AutoRF: Learning 3D Object Radiance Fields from Single View Observations

Norman Müller[1,3] Andrea Simonelli[2,3] Lorenzo Porzi[3] Samuel Rota Bulò[3]
Matthias Nießner[1] Peter Kontschieder[3]

Technical University of Munich[1] University of Trento[2] Meta Reality Labs Zurich[3]

Appendix

A. Implementation details

Encoder. Our encoder is based on a ResNet34 backbone where we replace all BatchNorm layers with InstanceNorm layers to support batch size of 1. The first four layers of this architecture are shared while the following two layers are replicated to form separate heads for shape and appearance encoding. For a $3 \times H \times W$ image, input for each encoding head is a feature map of shape $256 \times H \,/\, 16 \times W \,/\, 16$. These feature maps are passed through the individual heads and and adaptive max pooling is applied to obtain shape and appearance codes, each of dimension 128. We rescale the input images to a maximum of 320px in each dimension while preserving the aspect ratio.

Shape decoder. The shape decoder is a MLP that is made of 5 ResNet blocks with hidden dimension 128. At each layer, we feed the previous feature map and the positional encoding of the query points. In order to match the dimensionality of the positional encoding with the hidden dimensions of the MLP, we apply a single linear layer and aggregate the output with the intermediate feature maps by a simple per-channel mean pooling.

Color decoder. For decoding the color, we use a similar architecture as for the shape decoder: A MLP of 5 ResNet blocks with hidden dimensionality of 128 and additional linear aggregations for additional input: As for the shape decoder, we aggregate intermediate features with positional encodings by mean pooling. Furthermore, on the third layer we pass in the same way the output of the corresponding layer of the shape encoder. This enables the color decoder to incorporate estimated shape information. On the final two layers, we pass the view direction (encoded as 3-dimensional vector) to account for view-dependent effects.

Volumetric rendering. For each ray passing through the unit cube in the normalized object space, we compute the intersection segment and uniformly sample 64 points on this segment. During training, we randomly sample 1024 rays per input image and fix the rendering resolution to 80×120px. For the spatial coordinates, we use positional encoding from NeRF with 6 frequencies. At test time, we render each sample at a fixed resolution of 64×64px.

Hyperparameters. We train at a batch size of 1 and use the Adam optimizer with a learning of $10^{-}5$. For test-time optimization, we optimize shape, appearance and camera position using the Adam optimizer at learning rates 0.05, 0.02 and 0.02, respectively, for 32 iterations. We notice that a higher learning rate for color enables the AutoRF to focus on mainly adjusting color values while performing only slight modifications on shape and pose. This way, the optimization eschews strong overfitting to the input view and does not deviate too much from the learnt shape and color code manifold.

B. Additional ablation results

B.1. Auto-decoder variant of AutoRF

In order to better understand the role of the encoder we perform further ablation studies by using AutoRF in an auto-decoder fashion. To do so, we remove the encoder and only optimize the shape and appearance codes. The initialization of the codes is given by the averages computed on the training set. We optimize the auto-decoder version of our method (AutoRF AutoDecoder) for 128 rounds. The results of these ablations can be found in Tab. 1.

nuScenes cars	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓
AutoRF AutoDecoder on test	18.77	0.485	0.231	149.74
AutoRF AutoDecoder on train	18.81	0.487	0.228	134.91
AutoRF on test	18.94	0.491	0.223	145.10
AutoRF on train	**18.95**	**0.493**	**0.210**	**106.50**

Table 1. Comparison between AutoRF and its auto-decoder variant (AutoRF AutoDecoder) on nuScenes cars.

We observe that AutoRF AutoDecoder underperforms on all metrics in comparison with AutoRF. This validates the choice of having an encoder inside the architecture. Furthermore, we observe that the speed of convergence in AutoRF is much faster than its auto-decoder variant. This might be related to the fact that the encoder provides, in a single-shot, an already good initial estimate of the codes.

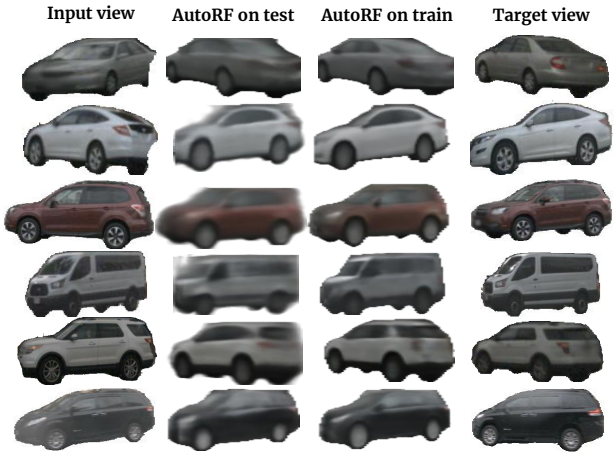

Figure 1. Comparison of AutoRF trained on nuScenes test images with machine-generated annotations and AutoRF trained on nuScenes train images with ground-truth annotations.

