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

 Norman Müller1,3 | Andrea Simonelli2,3 | Lorenzo Porzi3 | Samuel Rota Bulò3
 Matthias Nießner1 | Peter Kontschieder3
 Technical University of Munich1 | University of Trento2 | Meta Reality Labs Zurich3

3D detector, Panoptic

Training

Encoder

Shape code

Neural RF

Decoder

Appearance code

Inference

3D detector, Panoptic

Single input image

Synthetic novel views

Figure 1. Overview of AutoRF. Our model consists of an encoder that extracts a shape and an appearance code from an object’s image, which can be decoded into an implicit radiance field operating in normalized object space and leveraged for novel view synthesis. Object images are generated from real-world imagery by leveraging machine-generated 3D object detections and panoptic segmentation. At test time, we fit objects to respective target instances using a photo-metric loss formulation.

Abstract

We introduce AutoRF – a new approach for learning neural 3D object representations where each object in the training set is observed by only a single view. This setting is in stark contrast to the majority of existing works that leverage multiple views of the same object, employ explicit priors during training, or require pixel-perfect annotations. To address this challenging setting, we propose to learn a normalized, object-centric representation whose embedding describes and disentangles shape, appearance, and pose. Each encoding provides well-generalizable, compact information about the object of interest, which is decoded in a single-shot into a new target view, thus enabling novel view synthesis. We further improve the reconstruction quality by optimizing shape and appearance codes at test time by fitting the representation tightly to the input image. In a series of experiments, we show that our method generalizes well to unseen objects, even across different datasets of challenging real-world street scenes such as nuScenes, KITTI, and Mapillary Metropolis. Additional results can be found on our project page https://sirwyver.github.io/AutoRF/.

1. Introduction

In this work, we address the challenging problem of inferring 3D object information from individual images taken in the wild. Providing an objects’ 3D shape with 6DOF pose and corresponding appearance from a single image is key to enabling immersive experiences in AR/VR, or in robotics to decompose a scene into relevant objects for subsequent interaction. The underlying research problem is related to novel view synthesis or inverse graphics, and has recently gained a lot of attraction in our community [10,12,17,21,34,39,42,43], leading to remarkable improvements in terms of monocular 3D reconstruction fidelity.

Many existing works [10, 17, 21, 34, 42, 43] are limited in their applicability – in particular due to their imposed data and supervision requirements: The majority of works require multiple views and non-occluded visibility of the same physical object, near-perfect camera pose information, and the object of interest being central, at high resolution, and thus the most prominent content of the image. Due to a lack of real-world datasets providing such features (with the notable, recently released exception of [33]), an overwhelming number of methods have only shown experimental results on synthetic datasets, and thus under perfect data conditions, or require large datasets of CAD models to construct a shape prior. When applied to real data the existing domain gap becomes evident, typically leading to major performance degradation.

Our work investigates the limits of novel view synthesis from monocular, single-image real-world data. We focus on street-level imagery where objects like cars have high variability in scale and can be very small compared to the full image resolution. Also, such objects are often

Work was done during Norman’s and Andrea’s internships at Meta Reality Labs Zurich.
occluded or may suffer from motion blur as a consequence of the data acquisition setup. We only consider a single-view object scenario during both training and inference, i.e., we do not impose constraints based on multiple views of the same object. For supervision, we limit our method to learning only from machine-generated predictions, leveraging state-of-the-art and off-the-shelf, image-based 3D object detection [20, 35] and instance/panoptic segmentation algorithms [11, 18, 32], respectively. This data setting also enables us to benchmark our results on existing autonomous driving research datasets [2, 7, 28]. However, the absence of human quality control requires our method to cope with label noise introduced by machine-predictions from monocular 3D object detectors (itself addressing an ill-posed problem), and imperfect instance segmentation masks.

