6. Appendix

6.1. Volumetric rendering

We use the same volume rendering formula as NeRF [24], originally from [21], where the color of a pixel is represented as a sum over samples taken along the corresponding ray through the volume:

$$\sum_{i=1}^{N} \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right) \left(1 - \exp(-\sigma_i \delta_i)\right) \mathbf{c}_i \qquad (9)$$

where the first exp represents ray transmission to sample i, $1-\exp(-\sigma_i\delta_i)$ is the absorption by sample i, σ_i is the (post-activation) density of sample i, and $\mathbf{c}_i$ is the color of sample i, with distance δ_i to the next sample.

6.2. Per-scene results

Fig. 11 provides a qualitative comparison of methods for the Phototourism dataset, on the *Trevi fountain* scene. We also provide quantitative metrics for each of the three tasks we study, for each scene individually. Tab. 7 reports metrics on the static synthetic scenes, Tab. 8 reports metrics on the static real forward-facing scenes, Tab. 9 reports metrics on the dynamic synthetic monocular "teleporting camera" scenes, Tab. 10 reports metrics on the dynamic real forward-facing multiview scenes, and Tab. 11 reports metrics on the Phototourism scenes.

6.3. Ablation studies

Multiscale. In Tab. 4, we ablate our model on the static *Lego* scene [24] with respect to our multiscale planes, to assess the value of including copies of our model at different scales.

Feature length. In Tab. 5, we ablate our model on the static *Lego* scene with respect to the feature dimension M learned at each scale.

Time smoothness regularizer. Sec. 3.2 describes our temporal smoothness regularizer based on penalizing the norm of the second derivative over the time dimension, to encourage smooth motion and discourage acceleration. Tab. 6 illustrates an ablation study of this regularizer on the *Jumping Jacks* scene from D-NeRF [30].

7. Model hyperparameters

Our full hyperparameter settings are available in the config files in our released code, at `github.com/sarafridov/K-Planes`.

Scales (32 Feat. Each)	Explicit PSNR ↑	Hybrid PSNR ↑	# params ↓
64, 128, 256, 512	35.26	35.79	34M
128, 256, 512	35.29	35.75	33M
256, 512	34.52	35.37	32M
512	32.93	33.60	25M
64, 128, 256	34.26	35.07	8M

Scales (96 Feat. Total)	Explicit PSNR ↑	Hybrid PSNR ↑	# params ↓
64, 128, 256, 512	35.16	35.67	25M
128, 256, 512	35.29	35.75	33M
256, 512	34.50	35.16	47M
512	33.12	34.09	76M
64, 128, 256	34.26	35.07	8M

Table 4. **Ablation study over scales.** Including even a single lower scale improves performance, for both our explicit and hybrid models, even when holding the total feature dimension constant. Using lower scales only (excluding resolution 512^3) substantially reduces model size and yields quality much better than using high resolution alone, though slightly worse than including both low and high resolutions. This experiment uses the static *Lego* scene; in the top table each scale is allocated 32 features and in the bottom table a total of 96 features are allocated evenly among all scales.

Feature Length (M)	Explicit PSNR ↑	Hybrid PSNR ↑	# params ↓
2	30.66	32.05	2M
4	32.27	34.18	4M
8	33.80	35.12	8M
16	34.80	35.44	17M
32	35.29	35.75	33M
64	35.38	35.88	66M
128	35.45	35.99	132M

Table 5. **Ablation study over feature length M.** Increasing the feature length M learned at each scale consistently improves quality for both our models, with a corresponding linear increase in model size and optimization time. Our experiments in the main text use a mixture of $M = 16$ and $M = 32$; for specific applications it may be beneficial to vary M along this tradeoff between quality and model size. This experiment uses the static *Lego* scene with 3 scales: 128, 256, and 512.

