Control-NeRF: Editable Feature Volumes for Scene Rendering and Manipulation Supplementary Material

Verica Lazova[1,2]
verica.lazova@uni-tuebingen.de

Vladimir Guzov[1,2]
vladimir.guzov@mnf.uni-tuebingen.de

Kyle Olszewski[3]
kolszewski@snap.com

Sergey Tulyakov[3]
stulyakov@snap.com

Gerard Pons-Moll[1,2]
gerard.pons-moll@uni-tuebingen.de

[1]University of Tübingen [2]Max Planck Institute for Informatics, Saarland Informatics Campus [3]Snap Inc.

1. Training and Architecture Details

As noted in Sec. 3.1, we use a positional encoding function [8] on each dimension of the direction vector $\mathbf{d} \in \mathbb{R}^3$, to map this vector into a higher-dimensional space before passing it as input to our radiance function $F_\Theta : (\mathbf{v}_s, \gamma(\mathbf{d})) \to (\mathbf{c}, \sigma)$, where

$$\gamma(p) = \begin{pmatrix} \sin\left(2^0\pi p\right), \cos\left(2^0\pi p\right), \cdots, \\ \sin\left(2^{L-1}\pi p\right), \cos\left(2^{L-1}\pi p\right) \end{pmatrix}. \tag{1}$$

We use $L = 4$ in our experiments.

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{v}_s)\mathbf{c}(\mathbf{v}_s, \mathbf{d})dt\,, \tag{2}$$

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{v}_s)ds\right) \tag{3}$$

where $\mathbf{v}_s = \mathrm{S}(V_s, \mathbf{p})$ is the 64-dimensional sampled feature vector from the volume V_s for scene s at point $\mathbf{p} \in \mathbb{R}^3$, and S represents the trilinear sampling operation. The density values sampled from the network can thus be used to determine the probability of a ray terminating at the sampled point along the ray. In practice, following the example of [6], we use a discretized approximation of this integral, using a 2-stage process in which we optimize a coarse network $\hat{C}_c(\mathbf{r})$ that samples 64 points from evenly spaced bins along the ray length, followed by sampling these points plus another 64 points from our fine network $\hat{C}_f(\mathbf{r})$ using the coarse network opacity results to sample from more relevant portions of the scene volume (see Sec. 5.2 of [6]. For our experiments, the networks are trained using 1024 rays per batch sampled from the LLFF [5] multi-view image datasets, scaled to a resolution of 504×378. [1] The network architecture we use is overall based on that of [6], except that the input channels have been modified to accept our feature vector in place of the parameters representing the point to be sampled in the training scene. While they use a positionally encoded representation of each dimension in the the 3D position $\mathbf{p}$ sampled along the view ray (with $L = 10$, for a total of 60 parameters passed as input to represent this position in the scene as in Eq. 1), we pass the 64-dimensional feature vector sampled from the volume as described above into the network with no positional encoding.

[1]This differs slightly from the training parameters used in [6], as they use 4096 rays per batch sampled from 1008×752 images and an additional 128 samples from the fine network for training. See Sec. 3 of this document for comparisons to the results obtained when training both their and our network using the parameters described above.

2. Ablation

We performed an ablation study to evaluate the efficacy of the total variation loss and multi-resolution training techniques described earlier. We use the trained model to optimize the feature volumes for the *ferns* and *trex* scenes with and without the aforementioned techniques. We provide the per-scene results of these experiments both with and without the Total Variation loss and multi-resolution training described in Secs. 3.3.2 and 3.3.3, using the Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS) [9] metrics. As Table 1 shows, these results show that overall our final approach outperforms these less sophisticated alternatives per-scene in nearly all cases for each metric, and on average for all metrics. We also performed a second ablation to study how the generalization capabilities of our method are affected by the number of scenes used at training time. In our original experiments we train on 6 scenes (optimizing the rendering network parameters and the feature volumes simultaneously), and we only finetune on 2 scenes (optimizing only the feature volumes). The results are shown in Table 2, and as expected the more scenes are observed at training time the easier it is to generalize to unseen scenes.