Our proposed method follows an encoder/decoder architecture trained on images with machine-predicted 3D bounding boxes and corresponding 2D panoptic segmentation masks per image. The encoder learns to transform a training sample from its actual (arbitrary) pose and scale representation into two canonical, object-centric encodings representing shape and appearance, respectively. The decoder translates the object’s shape and appearance codes into an object-centric, implicit radiance field representation, which provides occupancy and color information for given 3D points and viewing directions in object space. Our training procedure benefits from the segmentation mask to gather information about the object’s foreground pixels and to cope with potential occlusions, while it leverages the pose information provided by the 3D bounding box to enforce the object-centric representation. At test time, we further optimize predicted latent codes to fit the representation tightly to the given input image by using a photometric loss formulation. Ultimately, our architecture can learn strong implicit priors that also generalize across different datasets. We provide insightful experimental evaluations and ablation studies on challenging real-world and controllable synthetic datasets, which has not been demonstrated so far. This enables corrections from potential domain shifts and to generalise across different datasets.

2. Related works

3D reconstruction from a single image. The task of extracting 3D information from a single image, also known as “inverse graphics” has received considerable attention in recent years. Several works focus on reconstructing the shape, or the shape and appearance of a single object per image [12, 17, 34, 39], while others attempt to extract multiple objects per image [6, 10, 21, 43] or to build a holistic representation of an entire scene [5, 42]. All of these approaches use differentiable rendering to formulate a reconstruction cost to compare the predicted 3D model to the 2D image while differing in the specific representation used to encode the 3D model. Common choices here include 3D meshes [10, 12, 17], signed distance functions [6, 21, 27], depth [39], and implicit models [34, 42, 43]. The last option, i.e., implicit models, will be the focus of the next section, and is the one we adopt in our work.

Most of these approaches exploit some form of shape prior, either learned in the form of an implicit model [42, 43], or constructed from some collection of template shapes [6, 21]. Either way, they utilize large libraries of CAD models to bootstrap their networks or as the sole form of training data, incurring a considerable domain gap when moving to real images. Similarly, multiple views of the same object are generally required at training time, further justifying the use of synthetic data. In contrast, our method can be trained with a single view per object, and entirely on real images. Among those mentioned above, the only works that overcome these limitations are those of Henderson et al. [12] and Wu et al. [39]. These, however, utilize high-resolution, unobstructed, and generally clean views of the objects of interest (e.g., well-lit frontal shots of faces [39]). In contrast, our method is trained and validated on occluded and/or low-resolution images.

Differentiable rendering and implicit models. A common denominator of many inverse graphics formulations is the use of “differentiable rendering”. Differentiable rendering encompasses a large array of techniques to produce 2D views of some 3D model, while also allowing for the back-propagation of gradients from the image domain to the model’s parameters. The earliest such formulations included forms of “differentiable rasterization” [3, 22, 24]; i.e., back-propagable extensions of traditional rendering algo-
rithms for (textured) 3D meshes. More recently, approaches based on volumetric rendering have found great success, in particular when coupled with so-called “implicit models” [4, 23, 26, 29, 36, 41]. Implicit models represent objects or scenes as functions (i.e., neural networks), which map from 3D points to some local set of properties of the entity being modeled, e.g., whether the entity occupies the given point [25]. While some approaches focus solely on encoding shape [9, 16, 25, 31], others [23, 26, 29, 36, 41] have shown success in encoding shape and appearance together, meaning that i) they can produce photo-realistic renderings of the object they encode; and ii) can be trained using only 2D images. In our work we follow the Neural Radiance Fields (NeRF) formulation of Mildenhall et al. [26], which belongs to the latter category. Differently from NeRF, our model can generalize across multiple objects, and can synthesize novel views of an object given a single input image. The capability of single-view, novel view synthesis has investigated before with Scene Representation Networks [36], ShaRF [19] and pixelNeRF of Yu et al. [41]. In [36], a scene presentation is learnt that can be fine-tuned to a test scene with given camera poses. In ShaRF, a shape generation network is pretrained on synthetic data and optimized via Generative Latent Optimization [1], whereas in this work, we do not use any geometric information. As pixelNeRF leverages local image features to synthesis novel views, it is (in contrast to ours) trained on at least two views of the same instance. In the recent work FiG-NeRF [40], Xie et al. introduce a 2-component, deformable neural radiance field for jointly modeling object categories and a foreground/background segmentation. In [30], Ost et al. learn a scene graph to represent automotive data enabling novel views. However, their method requires video data and remains in the scenario of over-fitting the model to single scenes. The latter limitation is addressed in [33] and [13], where large-scale posed video data is leveraged to learn object-category specific generalization for novel view rendering. In contrast to all these methods, we use solely a single view of each object instance, allowing us to leverage large-scale, unstructured data at training time. Related to ours, CodeNeRF [15] leverages a neural radiance field that learns to disentangle latent embeddings for shape and appearance in an object-specific way. However, since CodeNeRF works as an auto-decoder architecture, it requires optimizing shape and appearance codes at test time to photometrically match a given input image before novel view synthesis can be run. AutoRF does not have this limitation, because it can regress shape and appearance codes directly via the encoder, yet offering the possibility of refining the object encoding at test-time. We finally refer to [37] for a survey about recent advances in neural rendering.

