Car-Studio: Learning Car Radiance Fields from Single-View and Unlimited In-the-wild Images

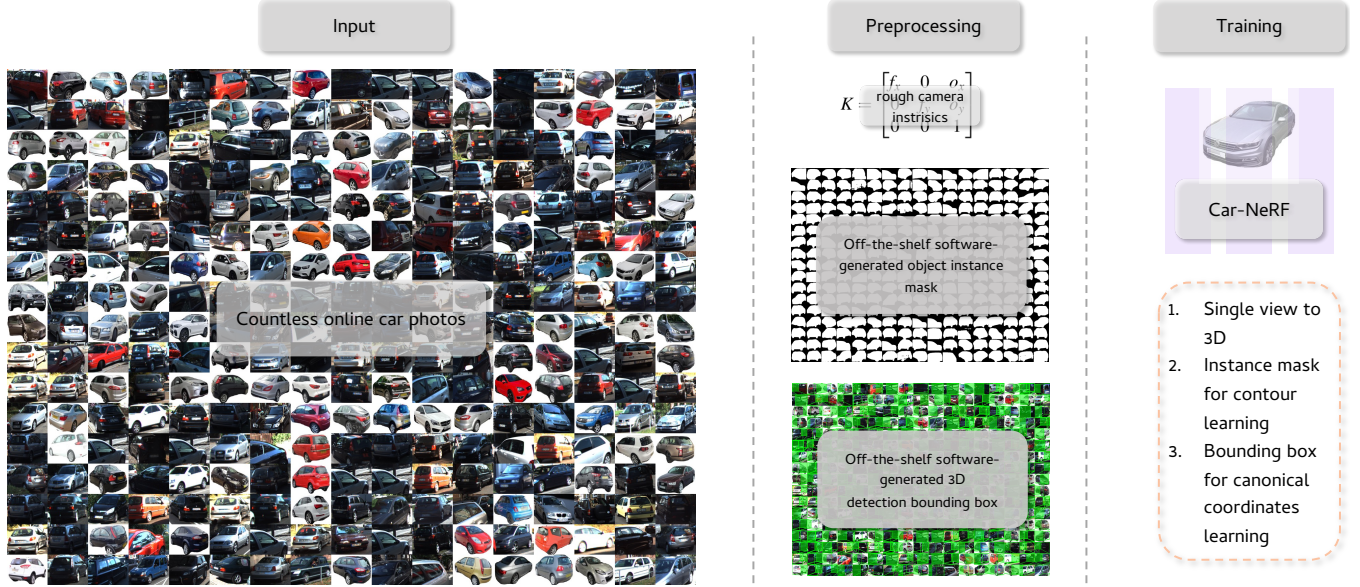Tianyu Liu[1], Hao Zhao[2], Yang Yu[3], Guyue Zhou[2], Ming Liu[1,3,4]

Fig. 1: **Car-Studio: learns a General Car-NeRF from Countless Car Images in the wild**. We extract car patches from unconstrained internet images using an off-the-shelf 2D object detector [1]. To preprocess the patches, we generate rough camera intrinsics, object instance masks, and 3D detection bounding boxes using a monocular 3D object detector [2] and a pixel-level segmenter [3]. Our Car-NeRF model takes the preprocessed information as input and learns from a unified canonical car model with 3D bounding boxes and pixel-level masks to output a high-quality car image from any 3D viewpoint based on a single-view image.

Abstract—Compositional neural scene graph studies have shown that radiance fields can be an efficient tool in an editable autonomous driving simulator. However, previous studies learned within a sequence of autonomous driving datasets, resulting in unsatisfactory blurring when rotating the car in the simulator. In this letter, we propose a pipeline for learning unconstrained images and building a dataset from processed images. To meet the requirements of the simulator, which demands that the vehicle maintain clarity when the perspective changes and that the contour remains sharp from the background to avoid artifacts when editing, we design a radiation field of the vehicle, a crucial part of the urban scene foreground. Through experiments, we demonstrate that our model achieves competitive performance compared to baselines. Using the datasets built from in-the-wild images, our method gradually presents a controllable appearance editing function. We will release the dataset and code on https://lty2226262.github.io/car-studio/ to facilitate further research in the field.