PSNR↑

	Fern	T-Rex	Avg.
Our method	**25.752**	**26.510**	**26.131**
w\o Multiscale	24.789	25.617	25.203
w\o Total Variation (TV)	25.039	25.504	25.271
w\o Multiscale, w\o TV	22.193	19.296	20.745

SSIM↑

	Fern	T-Rex	Avg.
Our method	**0.820**	**0.907**	**0.864**
w\o Multiscale	0.793	0.878	0.835
w\o Total Variation (TV)	0.804	0.869	0.836
w\o Multiscale, w\o TV	0.704	0.691	0.698

LPIPS↓

	Fern	T-Rex	Avg.
Our method	0.236	**0.153**	**0.195**
w\o Multiscale	0.279	0.209	0.244
w\o Total Variation (TV)	**0.230**	0.221	0.226
w\o Multiscale, w\o TV	0.324	0.366	0.345

Table 1: **Per-scene quantitative ablation results**.

3. Additional Evaluations

To demonstrate the generalization capacity of our networks, we provide additional results showing interactive scene composition, editing and novel view synthesis on multiple datasets from different domains. For these results, we use the same network used for our previous experiments, trained on the 6 LLFF scenes previously described. The feature volumes for each subject are then optimized using the same approach as before.

We use images from the DeepVoxels [7] dataset, which contains multiple calibrated images of static 3D objects such as furniture. We use 479 images captured from the full 360° field around an object, at a resolution of 512×512. We also use a multi-view dataset set of similarly rendered images of textured 3D models of scanned human subjects from AXYZ Design [2] (25 primarly frontal images per subject, at a resolution of 512×512). The feature volume optimization takes approximately 6 hours per subject.

Num. Scenes	PSNR↑	SSIM↑	LPIPS↓
1	24.1	0.79	0.29
3	25.13	0.83	0.26
6	**26.13**	**0.86**	**0.19**

Table 2: **Quantitative results on generalization w/ varying number of training scenes**.

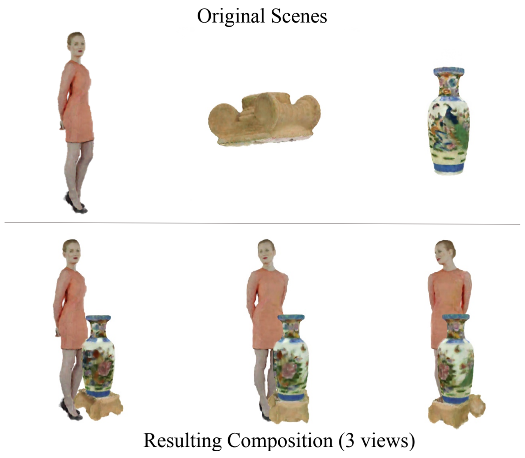

Figure 1: **Combining objects from various datasets**. Note that we use the rendering network trained on 6 scenes from LLFF [5] and finetune a volume for each object separately.

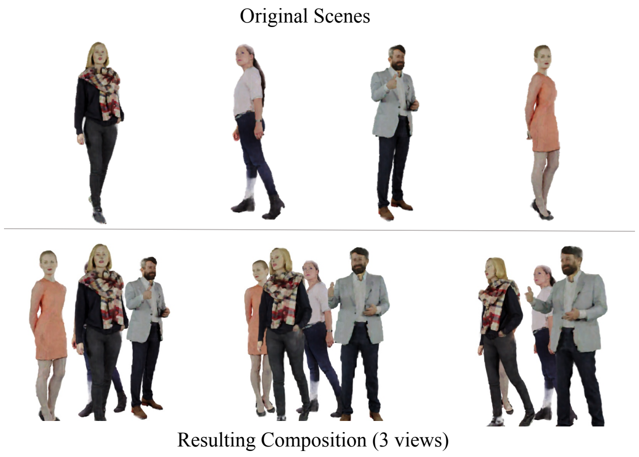

Figure 2: **Combining people from AXYZ [2] dataset**. Note that we use the rendering network trained on 6 scenes from LLFF [5] and finetune a volume for each subject separately.

Fig. 1 portrays the combination of the feature volumes for a vase and a stand with that of a human subject from the AXYZ dataset. We also we combine feature volumes for 4 human subjects (Fig. 2). Interestingly, despite the large difference in the appearance of the subjects in these datasets from the network training images, including a complete lack of humans in the LLFF images, and the relatively small number of scenes used for training the rendering network, the results are quite reasonable. This suggests that the initial network parameter training and feature volume optimization does indeed learn a disentangled representation that allows for a flexible approach to rendering novel content beyond that which is similar to what it has seen during training.

