A Model details

A.1 Implementation

We use a two-layer MLP with bias. The first layer has 64 neurons and a SiLU activation function [8]. For the RGB-only implementation, the output layers have 3 output neurons and a Sigmoid activation function. Instead, for the semantic segmentation implementation, the output layer has 67 output neurons, with a Sigmoid activation function for the first 3 channels, and a Softmax for the last 64. We use the Adam optimizer with $\varepsilon = 1 \times 10^{-15}$, a learning rate of 0.001 for the MLP and 0.0025 for the feature vectors, and $\lambda_{sem} = 0.001$ for the semantic segmentation model. The rest of the Gaussians' parameters and optimization hyperparameters follow 3DGS's [9] original implementation. We use the 3DGS [9] CUDA differentiable rasterizer. We adapt it to our model by skipping spherical harmonics and passing feature vectors as precomputed colors. To do it, we change the number of channels when compiling from 3 to 16 and 32 for FeatSplat-16 and FeatSplat-32 respectively. The model has been implemented on Pytorch 1.12 and CUDA SDK 11.6.2.

A.2 Embeddings ablation

We ablate the impact on the 32-dimensional model of concatenating to the pre-MLP feature vectors the 3D camera postion coordinates $\mathbf{x}_{cam}$, its rotation using Euler angles $\mathbf{x}_{rot}$, and the 2 dimension embedding of the relative position of each pixel $\mathbf{e}_p$. The evaluation is done on the scenes *Bicycle* from Mip-360 [1] and *Train* from T&T [13].

	Bicycle			Train		
FeatSplat – 32	SSIM	PSNR	LPIPS	SSIM	PSNR	LPIPS
w/o embeddings	0.7415	24.54	0.241	0.807	22.02	0.221
only $\mathbf{e}_p$	0.745	24.70	0.239	**0.815**	22.84	0.216
only $\mathbf{x}_{cam}$	0.748	24.84	0.239	0.808	22.09	0.219
only $\mathbf{x}_{rot}$	0.743	24.52	0.240	0.810	22.03	0.22
with $\mathbf{e}_p$ and $\mathbf{x}_{rot}$	0.746	24.69	0.239	0.813	22.748	0.215
with $\mathbf{x}_{cam}$ and $\mathbf{x}_{rot}$	0.748	24.75	0.240	0.813	22.58	0.216
with $\mathbf{e}_p$ and $\mathbf{x}_{cam}$	**0.751**	**25.02**	**0.237**	**0.815**	**22.85**	**0.215**
with $\mathbf{e}_p$, $\mathbf{x}_{cam}$ and $\mathbf{x}_{rot}$	0.749	25.01	0.238	0.813	22.84	0.216

Table 3: FeatSplat embbedings ablation on scenes *Bicycle* from Mip-360 [1] and *Train* from T&T [13]. We integrate in the final model only $\mathbf{e}_p$ and $\mathbf{x}_{cam}$.

The ablation results in Table 3 show that both $\mathbf{e}_p$ and $\mathbf{x}_{cam}$ have by themselves a positive impact on the metrics, while $\mathbf{x}_{cam}$ does not improve against not adding it. While $\mathbf{e}_p$ is responsible from almost all the improvement on *Train*, $\mathbf{x}_{cam}$ has a better performance on *Bicycle*, and both together achieve the best performance. On the other hand, $\mathbf{x}_{rot}$ alone does not improve the performance. Combining it with $\mathbf{x}_{cam}$ improves the performance of $\mathbf{x}_{cam}$ alone on *Train*, although less than $\mathbf{e}_p$. The results show that $\mathbf{x}_{rot}$ is redundant, with $\mathbf{e}_p$ and $\mathbf{x}_{cam}$ having the best performance.

