxel structure similar to Plenoctree but with trilinear instead of nearest-neighbor interpolation.

Metrics. We report the same perceptual metrics as in 4.2 and the time it takes to render a 720p image. We evaluate only foreground regions as Plenoctree and KiloNeRF assume bounded scenes. We also report any additional time needed to generate any additional data structures needed for rendering *beyond* the base model training time in the spirit of enabling fly-throughs within a day. As our renderer presents an initial coarse voxel-based estimate before progressively refining the image, we present an additional set of measurements, labeled as Mega-NeRF-Initial, to quantify the quality and latency of the initial reconstruction.

Results. We list our results in Table 3. Although Mega-NeRF-Plenoctree renders most quickly, voxelization has a large visual impact. Plenoxels provides better renderings but still suffers from the same finite resolution shortfalls and is blurry relative to the NeRF-based methods. Mega-NeRF-KiloNeRF comes close to interactivity at 1.1 FPS but still suffers from noticeable visual artifacts. Its knowledge distillation and finetuning processes also require over a day of additional processing. In contrast, Mega-NeRF-Dynamic remains within 0.8 db in PSNR of normal NeRF rendering while providing a 40x speedup. Mega-NeRF-Plenoctree and Mega-NeRF-Dynamic both take an hour to build similar octree structures.

5. Limitations

We discuss limitations and the societal impact of our work in the supplementary material.

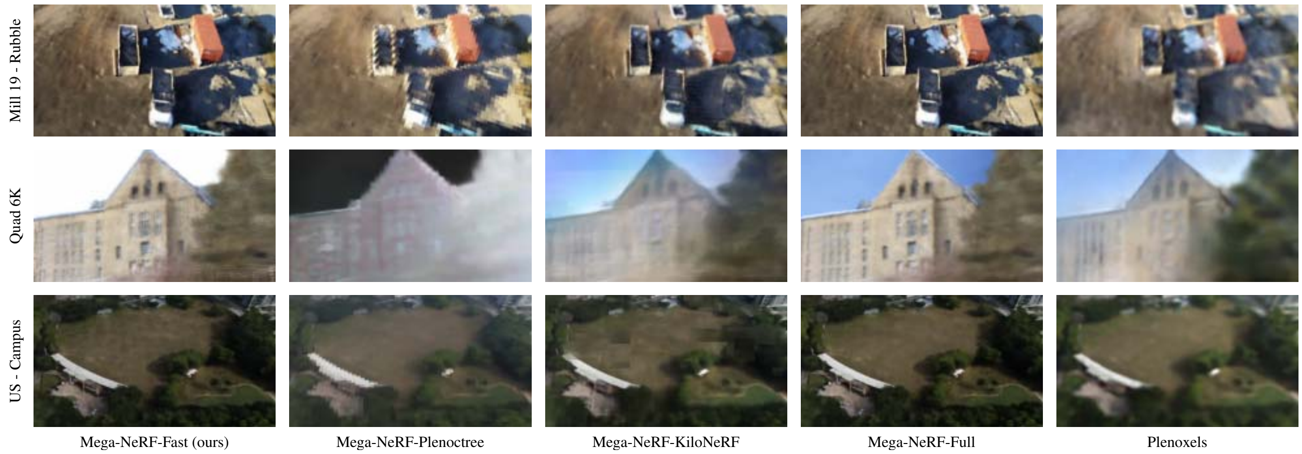


Figure 7. **Interactive rendering.** Plenoctree's approach causes significant voxelization and Plenoxel's renderings are blurry. KiloNeRF's results are crisper but capture less detail than Mega-NeRF-Dynamic and contain numerous visual artifacts.

best second-best	Mill 19					Quad 6k					UrbanScene3D				
	↑PSNR	↑SSIM	↓LPIPS	Preprocess Time (h)	Render Time (s)	↑PSNR	↑SSIM	↓LPIPS	Preprocess Time (h)	Render Time (s)	↑PSNR	↑SSIM	↓LPIPS	Preprocess Time (h)	Render Time (s)
Mega-NeRF-Plenoctree	16.27	0.430	0.621	1:26	**0.031**	13.88	0.589	0.427	1:33	**0.010**	16.41	0.498	0.530	**1:07**	**0.025**
Mega-NeRF-KiloNeRF	21.85	0.521	0.512	30:03	0.784	20.61	0.652	0.356	27:33	1.021	21.11	0.542	0.453	34:00	0.824
Mega-NeRF-Full	**22.96**	**0.588**	**0.452**	-	101	**21.52**	**0.676**	0.355	-	174	**24.92**	**0.710**	**0.393**	-	122
Plenoxels	19.32	0.476	0.592	-	0.482	18.61	0.645	0.411	-	0.194	20.06	0.608	0.503	-	0.531
Mega-NeRF-Initial	17.41	0.447	0.570	**1:08**	0.235	14.30	0.585	0.386	**1:31**	0.214	17.22	0.527	0.506	1:10	0.221
Mega-NeRF-Dynamic	22.34	0.573	0.464	**1:08**	3.96	20.84	0.658	**0.342**	**1:31**	2.91	23.99	0.691	0.408	1:10	3.219

