Appendix of SimNP: Learning Self-Similarity Priors Between Neural Points

1. Content

In this supplementary material, we present ablation studies of our method **SimNP** in Section 2. Section 3 deals with additional point cloud supervision signals at test time. We make an argument about the quality of the PSNR metric in Section 4. Additional visualizations of learned symmetries (including a detailed description of how they are obtained) are shown in Section 6. Sections 7 and 8 elaborate on details regarding the evaluation and architecture. Finally, we provide additional qualitative results in Section 9.

2. Ablation Studies

Representing Category-Level Self-Similarities. In Tab. 4, we present the results of our ablation studies with respect to shared features, attention definition, and number of embeddings. We can observe that the shared features are essential to train a high-quality category-level neural point renderer. They are further investigated qualitatively in Section 5. We also test to obtain our matrix $\mathbf{A}$ via dot products between optimized keys (one per embedding) and queries (one per neural point) instead of directly optimizing $\mathbf{A}$, which leads to a marginal drop in performance. As this alternative formulation is more flexible in terms of the number of neural points though, the results indicate potential for an extension of SimNP from object to scene level. Finally, with an increasing number of embeddings, PSNR and SSIM decrease slightly in turn for improved LPIPS. We explain this behavior with the effect of blur on the different metrics, which we further investigate in Sec. 4. The smaller the number of embeddings, the smoother are the reconstructions, up to the same number of embeddings as neural points (512). Even more embeddings result in less information sharing and therefore worse generalization to novel views. In total, we can observe that except for existence of shared features, our approach is very robust to changes in these hyperparameters as the quality of the results differs only slightly.

Coherent Point Cloud Prediction. We ablate the encoder used for point cloud supervision at test time with respect to additional input data and data augmentations during training. The results are shown in Tab. 1. The ray encodings [1] as well as the segmentation mask individually decrease the 3D Chamfer distance between the ground-truth point clouds and the ones predicted for view 64 of each test example. Given the ray encodings, the encoder is less likely to confuse the pose of the object, *e.g.*, in case of almost

Rays	Mask	Aug	CD↓ ($\times 10^{-3}$)
✗	✗	✗	1.2103
✓	✗	✗	1.1977
✗	✓	✗	1.1634
✓	✓	✗	1.1434
✓	✓	✓	1.1028

Table 1: **Point cloud prediction ablation.** The 3D Chamfer distance is computed between the ground-truth point cloud and the one obtained by decoding the ResNet18 output for view 64 of all test examples. By using ray encodings [1] (Rays) and the segmentation mask (Mask) as additional inputs for the encoder, we can improve the point cloud prediction. Furthermore, random color jitter and grayscale augmentations (Aug) result in better generalization.

Method	PSNR↑	SSIM↑	LPIPS↓
PixelNeRF [2]	<u>23.17</u>	0.905	0.112
Ours	23.00	<u>0.911</u>	**0.081**
Ours + Blur	**23.31**	**0.913**	<u>0.092</u>

Table 2: **Effect of blur.** By applying a Gaussian blur on our rendered images, we can boost our PSNR results to outperform the strongest baseline with respect to this metric. However, this comes with worse results in LPIPS.

Supervision	PSNR↑	SSIM↑	LPIPS↓
Mask	23.00	0.911	0.081
Mask + Depth	23.28	0.915	0.079
Point Cloud	23.69	0.920	0.077

Table 3: **Point cloud supervision on ShapeNet cars.** Utilizing a depth map can partly bridge the gap between our purely 2D point cloud supervision and the use of ground-truth point clouds.

Configuration			1 Input View			1 Input View (Sym.)			2 Input Views		
S	A	M	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
✗	✓	512	22.37	0.898	0.106	22.02	0.894	0.114	25.66	0.937	0.069
✓	✗	512	22.76	0.908	0.084	22.47	0.905	0.086	26.38	0.945	0.057
✓	✓	128	**23.09**	**0.913**	0.083	**22.81**	**0.911**	**0.084**	**26.92**	**0.949**	0.058
✓	✓	256	23.06	0.912	0.082	22.76	0.910	**0.084**	26.84	**0.949**	0.055
✓	✓	512	23.00	0.911	**0.081**	22.67	0.908	**0.084**	26.67	0.948	**0.053**
✓	✓	1024	22.81	0.909	0.083	22.47	0.905	0.087	26.37	0.947	0.054

Table 4: **Ablation studies on ShapeNet cars.** S represents if shared features $\mathbf{S}$ are being used, A represents direct parameterization of the attention map $\mathbf{A}$ (✓) vs. the common calculation of attention with (shared) keys and queries (✗), and M is the number of embeddings. The gray row shows the configuration from the main paper.

symmetric front and back sides of cars. Due to the lighting used for rendering the dataset, we have observed that some white cars fade into the white background. Therefore, we attribute the improvements gained by leveraging the segmentation mask as input to such examples. Finally, we employ color jitter and grayscale data augmentations during training to enhance generalization.