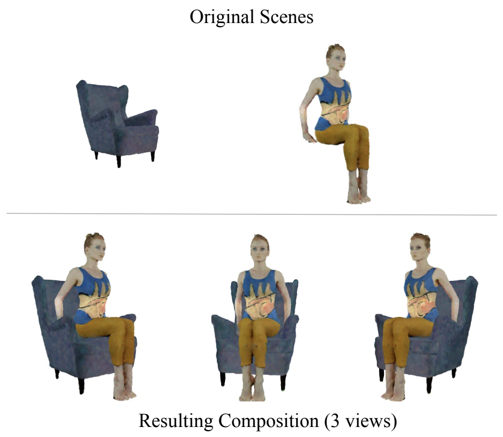

Figure 3: **Combining objects from various datasets**. Tight object fusion is achieved by choosing in each voxel the feature set with maximum L_2 norm among voxels of original scenes.

We also noticed that the color and density information carried by each voxel correlates with the L_2 norm of its feature vector. This allows us to fuse feature volumes by choosing feature vector with maximum L_2 norm among the voxels with the same coordinates from original scenes. This way, we can achieve tight contact between objects from different scenes without artifacts (Fig. 3).

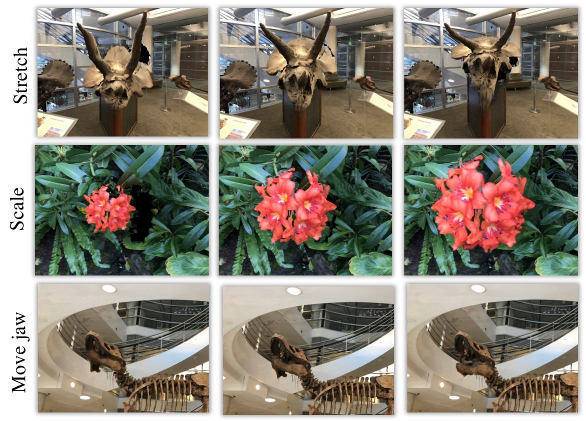

Figure 4: **Rigid and non-rigid transformations** of objects extracted from a scene. The middle column shows the original object/scene. Please zoom in and consult the supplementary video for further demonstrations.

3.1. User Study

We also conducted a user study to evaluate the scene manipulation capabilities of our method and Neural Point-Based Graphics (NPBG) [1]. Given 6 scenes and 10 pairs of edited images per scene (ours vs. NPBG), Amazon Mechanical Turk users were presented these pairs and asked to decide which image was preferable. 5 workers were asked per image pair, for a total of 300 questions asked (18 unique users participated). In 62% of the cases, users preferred our edited images.

	PSNR↑	SSIM↑	LPIPS↓
NeRF [6]	**28.045**	**0.881**	**0.137**
NeRF-multiscene	25.262	0.815	0.194
Ours	25.635	0.853	0.181

Table 3: **Quantitative comparison with NeRF [6]**. Metrics are averaged across test images for 8 scenes from from LLFF [5] dataset. Our method is trained on multiple scenes simultaneously while NeRF memorizes only one scene.

3.2. Scene Content Deformation

Our method allows for rigid and non-rigid transformations of objects by resampling the volume. Fig. 4 shows various rigid and non-rigid manipulations of objects extracted from these volumes and on entire scenes, obtained using the aforementioned volume deformation and resampling techniques. Please check the supplementary video for more animated results. Keep in mind that related methods such as NPBG [1] and NSVF [4] will struggle with this type of deformations. For NPBG this might require changing the density of the pointcloud which will affect the quality of the rendered results; And for NSVF, which relies on sparse voxels, it might require adding voxels to the empty regions, which is more difficult than pruning. While we show examples such as stretching and scaling specific scene content, the flexibility of our editing framework allows for arbitrary manipulations that can be specified as local or global modifications to a scene's feature volume.

3.3. More Comparisons

Multi-scene NeRF. In Table 3 we show a comparison to NeRF [6] on NVS. As NeRF is a scene-specific method, while our rendering network generalizes across scenes, we adapted it for a multi-scene scenario by associating each scene with a one-hot encoding vector. NeRF is then conditioned on this code in order to generate the specific scene. We have increased the capacity of the network accordingly to accommodate multiple scenes. While the original NeRF performs better than our method, we outperform NeRF in the multiscene scenario.

