Appendix

A1. Related Work

Alpha and Bad Weather. Early works on weather modeling [22, 23] use volume rendering to characterize the effect of rain and fog. Dehazing methods [7, 9], apart from separating the base scene albedo from the foggy airlight, use the estimated alpha mask to compute an up-to-scale depth map proportional to $-\log(1-\alpha)$. The notion of alpha invariance is implied in this step.

NeRF vs MPI. NeRF uses volume rendering [17] for view synthesis [20] and other inverse rendering tasks . Unlike prior works [25, 29, 39, 40] that directly model the alphas of a fixed set of multi-plane images (MPI), NeRF optimizes the continuous volumetric densities $\sigma(x)$. MPI fixes the discretization in advance, whereas the density parameterization in NeRF makes it easy to adjust discretization and resample along a ray. The flipside of this flexibility, however, is that the magnitude of $\sigma(x)$ is tied to the domain of integration. Our work emphasizes that even though $\sigma(x)$ and distance change to compensate for each other, the alpha value of a particular discretized segment should remain constant. The opacity is an invariant property of local geometry.

Volume-rendered SDF. Signed distance function (SDF) is suitable for procedural content authoring and surface extraction. SDF rendering used to rely on finding surface intersections by sphere tracing [24, 36], but recent methods such as NeuS [33] and VolSDF [37] move to the "fuzzier" volume rendering for easier optimization. To perform volume rendering, the distance function is first transformed into volumetric densities before alpha compositing. VolSDF learns scaling and shrinking coefficients on the distance-transformed volume densities. NeuS instead demands the CDF of volume rendering to match the shape of a scaled, horizontally flipped `sigmoid` i.e. $1 - T(t) = \texttt{sigmoid}(s \cdot -\texttt{SDF}(t))$, so that the CDF's derivative, in other words the weighting function $w(t)$, has its local maxima located at SDF value 0 for precise surface level-set extraction. The learned parameter s in the `sigmoid` acts as a global scaling coefficient on the implied volume densities. In this sense both formulations are alpha invariant by default, with the extra constraint that the magnitude of volume density function is globally tied to its sharpness. The assumption is restrictive but fine for most use cases.

Gaussian Splatting. Gaussian splatting [12, 15, 35] renders an image by alpha compositing point primitives. Since the scene representation is by nature discrete, the concept of volume density does not apply, as is the case with MPIs. These explicit primitives can be efficiently rasterized using GPU pipelines. Whereas NeRF could use ray sampling strategies on the continuous density field to refine local details and discover missing structures, 3D gaussians / metaballs [12–14] are less flexible; it relies on SfM initialization in some cases, and needs to explicitly optimize point shape, location, and periodically remove, repopulate and merge-split the primitives. 3DGS [12] uses an exponent activation to scale the shape of each primitive, and it shares a similar motivation in terms of handling arbitrary scene scaling.

A2. Transmittance in Discrete Setting

In Sec. 3, we write down the expression for high transmittance initialization in the continuous setting. The expression is the same when the ray is cut into discrete intervals. The tree-branching analogy in Fig. 1 shows that the transmittance / survival probability for each segment is $1-\alpha_i$, with overall survival probability $\prod(1-\alpha_i) = \prod e^{-\sigma_i d_i} = e^{-\sum \sigma_i d_i} = \exp\left(-\int_0^L \sigma ds\right)$. The same steps as Eq. (5) from Sec. 3 follow.

A3. Additional Results

Surface Statistics. We provide additional visualizations verifying alpha invariance in Fig. A2. Similar to Fig. 4, we shoot rays through each pixel to obtain the density histograms $\{w_i\}$, and query the spatial location at the 50-th percentile of the probability CDF of each histogram for its σ value. Here, we showcase various scene types from different datasets, and generate these visualizations with different architectures, demonstrating how inverse scaling between distance and volume density is a generalizable phenomenon in radiance fields.

