Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields Supplemental Materials

Jonathan T. Barron[1] Ben Mildenhall[1] Dor Verbin[1,2]
Pratul P. Srinivasan[1] Peter Hedman[1]
[1]Google [2]Harvard University

1. Additional Model Details

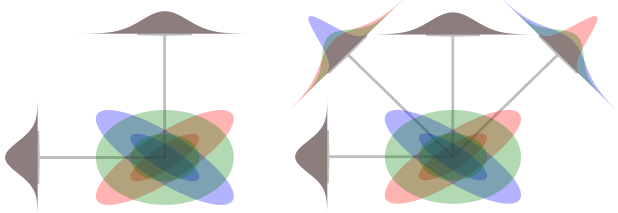
Our model contains some small components not discussed in the main paper that improve performance slightly.

Off-Axis Positional Encoding. When constructing integrated positional encoding features, we must select a basis $\mathbf{P}$. In mip-NeRF [1], this basis is selected as the identity matrix. This is convenient, because it means that only the diagonal of the covariance matrix $\mathbf{\Sigma}$ is required to construct IPE features, and off-diagonal components need not be computed. However, the reparameterization used by our model requires access to a full covariance matrix, as otherwise the Kalman-like warping we use would be inaccurate in the presence of highly anisotropic Gaussians (which are frequent in distant parts of the scene). So given that we are required to construct a full $\mathbf{\Sigma}$ matrix, we take advantage of the extra information presented therein, and encode not just axis-aligned IPE features but off-axis IPE features as well. As our basis $\mathbf{P}$, instead of an identity matrix we use a large skinny matrix that contains the unit-norm vertices of a twice-tessellated icosahedron, where redundant negative copies of vertices are removed. For reproducibility's sake this matrix is:

$$\mathbf{P} = \begin{bmatrix} 0.8506508 & 0 & 0.5257311 \\ 0.809017 & 0.5 & 0.309017 \\ 0.5257311 & 0.8506508 & 0 \\ 1 & 0 & 0 \\ 0.809017 & 0.5 & -0.309017 \\ 0.8506508 & 0 & -0.5257311 \\ 0.309017 & 0.809017 & -0.5 \\ 0 & 0.5257311 & -0.8506508 \\ 0.5 & 0.309017 & -0.809017 \\ 0 & 1 & 0 \\ -0.5257311 & 0.8506508 & 0 \\ -0.309017 & 0.809017 & -0.5 \\ 0 & 0.5257311 & 0.8506508 \\ -0.309017 & 0.809017 & 0.5 \\ 0.309017 & 0.809017 & 0.5 \\ 0.5 & 0.309017 & 0.809017 \\ 0.5 & -0.309017 & 0.809017 \\ 0 & 0 & 1 \\ -0.5 & 0.309017 & 0.809017 \\ -0.809017 & 0.5 & 0.309017 \\ -0.809017 & 0.5 & -0.309017 \end{bmatrix}^{\mathrm{T}}. \quad (1)$$

These off-axis features allow the model to encode the shape of anisotropic Gaussians (with a similar intuition as the random Fourier features explored by Tancik *et al*. [17]) which otherwise are indistinguishable using axis-aligned IPE features, as shown in Figure 1. Ablating these off-axis features reduces performance slightly, with SSIM for the *bicycle* scene falling from 0.687 to 0.664.

Computing IPE features with a large $\mathbf{P}$ matrix using the procedure described in mip-NeRF ($\mathrm{diag}(\mathbf{P}\mathbf{\Sigma}\mathbf{P}^{\mathrm{T}})$) is prohibitively expensive. A tractable alternative is to instead compute the equivalent expression $\mathrm{sum}(\mathbf{P}^{\mathrm{T}} \circ (\mathbf{\Sigma}\mathbf{P}^{\mathrm{T}}), 0)$ where $\circ$ is an element-wise product and $\mathrm{sum}(\cdot, 0)$ is summation over rows. With this small optimization, off-axis IPE features are only modestly more expensive to compute than the axis-aligned IPE features used in mip-NeRF.



(a) Axis-Aligned IPE [1] (b) Off-Axis IPE

Figure 1. The axis-aligned positional encoding used by mip-NeRF [1] does not capture the covariance of the Gaussian being encoded. Here we plot three bivariate Gaussians colored red, green, and blue (a) with axis-aligned IPE and (b) with our off-axis IPE, and show the marginal distributions produced by projecting each Gaussians onto the basis used by each encoding. Because these Gaussians have identical marginal distributions, mip-NeRF's axis-aligned IPE produces identical features, while the off-axis projections of our approach allow them to be disambiguated.

Annealing. Before resampling ray-intervals from proposal weights $\hat{\mathbf{w}}$, we anneal those weights by raising them to a power. With N training steps, at step n we compute

$$\hat{\mathbf{w}}_n \propto \hat{\mathbf{w}}^{\frac{bn/N}{(b-1)n/N+1}} \tag{2}$$

and use $\hat{\mathbf{w}}_n$ when drawing samples. The exponent is Schlick's bias function [15] applied to ${}^{n}/_{N} \in [0, 1]$, which curves the exponent such that it quickly rises from 0 and saturates towards 1. We set the bias hyperparameter $b = 10$ in all experiments. At the beginning of training the exponent is 0, which yields a flat distribution ($\hat{\mathbf{w}}_0 \propto \mathbf{1}$), and at the end of training that power is 1, which yields the proposal distribution ($\hat{\mathbf{w}}_N = \hat{\mathbf{w}}$). This annealing encourages "exploration" during training, by causing the NeRF MLP to be presented with a wider range of proposal intervals than it otherwise would towards the beginning of training. Annealing has a modest positive effect: ablating it causes SSIM on the *bicycle* scene to decrease from 0.687 to 0.679.