3. Method

Given a single input image, our goal is to encode each 3D object that is present in the scene into a compact representation that allows to, e.g., efficiently store the objects into a database and re-synthesize them from different views/contexts in a later stage. While this problem has been already addressed in the past [15, 41], we focus on a more challenging scenario when it comes to training such an encoder. As opposed to the vast majority of methods that assume to have access to at least a second view of the same object instance when training such an encoder, we focus on addressing the more challenging setting where object instances can be observed from a single view only. Moreover, no other prior knowledge about the object’s geometry (e.g., CAD models, symmetries, etc.) is exploited. Finally, we train our model using real-world images that have not been curated for the specific task at hand. E.g., we might leverage 3D object detection datasets to train, where images contain multiple, possibly occluded object instances, each having possibly different scales and thus resolutions.

To be able to train the encoder in the underconstrained scenario mentioned above, we take advantage of pre-trained instance or panoptic segmentation algorithms to identify in the image 2D pixels belonging to the same object instance as well as pre-trained monocular 3D object detectors in order to get a prior about the objects’ poses in 3D space. Accordingly, both at training and test time, we assume to get for each image a set of 3D bounding boxes with associated 2D masks, which represent the detected object instances, and information about camera calibration.

By leveraging the information about the object 3D bounding box, we can disentangle the object representation from the actual object pose and scale. Indeed, we obtain a normalized, object-centric encoding, which is factored into a shape and an appearance component. Akin to a conditional NeRF model, the shape code is used to condition an occupancy network, which outputs a density given a 3D point in normalized object space and the appearance code is used to condition an appearance network that provides a RGB color given a 3D point and a viewing direction in normalized object space. The two networks yield an implicit representation of the 3D object.

3.1. Preliminaries

Image I. Given a 2D image where multiple objects of interest are present, we run a 3D object detector along with panoptic segmentation in order to extract for each object instance a 3D bounding box and an instance mask (see Fig. 2). The bounding box and the mask are used to produce a masked 2D image \( \hat{I} \) of the detected object instance, fitting a fixed input resolution. In addition, the 3D bounding box captures extent, position, and rotation of an object in cam-
the scene. The RGB color of pixel \( p \) is represented in \( \mathbb{R}^3 \) space, while the segmentation mask provides per-pixel information about possible occlusions w.r.t. other objects in the scene. The RGB color of pixel \( u \in U \) in image \( I \) is denoted by \( I_u \in \mathbb{R}^3 \), where \( U \) represents its set of pixels.