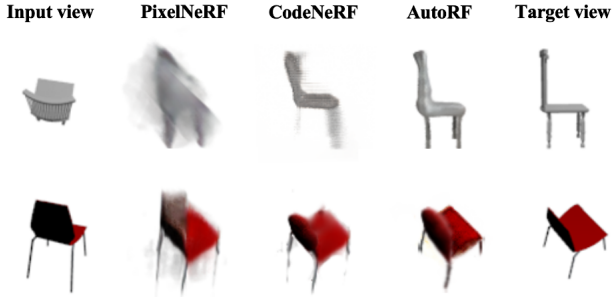

Figure 2. Qualitative comparison on SRN-chairs trained on single views.

SRN-Chairs	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓
pixelNeRF [41]	17.73	0.726	0.218	162.9
CodeNeRF [15]	18.14	0.763	0.187	137.6
AutoRF (no opt.)	18.08	0.761	0.180	134.3
AutoRF	**18.64**	**0.803**	**0.148**	**133.2**

Table 2. Evaluation of novel-view synthesis on the SRN-Chairs dataset from [36].

B.2. Evaluation on other categories

Tab. 2 and Fig. 9, we demonstrate qualitatively and quantitatively that AutoRF performs well also on indoor classes, providing results for SRN-chairs. We follow the same protocol as described in Section 4.2). We notice that pixelNeRF tends to produce more fuzzy radiance fields compared to ours.

B.3. Shape reconstruction

Furthermore, we provide qualitative results of our shape reconstructions in Figure 3. For this, we create 40 novel views of the same instance with a camera orbiting around and facing the object center and apply TSDF-fusion on the resulting pairs of depth and color images. We observe consistent depth and color images enabling creating of accurate meshes.

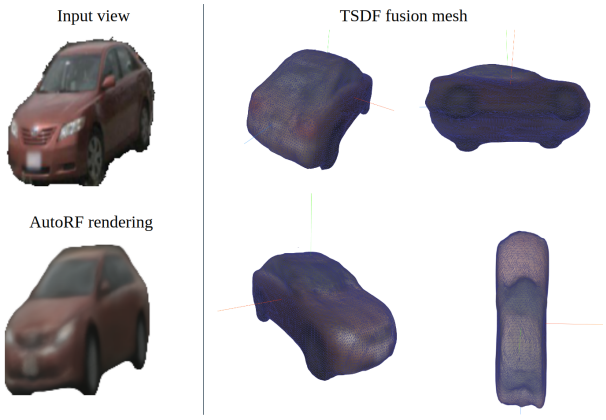

Figure 3. Explicit meshing: Given a single input view, we render depth and color from a multiple views using AutoRF and apply TSDF-fusion in order to create a 3D mesh.

B.4. Run time

In this section, we provide further details related to the run time of AutoRF. We observe that the rendering of a 64×64 image takes approximately 0.23 seconds. Regarding test-time optimization, we observed that the rendering of a 32×32 image takes approximately 0.11 seconds. Overall the test optimization, which is usually made of 32 steps, takes approximately 3.3 seconds per object.

C. Further details on the creation of nuScenes novel view data

We train and evaluate on nuScenes, a data set consisting of 168k training images, 36 validation images, and 36k test images. After filtering for daytime scenes with dry weather, we run the pre-trained 2D panoptic segmentation model [32] to obtain segmentation masks for all remaining images.

For the training and validation data, we match provided 3D bounding box annotations with the instance masks resulting from the panoptic segmentation. For the test data, we first run the 3D monocular detection model from [35], filter for detections with a score above 0.7, and match the resulting 3D annotations. We classify each pixel into instance foreground, background, or unknown region based on their semantic mask. As we do not leverage depth information, we rely on a simple heuristic: Semantic classes that cannot occlude any foreground instance (street, sky, sidewalk, manhole, crosswalks) are considered background while others are considered "unknown" if the do not belong to the object in focus. For those pixels, we do not compute any loss

and exclude them during optimization. For the generation of the validation set, we leverage the tracking information provided in the ground-truth annotations of nuScenes in order to create image pairs of the same instance at different time steps. We then filter for sufficiently visible instances: we only consider instances where the segmentation mask occupies at least 60% of the instance 2D bounding box, the instances are no further than 40m distant to the input camera and the overall resolution has to be at least 40px. Based on the remaining images, we randomly select 10k pairs for novel view evaluation.