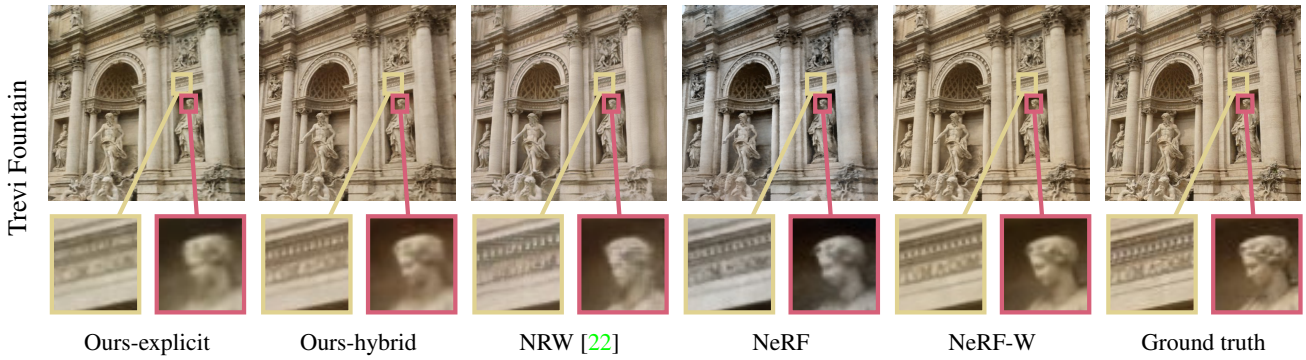

Figure 11. **Qualitative results from Phototourism dataset**. We compare our model with strong baselines. Our method captures the geometry and appearance of the scene, but produces slightly lower resolution results than NeRF-W. Note that our model optimizes in just 35 minutes on a single GPU compared to NeRF-W, which takes 2 days on 8 GPUs.

Time Smoothness Weight (λ)	Explicit PSNR ↑	Hybrid PSNR ↑
0.0	30.45	30.86
0.001	31.61	32.23
0.01	32.00	32.64
0.1	31.96	32.58
1.0	31.36	32.22
10.0	30.45	31.63

Table 6. **Ablation study over temporal smoothness regularization.** For both models, a temporal smoothness weight of 0.01 is best, with PSNR degrading gradually with over- or under-regularization. This experiment uses the *Jumping Jacks* scene with 4 scales: 64, 128, 256, and 512, and 32 features per scale.

PSNR ↑

	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Mean
Ours-explicit	34.82	25.72	31.2	36.65	35.29	29.49	34.00	30.51	32.21
Ours-hybrid	34.99	25.66	31.41	36.78	35.75	29.48	34.05	30.74	32.36
INGP [25]	35.00	26.02	33.51	37.40	36.39	29.78	36.22	31.10	33.18
TensoRF [6]	35.76	26.01	33.99	37.41	36.46	30.12	34.61	30.77	33.14
Plenoxels [10]	33.98	25.35	31.83	36.43	34.10	29.14	33.26	29.62	31.71
JAXNeRF [7, 24]	34.20	25.27	31.15	36.81	34.02	30.30	33.72	29.33	31.85

SSIM ↑

	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Mean
Ours-explicit	0.981	0.937	0.975	0.982	0.978	0.949	0.988	0.892	0.960
Ours-hybrid	0.983	0.938	0.975	0.982	0.982	0.950	0.988	0.897	0.962
INGP	-	-	-	-	-	-	-	-	-
TensoRF	0.985	0.937	0.982	0.982	0.983	0.952	0.988	0.895	0.963
Plenoxels	0.977	0.933	0.976	0.980	0.975	0.949	0.985	0.890	0.958
JAXNeRF	0.975	0.929	0.970	0.978	0.970	0.955	0.983	0.868	0.954

Table 7. **Full results on static synthetic scenes [24].** Dashes denote values that were not reported in prior work.

PSNR ↑

	Room	Fern	Leaves	Fortress	Orchids	Flower	T-Rex	Horns	Mean
Ours-explicit	32.72	24.87	21.07	31.34	19.89	28.37	27.54	28.40	26.78
Ours-hybrid	32.64	25.38	21.30	30.44	20.26	28.67	28.01	28.64	26.92
NeRF [24]	32.70	25.17	20.92	31.16	20.36	27.40	26.80	27.45	26.50
Plenoxels [10]	30.22	25.46	21.41	31.09	20.24	27.83	26.48	27.58	26.29
TensoRF (L) [6]	32.35	25.27	21.30	31.36	19.87	28.60	26.97	28.14	26.73
DVGOv2 [33]	-	-	-	-	-	-	-	-	26.34

SSIM ↑

	Room	Fern	Leaves	Fortress	Orchids	Flower	T-Rex	Horns	Mean
Ours-explicit	0.955	0.809	0.738	0.898	0.665	0.867	0.909	0.884	0.841
Ours-hybrid	0.957	0.828	0.746	0.890	0.676	0.872	0.915	0.892	0.847
NeRF [24]	0.948	0.792	0.690	0.881	0.641	0.827	0.880	0.828	0.811
Plenoxels [10]	0.937	0.832	0.760	0.885	0.687	0.862	0.890	0.857	0.839
TensoRF (L) [6]	0.952	0.814	0.752	0.897	0.649	0.871	0.900	0.877	0.839
DVGOv2 [33]	-	-	-	-	-	-	-	-	0.838

Table 8. **Full results on static forward-facing scenes [23].** Dashes denote values that were not reported in prior work.