Dilation. We slightly "dilate" each proposal histogram $(\hat{\mathbf{t}}, \hat{\mathbf{w}})$ before resampling it. This reduces aliasing artifacts, likely because the proposal MLP is supervised using only rays that correspond to input pixels, so its predictions may only hold for certain angles — in a sense, the proposal network is rotationally aliased. By widening the intervals of the proposal MLP we help counteract this aliasing. To dilate a histogram $(\hat{\mathbf{s}}, \hat{\mathbf{w}})$ we first compute $\hat{\mathbf{p}}$ where $\hat{p}_i = \hat{w}_i / (\hat{s}_{i+1} - \hat{s}_i)$, giving us a probability density that integrates to 1 rather than a histogram that sums to 1. We then dilate this by computing

$$\max_{s-\epsilon \leq s' < s+\epsilon} \hat{\mathbf{p}}_{\hat{\mathbf{s}}}(s') \tag{3}$$

where $\hat{\mathbf{p}}_{\hat{\mathbf{s}}}(s)$ is interpolation into the step function defined by $\hat{\mathbf{s}}, \hat{\mathbf{p}}$ at s. Equation 3 can be computed efficiently by constructing a new set of intervals whose endpoints are $\text{sort}(\hat{\mathbf{s}} \cup \hat{\mathbf{s}} - \epsilon \cup \hat{\mathbf{s}} + \epsilon\})$ and computing the max of all intervals in that expanded set. After dilation, we convert the dilated $\hat{\mathbf{p}}$ back into a histogram by multiplying each element by the size of its interval, and then normalizing the resulting histogram to sum to 1.

When dilating each proposal histogram, we set the dilation factor ϵ to a function of the expected size of a histogram bin. That is, for each coarse-to-fine resampling level k in which we resample n_k fine intervals from the coarse histogram that preceded it, we compute the product of all sample counts that preceded level k and set epsilon to an affine function of the inverse of that product:

$$\epsilon_k = \frac{a}{\prod_{k'=1}^{k-1} n_k} + b \tag{4}$$

where $a = 0.5$ and $b = 0.0025$ are hyperparameters that respectively determine how much a histogram is dilated relative to the expected size of a histogram bin (which is the inverse of the product of prior sample counts) and in absolute terms.

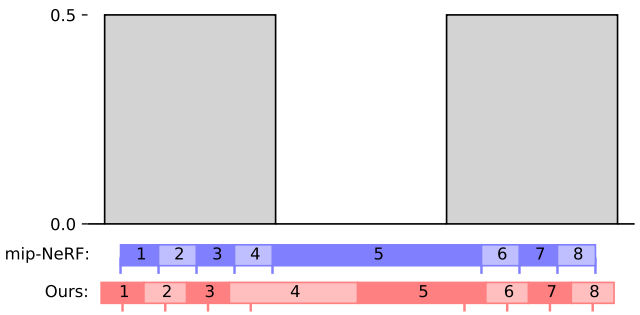
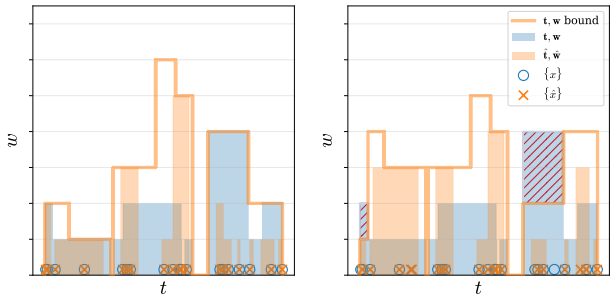


Figure 2. When resampling a histogram (such as the toy 3-bin histogram shown here in gray) into a set of n intervals, mip-NeRF [1] samples $n+1$ sorted random values (blue ticks) from the histogram for use as the endpoints of intervals (blue boxes), which yields intervals that do not cover the front and back of histogram modes, and asymmetrically span gaps between modes. We instead draw $n + 1$ sorted random values (red ticks) and use the midpoints of those samples as the endpoints of intervals (red boxes), resulting in less irregular resampling.

Sampling. Mip-NeRF generated n "fine" intervals along each ray by sampling $n + 1$ distances from the "coarse" histogram weights $\hat{\mathbf{t}}, \hat{\mathbf{w}}$ and using those sorted sampled distances as the endpoints of a set of n intervals. This approach of using sampled points for use as interval endpoints can produce unusual results, as evidenced by Figure 2 — it effectively "erodes" the coarse histogram, as the samples are unlikely to span the extent of each coarse histogram bin. For this reason we use a slightly modified resampling procedure: We sample n sorted values from the coarse histogram, and then use the midpoints of each adjacent sample pair as the endpoints of our new set of "fine" intervals (and we reflect the first and last sample around the first and last endpoint to deal with boundary conditions). This change has little quantitative effect on rendering quality, but we found that it qualitatively reduced aliasing.