Index Terms—Autonomous Driving, Neural Radiance Field, 3D Reconstruction from Single Image

[1] The Hong Kong University of Science and Technology, Hong Kong SAR, China. [2] AiR, Tsinghua University, Beijing, China. [3]The Hong Kong University of Science and Technology (Guangzhou), Guangdong, China. [4] HKUST Shenzhen-Hong Kong Collaborative Innovation Research Institute, Futian, Shenzhen, China.

I. INTRODUCTION

NeRF [4] has emerged as a powerful art for generating novel views in both computer vision and computer graphics, and has recently garnered attention for its potential in autonomous driving [5]–[17]. By modeling the color and density of every point within the relevant 3D space, NeRF can create highly realistic images of a scene from any viewpoint. This capability is particularly useful in the context of autonomous vehicle testing [18], [19], as it enables the replay of a scene multiple times with different viewpoints even though it's not recorded before. By doing so, NeRF can help improve the safety and reliability of autonomous vehicles by enabling more thorough testing and evaluation.

In this letter, our objective is to develop a module for building a photorealistic autonomous driving simulation platform that is capable of generating videos through the editing of recorded urban scene footage. This simulation platform offers significant benefits, particularly in the generation of

AI-driven potential accident videos [20]. These videos greatly enhance the efficiency of validating autonomous driving algorithms [21], [22]. Creating a photorealistic simulation platform through the process of editing radiance fields offers noteworthy advantages, including the reduction of appearance inaccuracies and the bridging of the gap between simulated and real urban scenes [23]–[25]. However, reconstructing editable urban scenes in the wild poses several challenges.

One major challenge is the inconsistent movement over time between the background and foreground. Previous works, such as [5]–[14], [16], have addressed this issue by decomposing the scene into background and foreground and learning foreground nodes based on categories, speed, distance, or a combination of these factors. In this letter, we focus on the challenge of reconstructing editable car instances and address it by parsing the foregrounds based on categories to achieve flexible editing.

Another challenge in reconstructing editable urban scenes is constrained views. Prior works discussed in the previous paragraph were trained on autonomous driving street datasets, where photos are captured from egocentric video streams that mostly show the front or rear of the car facing the camera. Consequently, these models may not perform well when the target is viewed from an unseen angle, such as the side view of a car. Some existing works have introduced pre-training generative branches [9], [11], [17] to generate unseen views, but these indirect approaches do not address the fundamental problem of a shortage of side view data. To overcome the challenge of constrained views, we propose leveraging online photos captured in the wild, specifically unlimited Internet images. However, using these images presents a new set of challenges, such as unknown illuminations, environments, camera intrinsic parameters, 3D poses, and dimensions. Therefore, we propose a novel toolchain that leverages in-the-wild 2D photos for favorable 3D neural radiance field training. Our approach aims to address the fundamental shortage of side view data and overcome the challenges of using online photos for 3D object reconstruction.

Our toolchain, named **Car-Studio** (as depicted in Fig 1), is designed to utilize in-the-wild car images to train a category-based NeRF [4] model for cars. To generate a car instances dataset, we use three in-the-wild datasets: KITTI dataset multi-object tracking track (KITTI-MOT) [26], KITTI dataset object detection track (KITTI-DET), and DVM-Cars [27] dataset. We provide a comparison of these datasets in Tab I. While using in-the-wild data presents challenges such as messy objects with backgrounds, no camera parameters, unknown car poses, and single-view-only instances without multi-view supervision, it also provides benefits. For example, it offers different types of car instances with various orientations that supplement the urban benchmark such as [26], which lacks side-views.

Based on our analysis of the combined datasets, we observed that the 2D car patches exhibit variations in size and possess boundaries that are challenging to discern, particularly in scenarios where the cars are in areas of low light or have hues that are similar to the surrounding environment. To overcome these challenges, we developed **Car-Nerf**, which leverages the anti-aliasing advantages of mip-NeRF [28] to enable

continuous scaling editing from far to near in simulation, and a segmentation mask that supervises the rendered accumulated radiance field weights for sharp contours. Additionally, we employ a canonical car-centric coordinate system to learn 3D spatial features from single-view-only instances. Our proposed approach produces plausible rendering results in novel views and enables controllable spatial and appearance editing during scenes.