Single-scene NeRF. Below we provide further details on the comparisons with the original implementation of NeRF [6], using the training parameters described in Sec. 1 of this document on the LLFF dataset. For these compar-

PSNR↑

	Per-Scene, Training							Per-Scene, Novel			
	Room	Leaves	Fortress	Orchids	Flower	Horns	Avg.	Fern	T-Rex	Avg.	Total Avg.
NeRF [6]	**33.965**	**22.562**	**33.099**	**21.276**	**28.564**	**29.484**	**28.158**	**26.843**	**28.567**	**27.705**	**28.045**
Ours	30.938	18.438	28.930	21.182	27.526	25.807	25.470	25.752	26.510	26.131	25.635

SSIM↑

	Per-Scene, Training							Per-Scene, Novel			
	Room	Leaves	Fortress	Orchids	Flower	Horns	Avg.	Fern	T-Rex	Avg.	Total Avg.
NeRF [6]	**0.963**	**0.814**	**0.932**	0.744	**0.893**	**0.912**	**0.876**	**0.856**	**0.930**	**0.893**	**0.881**
Ours	0.943	0.770	0.861	**0.764**	0.883	0.875	0.849	0.820	0.907	0.864	0.853

LPIPS↓

	Per-Scene, Training							Per-Scene, Novel			
	Room	Leaves	Fortress	Orchids	Flower	Horns	Avg.	Fern	T-Rex	Avg.	Total Avg.
NeRF [6]	**0.093**	**0.186**	**0.068**	0.204	**0.110**	**0.133**	**0.132**	**0.168**	**0.137**	**0.152**	**0.137**
Ours	0.131	0.227	0.207	**0.178**	0.123	0.190	0.176	0.236	0.153	0.195	0.181

Table 4: **Per-scene quantitative results compared with the results of the original NeRF implementation**. For NeRF, the networks are trained per-scene until convergence. Ours, in contrast, uses a single rendering network trained for all scenes.

isons, we measure the difference between synthesized novel views and ground-truth images withheld for each scene during training, as in their evaluations. In our experiments, the total amount of computation time required to optimize the radiance function parameters for NeRF for a single scene until convergence, which required approximately 48 hours, or roughly 2 days. [2] Thus, training for all of the 8 scenes in the results depicted in Tab. 4, a total of approximately 16 days of computation was required (though this was performed in parallel on multiple systems) using an NVIDIA V100 GPU for each scene. In contrast, training our *single* rendering network on the set of 6 training scenes took a total of roughly 36 hours. This network can then be applied to novel scenes using our feature volume optimization process. For the novel scenes, an average of 5.5 hours (also performed in parallel on multiple systems) was required to compute their feature volumes, meaning that for these 8 scenes in total roughly 47 hours, or slightly less than 2 days of total computation was required. We further note that, while NeRF essentially memorizes a representation of the training scene that allows for a limited range of novel view synthesis, our approach additionally allows for the intuitive manipulation and combinination of data from multiple scenes, as demonstrated in our experimental results. Thus, given that computational efficiency in training a single network for multiple scenes networks is an advantage of our approach, we conduct additional evaluations in which we examine how well NeRF performs with similar computational resources. We trained the NeRF network for each of the 8 scenes for 100K iterations, or approximately 5.5 hours, which is comparable to the time required to run the novel scene feature volume optimization for a single scene using our approach. After this point, in our experiments the performance improved slowly until converging after approximately 2 days to the aforementioned results. The results are depicted in Tab. 5, with the results for Neural Point-Based Graphics (NPBG) [1] included for further comparison. As seen in these tables, while NeRF does produce results that are slightly more visually appealing when given unrestricted computational resources, when the training time is restricted the results are comparable, with ours outperforming each alternative on average in 2 out of 3 metrics. We also note that these 2 metrics, SSIM and LPIPS, are generally regarded to correspond better to realistic and more higher quality images for the human visual system [3].

[2]These numbers are slightly different than those reported in [6], but as noted above, we use a different training image resolution and number of samples in the fine network $\hat{C}_f(\mathbf{r})$ in our experiments. As pre-trained models for each of these scenes were not available for their implementation, we trained the models for each scene using the above parameters for a more direct comparison.

NPBG and NSVF. Additionally, in Table 6, we show more detailed comparisons against Neural Point-Based Graphics (NPBG) [1] and Neural Sparse Voxel Fields [4]. Please keep in mind that NSVF doesn't perform well on complex real scenes or on front-faced scenes where you don't see the object from all sides. This is mentioned by the authors in the paper. In order to deal with complex real scenes they rely on RGBD images as input. For this comparison both our method and NSVF are trained only on posed images without any depth information. NPBG requires a point-cloud as input alongside the posed images.