Voxel Variants, continued. DVGO has a strategy of fixing the scene size to some canonical scale where the interval length between each sampled point is dependent on the current voxel grid resolution's ratio to the base resolution, which empirically equates to either 0.5 or 1, depending on the stages of optimization progress. We also note that DVGO's sampling procedure is stochastic, as each ray has a potentially unique number of samples. The implication is that every ray will have a different ray length purely determined by its number of samples as the interval length between any two contiguous samples is hardcoded to either 0.5 or 1. As such, the term "ray length" loses its meaning in DVGO's context as there are no deterministic near and far planes; every ray is simply deconstructed into its constituent samples. We observe that disabling this heuristic (i.e., setting the interval lengths to be the true physical distances and forcing each ray to be bounded within a predetermined global near and far plane) produces significantly worsened rendering quality, as shown in Tab. A2.

For Plenoxels, only rectifying σ with exp is insufficient to maintain convergence at various scene scales; a high transmittance offset is needed even at $k = 1$. See Tab. A3 for numerical results. We note that our results are worse ($\sim$ 2-

	Chair										Ship									
	Default					Ours					Default					Ours				
$k = 0.1$	31.20	31.14	31.04	14.04	28.82	31.20	31.27	31.07	30.99	31.25	5.88	5.88	27.59	27.61	5.88	27.20	27.06	27.56	27.51	27.31
$k = 0.4$	28.96	31.32	31.26	31.17	31.26	31.00	31.29	31.02	31.24	31.09	27.57	27.46	27.59	27.35	27.49	27.11	27.14	27.00	27.11	27.84
$k = 1.0$	14.04	31.32	14.04	14.04	28.82	31.24	31.30	31.11	31.39	31.23	27.56	25.64	27.27	25.57	27.60	27.18	26.99	27.16	27.30	27.11
$k = 2.5$	14.04	31.37	28.99	31.38	31.32	31.14	31.23	31.15	31.03	31.23	27.50	27.45	5.88	27.49	27.56	27.56	27.31	27.35	27.18	27.16
$k = 10.0$	14.04	30.97	28.76	31.00	9.75	31.29	31.19	31.16	31.10	31.22	5.88	27.57	27.35	24.12	27.43	26.99	27.00	27.16	27.27	27.28
average	25.76 ± 7.79					31.18 ± 0.10					22.89 ± 8.54					27.23 ± 0.20				

Table A1. PSNR ↑ values of Vanilla-NeRF on the chair and ship scenes from the Blender dataset. Here, we run 5 experiments for 5 different k-values, and compare the results from the default NeRF baseline against our exp parametrization on σ and high transmittance initialization. We observe consistent rendering quality across all runs for both scenes with our method, but identify inconsistent rendering quality for the default configuration, with failure modes and poor convergence marked here in red.

