Supplemental Materials

A Videos

We include several videos as supplementary materials under the **vids** folder. We refer reviewers to **index.html** for descriptions and further details.

B Space Efficiency

As mentioned in Section 5, a downside of our method is an increased serialization footprint due to training a hierarchy of spatial grid NeRFs. We explore several mitigations:

Saving memory on coarser levels. The learned feature grids used in fast NeRF methods usually comprise the bulk of their memory footprint. In the case of iNGP [21] (which we use as the backbone model in our experiments), the feature grid is stored as a multi-resolution hash table. One possible optimization is to use smaller hash tables for lower-resolution levels. Assume that S_l is the hash table size of the finest-resolution level l. By setting $S_{l-1} = S_l/2$, we can reduce the overall storage cost of our pyramid from lS_l (when each level has the same capacity) to less than $2S_l$.

Space-efficient grid implementations. Storing features in an explicit voxel grid is expensive at high resolutions as the space complexity is $\mathcal{O}(n^3)$ for spatial scenes and $\mathcal{O}(n^4)$ spatio-temporal scenes. iNGP [21] instead uses hash tables that are smaller that the explicit grids. As an alternative, TensoRF [7] uses CANDECOMP/PARAFAC (CP) [6] or Vector-Matrix (VM) decomposition to store features in a highly space-efficient manner. We test both decomposition methods as alternatives to our default hash table encoding.

Existing grid hierarchy. Note that multi-resolution grids such as those used by iNGP [21] or K-Planes [10] already define a scale hierarchy that is a natural fit for PyNeRF. Rather than learning a separate feature grid for each model in our pyramid, we can reuse the same multi-resolution features across levels (while still training different MLP heads).

We measure the perceptual quality and memory footprint of these strategies across the datasets defined in Section 4.2 and Section 4.3 and compare to our default PyNeRF method and the base TensoRF-CP, TensoRF-VM, and iNGP implementations. We summarize our results in Table 5. Using smaller tables for coarser levels reduces the overall model size by over 2× but comes at a 0.5 db cost in PSNR. Sharing the feature grid across levels reduces the overall size even further (by 4×) without a noticeable degradation in quality. Using TensoRF instead of iNGP hash tables also reduces disk space, especially with CP decomposition (300× reduction) although this is offset by a significant decrease in quality. All PyNeRF implementations perform significantly better than their base models.

Table 5: **Space Efficiency.** Using smaller tables for coarser levels or sharing the feature grid across hierarchy levels both decrease model size, with the latter strategy resulting in no apparent degradation in quality. Using TensoRF as the backbone instead of iNGP also reduces model size but reduces quality significantly. All PyNeRF implementations perform better than the base models.

Strategy	↑PSNR	↑SSIM	↓LPIPS	↓Avg Error	↓ Model Size (MB)
PyNeRF - smaller coarse tables	29.06	0.773	0.236	0.068	410
PyNeRF - TensoRF-CP	22.19	0.591	0.524	0.141	3
PyNeRF - TensoRF-VM	23.88	0.642	0.412	0.118	531
PyNeRF - shared iNGP table	**29.49**	0.778	**0.230**	**0.066**	222
TensoRF-CP [7]	21.63	0.580	0.529	0.150	**0.4**
TensoRF-VM [7]	22.81	0.588	0.461	0.132	67
iNGP [21]	23.96	0.697	0.354	0.096	214
PyNeRF	**29.49**	**0.780**	0.233	**0.066**	912