3. Additional Point Cloud Supervision

By leveraging the autodecoder framework, we allow for flexible supervision at test time. Table 3 compares different forms of point cloud supervision. SimNP can effectively utilize additional depth maps or ground truth point clouds.

4. Effect of Blur

To support our claim that the state-of-the-art single-view PSNR results are due to the metric favoring blurry reconstructions, we postprocess our test predictions by applying a Gaussian filter with standard deviation 0.6. Table 2 shows that simply blurring our renderings is enough to raise the PSNR above the one of PixelNeRF [2]. Interestingly, SSIM is also affected positively, in contrast to LPIPS, which gets worse. We conclude that the two standard image quality metrics are rather biased towards blurry images such that we suggest to focus more on perceptual metrics like LPIPS.

5. Category-Level Template

In order to investigate the effect of the shared features, we set the instance-specific embeddings to zero. Figure 1 shows the templates learned by the shared features $\mathbf{S}$. Besides the general shape of a car for a given point cloud including details like side mirrors, the shared features also encode common textures like wheels, windows, and lights. Furthermore, by decoding random point cloud latent codes $\mathbf{z}$, we always obtain plausible point clouds indicating that point latent plus shared components learn a deformable category-level template, which can be filled with individual details by fitting embeddings $\mathbf{E}$ to observations.

6. Learned Symmetries

We visualize the attention scores for seven more embeddings in Fig. 2. To evaluate a pixel's radiance $\mathbf{c} \in \mathbb{R}^3$, the usual volume rendering formulation proposed by NeRF accumulates the radiance for K sample points $\{\mathbf{x}_i \in \mathbb{R}^3\}_{i=1}^K$ along the ray though the pixel as:

$$\mathbf{c} = \sum_{i=1}^{K} \tau_i (1 - \exp(-\sigma_i \Delta_i)) \mathbf{r}_i \tag{1}$$

$$\tau_i = \exp\left(- \sum_{j=1}^{i-1} \sigma_j \Delta_j \right) \tag{2}$$

$$\Delta_i = \|\mathbf{x}_i - \mathbf{x}_{i-1}\|_2 , \tag{3}$$

where σ_i and $\mathbf{r}_i$ are the density and radiance of sample $\mathbf{x}_i$. In order to obtain the influence of each neural point on the pixel, we simply replace the radiance $\mathbf{r}_i$ in Eq. 1 with the normalized inverse distances $\mathbf{w}_i \in \mathbb{R}^N$ between the sample point and each neural point neighbor:

$$\mathbf{w}_i[j] = \begin{cases} \frac{w(\mathbf{x}_i, \mathbf{p}_j)}{W}, & \text{if } j \in \mathcal{N}(\mathbf{x}_i) \\ 0, & \text{otherwise,} \end{cases} \tag{4}$$

where $\mathcal{N}$ is the k-nearest neural point function, $\mathbf{p}_j$ the coordinates of the j-th neural point, w the inverse point distance, and W the sum of these weights in the neighborhood, as defined in Section 3.2 of the paper. Once we have rendered the neural point weights $\mathbf{c} \in \mathbb{R}^N$ for each pixel, the influence $\mathbf{i} \in \mathbb{R}^M$ of each embedding can be obtained by multiplying the learned attention scores:

$$\mathbf{i} = \text{softmax}(\mathbf{A}) \cdot \mathbf{c}. \tag{5}$$

7. Evaluation Details

Evaluation of Render Time. Table 1 of the paper provides time measurements for rendering a single view. For