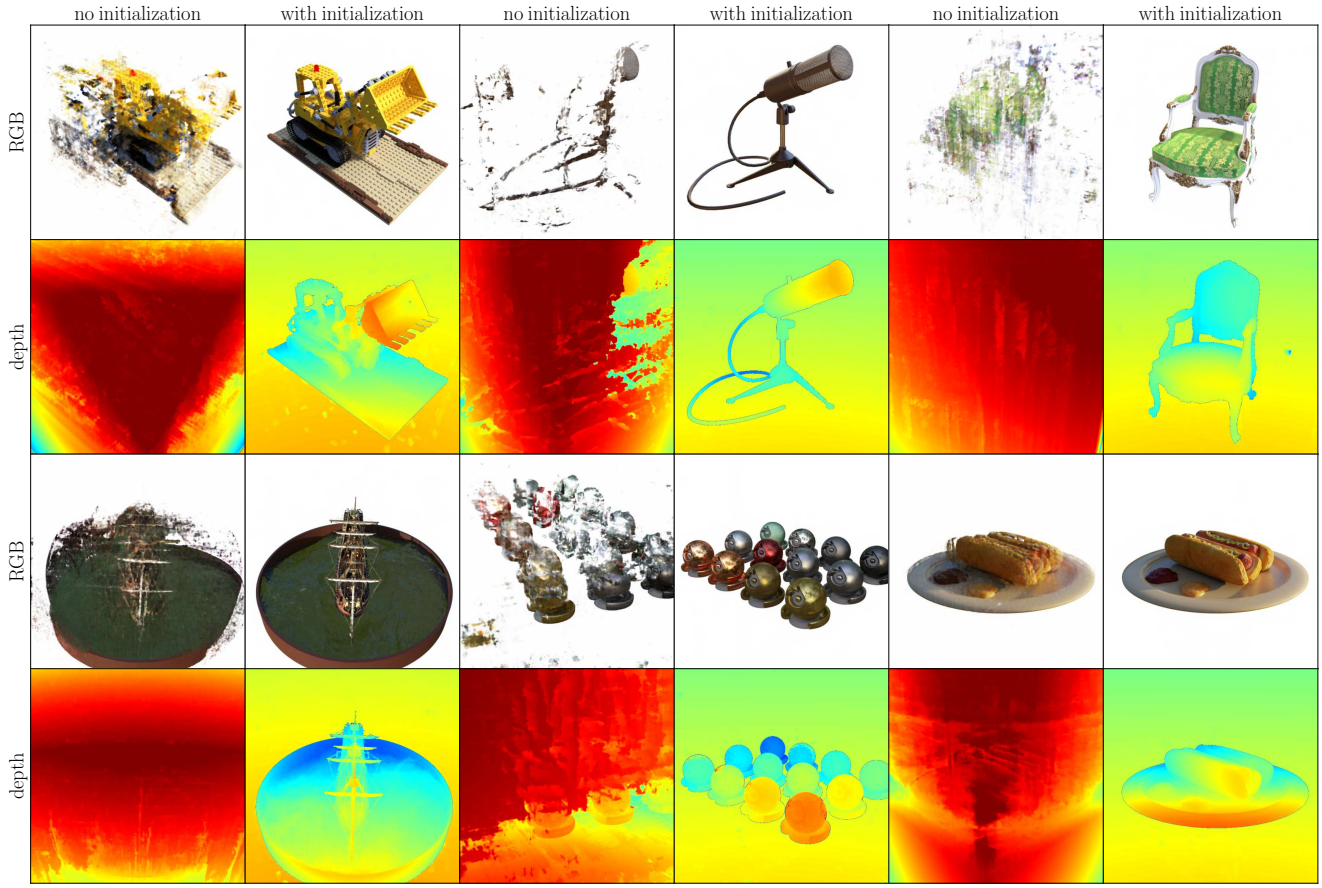

Figure A1. RGB-image and depth-maps produced from TensoRF on the lego, mic, chair, ship, materials, and hotdog scenes from the Blender dataset at a large scene scaling $k = 25$. When using `exp` activation to parameterize σ, not using our high transmittance initialization strategy causes the optimization to get stuck with cloudy floaters.

3 dB) than the results presented in Plenoxels [8]. We had to significantly lower the learning rate to be more suitable for the exp activation, and believe that other changes in the hyperparameters are also needed to match the default performance. We leave this as future work; our results still demonstrate consistent rendering quality and a need for our high transmittance initialization strategy.

Benefit of High Transmittance Initialization. We provide additional RGB-image and depth-map visuals in Fig. A1 demonstrating the benefits of our high transmittance initialization in handling large scene scales when using exp activation.

A4. Reproducibility

In Tab. 2, we observe random failure modes when training the vanilla 8-layer MLP on various scenes using NeRF-

	chair	drums	ficus	hotdog	lego	materials	mic	ship
disabled	fail	fail	fail	fail	fail	fail	fail	fail
default	34.10	25.40	32.56	36.67	34.53	29.71	33.23	28.76

	fern	flower	fortress	horns	leaves	orchids	room	trex
disabled	15.74	17.38	20.77	20.62	16.96	11.77	21.44	20.62
default	24.49	27.61	29.91	27.01	20.41	19.95	30.87	26.41

	bicycle	bonsai	counter	garden	kitchen	room	stump
disabled	fail	fail	fail	fail	fail	fail	fail
default	21.98	27.33	25.41	24.41	25.81	28.14	23.51