Our contributions can be summarized as follows:

- A curated dataset **CarPatch3D** of hundreds of thousands of 2D images and 3D spatial information. This dataset provides favorable information for training a category-based NeRF model for cars. Its availability enables the development of more effective urban NeRF foreground models.
- We developed **Car-NeRF**, a specialized model designed for the car category, aiming to enhance boundary clarity and anti-aliasing quality. It demonstrates state-of-the-art performance in tasks such as image reconstruction and novel view synthesis.
- We designed a pipeline called **Car-Studio** that can learn from single-view in-the-wild car images to generate 3D surrounding views and enable plausible controllable spatial and appearance editing.

II. RELATED WORK

Urban NeRF for autonoumous driving. Urban NeRF has recently attracted considerable attention, with numerous methods and models developed to address the challenges posed by foreground objects, particularly cars. While some methods, such as BlockNerf [15], employ foreground masks to ignore the foreground objects and focus on large-scale background reconstruction, our work focuses on the foreground cars and their interactions with the environment. Existing methods [6]–[8], [10], [14], [29] use the same auto-decoder architecture as the seminal NeRF [4]. While these methods are effective for representing different instances, they lack the ability for zero-shot learning and controllable appearance editing. Recent approaches [9], [11], [17], [30]–[34] have utilized generative models to learn foreground models. In contrast, our approach adopts the decoder-encoder architecture used in previous works such as PixelNerf [35] and AutoRF [5], without relying on explicit generators. Moreover, our approach extends the processing of images to include pose-free and even camera intrinsics-free in-the-wild images and introduces a controllable appearance editing approach to enable more flexible editing capabilities.

Single view to 3D NeRF. Inferring 3D shapes from 2D images is a long-standing challenge in computer vision and graphics. Existing methods present good results with different approaches. Deformation-based methods [36]–[39] learn the 3D shape by mesh deformation based on a primitive mesh. Unsup3D [40] assumes symmetric and deformable shapes to learn 3D shapes. Mesh R-CNN [41] predicts the 3D mesh by a 3D voxel prediction followed by a mesh refine branch. PiFu [42] learns the occupancy field in 3D directly by the occupancy implicit functions. Some works learn shape

TABLE I: Comparison of three dataset sources for CarPatch3D.

Dataset	Background	Unknown camera intrinsics and poses	Car orientaion	Viewpoint diversity	Filtered(details in III-B) car patches number	Filtered(details in III-B) car instances number
KITTI-MOT	✓	✗	Mostly front/back-view	Single view + multi-view	4481	383
KITTI-DET	✓	✗	Mostly front/back-view	Single view only	4945	4945
DVM-Cars	✗	✓	Many side-view	Single view + multi-view	521275	201075

codes for different types of object priors like CAD model [43], [44], SDF-based Database [45], activation code library [46]. Recently, single-view to 3D NeRF has emerged as a promising approach to tackling this problem, with impressive results achieved by methods such as MVSNerf [47] and PixelNerf [35] using multi-view supervision, obtaining multi-view images is often more expensive and impractical compared to single-view images, especially in real-world scenarios. In contrast, our proposed method can process both multi-view and single-view image input, making it more versatile and practical for a wider range of applications. Recent works have explored the use of transformer-based architectures rather than MLPs to develop methods for view synthesis from a single input image, offering more fine-grained control over the reconstruction process [48], [49]. CodeNeRF [50] initializes the implicit representation with a pre-trained neural network and optimizes it using a differentiable renderer. Our proposed method differs in that ours involves two stages: an encoder that extracts features from the input image and uses them to initialize the implicit representation, followed by a similar optimization process as CodeNeRF to refine it further. This enables our model to extract individual instance latent codes from images with backgrounds, while CodeNeRF is restricted to test optimization only.

III. METHODOLOGY

A. Car-Studio Pipeline Overview

We propose **Car-Studio**, a three-phase pipeline designed to learn a general car neural rendering model from online car photos, as shown in Fig. 1. In the first phase, **Input**, we use a pre-trained 2D detector [1] to identify patches of interest. We filter out patches that are too small or whose classification results indicate a non-vehicle object to ensure high-quality data. In the subsequent **Preprocessing** phase, we refine the data further to prepare it for use in our model. We describe the preprocessing phase in detail in Section III-B. In the third phase, **Training**, we use our **Car-NeRF** model, discussed in Section III-C, to learn from the preprocessed data.