Table 3. **Interactive rendering.** We evaluate two existing fast renderers on top of our base model, Mega-NeRF-Plenoctree and Mega-NeRF-KiloNeRF, relative to conventional rendering, labeled as Mega-NeRF-Full, Plenoxels, and our novel renderer (**below**). Although PlenOctree achieves a consistently high FPS, its reliance on a finite-resolution voxel structure causes performance to degrade significantly. Our approach remains within 0.8 db in PSNR quality while accelerating rendering by 40x relative to conventional ray sampling.

	Mill 19			Quad 6k			UrbanScene3D		
	↑PSNR	↑SSIM	↓LPIPS	↑PSNR	↑SSIM	↓LPIPS	↑PSNR	↑SSIM	↓LPIPS
Mega-NeRF-no-embed	20.42	0.500	0.561	16.16	0.544	0.643	19.45	0.587	0.545
Mega-NeRF-embed-only	21.48	0.494	0.566	17.91	0.559	0.638	22.79	0.611	0.537
Mega-NeRF-no-bounds	22.14	0.534	0.522	18.02	0.565	0.616	23.42	0.636	0.511
Mega-NeRF-dense	21.63	0.504	0.551	17.94	0.562	0.627	22.44	0.605	0.558
Mega-NeRF-joint	21.10	0.490	0.574	17.43	0.560	0.616	21.45	0.595	0.567
Mega-NeRF	**22.34**	**0.540**	**0.518**	**18.08**	**0.566**	**0.602**	**23.60**	**0.641**	**0.504**

Table 4. **Diagnostics.** We compare Mega-NeRF to various ablations after 24 hours of training. Each individual component contributes significantly to overall model performance.

6. Conclusion

We present a modular approach for building NeRFs at previously unexplored scale. We introduce a sparse and spatially aware network structure along with a simple geometric clustering algorithm that partitions training pixels into different NeRF submodules which can be trained in parallel. These modifications speed up training by over 3x while significantly improving reconstruction quality. Our empirical evaluation of existing fast renderers on top of Mega-NeRF suggests that interactive NeRF-based rendering at scale remains an open research question. We advocate leveraging temporal smoothness to minimize redundant computation between views as a valuable first step.

Acknowledgments

This research was supported by the National Science Foundation (NSF) under grant number CNS-2106862, the Defense Science and Technology Agency of Singapore (DSTA), and the CMU Argo AI Center for Autonomous Vehicle Research. Additional support was provided by CableLabs, Crown Castle, Deutsche Telekom, Intel, InterDigital, Microsoft, Seagate, VMware, Vodafone, and the Conklin Kistler family fund. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view(s) of their employers or the above funding sources.

References

[1] Gen 3.6 search and rescue. https://www.faa.gov/air_traffic/publications/atpubs/aip_html/part1_gen_section_3.6.html. Accessed: 2021-10-15. 2

[2] Pix4dmapper. https://www.pix4d.com/product/pix4dmapper-photogrammetry-software. Accessed: 2021-11-01. 11

[3] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M. Seitz, and Richard Szeliski. Building rome in a day. *Commun. ACM*, 54(10):105–112, oct 2011. 3

[4] David Crandall, Andrew Owens, Noah Snavely, and Daniel Huttenlocher. Discrete-continuous optimization for large-scale structure from motion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. 5, 13, 14

[5] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B. Tenenbaum, and Jiajun Wu. Neural radiance flow for 4d view synthesis and video processing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 12

[6] J. Eyerman, G. Crispino, A. Zamarro, and R Durscher. Drone efficacy study (DES): Evaluating the impact of drones for locating lost persons in search and rescue events. *Brussels, Belgium: DJI and European Emergency Number Association*, 2018. 1

[7] L-CCGP Florian and Schroff Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR). IEEE/CVF*, 2017. 6

[8] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2367–2376, 2019. 6

[9] Thomas A. Funkhouser and Carlo H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, page 247–254, New York, NY, USA, 1993. Association for Computing Machinery. 3

[10] Yasutaka Furukawa, Brian Curless, Steven M. Seitz, and Richard Szeliski. Towards internet-scale multi-view stereo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1434–1441, 2010. 11

[11] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021. 12

[12] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14346–14355, 2021. 3

[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. 3

[14] Yoonwoo Jeong, Seokjun Ahn, Christopher Choy, Animashree Anandkumar, Minsu Cho, and Jaesik Park. Self-calibrating neural radiance fields. In *ICCV*, 2021. 11