Table A2. PSNR ↑ values of DVGO on the Blender (top row), LLFF (middle row), and Mip-NeRF 360 (bottom row) datasets. DVGO sets the interval lengths d to the ratio of the current voxel grid resolution to the base voxel grid resolution in order to to make the model independent of scene size; this is referred to as 'default' in the table. We observe that disabling this heuristic (i.e., setting the interval lengths in the volume rendering equation to be the true physical distances between any two sample points) results in DVGO failing to render at a high quality. These failure modes are marked with red in the rows titled 'disabled'.

	chair	drums	ficus	hotdog	lego	materials	mic	ship
$k = 0.1$	31.80 / 31.89	24.23 / 24.26	29.41 / 29.58	34.12 / 34.31	31.20 / 31.31	28.21 / 28.41	31.11 / 31.31	28.16 / 28.25
$k = 0.4$	30.33 / 31.83	23.61 / 24.23	27.65 / 29.12	31.76 / 34.46	29.61 / 31.22	26.06 / 28.02	29.04 / 31.13	27.01 / 28.13
$k = 1.0$	27.06 / 32.00	21.66 / 24.37	24.77 / 29.39	27.31 / 34.56	26.22 / 31.35	23.14 / 28.31	24.85 / 31.29	24.58 / 28.19
$k = 2.5$	17.49 / 32.43	17.24 / 24.44	18.89 / 29.83	20.20 / 34.72	16.64 / 31.53	15.51 / 28.64	15.89 / 31.61	16.67 / 28.22
$k = 10.0$	13.05 / 32.25	10.84 / 24.45	11.40 / 30.23	13.03 / 34.93	11.64 / 31.69	9.57 / 28.79	15.89 / 31.61	11.27 / 28.21

Table A3. PSNR ↑ for Plenoxels [baseline / ours] at different scene scaling k on the Blender dataset. By default, Plenoxels applies a very large learning rate directly on the coefficients of a voxel grid, where σ is queried via trilinear interpolation, followed by ReLU to ensure non-negative density values. The baseline Plenoxels model here replaces ReLU with `exp` activation and uses a reduced learning rate ($\eta_{\text{init}} = 0.05$, $\eta_{\text{final}} = 0.005$) with no reverse cosine delay. Our model in addition applies a high transmittance offset. As shown, this transmittance offset is required for high rendering quality at various scene sizes, even at $k = 1$.

Pytorch [38] at git commit hash `63a5a63`. To get a better sense of the *frequency* of these failure modes, we train Vanilla-NeRF on chair and ship scenes 5 times for each k-value, comparing the default `ReLU` parametrization on σ against our high transmittance initialization in combination with exp. Full results are provided in Tab. A1. We attribute the random failure modes to poor initialization of the MLP layers (the initial output distribution has 0 mean and 0 variance) and the inability of `ReLU` to provide a smooth transition from low to high α values. See Fig. 2 for a visualization. However, even with such a poorly initialized MLP, our parametrization on σ is enough to guarantee convergence on all runs across all tested k values.

We train NeRF-Pytorch for only 200k iterations. A longer training schedule of 500k iterations would push the NeRF-Pytorch performance closer, but still slightly below the original NeRF numbers. The overall conclusions do not change.