Background Colors. NeRF and mip-NeRF assume a known background color, which is usually set to black or white. This often results in scene reconstructions in which the background is incorrectly represented as semi-transparent instead of opaque. These semi-transparent backgrounds may still allow for realistic view synthesis, but they tend to yield less meaningful mean or median ray termination distances, which results in less accurate depth maps. For this reason, when compositing a pixel color during training we draw a random RGB background color from $[0, 1]^3$ which encourages training to reconstruct a fully-



(a) $\mathcal{L}_{\text{prop}}(\mathbf{t}, \mathbf{w}, \hat{\mathbf{t}}, \hat{\mathbf{w}}) = 0$ (b) $\mathcal{L}_{\text{prop}}(\mathbf{t}, \mathbf{w}, \hat{\mathbf{t}}, \hat{\mathbf{w}}) = 0.75$

Figure 3. A visualization of the motivation behind $\mathcal{L}_{\text{prop}}$, the loss used to train our proposal MLP to bound the weights emitted by our NeRF MLP. In both plots we have two different histograms $(\mathbf{t}, \mathbf{w})$ (shown in orange) and $(\hat{\mathbf{t}}, \hat{\mathbf{w}})$ (shown in blue) generated from points $\{x\}$ and $\{\hat{x}\}$ respectively, as well as a plot of the bound described in the paper. (a) If $\{x\} = \{\hat{x}\}$, the bound implied by $(\hat{\mathbf{t}}, \hat{\mathbf{w}})$ is guaranteed to be an upper bound on $(\mathbf{t}, \mathbf{w})$, and our loss must be zero. (b) If $\{x\} \neq \{\hat{x}\}$ (in this case, only 16 of 20 points are shared between $\{x\}$ and $\{\hat{x}\}$) then $(\mathbf{t}, \mathbf{w})$ may exceed the upper bound implied by $(\hat{\mathbf{t}}, \hat{\mathbf{w}})$, and a loss may be incurred (shown in red). From this we see how minimizing $\mathcal{L}_{\text{prop}}$ encourages the proposal weights $(\hat{\mathbf{t}}, \hat{\mathbf{w}})$ to describe the same distribution as the NeRF weights $(\mathbf{t}, \mathbf{w})$, despite their histogram bin endpoints being different.

opaque background of the scene (at test-time we set the background color to $(0.5, 0.5, 0.5)$). We use randomized backgrounds for our 360 dataset and the LLFF dataset, but for the Blender dataset we use the same fixed/known white background as in prior work.

2. Implementation Details

The Jacobian $\mathbf{J}_f(\boldsymbol{\mu})$ used by our Kalman-like reparameterization can be computed straightforwardly using most autodiff frameworks. A less expensive alternative (as it does not require the explicit construction of a Jacobian matrix) is to instead construct and apply a function whose application corresponds to matrix multiplication with $\mathbf{J}_f(\boldsymbol{\mu})$. In Jax [2], this can be accomplished using the `linearize` operator, and applying it twice in sequence to $\boldsymbol{\Sigma}$, with the dimensions of the covariance matrix transposed after each application.

3. Proposal Supervision Visualization

The loss used to supervise our proposal MLP is motivated by bounds that can be established between histograms of 1D data. The bound used by our loss is guaranteed to hold if two histograms are constructed from the same underlying "true" distribution of data. By minimizing any excess histogram mass that violates this bound, we can encourage two histograms with differently-spaced bin locations to be consistent with each other. In Figure 3 we provide an illustration of this concept, and the supplemental video contains additional explanatory illustrations.

4. Additional Results

Our Dataset. We captured our dataset using two different mirrorless digital cameras. The outdoor scenes were captured with a Sony NEX C-3 equipped with a 18-55mm lens, using the widest possible zoom level. For the indoor scenes, we used a Fujifilm X100V camera with a fixed 22mm lens. For each scene, we used the first camera location as a reference view, where we configured ISO, white balance, shutter speed, aperture size, and focus. We then kept these settings locked during capture, to limit the photometric variation between images of the same scene. To further limit color harmonization issues, we captured the outdoor scene when the sky was overcast, making sure that the camera operator cast soft shadows that minimally affected the illumination in the scene. For the indoor scenes, we relied on large diffuse light sources (e.g. daylight reflecting off white walls) and avoided casting shadows onto the scene.

We captured between 100 and 330 images in each scene. This took between 1 and 20 minutes, depending on whether we used burst mode or not. To obtain camera poses, we use the publicly available COLMAP software [16]. We use shared intrinsics between all images in a scene, and calibrate using the OpenCV radial distortion model. Before training a NeRF, we use COLMAP to undistort the images, and downsample them to a resolution of 1.0 – 1.6 megapixels using ImageMagick. We use 1 in 8 of the input images as our test set, regularly subsampled to cover as many viewpoints as possible.

Post-capture, we apply a rigid transform and rescaling to COLMAP's reconstructed poses in order to better fit the captured scene content to our parameterization. In order to match the global coordinate frame to the capture pattern (assumed to be approximately circular rings orbiting a fixed point in space), we subtract the mean camera position and calculate the principal components of the recentered camera position vectors. We then use these three orthogonal vectors to form a new basis where the smallest principal component becomes the world-space "up" vector. After recentering all camera poses using this transformation, we rescale the camera positions such that they lie within the $[-1, 1]^3$ cube. If the input poses lie approximately on a sphere, this usually causes them to lie within the uniformly parameterized region of space contained by the sphere of radius 1.