**Normalized Object Coordinate Space** \( O \). Each object instance has an associated 3D bounding box \( \beta \) that identifies a rectangular cuboid in camera space describing the pose and extent of the associated object. The 3D points contained in a 3D bounding box \( \beta \) can be mapped via a diffeomorphism to the (centered) unit cube \( O := [-\frac{1}{2}, \frac{1}{2}]^3 \) called Normalized Object Coordinate Space (NOCS). Indeed, every 3D bounding box can be translated, rotated and scaled into a centered unit cube. In light of this fact, we will use directly \( \beta \) to represent the aforementioned diffeomorphism and, hence, to map points from camera space to the NOCS.

**Object-Centric Camera** \( \gamma \). Each image \( I \) depicting a 3D scene has an associated camera denoted by \( \rho \). Camera \( \rho \) maps pixels \( u \in U \) to unit-speed rays in camera space denoted by \( \rho_u : \mathbb{R}_+ \rightarrow \mathbb{R}^3 \), where \( \rho_u(t) \) gives the 3D point along the ray at time \( t \). By leveraging the bounding box \( \beta \) of a given object, we can map each ray \( \rho_u \) from camera space to NOCS yielding an object-centric ray \( \gamma_u \). Specifically, \( \gamma_u \) is a unit speed reparametrization of the remapped ray \( \beta \circ \rho_u \). We refer \( \gamma \) as the object-centric camera for a given object.

**Occupancy Mask** \( Y \). We use panoptic segmentation to produce a 2D occupancy map \( Y \) associated with an object’s image \( I \). An occupancy mask \( Y \) provides for each pixel \( u \in U \) a class label \( Y_u \in \{+1, 0, 1\} \). Foreground pixels, \( i.e., \) pixels belonging to the object instance mask, are assigned label \(+1\). Background pixels, \( i.e., \) pixels that are not occluding the object of interest, are assigned label \(-1\). Pixels for which it is not possible to determine whether they occlude the object or not are assigned label \(0\). A pixel is assigned the background label if it belongs to a semantic category that is not supposed to occlude the object of interest (\( e.g., \) for car, we have sky, road, sidewalk, \( etc. \)). See Fig. 2 for an example.

### 3.2. Architecture Overview

We provide an overview of our architecture in Fig. 3 and provide a description below.

**Inputs** \((I, \gamma, Y)\). Our architecture takes as input the image \( I \) of an object that has been detected, the corresponding camera \( \gamma \) in NOCS, which has been derived by exploiting the information about the object’s 3D bounding box, and the occupancy mask \( Y \) obtained by leveraging panoptic segmentation. Details about \( I, \gamma \) and \( Y \) have been provided in Sec. 3.1. Examples of object images, occupancy masks and 3D bounding boxes are given in Fig. 2.

**Shape and Appearance Encoder** \( \Phi_B \). We encode an input image \( I \) depicting a given object of interest into a shape code \( \phi_S \) and an appearance code \( \phi_A \) via a neural network \( \Phi_B \); \( i.e., \) \( (\phi_S, \phi_A) := \Phi_B(I) \). The encoder comprises a CNN feature extractor that outputs intermediate features that are fed to two parallel heads, responsible for generating the shape and appearance code, respectively. Implementation details of the encoder and decoders that follow can be found in the supplementary material.

**Shape Decoder** \( \Psi_B \). The shape code \( \phi_S \) is fed to a decoder network \( \Psi_B \), which implicitly outputs an occupancy network \( \sigma; \ i.e., \ \sigma := \Psi_B(\phi_S) \). The occupancy network \( \sigma : O \rightarrow \mathbb{R}_+ \) outputs a density for a given 3D point \( x \in O \) expressed in NOCS.

**Appearance Decoder** \( \Psi_A \). As opposed to the shape decoder, the appearance decoder \( \Psi_A \) takes in input both shape and appearance codes and implicitly outputs an appearance network \( \xi \); \( i.e., \) \( \xi := \Psi_A(\phi_S, \phi_A) \). The appearance network \( \xi : O \times \mathbb{S}^2 \rightarrow \mathbb{R}^3 \) outputs an RGB color for a given 3D point \( x \in O \) and a viewing direction \( d \) on the unit 3D sphere \( \mathbb{S}^2 \).