B. Preprocessing

The NeRF model [4] requires images, camera parameters, and camera poses as input. Obtaining accurate camera parameters from a single image can be challenging when there is no known size calibration target in the field of view. Therefore, we use a rough focal length based on [26] and the optical center equal to half of the image shape size, and pass them to a 3D detection model [2] to solve for the camera pose. To ensure high-quality patches, we apply a filtering process that consists of the following steps:
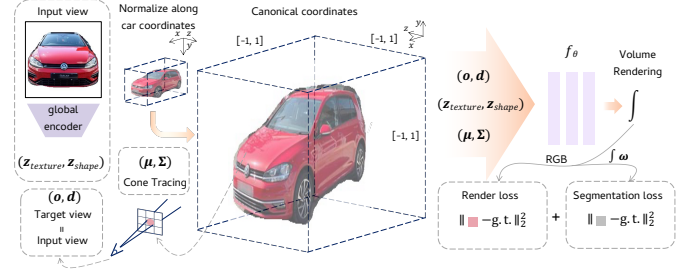

Fig. 2: The Car-NeRF architecture. We use a global encoder to obtain texture ($z_{texture}$) and shape (z_{shape}) latent vectors from the input image. We then scale the world coordinates into object-centric normalized canonical coordinates. Cone tracing is performed at the input view's position o along direction d to obtain the mean distance μ and variance Σ along the axis of the cone, which are passed to our model f_θ. The model outputs the rendered image's RGB value, which is used to calculate render loss, while the segmentation losses are calculated using the output accumulated weights along axis $\int \omega$.

1) We filter out images with a low intersection ratio between the 2D detector output and the minimum 2D envelope box for reprojecting the 3D box, as this indicates inconsistencies between the outputs of the 2D and 3D detectors.

2) We filter out images with a low pixel count ratio between the result of the 2D segmentor Segment Anything [3] pixel-level segmentation and the 2D detector's result, which indicates a high occlusion rate.

3) We filter out images with low prediction confidence from the 2D detector, 3D detector, and 2D segmentor.

4) We manually filter out 3D prediction results that are inconsistent with the image.

We carefully followed the aforementioned steps to ensure high-quality preprocessing of the data, making it suitable for use in our pipeline. This effort resulted in a new dataset named **CarPatch3D**. CarPatch3D comprises a set of car patches, selected using the semi-automatic method described above, along with binary masks generated by the SAM [3], roughly estimated camera parameters, and 3D position and orientation of the vehicles generated by DD3D [2]. This rich set of data makes CarPatch3D highly suitable for the NeRF training of cars in autonomous driving.

C. Car-NeRF Model

The architecture of our model is shown in Fig 2. We learn the global latent codes z using a global encoder, which is a ResNet-34 [51] pre-trained on ImageNet [52]. We then decouple the global latent codes into a shape component z_{shape} and a texture component $z_{texture}$ using two one-layer independent MLPs.

Normalization to car-centric canonical coordinate. To learn 3D shapes from single-view image supervision, we convert the camera-centric coordinates to canonical coordinates centered at the car's centroid and oriented along its principal axes. We obtain the dimensions of the car (length l, height h, and width w) from the preprocessing 3D detection module's result. The scaled [-1, 1] camera pose with respect to the camera coordinate T_{cam}^{scaled} is given by:

$$T_{cam}^{scaled} = ([\frac{2}{l}, \frac{2}{h}, \frac{2}{w}, 1]^T \mathbf{1}_{1\times4}) \circ \mathbf{I}_{4\times4} \qquad (1)$$

, where $\circ$ is the Hadamard product.

The camera pose with respect to the canonical coordinate system is obtained by the scaled camera pose with respect to the camera coordinate system (T_{cam}^{scaled}), and the car pose with respect to the camera coordinate system (T_{cam}^{car}). The resulting camera pose with respect to the canonical coordinate system T_{canon}^{cam} is given by:

$$T_{canon}^{cam} = T_{cam}^{scaled} (T_{cam}^{car})^{-1} T_{canon}^{scaled} \qquad (2)$$

, where T_{canon}^{scaled} is scaling matrices that map the canonical coordinates to camera-centric scaled coordinates whose diagonal values are (1, -1, -1, 1) according to Fig 2.