In Table 3 we show an expanded table of results for our dataset where we enumerate PSNRs, SSIMs, and LPIPS scores for each individual scene. Each technique's per-scene performance is roughly consistent with its average performance as reported in the main paper.

As discussed in the paper, Stable View Synthesis [14] is the only baseline model we evaluate against that outperforms our model on any metric, which is LPIPS [19]. Upon

visually inspecting the results of SVS on our dataset, we observed that LPIPS is often dramatically inconsistent with our own visual perception. See Figure 6, where we visualize the renderings (and depths) of our model versus SVS on one scene where SVS yielded a lower LPIPS metric than our model. Contrary to what the LPIPS scores indicate, our model's rendering is significantly more realistic and exhibits significantly fewer artifacts than SVS, particularly in the background of the scene. We believe this is due to SVS having been trained to minimize a perceptual loss that resembles LPIPS, causing it to produce results that are able to minimize LPIPS effectively despite being visually unsatisfying. This is consistent with recent work that has demonstrated vulnerabilities in LPIPS [7].

NeRF's Blender Dataset. For completeness, in Table 1 we evaluate our model on the Blender dataset from Mildenhall *et al*. [12], for which mip-NeRF is the current state-of-the-art. This dataset consists entirely of small synthetic objects in front of a white background, unlike the large and unbounded scenes which motivated our model's design. Because this task is easier than our 360 scenes, we use a simplified version of our model for the sake of speed: only one round of sampling, 128 samples for the proposal MLP, 32 samples for the NeRF MLP, a proposal MLP with 4 layers and 256 hidden units, a NeRF MLP with 8 layers and 256 (or 512) hidden units, axis-aligned IPE, MSE loss, and no distortion regularizer. Our model is not designed to improve accuracy on these scenes, and as such our model's accuracy is comparable to mip-NeRF across all error metrics. However, we see that (due to our use of proposal networks) our model is significantly faster to train than mip-NeRF, and that this relative speedup increases as model capacity rises.

	# hidden	PSNR ↑	SSIM ↑	LPIPS ↓	Time (hrs)	# Params
mip-NeRF [1]	256	33.09	0.961	0.043	2.89	0.61M
Our Model		32.96	0.960	0.043	1.86	0.84M
mip-NeRF [1]	512	33.03	0.964	0.037	7.03	2.27M
Our Model		33.25	0.962	0.039	3.42	3.23M

Table 1. Performance on the Blender dataset used in NeRF [12] as we vary the number of hidden units in the NeRF MLP.

The LLFF Dataset. In Table 2 we evaluate against mip-NeRF on the "front-facing" scenes presented in the LLFF paper [11]. We use the same model as in our Blender evaluation, except we do not disable our distortion regularizer and we use Charbonnier loss instead of MSE. Our model does not outperform mip-NeRF in terms of PSNR, but yields improved SSIM and LPIPs metrics and a significant speedup when the NeRF MLP is large.

The Tanks and Temples Dataset. The "Tanks and Temples" dataset is a popular dataset for 3D geometry and view synthesis tasks [8]. It contains several scenes with a large central object with the camera moving around that object. At first glance this dataset may appear to be ideal for our purposes, but it has significant issues that motivated the construction of our own dataset. As shown in Figure 4, the photometric properties of the camera are not constant across each scene capture. We believe this is due to the camera's autoexposure or auto white balance being allowed to vary between images. Additionally, many images are overexposed, resulting in "clipped" RGB values (also visible in Figure 4). These issues make evaluation difficult, as measuring the accuracy of a view synthesis algorithm becomes an ill-posed task when faced with photometric variation — which photometric condition should the model attempt to replicate? This challenge posed by "in the wild" images has been investigated by Martin-Brualla *et al*. [10] who constructed specialized training and evaluation procedures for dealing with it. But we view this challenge as orthogonal to the challenges posed by the unbounded nature of a scene, hence the construction of our own dataset where our camera is photometrically fixed within each capture, and where scenes are chosen to minimize saturated pixels.

	# hidden	PSNR ↑	SSIM ↑	LPIPS ↓	Time (hrs)	# Params
mip-NeRF [1]	256	26.93	0.830	0.177	2.48	0.61M
Our Model		26.68	0.847	0.150	2.39	0.84M
mip-NeRF [1]	512	27.01	0.845	0.148	6.15	2.27M
Our Model		26.86	0.858	0.128	3.84	3.23M

Table 2. Performance on the dataset presented with LLFF [11] as we vary the number of hidden units in the NeRF MLP.