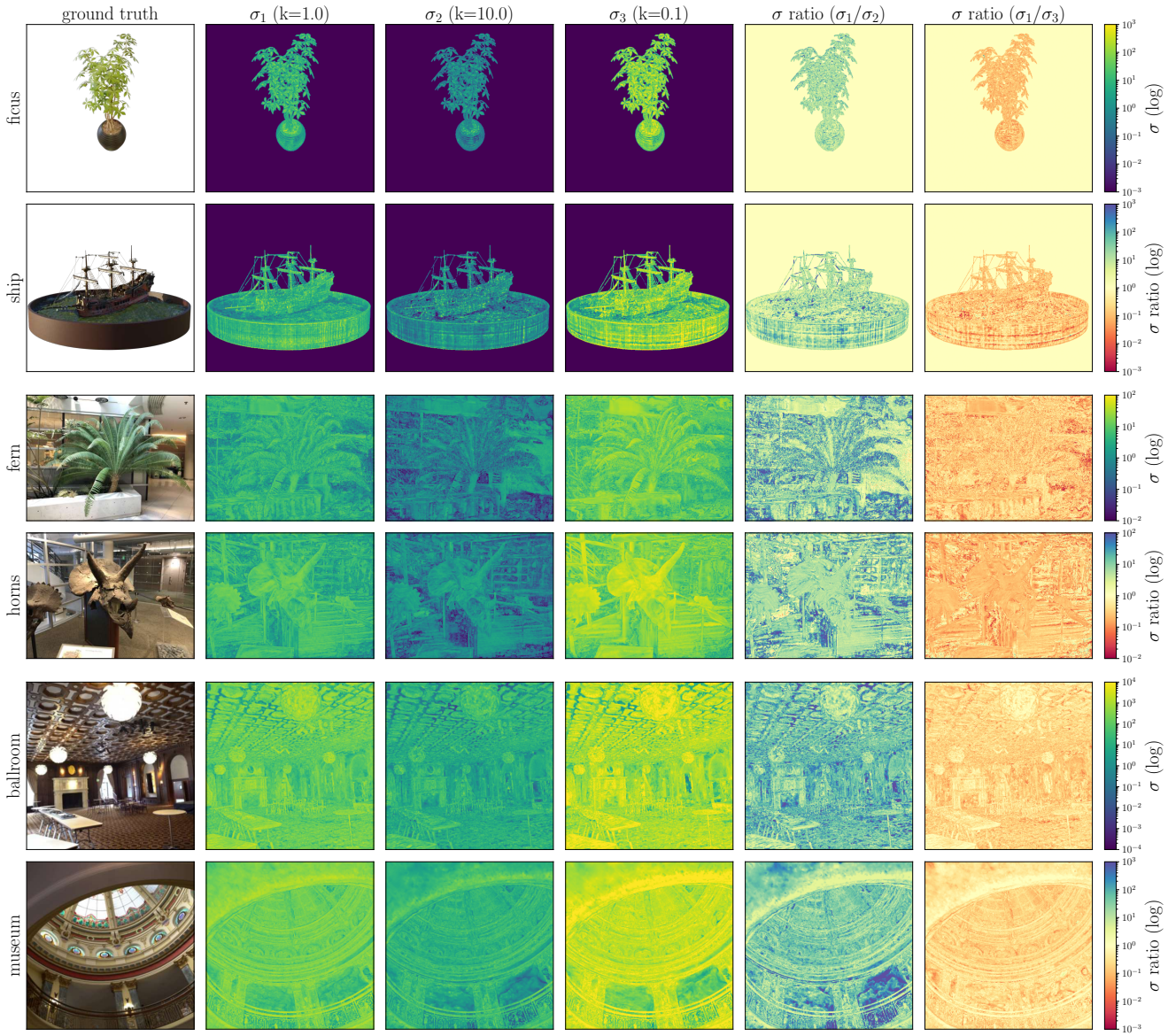

Figure A2. Visualization of σ-image produced at the 50-th percentile location of each ray's density histogram CDF. We produce a division image that shows the global difference of numerical range across different scene scaling factor k. The first four rows are produced from the TensoRF architecture (the top two rows are the ficus and ship scenes from the blender dataset; the middle two rows are the fern and horns scenes from the LLFF dataset). The bottom two rows are produced from the Nerfacto architecture on the ballroom and museum scenes from the Tanks-and-Temples dataset. Across a variety of scenes and NeRF architectures, these visualizations demonstrate that the phenomenon of alpha invariance holds to a very strong degree.