	ScanNet++			
	SSIM	PSNR	LPIPS	FPS
HW GTS - RPBG	0.873	24.355	0.280	~ 1
FeatSplat–32 (Ours)	0.869	24.247	0.314	50
Nerfacto	0.861	24.049	0.342	0.3
TensoRF	0.849	23.978	0.407	–
3D Gaussian Splatting	0.871	23.891	0.319	155
iNGP	0.859	23.812	0.375	0.3

Table 4: ScanNet++ Novel View Synthesis Benchmark

B ScanNet++ Novel View Synthesis Benchmark

C Datasets

C.1 Mip360, T&T and DB.

The three typical datasets we use amount to a total of 12 scenes: Mip360[1], with 9 scenes, 5 outdoor at 1237×822 (*Bicycle*, *Flowers*, *Garden*, *Stump*, *Treehill*,) and 4 indoor at 1558×1039 (*Bonsai*, *Counter*, *Kitchen*, *Room*), with an average of 216 views per scene; Tanks and Temples [13] with 2 outdoor scenes at 979×546 (Train and Truck) and an average of 276 views per scene; and Deep Blending [7] with 2 indoor scenes (*DrJohnson* and *Playroom*) at 1264×832 and an average of 244 views per scene. In these datasets the testing set is selected sampling one every eight frames of the camera trajectory, using the other 7 frames to train.

C.2 ScanNet++

ScanNet++ is large-scale dataset with 460 indoor scenes and high-quality RGB and semantic annotations. It has been designed for evaluating high resolution novel view synthesis, with a resolution of undistorted images of 1752×1168 and a test set generated from an independent camera trajectory. The subset of used scenes has been randomly selected ensuring the fit in memory 24 gigabytes of vRAM in a single RTX 3090 GPU as Float-Point 32. As a consequence, the selected subset of scenes has an average of 408 views per scene, higher than the others datasets used. The evaluated scenes are: *0a5c013435*, *f07340dfea*, *7bc286c1b6*, *d2f44bf242*, *85251de7d1*, *0e75f3c4d9*, *98fe276aa8*, *7e7cd69a59*, *f3685d06a9*, *21d970d8de*, *8b5caf3398*, *ada5304e41*, *4c5c60fa76*, *ebc200e928*, *a5114ca13d*, *5942004064*, *1ada7a0617*, *f6659a3107*, *1a130d092a*, *80ffca8a48*, *08bbbdcc3d*,

C.3 Test and Train cameras plots

Figures 9, 8, 10 and 11 show the distribution of train and test cameras for all evaluated scenes.