Despite the mismatch between this dataset and the goals of our work, we evaluated our model on this dataset against our NeRF-like baselines and against SVS (which is both the state of the art for this dataset as well as the most competitive baseline for our own dataset), the results of which are

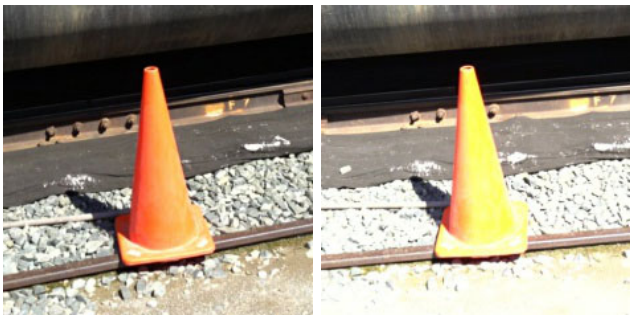


Figure 4. Crops from two images taken from the "Tanks and Temples" dataset. The capture process used in acquiring this dataset seems to have allowed autoexposure and/or auto white balance to vary across images, which results in the same object having a different appearance across scenes. This issue partially motivated the construction of our own dataset, in which great care is taken to prevent such photometric variation.

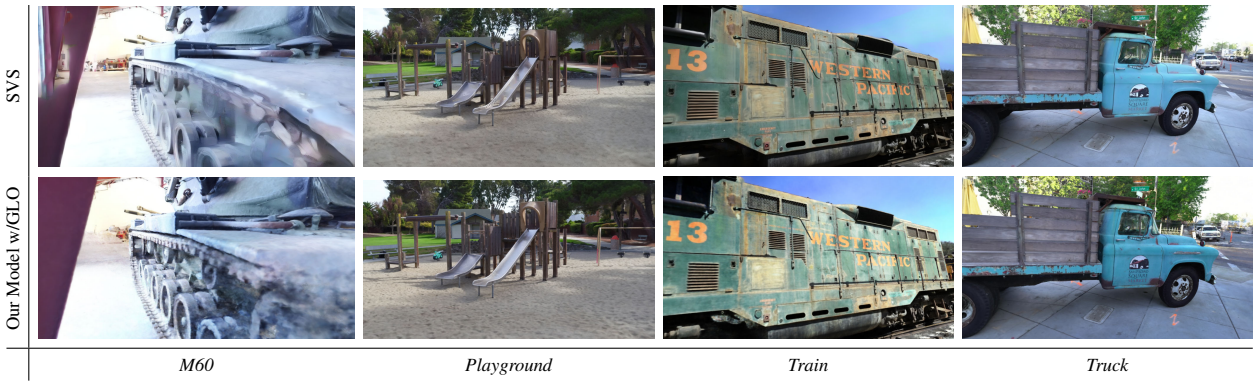


Figure 5. A visualization of our model with Stable View Synthesis [14] on scenes from the Tanks and Temples dataset [8]. Image quality is roughly comparable across the two techniques, though our renderings exhibits different failure modes than SVS's in the absence of observations (as in *M60*) and, because our model neutralizes most photometric variation during training, our renderings may have a different global brightness or color shift (as in *Train*).

shown in Tables 4 and 5, and visualized in Figure 5. The metrics used elsewhere in this paper (PSNR, SSIM, and LPIPS) are difficult to draw meaningful conclusions from due to the aforementioned photometric variation. In particular, our top-performing "w/GLO" model variant performs quite poorly according to those metrics, because that model variant learns a per-image embedding for each scene and uses that embedding within the NeRF MLP when predicting color. When we evaluate this model variant at test-time, we set the embedding vector to $\mathbf{0}$. This gives us a pleasing looking reconstruction that roughly corresponds to the photometric average of all input cameras, and that is consistent across all images, which we believe to be a good goal for view synthesis. However, SVS (and to a lesser extent, the non-GLO NeRF baselines) do not behave this way, and instead attempt to "explain away" photometric variation due to the camera by modifying the brightness and color of the scene as a function of viewing direction. Effectively, SVS does not attempt to synthesize a view, it attempts to synthesize a view *and* the most likely camera settings for that view. This motivated us to construct "color corrected" error metrics: before evaluating each metric we solve a per-image least squares problem that fits a quadratic polynomial expansion of the rendering's RGB values to the true image, while ignoring saturated pixels. This partially reduces the effect of photometric variation on this data, and yields results in which SVS and our model (with GLO) are roughly quantitatively comparable.

When using these color-corrected metrics, our model slightly outperforms SVS in terms of PSNR, but underperforms SVS on SSIM and LPIPS. With this in mind, it is worth reiterating the advantages that SVS has over our model on this benchmark: 1) SVS has been trained on the training set of this dataset, while our model does not use that external training data — and indeed uses no external training data at all. 2) SVS relies on a proxy geometry produced by an external system (and may fail when that geometry is incorrect), while we use no proxy geometry and in fact produce high-quality depth maps ourselves. 3) SVS has been trained with a perceptual loss, while our model is trained using only a per-pixel loss on RGB. 4) Our model is extremely compact, and requires only 10 million parameters to perform view synthesis, while SVS requires multiple large CNNs and access to all training images (because it operates by blending training images together) to render novel views.

From Table 5 we see that our model's improvement over SVS is most significant on the *playground* scene. Notably, this is the only test-set scene that mostly consists of natural content, while the other three scenes predominately feature large vehicles. We speculate that SVS may be better-suited to large piecewise planar objects (which makes sense, given SVS's reliance on a proxy geometry that is itself a piecewise planar mesh) while ours may be better suited to scenes that contain natural content (trees, grass, flowers, etc).