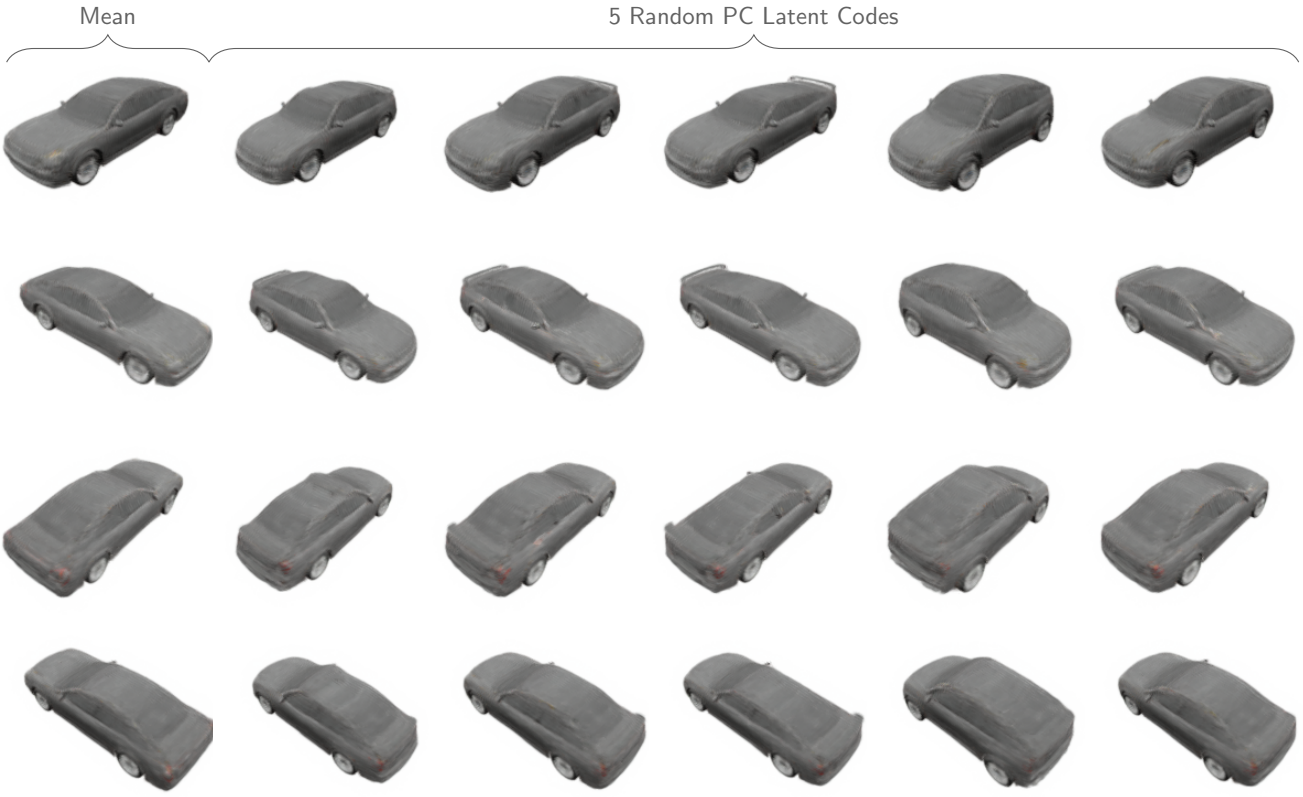

Figure 1: **Learned car template.** By rendering all-zero embeddings $\mathbf{E}$, we visualize the template learned by the shared features $\mathbf{S}$. The left column shows results for a zero point cloud latent code $\mathbf{z}$, whereas the remaining ones are obtained by sampling random vectors for point clouds.

a fair comparison, we separated the actual rendering functionality from all preceding inference steps in the code provided by the authors. For example, for VisionNeRF and PixelNeRF we only count the ray casting, feature sampling, and rendering MLPs as rendering, not the inference of feature maps. We evaluate each method on all 251 views of five random test examples and average the timings per view. None of the methods use any form of radiance caching or amortized rendering but render each view individually. The experiments were performed on a single RTX 8000 GPU.

Sym. Setup for Symmetric Views. Besides the render time, Table 1 of the paper also presents results for single-view reconstruction of views that show mostly the object side opposite to the input view. To be more precise, we choose the view index intervals 0-33, 74-112, and 152-191. These are all views with the camera being on the right side of the car up to a certain height, as the input view shows the object from the front left. Note that this subset also contains views of the rear, which are more challenging for our method because of missing similarities to observed areas.

8. Architecture Details

The architecture is composed of the point cloud prediction network, the attention representing the category-level symmetries, and the rendering network. For point cloud prediction, we use a four-layer MLP with hidden dimensions 256, 128, 64, and ReLU activation function. As input, it gets the latent code $\mathbf{z} \in \mathbb{R}^l$ with $l = 512$. The final layer outputs the coordinates of all $N = 512$ neural points inside a cube of side length 2 using the hyperbolic tangent.

The rendering network consists of the kernel K_θ and density and radiance function F_ψ. K_θ is implemented as a five-layer MLP with output dimension 256. F_ψ consists of two separate branches: another five-layer MLP for radiance prediction and a two-layer MLP for the density. All MLPs of the rendering network use 256 as the number of hidden dimensions and LeakyReLU as non-linearity.