**Volume Renderer** \( V \). The occupancy network \( \sigma \) and the appearance network \( \xi \) form a radiance field representing an object in NOCS. We can compute the color associated with \( u \) by rendering the object-centric ray \( \gamma_u \) using the approach proposed in [26]. However, since we are interested in modeling only the object of interest, the object-centric ray is limited to points intersecting \( O \). This yields the following volume rendering formula:

\[
V(\gamma_u | \sigma, \xi) := -\int_{a_u}^{b_u} \alpha_t(\gamma_u, \sigma) \xi(\gamma_u(t), d_u) dt ,
\]

\(^1\) The formula differs at first sight from the one in [26], but the two become equivalent after computing the time derivative \( \alpha_t \).
where \([a_u, b_u]\) is the time-window where \(\gamma_u\) intersects \(O\) and \(d_u \in \mathbb{S}^2\) is the unit velocity of \(\gamma_u\). Moreover, \(\dot{\alpha}_t\) denotes the time derivative of the accumulated transmittance along the ray \(\gamma_u\) in the range \([a_u, t]\) defined as

\[
\alpha_t(\gamma_u, \sigma) := \exp \left( - \int_{a_u}^{t} \sigma(\gamma_u(s)) ds \right).
\]

The integral in the volume renderer \(V\) can be solved with a quadrature rule by leveraging sampled points along the ray (see, [26] for more details).

### 3.3. Training

To train our architecture we rely on two loss terms: a photometric loss and an occupancy loss. We provide the losses for a given training example \(\Omega = (I, \gamma, Y)\) comprising the image \(I\), the occupancy mask \(Y\), the object-centric camera \(\gamma\). Moreover, we assume that the radiance field \((\sigma, \xi)\) for the object has been computed from \(I\) using the encoder \(\Phi_E\) and the decoders \(\Psi_S\) and \(\Psi_A\). Finally, we denote by \(\Theta\) all learnable parameters involved in the architecture.

**Photometric Loss** \(L_{\text{RGB}}\). The photometric loss term resembles an autoencoder loss, for it forces the model to fit the input it is given after encoding it into a shape and appearance code with \(\Phi_E\), decoding it into an object radience field by using \(\Psi_S\) and \(\Psi_A\), and finally rendering it with the volume renderer \(V\). The loss is formally defined as follows

\[
L_{\text{RGB}}(\Theta|\Omega) := \frac{1}{|W_{\text{RGB}}|} \sum_{u \in W_{\text{RGB}}} \|I_u - V(\gamma_u|\sigma, \xi)\|^2,
\]

where \(W \subset \mathcal{U}\) contains only foreground pixels; i.e., \(Y_u = 1\) whose object-centric ray \(\gamma_u\) intersect \(O\).

**Occupancy Loss** \(L_{\text{OCC}}\). We use panoptic segmentation to infer whether a pixel is a foreground, background or unknown pixel. This information is encoded in the occupancy mask \(Y\), which is used to directly supervise the accumulated transmittance component \(\alpha\) of the volume rendering equation \(V\). Indeed, \(\alpha(\gamma_u, \sigma) := \alpha_{\text{OCC}}(\gamma_u, \sigma)\) represents the probability that the object does not intersect ray \(\gamma_u\), or in other terms that \(u\) is potentially a background pixel. Similarly \(1 - \alpha(\gamma_u, \sigma)\) is the probability of \(u\) to be a foreground pixel. We can therefore implement a classification loss directly on the accumulated transmittance as follows:

\[
L_{\text{OCC}}(\Theta|\Omega) := -\sum_{u \in W_{\text{OCC}}} \log \left[ \frac{Y_u(\frac{1}{2} - \alpha(\gamma_u, \sigma)) + \frac{1}{2}}{|W_{\text{OCC}}|} \right],
\]

where \(W_{\text{OCC}} \subset \mathcal{U}\) contains only foreground or background pixels; i.e., \(Y_u \neq 0\), with rays \(\gamma_u\) intersecting \(O\).