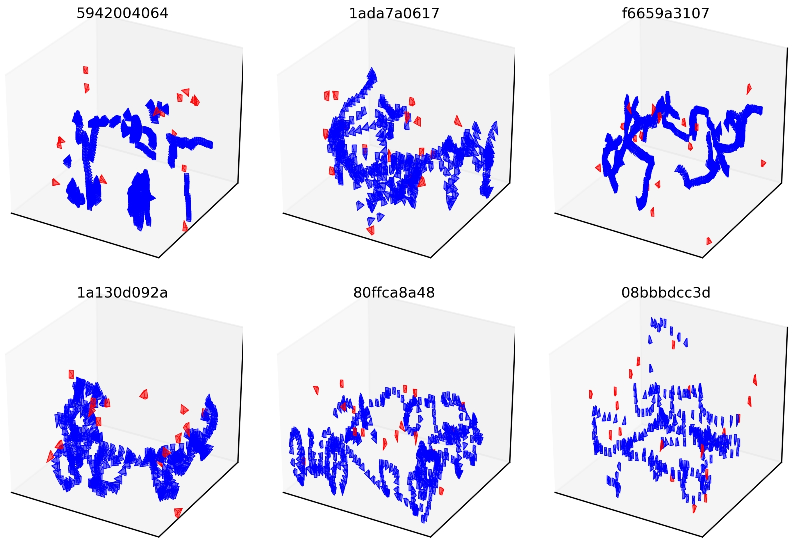

Figure 8: Train (blue) and test (red) and camera for ScanNet++

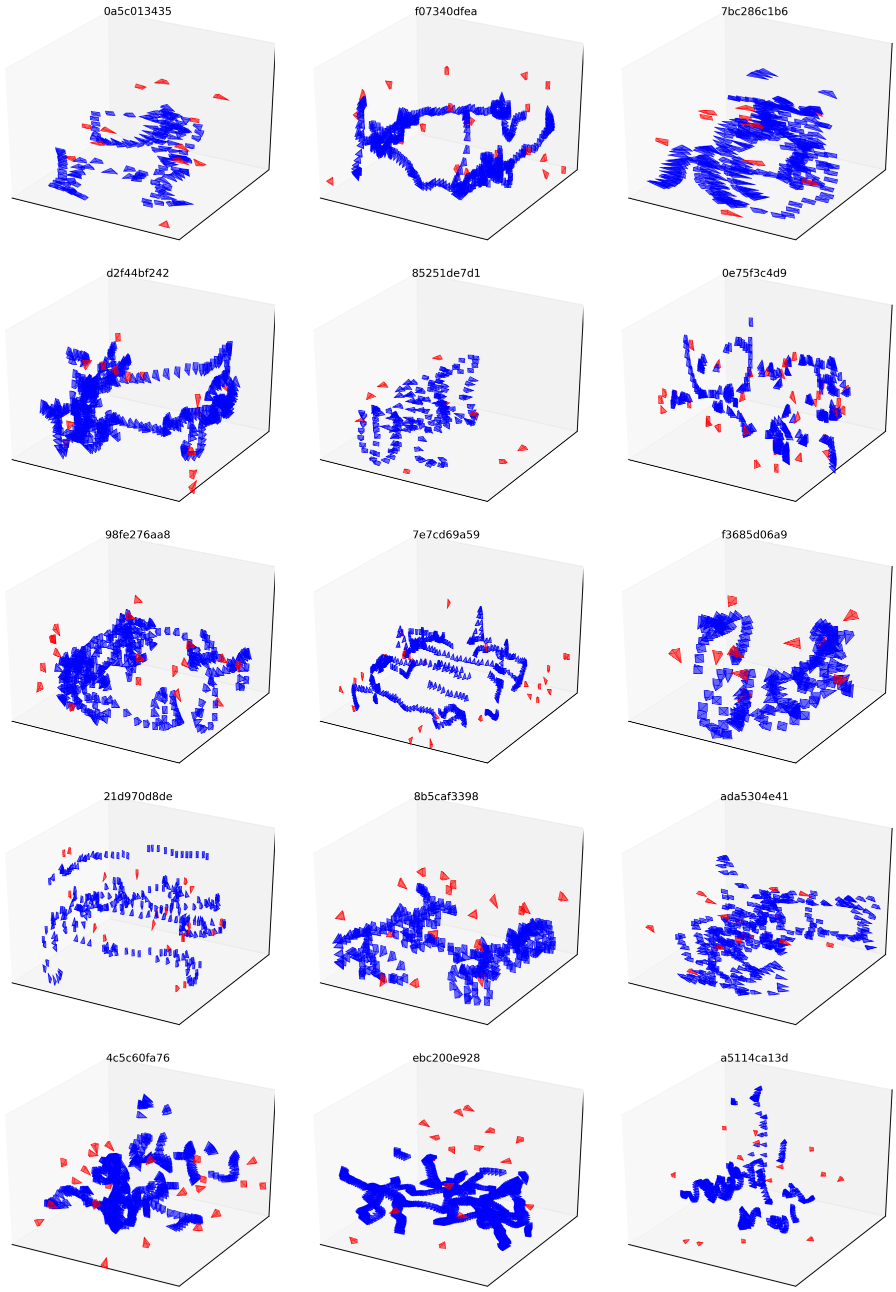

Figure 9: Train (blue) and test (red) and camera for ScanNet++