9. Additional Qualitative Results

We present results for coherent point cloud prediction in Figure 3. These results are obtained using the **Ours** setup from the main paper. The point colors encode point identity

over multiple subjects. It can be seen that the resulting point clouds behave *coherent* such that individual points represent the same object parts over multiple instances.

Further, we present additional qualitative results for single-view reconstruction in Figure 4, two-view reconstruction in Figure 5, and interpolation in Figure 6.

References

[1] Daniel Watson, William Chan, Ricardo Martin Brualla, Jonathan Ho, Andrea Tagliasacchi, and Mohammad Norouzi. Novel view synthesis with diffusion models. In *Int. Conference on Learning Representations (ICLR)*, 2023.

[2] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

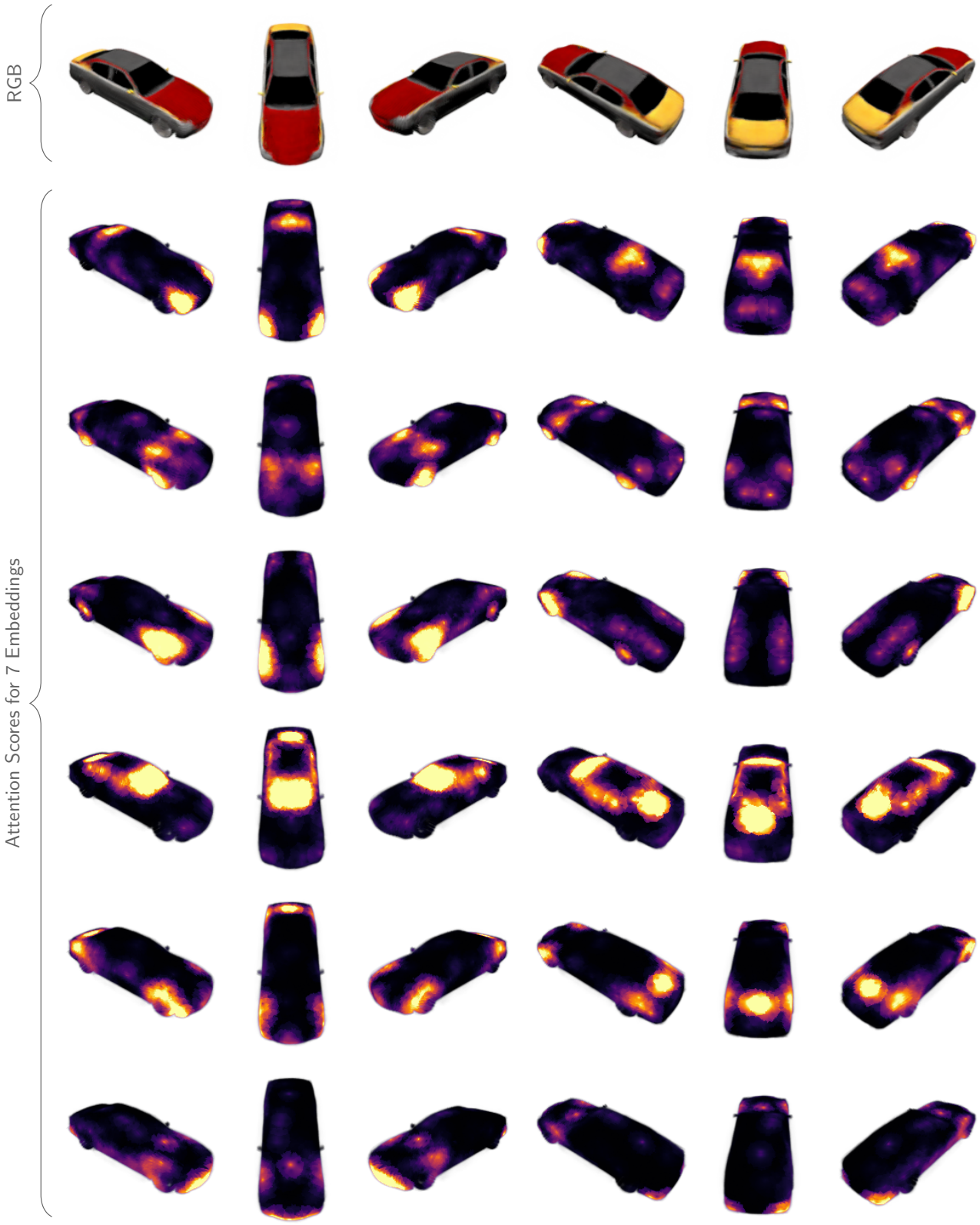

Figure 2: **Attention visualization.** We render the influence of seven different embeddings (one per row) on each ray.