**Final Loss** \(L\). The final loss that we use to train our network is a linear combination of the two losses described above:

\[
L(\Theta|\Omega) = L_{\text{RGB}}(\Theta|\Omega) + \lambda L_{\text{OCC}}(\Theta|\Omega),
\]

where the occupancy loss is modulated with a hyperparameter \(\lambda \geq 0\).

### 3.4. Test-Time Optimization

Our method enables a forward encoding of an object at test-time due to the presence of an ad-hoc encoder \(\Phi_E\). Nevertheless, we can further refine the regressed codes or even the object’s prior pose to make the output more robust, for instance, to domain shifts in the object’s appearance or errors in the predicted 3D bounding box. To this end, we keep optimizing our loss \(L\) at test time, but with the object’s shape/appearance codes \((\phi_S, \phi_A)\) and the 3D bounding box \(\beta\) being regarded as variables to be optimized. Since the optimization requires a good initial estimate to converge towards a good solution, we use the initial 3D bounding box prediction from the 3D object detector and the object’s encoding provided by our encoder \(\Phi_E\) to initialize the variables. Our formulation allows also to optimize a subset of those variables by keeping the other fixed (e.g., fine-tune appearance only by optimizing \(\phi_A\), while keeping \(\phi_S\) and \(\beta\) fixed). It is worth mentioning that in a monocular setting optimizing the bounding box \(\beta\) is not well-defined, because of the scale-depth ambiguity. In practice, we keep the size component of the bounding box fixed to the one regressed by the 3D object detection and optimize only the bounding box pose; i.e., rotation and translation.
4. Experiments

We quantitatively evaluate our approach for the task of novel view synthesis from a single view on the nuScenes dataset [2], and on the SRN-Cars synthetic car dataset introduced in [36]. Finally, we also evaluate our model trained on nuScenes data on images taken from the KITTI [8] and Mapillary Metropolis datasets, respectively. In Fig. 4 we provide reconstruction results together with synthesised, novel views after training on nuScenes for 1) nuScenes validation (top), 2) KITTI validation (middle), and 3) Mapillary Metropolis validation (bottom). Please note that the model has never seen any data from KITTI or Metropolis during training.

Baseline. We compare quantitatively and qualitatively to PixelNeRF [41] on the task of one-view, 2D-supervised reconstruction. For experiments on nuScenes, we extend their method to support training only on foreground and background pixels, and transform the camera system into the normalized object space in order to leverage 3D object annotations. As PixelNeRF is trained in a multi-view setup, we provide an additional view during training time leveraging provided tracking annotations. In contrast, we train our model using only a single observation of the same instance.

Metrics. We report the standard image quality metrics PSNR (Peak Signal to Noise Ratio) and SSIM (Structural Similarity Index Measure) [38] for all evaluations. Furthermore, we include LPIPS [44] and FID [14] scores to more accurately reflect human perception.

Implementation Details. Similar to PixelNeRF [41], our image encoding uses a ResNet34 backbone pre-trained on ImageNet, while each decoder consists of five fully-connected residual blocks. For an in-depth description of our architecture, we refer to the supplementary material.

4.1. Evaluation on nuScenes

The nuScenes dataset is a large-scale driving dataset with 3D detection and tracking annotations for 7 object classes. It contains 700 training, 150 validation, and 150 test sequences, comprising 168k training images, 36k validation images, and 36k test images. As this dataset is commonly used for perception tasks in autonomous driving research, we pre-process the data to make it suitable for the task of novel view synthesis: We filter for sequences at daytime (provided as meta-information) and we run a pre-trained 2D panoptic segmentation model [32] as nuScenes does not provide 2D segmentation masks.