[15] Yuhe Jin, Dmytro Mishkin, Anastasiia Mishchuk, Jiri Matas, Pascal Fua, Kwang Moo Yi, and Eduard Trulls. Image matching across wide baselines: From paper to practice. *International Journal of Computer Vision*, 129(2):517–547, 2021. 3, 14

[16] DP Kingma, J Ba, Y Bengio, and Y LeCun. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, 2015. 6

[17] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4), 2017. 1, 14

[18] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *IEEE International Conference on Computer Vision (ICCV)*, 2021. 11

[19] Philipp Lindenberger, Paul-Edouard Sarlin, Viktor Larsson, and Marc Pollefeys. Pixel-Perfect Structure-from-Motion with Featuremetric Refinement. In *ICCV*, 2021. 6, 11

[20] Yilin Liu, Fuyou Xue, and Hui Huang. Urbanscene3d: A large scale urban scene dataset and simulator. *arXiv preprint arXiv:2107.04286*, 2021. 6, 13, 14

[21] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR*, 2021. 3, 4, 6, 14

[22] Quan Meng, Anpei Chen, Haimin Luo, Minye Wu, Hao Su, Lan Xu, Xuming He, and Jingyi Yu. GNeRF: GAN-based Neural Radiance Field without Posed Camera. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 11

[23] 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 Transactions on Graphics (TOG)*, 2019. 14

[24] 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, 4, 6, 14

[25] 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*, 40(4), 2021. 3

[26] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. *ICCV*, 2021. 12

[27] 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 H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 13

[28] D. Rebain, W. Jiang, S. Yazdani, K. Li, K. Yi, and A. Tagliasacchi. Derf: Decomposed radiance fields. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14148–14156, Los Alamitos, CA, USA, jun 2021. IEEE Computer Society. 2, 3

[29] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14335–14345, 2021. 2, 3

[30] Konstantinos Rematas, Andrew Liu, Pratul P. Srinivasan, Jonathan T. Barron, Andrea Tagliasacchi, Tom Funkhouser, and Vittorio Ferrari. Urban radiance fields. *CVPR*, 2022. 3

[31] Gernot Riegler and Vladlen Koltun. Stable view synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 6

[32] Sara Fridovich-Keil and Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022. 7

[33] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 6

[34] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul Srinivasan, Jonathan T. Barron, and Henrik Kretzschmar. Block-NeRF: Scalable large scene neural view synthesis. *arXiv*, 2022. 3

[35] Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P. Srinivasan, Jonathan T. Barron, and Ren Ng. Learned initializations for optimizing coordinate-based neural representations. In *CVPR*, 2021. 3

[36] Seth J. Teller and Carlo H. Séquin. Visibility preprocessing for interactive walkthroughs. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '91, page 61–70, New York, NY, USA, 1991. Association for Computing Machinery. 3

[37] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2021. 12

[38] Alex Trevithick and Bo Yang. Grf: Learning a general radiance field for 3d representation and rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15182–15192, 2021. 3, 13

[39] Junjue Wang, Brandon Amos, Anupam Das, Padmanabhan Pillai, Norman Sadeh, and Mahadev Satyanarayanan. A scalable and privacy-aware iot service for live video analytics. In *Proceedings of the 8th ACM on Multimedia Systems Conference*, MMSys'17, page 38–49, New York, NY, USA, 2017. Association for Computing Machinery. 13

[40] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *CVPR*, 2021. 3

[41] 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. 6

[42] William T. Weldon and Joseph Hupy. Investigating methods for integrating unmanned aerial systems in search and rescue operations. *Drones*, 4(3), 2020. 1

[43] Yuanbo Xiangli, Linning Xu, Xingang Pan, Nanxuan Zhao, Anyi Rao, Christian Theobalt, Bo Dai, and Dahua Lin. Citynerf: Building nerf at city scale. *arXiv preprint arXiv:2112.05504*, 2021. 3

[44] Guo-Wei Yang, Wen-Yang Zhou, Hao-Yang Peng, Dun Liang, Tai-Jiang Mu, and Shi-Min Hu. Recursive-NeRF: An efficient and dynamically growing nerf. *arXiv preprint arXiv:2105.09103*, 2021. 4, 13

[45] 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. 3, 4, 5, 14

[46] A. Yu, V. Ye, M. Tancik, and A. Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4576–4585, Los Alamitos, CA, USA, jun 2021. IEEE Computer Society. 3, 13

[47] Kaan Yücer, Alexander Sorkine-Hornung, Oliver Wang, and Olga Sorkine-Hornung. Efficient 3D object segmentation from densely sampled light fields with applications to 3D reconstruction. *ACM Transactions on Graphics*, 35(3), 2016. 14

[48] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv:2010.07492*, 2020. 3, 4, 6, 14

[49] 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. 6