PSNR ↑

	Hell Warrior	Mutant	Hook	Balls	Lego	T-Rex	Stand Up	Jumping Jacks	Mean
Ours-explicit	25.60	33.56	28.21	38.99	25.46	31.28	33.27	32.00	31.05
Ours-hybrid	25.70	33.79	28.50	41.22	25.48	31.79	33.72	32.64	31.61
D-NeRF [30]	25.02	31.29	29.25	32.80	21.64	31.75	32.79	32.80	29.67
T-NeRF [30]	23.19	30.56	27.21	32.01	23.82	30.19	31.24	32.01	28.78
Tensor4D [31]	-	-	-	-	26.71	-	36.32	34.43	-
TiNeuVox [9]	28.17	33.61	31.45	40.73	25.02	32.70	35.43	34.23	32.67
V4D [11]	27.03	36.27	31.04	42.67	25.62	34.53	37.20	35.36	33.72

SSIM ↑

	Hell Warrior	Mutant	Hook	Balls	Lego	T-Rex	Stand Up	Jumping Jacks	Mean
Ours-explicit	0.951	0.982	0.951	0.989	0.947	0.980	0.980	0.974	0.969
Ours-hybrid	0.952	0.983	0.954	0.992	0.948	0.981	0.983	0.977	0.971
D-NeRF [30]	0.95	0.97	0.96	0.98	0.83	0.97	0.98	0.98	0.95
T-NeRF [30]	0.93	0.96	0.94	0.97	0.90	0.96	0.97	0.97	0.95
Tensor4D [31]	-	-	-	-	0.953	-	0.983	0.982	-
TiNeuVox [9]	0.97	0.98	0.97	0.99	0.92	0.98	0.99	0.98	0.97
V4D [11]	0.96	0.99	0.97	0.99	0.95	0.99	0.99	0.99	0.98

Table 9. **Full results on monocular "teleporting-camera" dynamic scenes.** We use the synthetic scenes from D-NeRF [30], which we refer to as monocular "teleporting-camera" because although there is a single training view per timestep, the camera can move arbitrarily between adjacent timesteps. Dashes denote unreported values. TiNeuVox trains in 30 minutes, V4D in 4.9 hours, D-NeRF in 2 days, and Tensor4D for an unspecified duration (Tensor4D reports iterations rather than time). Our reported results were obtained after roughly 1 hour of optimization on a single GPU. Like D-NeRF and TiNeuVox, we train and evaluate using half-resolution images (400 by 400 pixels).

PSNR ↑

	Coffee Martini	Spinach	Cut Beef	Flame Salmon[1]	Flame Steak	Sear Steak	Mean
Ours-explicit	28.74	32.19	31.93	28.71	31.80	31.89	30.88
Ours-hybrid	29.99	32.60	31.82	30.44	32.38	32.52	31.63
LLFF [23]	-	-	-	23.24	-	-	-
DyNeRF [16]	-	-	-	29.58	-	-	-
MixVoxels-L† [37]	29.36	31.61	31.30	29.92	31.21	31.43	30.80

SSIM ↑

	Coffee Martini	Cook Spinach	Cut Beef	Flame Salmon[1]	Flame Steak	Sear Steak	Mean
Ours-explicit	0.943	0.968	0.965	0.942	0.970	0.971	0.960
Ours-hybrid	0.953	0.966	0.966	0.953	0.970	0.974	0.964
LLFF	-	-	-	0.848	-	-	-
DyNeRF	-	-	-	0.961	-	-	-
MixVoxels-L	0.946	0.965	0.965	0.945	0.970	0.971	0.960

† Very recent/concurrent work. MixVoxels was released in December 2022. [1]Using the first 10 seconds of the 30 second long video.

Table 10. **Full results on multiview dynamic scenes [16].** Dashes denote unreported values. Note that our method optimizes in less than 4 GPU hours, whereas DyNeRF trains on 8 GPUs for a week, approximately 1344 GPU hours.

PSNR ↑				
	Brandenburg Gate	Sacre Coeur	Trevi Fountain	Mean
Ours-explicit	24.85	19.90	22.00	22.25
Ours-hybrid	25.49	20.61	22.67	22.92
NeRF-W [20]	29.08	25.34	26.58	27.00
NeRF-W (public)†	21.32	19.17	18.61	19.70
LearnIt [35]	19.11	19.33	19.35	19.26
MS-SSIM ↑				
	Brandenburg Gate	Sacre Coeur	Trevi Fountain	Mean
Ours-explicit	0.912	0.821	0.845	0.859
Ours-hybrid	0.924	0.852	0.856	0.877
NeRF-W	0.962	0.939	0.934	0.945
Nerf-W (public)†	0.845	0.752	0.694	0.764
LearnIt	-	-	-	-

† Open-source version https://github.com/kwea123/nerf_pl/tree/nerfw where we implement the test-time optimization ourselves exactly as for k-planes. NeRF-W code is not public.

Table 11. **Full results on phototourism scenes.** Note that our results were obtained after about 35 GPU minutes, whereas NeRF-W trains with 8 GPUs for two days, approximately 384 GPU hours.