D. Analysis of data quality

In this section, we further discuss the impact of having machine-generated annotations as opposed to manually annotated ones. An initial discussion with quantitative evaluations can be found in the main paper in Sec. 4.3 and Tab. 4. Here we provide qualitative results in Fig. 1, where it can be seen that results on test (second column) experience a slight increase of blurriness with respect to the ones on train (third column). It is important to note that, despite the limited decrease in sharpness, AutoRF is able to reliably synthesize novel views on never-seen objects present in the validation data. This is clear from the results shown in e.g. the first row, where AutoRF can recover a plausible car back having observed only its front.

E. Additional qualitative results

In this section we provide further qualitative results, aimed at highlighting AutoRF's ability to effectively provide meaningful object representations.

E.1. Novel-view synthesis

The natural output of AutoRF is the rendering of the input image in its original input view. A more interesting output is instead represented by the rendering of the input image from a novel (never-seen) view. Examples of such novel-view synthesis can be found in Fig. 6 and Fig. 7. It must be noted that in our experiments, AutoRF is focused on learning car representations, so the background, as well as additional objects, are not included into the novel view synthesis.

E.2. Code interpolation

AutoRF's explicit disentanglement of the shape and appearance allows to synthesize novel objects by performing a trivial interpolation of the codes. As an example, the appearance code of a red car can be interpolated to the one of a silver van. This results in the synthesis of a novel object smoothly transitioning it's color between red and silver. Interestingly, the same interpolation can be performed on the shape property of the object. Examples of such novel-view synthesis can be found in Fig. 4.

E.3. Scene editing

Novel-view synthesis covers the ability of synthesizing a static scene from novel camera poses. Another interesting ability is to synthesize novel scenes by arbitrarily changing object poses, properties as well as camera poses. Examples of such novel-view synthesis can be found in Fig. 5.

Figure 4. Color interpolation: starting from the properties of the objects in the input view (top-left), AutoRF is able to modify the properties of each object in the scene.

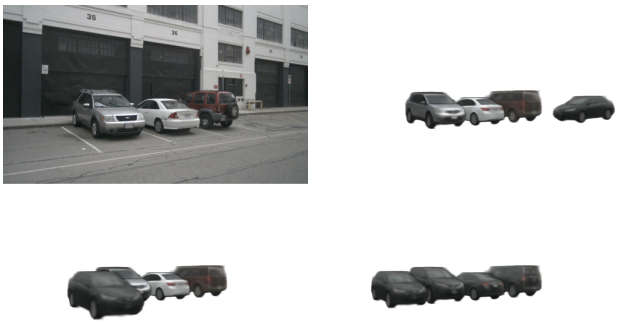

Figure 5. Scene editing: starting from the objects in the input view (top-left), AutoRF is able to apply arbitrary modifications and also include novel objects. Results on an unseen image of the nuScenes dataset.