PSNR↑

	Per-Scene, Training							Per-Scene, Novel			
	Room	Leaves	Fortress	Orchids	Flower	Horns	Avg.	Fern	T-Rex	Avg.	Total Avg.
NPBG [1]	26.058	17.854	19.172	17.535	22.106	20.651	20.563	18.285	20.407	19.346	20.259
NeRF [6] 100K	**32.492**	**21.952**	**31.957**	21.172	27.478	**27.922**	**27.162**	**26.370**	**27.209**	**26.790**	**27.069**
Ours	30.938	18.438	28.930	**21.182**	**27.526**	25.807	25.470	25.752	26.510	26.131	25.635

SSIM↑

	Per-Scene, Training							Per-Scene, Novel			
	Room	Leaves	Fortress	Orchids	Flower	Horns	Avg.	Fern	T-Rex	Avg.	Total Avg.
NPBG [1]	0.890	0.668	0.804	0.572	0.769	0.794	0.750	0.706	0.774	0.740	0.747
NeRF [6] 100K	**0.953**	**0.776**	**0.908**	0.718	0.859	0.870	0.847	**0.830**	0.903	**0.867**	0.852
Ours	0.943	0.770	0.861	**0.764**	**0.883**	**0.875**	**0.849**	0.820	**0.907**	0.864	**0.853**

LPIPS↓

	Per-Scene, Training							Per-Scene, Novel			
	Room	Leaves	Fortress	Orchids	Flower	Horns	Avg.	Fern	T-Rex	Avg.	Total Avg.
NPBG [1]	0.163	0.272	0.200	0.301	0.204	0.222	0.227	0.267	0.231	0.249	0.232
NeRF [6] 100K	**0.126**	0.235	**0.107**	0.250	0.155	0.199	0.179	**0.212**	0.178	**0.195**	0.183
Ours	0.131	**0.227**	0.207	**0.178**	**0.123**	**0.190**	**0.176**	0.236	**0.153**	**0.195**	**0.181**

Table 5: **Full per-scene quantitative results**. We report the per-scene and average results on the initial training scenes as well as on the novel scenes used in our generalization process, as well as the average across both datasets.

PSNR ↑

	Fern	Leaves	Fortress	Orchids	Flower	Trex	Horns	Average
NSVF	20.594	17.316	26.901	14.309	22.930	17.467	23.380	20.414
NPBG	18.285	17.854	19.172	17.535	22.106	20.407	20.651	19.430
Ours	**25.752**	18.438	28.930	**21.182**	**27.526**	**26.510**	**25.807**	**24.878**
Ours (single scene)	25.082	**20.554**	**29.618**	20.374	26.260	24.753	25.425	24.581

SSIM ↑

	Fern	Leaves	Fortress	Orchids	Flower	Trex	Horns	Average
NSVF	0.575	0.402	0.721	0.250	0.629	0.490	0.682	0.536
NPBG	0.706	0.668	0.804	0.572	0.769	0.774	0.794	0.727
Ours	**0.820**	**0.770**	**0.861**	**0.764**	**0.883**	**0.907**	**0.875**	**0.840**
Ours (single scene)	0.792	0.738	0.854	0.704	0.840	0.868	0.834	0.804

LPIPS ↓

	Fern	Leaves	Fortress	Orchids	Flower	Trex	Horns	Average
NSVF	0.448	0.519	0.346	0.571	0.385	0.445	0.431	0.449
NPBG	0.267	0.272	0.200	0.301	0.204	0.231	0.222	0.242
Ours	**0.236**	**0.227**	0.207	**0.178**	**0.123**	**0.153**	**0.190**	**0.188**
Ours (single scene)	0.272	0.261	**0.198**	0.245	0.173	0.195	0.238	0.226

Table 6: **Quantitative comparison with NPBG [1] and NSVF [4]**. Metrics are computed across test images for scenes from from LLFF [5] dataset. "Ours" is our method trained on 6 scenes simultaneously as in our original setup. "Ours (single scene)" is our method trained for one scene at a time.

References

[1] Kara-Ali Aliev, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. *arXiv preprint arXiv:1906.08240*, 2(3):4, 2019. 3, 4, 5

[2] AXYZ DESIGN. AXYZ Design. 2

[3] A. Horé and D. Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th International Conference on Pattern Recognition*, pages 2366–2369, 2010. 4

[4] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020. 3, 4, 5

[5] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019. 1, 2, 3, 5

[6] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, pages 405–421. Springer, 2020. 1, 3, 4, 5

[7] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2019. 2

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 1

[9] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018. 1