5. Potential Negative Impact

The broad use of neural rendering techniques carries with it several potential negative societal impacts. NeRF-like models have recently been incorporated into generative modeling approaches [4], and generative modeling techniques can be used to synthesize "deep fakes" that could be used to mislead people. Though our work does not directly concern generative modeling and instead aims to reconstruct accurate physical models of a scene from which new views can be generated, our contributions may be useful for generative approaches that build on NeRF.

The ability to reconstruct accurate models of a scene from photographs may have modest potential negative impacts. Our technique could conceivably be used to construct

a surveillance system, and such a system could have negative impact if used negligently or maliciously. Additionally, our system could be used to generate visual effects (a task that is currently labor intensive) and as such it may negatively affect job opportunities for artists.

Training a NeRF is computationally demanding, and requires multiple hours of optimization on an accelerator (though test-time rendering can be accelerated significantly [6]). This expensive training requires energy, and this may be of concern if that energy was produced in a way that damages the climate.

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. *ICCV*, 2021. 1, 2, 4, 8, 9

[2] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. `http://github.com/google/jax`. 3

[3] Boyang Deng, Jonathan T. Barron, and Pratul P. Srinivasan. JaxNeRF: an efficient JAX implementation of NeRF, 2020. `http://github.com/google-research/google-research/tree/master/jaxnerf`. 8, 9

[4] Jiatao Gu, Lingjie Liu, Peng Wang, and Christian Theobalt. Stylenerf: A style-based 3d-aware generator for high-resolution image synthesis, 2021. 5

[5] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *SIGGRAPH Asia*, 2018. 8

[6] Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. *ICCV*, 2021. 6

[7] Markus Kettunen, Erik Härkönen, and Jaakko Lehtinen. E-lpips: robust perceptual image similarity via random transformation ensembles. *arXiv:1906.03973*, 2019. 4

[8] 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. 4, 5, 9

[9] Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. Point-based neural rendering with per-view optimization. *Computer Graphics Forum*, 2021. 8

[10] 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. *CVPR*, 2021. 4

[11] 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. 4

[12] 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. *ECCV*, 2020. 4, 8, 9

[13] 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. 8, 9

[14] Gernot Riegler and Vladlen Koltun. Stable view synthesis. *CVPR*, 2021. 3, 5, 7, 8, 9

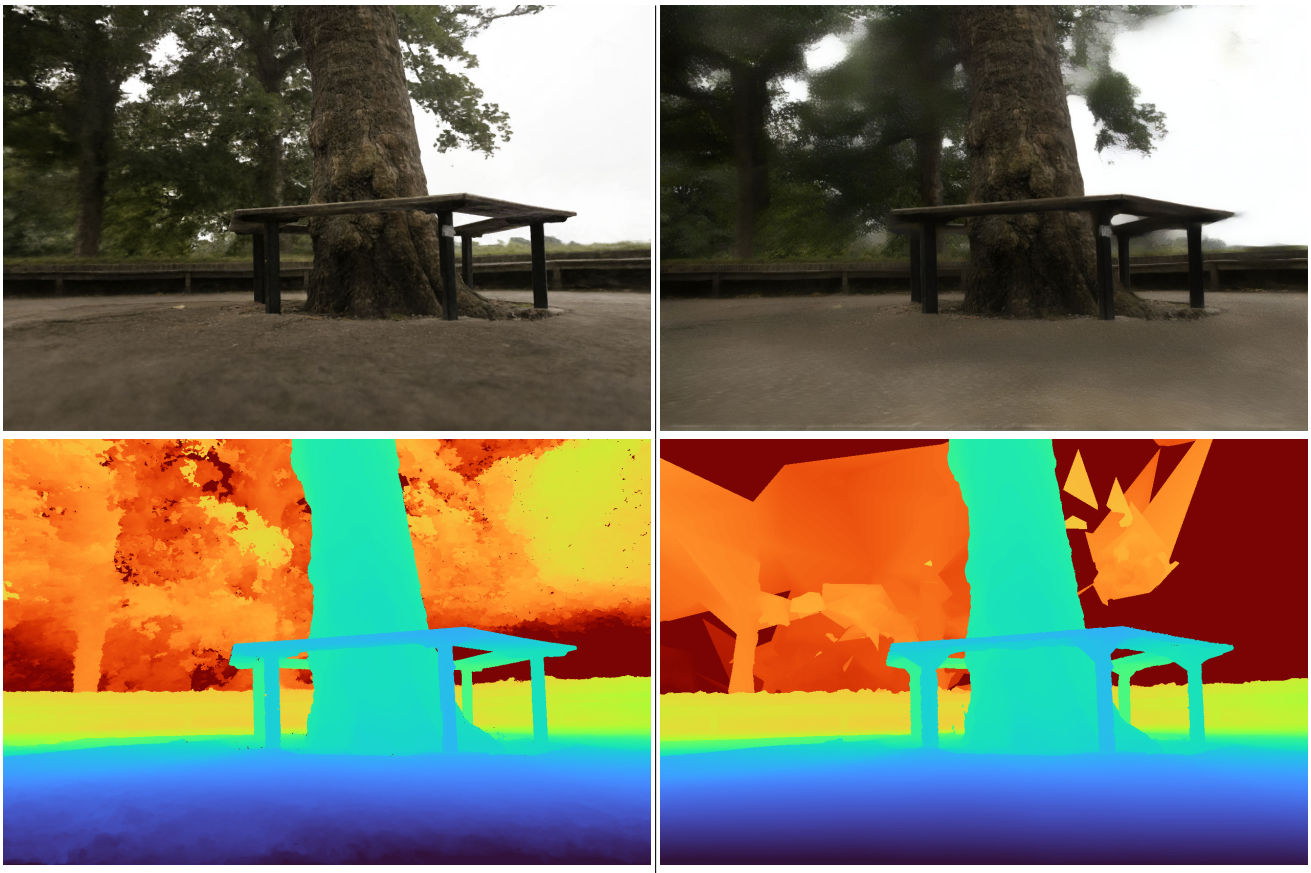
[15] Christophe Schlick. Fast alternatives to perlin's bias and gain functions. *Graphics Gems IV*, 1994. 2