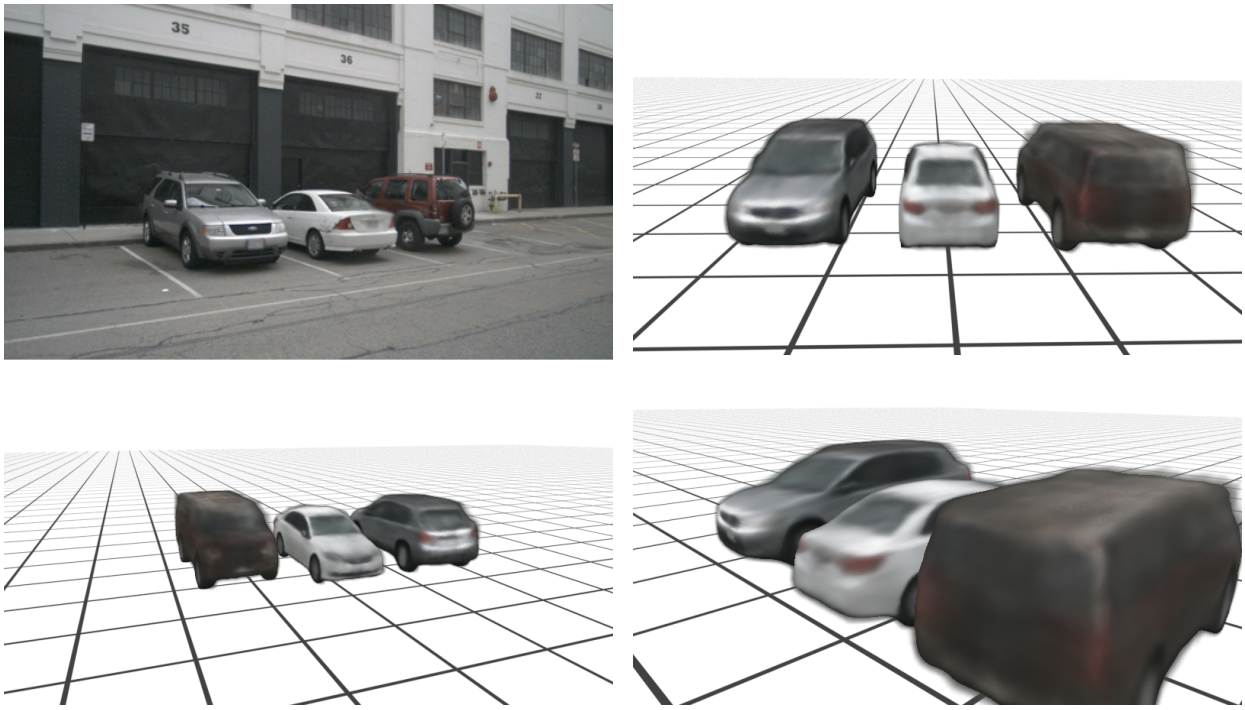

Figure 6. Novel-view synthesis: by only observing the input view (top-left), AutoRF is able to synthesize the objects in novel views.

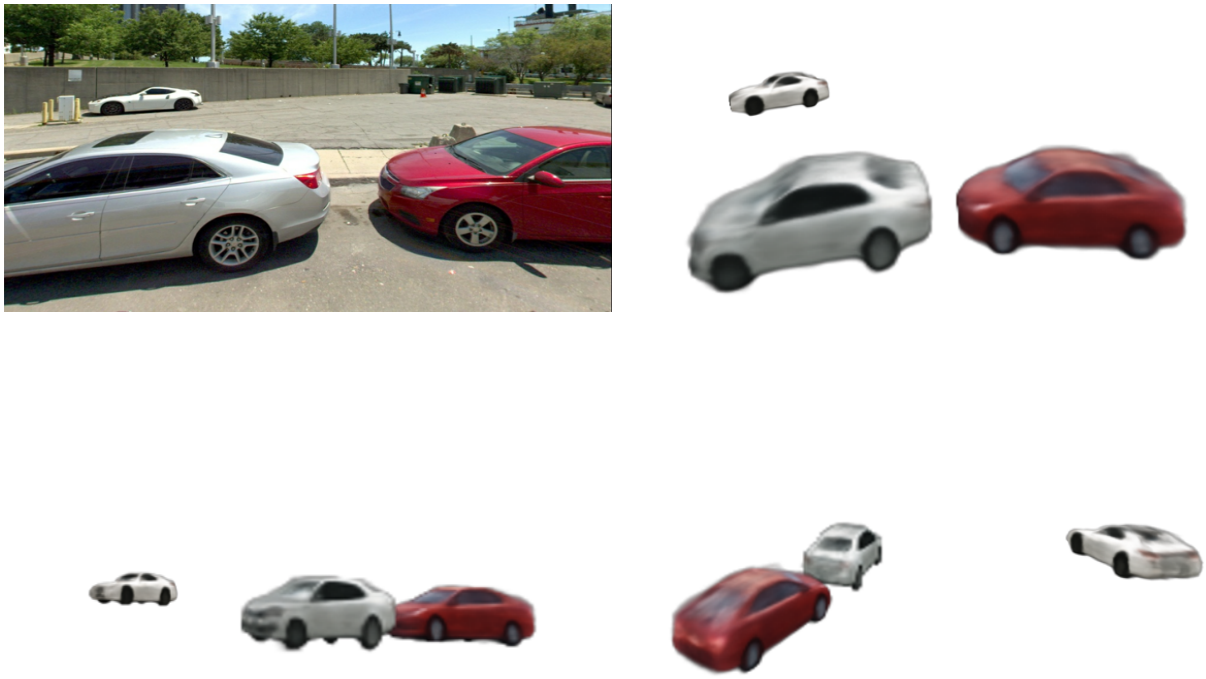

Figure 7. Novel view synthesis: further results on the unseen Mapillary Metropolis dataset. The model is solely trained on nuScenes test images with different camera setting.