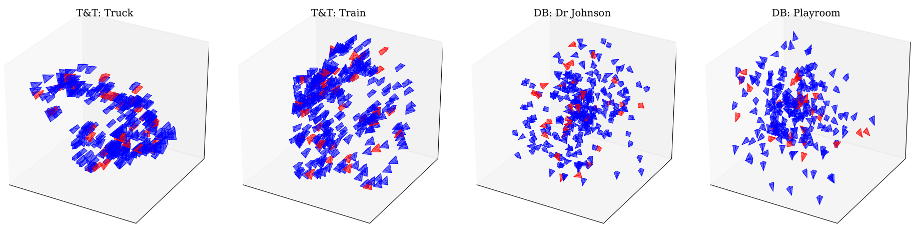

Figure 10: Train (blue) and test (red) and camera for T&T and DB

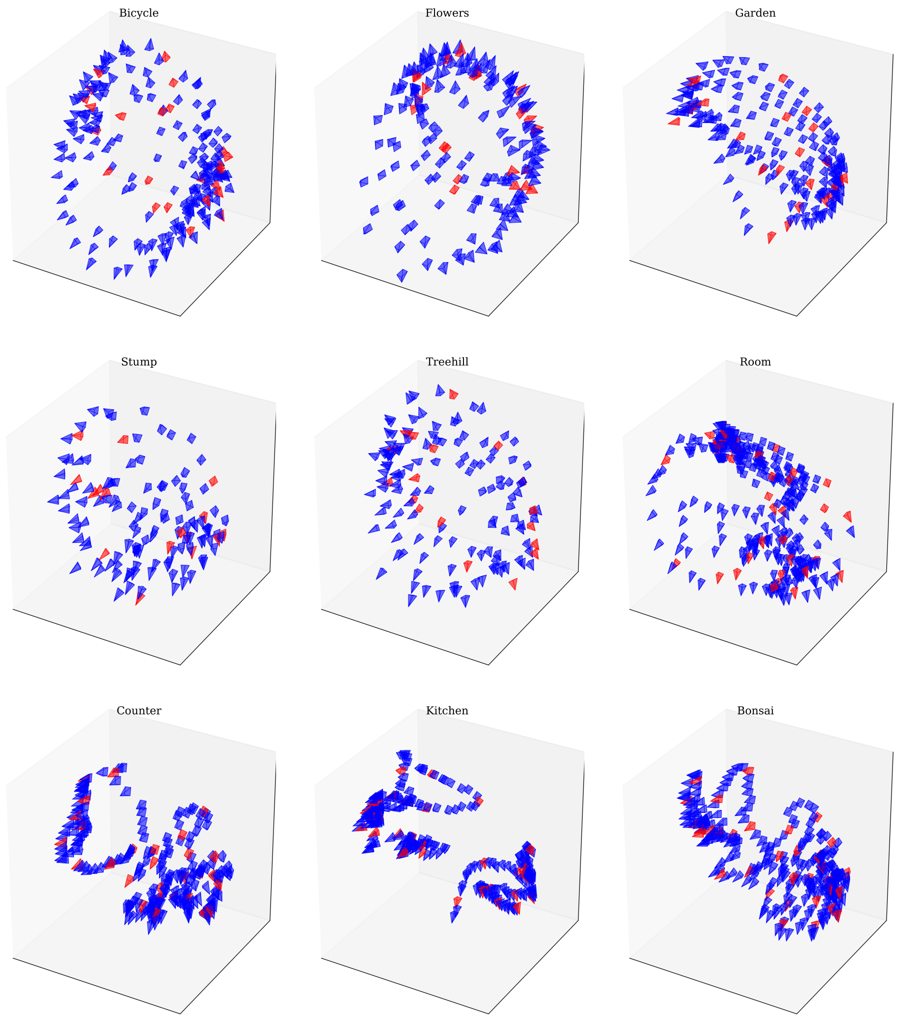

Figure 11: Train (blue) and test (red) and camera for Mip-360