[16] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. *CVPR*, 2016. 3, 7

[17] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020. 1

[18] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. NeRF++: Analyzing and Improving Neural Radiance Fields. *arXiv:2010.07492*, 2020. 8, 9

[19] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. *CVPR*, 2018. 3



(a) Our Model, PSNR = 16.67, SSIM = 0.493, LPIPS = 0.422

(b) SVS [14], PSNR = 16.11, SSIM = 0.488, LPIPS = 0.396

Figure 6. A rendering from (a) our model, and (b) Stable View Synthesis [14] on a scene from our dataset. The PSNR, SSIM, and LPIPS metrics for *this image* are shown in each subcaption. Despite SVS producing a blurry background, it achieves a lower LPIPS score, suggesting that this metric may be an unreliable signal in this setting. We also visualize (a) the depth map produced by our model alongside (b) the depth map produced by COLMAP [16] which is used by SVS. The poor reconstruction quality of COLMAP in the distant trees may explain why SVS struggles with this scene.

PSNR

	Outdoor					Indoor			
	bicycle	*flowers*	*garden*	*stump*	*treehill*	*room*	*counter*	*kitchen*	*bonsai*
NeRF [3, 12]	21.76	19.40	23.11	21.73	21.28	28.56	25.67	26.31	26.81
NeRF w/ DONeRF [13] param.	21.67	19.48	23.29	23.38	21.70	28.28	25.74	25.42	27.32
mip-NeRF [1]	21.69	19.31	23.16	23.10	21.21	28.73	25.59	26.47	27.13
NeRF++ [18]	22.64	20.31	24.32	24.34	22.20	28.87	26.38	27.80	29.15
Deep Blending [5]	21.09	18.13	23.61	24.08	20.80	27.20	26.28	25.02	27.08
Point-Based Neural Rendering [9]	21.64	19.28	22.50	23.90	20.98	26.99	25.23	24.47	28.42
Stable View Synthesis [14]	22.79	20.15	25.99	24.39	21.72	28.93	26.40	28.49	29.07
mip-NeRF [1] w/bigger MLP	22.90	20.79	25.85	23.64	21.71	30.67	28.61	29.95	31.59
NeRF++ [18] w/bigger MLPs	23.75	21.11	25.91	25.48	22.77	30.13	27.79	29.85	30.68
Our Model	24.37	21.73	26.98	26.40	22.87	31.63	29.55	32.23	33.46
Our Model w/GLO	23.95	21.60	25.09	25.98	21.99	28.24	28.40	30.81	30.27

SSIM

	Outdoor					Indoor			
	bicycle	*flowers*	*garden*	*stump*	*treehill*	*room*	*counter*	*kitchen*	*bonsai*
NeRF [3, 12]	0.455	0.376	0.546	0.453	0.459	0.843	0.775	0.749	0.792
NeRF w/ DONeRF [13] param.	0.454	0.379	0.542	0.522	0.461	0.841	0.776	0.678	0.813
mip-NeRF [1]	0.454	0.373	0.543	0.517	0.466	0.851	0.779	0.745	0.818
NeRF++ [18]	0.526	0.453	0.635	0.594	0.530	0.852	0.802	0.816	0.876
Deep Blending [5]	0.466	0.320	0.675	0.634	0.523	0.868	0.856	0.768	0.883
Point-Based Neural Rendering [9]	0.608	0.487	0.735	0.651	0.579	0.887	0.868	0.876	0.919
Stable View Synthesis [14]	0.663	0.541	0.818	0.683	0.606	0.905	0.886	0.910	0.925
mip-NeRF [1] w/bigger MLP	0.612	0.514	0.777	0.643	0.577	0.903	0.877	0.902	0.928
NeRF++ [18] w/bigger MLPs	0.630	0.533	0.761	0.687	0.597	0.883	0.857	0.888	0.913
Our Model	0.685	0.583	0.813	0.744	0.632	0.913	0.894	0.920	0.941
Our Model w/GLO	0.687	0.582	0.800	0.745	0.619	0.907	0.890	0.916	0.932