We match provided 3D bounding box annotations with the resulting instance masks and categorize the panoptic results into foreground (visible part of the object), background (non-occluding semantic categories like street, sky, sidewalk), and unknown regions (potentially occluding categories like people, vehicles or vegetation) as we do not rely on depth information in order to resolve occlusions. Furthermore, we filter for sufficiently visible instances and use tracking information for evaluation purposes. For training, we select from each instance a single view at random to train on (two views for the baseline) and evaluate on a pre-determined subset of 10k pairs of views of car instances in the validation split. We refer to the supplementary material for details about the data generation process.
Single-view synthesis results. We report our performance in comparison to state-of-the-art baselines pixelNeRF and CodeNeRF in Tab. 1. Even without any multi-view information, our model is able to synthesize higher quality results compared to the baseline trained with multi-view information. Test-time optimization allows the model to recover instance-specific details while preserving shape and overall appearance. In Fig. 5 we demonstrate qualitatively that our model produces overall sharper results and more natural shapes, while pixelNeRF struggles to synthesize views that are significantly different from the input view. We quantify this observation by plotting PSNR and LPIPS values against the rotational difference between input and target view in Fig. 6: While the performance of the models evaluated on views close to the input view is similar, pixelNeRF degrades significantly with increasing change of perspective but to the maximum rotational error, where the models can leverage similarities to the input view (e.g., a car seen from the left and right side).

<table>
<thead>
<tr>
<th>nuScenes cars</th>
<th>PSNR↑</th>
<th>SSIM↑</th>
<th>LPIPS↓</th>
<th>FID↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>pixelNeRF [41]</td>
<td>18.25</td>
<td>0.459</td>
<td>0.236</td>
<td>160.60</td>
</tr>
<tr>
<td>CodeNeRF [15]</td>
<td>18.44</td>
<td>0.462</td>
<td>0.241</td>
<td>146.32</td>
</tr>
<tr>
<td>AutoRF (no opt.) on test</td>
<td>18.69</td>
<td>0.479</td>
<td>0.227</td>
<td>138.23</td>
</tr>
<tr>
<td>AutoRF on test</td>
<td>18.94</td>
<td>0.491</td>
<td>0.223</td>
<td>145.10</td>
</tr>
</tbody>
</table>

Table 1. Overview of novel-view synthesis results on nuScenes cars from the validation set.

4.1.1 Shape reconstruction quality

We additionally evaluate our methods’ shape reconstruction quality on the nuScenes validation split by comparing the resulting depth renderings against ground truth (GT) object LiDAR points. We crop the LiDAR recordings according to the oriented GT 3D bounding-box annotations, remove points in the lower 10% of the bounding box (to exclude LiDAR points belonging to the street), and finally evaluate on samples with at least 20 remaining points. Tab. 2 shows that our model trained on single views and from auto-generated 3D detection and segmentation results produces more precise surfaces in terms of L1 and RMSE metrics compared to the pixelNeRF baseline trained with GT annotations and multiple views per instance (additional qualitative reconstructions can be found in the supplementary document).

<table>
<thead>
<tr>
<th>nuScenes cars</th>
<th>L1↓</th>
<th>RMSE↓</th>
<th>Avg. perturb.</th>
<th>PSNR↑</th>
<th>LPIPS↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>pixelNeRF [41]</td>
<td>0.357</td>
<td>0.984</td>
<td>0°/0cm</td>
<td>18.95</td>
<td>0.210</td>
</tr>
<tr>
<td>CodeNeRF [15]</td>
<td>0.239</td>
<td>0.641</td>
<td>5°/10cm</td>
<td>18.47</td>
<td>0.216</td>
</tr>
<tr>
<td>AutoRF (no opt.)</td>
<td>0.209</td>
<td>0.632</td>
<td>10°/20cm</td>
<td>17.95</td>
<td>0.269</td>
</tr>
<tr>
<td>AutoRF</td>
<td>0.204</td>
<td>0.614</td>
<td>20°/40cm</td>
<td>16.83</td>
<td>0.348</td>
</tr>
</tbody>
</table>

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

4.2. Evaluation on synthetic data

We evaluate our method against the baselines on the SRN dataset introduced in [36]. The SRN-Cars dataset contains 3514 samples of car renderings (based on shapes from 3D Warehouse) with a predefined split across object instances. While each model is rendered from 50 random views per object instance, we select a single random frame for training our method and CodeNeRF (and two random frames for the pixelNeRF baseline). For each object in the test set, we evaluate novel view synthesis from 251 views sampled on an Archimedean spiral based on a single, randomly chosen view as input for our method. We refer to the supplementary for evaluation on additional categories.