Figure 3: **Coherent point cloud prediction.** The point color encodes the point identity given by the order of the output tensor that is predicted by the point MLP. It can be seen that these identities behave coherent over multiple instances. This allows the formulation of shared features $\mathbf{S}$. Also, these coherent identities provide correspondences between instances.

Figure 4: **Single-view reconstruction.** Additional qualitative results that show that our method is better in replicating details on the symmetric side of the object.

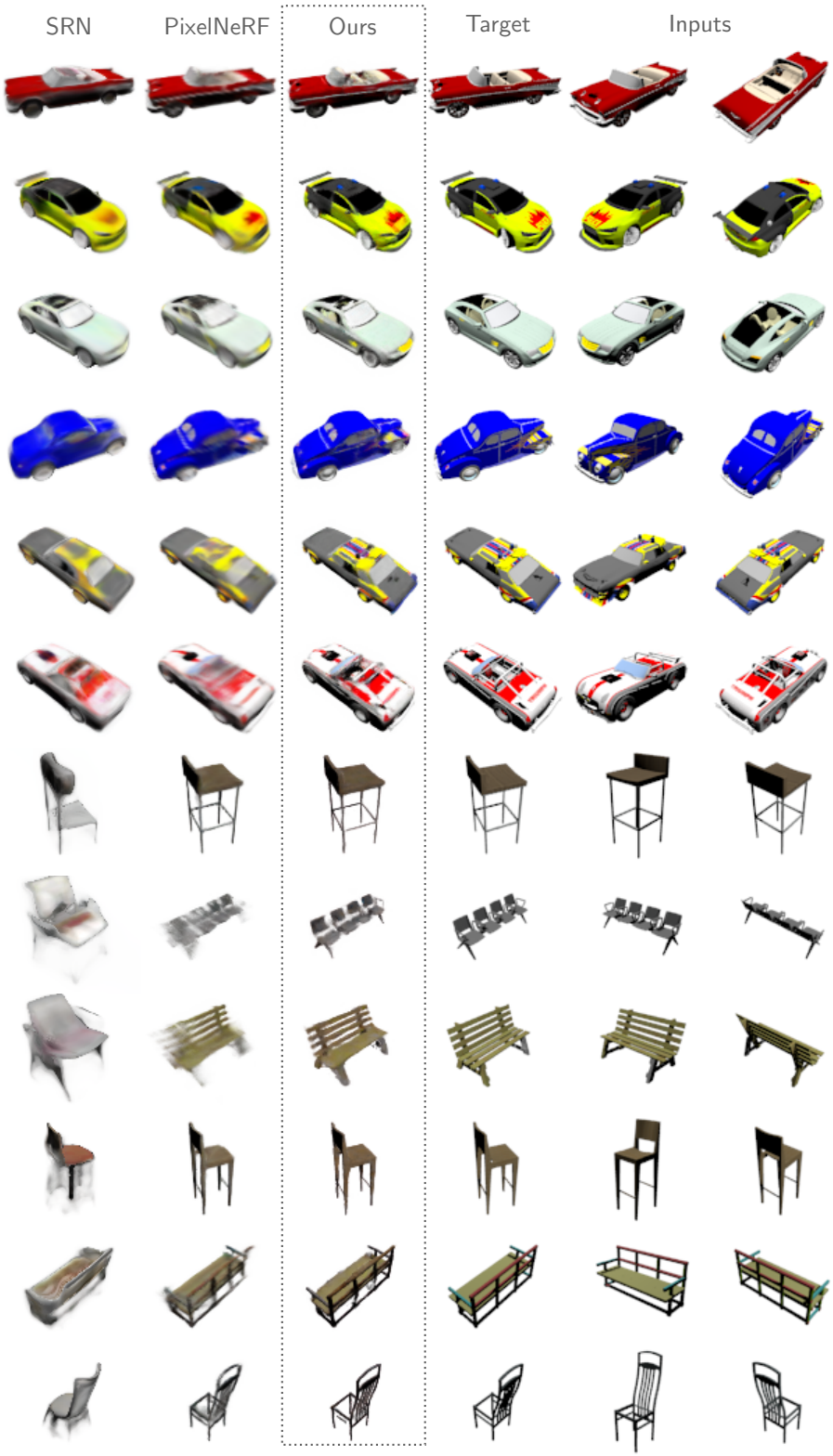

Figure 5: **Two-view reconstruction.** Additional qualitative results that show our method is better in representing highly detailed objects from just two views.

Figure 6: **Disentangled interpolation.** Additional qualitative results that highlight the ability of interpolating embeddings, point clouds and both together.