LPIPS

	Outdoor					Indoor			
	bicycle	*flowers*	*garden*	*stump*	*treehill*	*room*	*counter*	*kitchen*	*bonsai*
NeRF [3, 12]	0.536	0.529	0.415	0.551	0.546	0.353	0.394	0.335	0.398
NeRF w/ DONeRF [13] param.	0.542	0.539	0.436	0.492	0.545	0.368	0.394	0.410	0.368
mip-NeRF [1]	0.541	0.535	0.422	0.490	0.538	0.346	0.390	0.336	0.370
NeRF++ [18]	0.455	0.466	0.331	0.416	0.466	0.335	0.351	0.260	0.291
Deep Blending [5]	0.377	0.476	0.231	0.351	0.383	0.266	0.258	0.246	0.275
Point-Based Neural Rendering [9]	0.313	0.372	0.197	0.303	0.325	0.216	0.209	0.160	0.178
Stable View Synthesis [14]	0.243	0.317	0.137	0.281	0.286	0.182	0.168	0.125	0.164
mip-NeRF [1] w/bigger MLP	0.372	0.407	0.205	0.357	0.401	0.229	0.239	0.152	0.204
NeRF++ [18] w/bigger MLPs	0.356	0.395	0.223	0.328	0.386	0.270	0.270	0.177	0.230
Our Model	0.301	0.344	0.170	0.261	0.339	0.211	0.204	0.127	0.176
Our Model w/GLO	0.296	0.343	0.173	0.258	0.338	0.208	0.206	0.129	0.182

Table 3. Here we present an expanded version of Table 1 from the main paper, where we evaluate our model and multiple NeRF and non-NeRF baselines on our new dataset, but where we report metrics for each scene separately. Though some scenes are more challenging than others, the overall ranking of all techniques on each scene is generally consistent with the ranking suggested by the average metrics.

				Color Corrected				
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Time (hrs)	# Params
NeRF [3, 12]	18.72	0.609	0.473	19.67	0.616	0.473	4.15	1.5M
NeRF w/ DONeRF [13] param.	18.85	0.618	0.477	20.00	0.624	0.477	4.70	1.4M
mip-NeRF [1]	18.86	0.620	0.463	19.93	0.625	0.464	3.23	0.7M
NeRF++ [18]	19.32	0.647	0.425	20.52	0.652	0.427	9.71	2.4M
mip-NeRF [1] w/bigger MLP	19.85	0.697	0.340	21.09	0.702	0.343	22.75	9.0M
NeRF++ [18] w/bigger MLPs	19.83	0.693	0.358	21.15	0.697	0.362	19.94	9.0M
Stable View Synthesis [14]	21.13	0.777	0.209	22.76	0.778	0.216	-	-
Our Model	20.52	0.734	0.301	21.98	0.737	0.304	6.61	9.0M
Our Model w/GLO	19.65	0.731	0.280	22.78	0.761	0.272	7.09	9.0M

Table 4. The average performance of our model and all NeRF baselines, as well as the top-performing non-NeRF baseline on our own dataset (Stable View Synthesis), on the "Tanks and Temples" dataset [8]. This dataset exhibits significant photometric variation across images (see Figure 4), making it ill-suited to our goals. To partially ameliorate this we present additional "color corrected" metrics, in which this photometric variation has been ameliorated. Our model outperforms all NeRF baselines, but is slightly outperformed by SVS (which was designed for this dataset, and which was trained on the training set of this dataset), though this appears to be partially due to SVS being better able to predict the photometric variation of this dataset, while the "w/ GLO" variant of our model learns to be invariant to that photometric variation.

Color Corrected PSNR

	M60	*Playground*	*Train*	*Truck*
NeRF [3, 12]	17.59	21.72	19.17	20.21
NeRF w/ DONeRF [13] param.	17.31	23.13	18.76	20.81
mip-NeRF [1]	17.58	22.21	19.42	20.50
NeRF++ [18]	18.09	23.05	19.50	21.44
mip-NeRF [1] w/bigger MLP	19.14	23.65	19.82	21.74
NeRF++ [18] w/bigger MLPs	18.81	24.01	19.84	21.94
Stable View Synthesis [14]	19.94	25.50	21.76	23.85
Our Model	19.28	26.41	18.23	24.01
Our Model w/GLO	19.50	27.00	22.15	22.48

Color Corrected SSIM

	M60	*Playground*	*Train*	*Truck*
NeRF [3, 12]	0.619	0.624	0.575	0.646
NeRF w/ DONeRF [13] param.	0.622	0.659	0.559	0.657
mip-NeRF [1]	0.629	0.638	0.582	0.650
NeRF++ [18]	0.644	0.676	0.586	0.704
mip-NeRF [1] w/bigger MLP	0.694	0.726	0.642	0.747
NeRF++ [18] w/bigger MLPs	0.682	0.724	0.630	0.751
Stable View Synthesis [14]	0.756	0.788	0.731	0.836
Our Model	0.714	0.781	0.635	0.818
Our Model w/GLO	0.720	0.798	0.723	0.804

Color Corrected LPIPS

	M60	*Playground*	*Train*	*Truck*
NeRF [3, 12]	0.466	0.473	0.493	0.458
NeRF w/ DONeRF [13] param.	0.466	0.458	0.514	0.468
mip-NeRF [1]	0.462	0.461	0.483	0.449
NeRF++ [18]	0.432	0.418	0.473	0.387
mip-NeRF [1] w/bigger MLP	0.367	0.330	0.379	0.296
NeRF++ [18] w/bigger MLPs	0.383	0.348	0.409	0.308
Stable View Synthesis [14]	0.251	0.212	0.247	0.152
Our Model	0.341	0.264	0.389	0.223
Our Model w/GLO	0.330	0.246	0.284	0.228

Table 5. Performance on each individual test-set scene from the "Tanks and Temples" dataset, using our color corrected metrics.