Single-view synthesis results. We compare our method in Tab. 4 to pixelNeRF trained on additional views and CodeNeRF. We notice that our model outperforms the baselines on all metrics while not requiring multi-view constraints at training time. Additionally, we provide qualitative results in Fig. 7 illustrating our method’s high-fidelity reconstruction results, and how we are preserving fine-grained details such as differently colored roof-tops of cars.
Table 4. Evaluation of novel-view synthesis on the SRN-Cars dataset from [36].

Table 5. Novel-view evaluation on the nuScenes validation set.

Figure 7. Qualitative comparison on SRN-Cars dataset, illustrating our high-fidelity, single-view reconstruction results compared to the 2-view pixelNeRF baseline.

Figure 8. Scene editing examples on nuScenes. Starting from the input view, we can change the codes of the objects and synthesize novel scene layouts.

4.3. Ablations

Data quality. In Tab. 3, we report novel-view synthesis results of AutoRF trained with random perturbations of the ground truth annotations in terms of average different rotation and translation errors. We note that smaller inaccuracies have minor impact. Furthermore, we investigate the performance when we train AutoRF on human-annotated data on the nuScenes train split and evaluate the results against our model fully trained on machine-annotated single-view data. The results are summarized in Tab. 5 and show that leveraging high-quality annotations does not significantly improve the novel view synthesis results. While PSNR and SSIM are very similar, the main improvements are gained in terms of perceptual losses (LPIPS and FID). Qualitative analyses show that the model trained on GT annotations are slightly less blurry, which we assign to the fact that inaccurate pose annotations result in imprecise ray sampling in NOC space.

Domain transfer. While trained solely on the nuScenes street-level dataset, we show qualitative results in Fig. 4 demonstrating that the learnt object radiance field priors generalize well to novel datasets. Examples on nuScenes, KITTI, and Mapillary Metropolis show that AutoRF (no opt.) can reliably assign matching object priors and that test time optimization consistently preserves fine-grained details also in novel views.

Scene editing. Our approach naturally decomposes an object into pose, shape, and appearance. This directly enables 3D scene editing, where the objects observed in an input view can be rendered with novel poses, shapes and/or appearances, effectively creating a novel scene. We provide an example of the scene editing capability in Fig. 8 and refer to the supplementary document for further demonstrations.

5. Conclusions

In this work, we proposed a new approach for learning neural 3D object representations that in contrast to the majority of existing works exploits exclusively single views of object instances during training, without leveraging other 3D object shape priors such as CAD models or resorting to curated datasets. To address this challenging training setting, our method leverages machine-generated labels, namely 3D object detection and panoptic segmentation, to learn a normalized object-centric representation, which is pose independent and factorizes into a shape and an appearance component. These two components are decoded into an implicit radiance field representation for the object, which can then be rendered into novel target views. We show that our approach generalizes well to unseen objects, even across different datasets of real-world street scenes.

Societal impact and limitations. Our work helps to further investigate the possibilities of leveraging real-world, large-scale data for building representations needed in future AR/VR applications. As for the limitations, our work requires significant computational efforts for producing renderings of novel views, akin to related works from neural representation learning. Further, we will investigate AutoRF’s applicability to more articulated object categories.
References


[27] Norman Müller, Yu-Shiang Wong, Niloy J. Mitra, Angela Dai, and Matthias Nießner. Seeing behind objects for 3d...