Cone tracing and IPE. In our architecture, the encoder takes Internet images processed by a 2D detector as input. Since the patch size of the input images can vary greatly, the region that each pixel represents can also vary greatly. To address this, we use cone tracing instead of ray tracing for a continuous scaling that provides anti-aliasing [28]. A cone at apex $\boldsymbol{o}$ with direction $\boldsymbol{d}$ is represented by a multivariate Gaussian distribution with mean $\boldsymbol{\mu}(\boldsymbol{o}, \boldsymbol{d})$ and variance $\boldsymbol{\Sigma}(\boldsymbol{d})$.

To build a continuous scaling along the axis of cones, we use the integrated positional encoding (IPE) [28] instead of positional encoding (PE) [4]. The IPE incorporates the multivariate Gaussian representation into the encoding process, resulting in improved performance. The IPE of positions is denoted by γ_{pos}, while the IPE of directions is denoted by γ_{dir}.

Sampling and model architecture. We adopt a coarse-to-fine pixel sampling strategy and a shared model architecture similar to mip-NeRF [28]. The model f_θ consists of a shape component $f_{\theta,shape}$ and a texture component $f_{\theta,texture}$. To embed the latent vectors into the model, we introduce the following modifications:

$$\mathbf{f}_{out}, \sigma = f_{\theta,shape}(\gamma_{pos}(\boldsymbol{\mu}(\boldsymbol{o}, \boldsymbol{d}), \boldsymbol{\Sigma}(\boldsymbol{d})) + \boldsymbol{z}_{shape}) \qquad (3)$$

Here, $\boldsymbol{z}_{shape}$ is the shape latent vector, σ is the density, and $\mathbf{f}_{out}$ is the output feature of the shape component. The RGB color, denoted by c, can be calculated using the texture latent vector $\boldsymbol{z}_{texture}$ as follows:

$$\text{RGB} = c = f_{\theta,texture}(\mathbf{f}_{out} + \gamma_{dir}(\boldsymbol{d}) + \boldsymbol{z}_{texture}) \qquad (4)$$

The accumulated weights $\int \boldsymbol{\omega}$ from near t_n to far t_f along the axis $\boldsymbol{r} = \boldsymbol{o} + t\boldsymbol{d}$ of the cone is given by:

$$\int \boldsymbol{\omega}(\boldsymbol{r}) = \int_{t_n}^{t_f} \exp(- \int_{t_n}^t \sigma(\boldsymbol{r}(s)ds))\sigma(\boldsymbol{r}(t))dt \qquad (5)$$

Volume Rendering and Loss Functions. We employ the volume rendering technique described in [4] to compute the estimated rendered color $\hat{\text{RGB}}$ and the estimated accumulated weights $\hat{\int \boldsymbol{\omega}}$ along the cone axis defined by the apex $\boldsymbol{o}$ and the direction $\boldsymbol{d}$ at the given pixel location. At this pixel location, the ground truth color is denoted as RGB, and the corresponding binary mask obtained from the 2D segmentor described in Section III-B is denoted as α. The photometric L2 render loss $\mathcal{L}_r$ is defined as follows:

$$\mathcal{L}_r = \alpha(||\hat{\text{RGB}}_f - \text{RGB}||_2^2 + \lambda_c ||\hat{\text{RGB}}_c - \text{RGB}||_2^2) \qquad (6)$$

Here, the estimated rendered color obtained from fine sampling is denoted as $\hat{\text{RGB}}_f$, while that obtained from coarse sampling is denoted as $\hat{\text{RGB}}_c$. The hyperparameter λ_{coarse} is used to balance the loss between two sampling stages. To prevent our model from learning the background color or shapes, we incorporate the binary mask α into the loss function. Similarly, the segmentation loss $\mathcal{L}_s$ is:

$$\mathcal{L}_s = ||\hat{\int \boldsymbol{\omega}}_f - \alpha||_2^2 + \lambda_c ||\hat{\int \boldsymbol{\omega}}_c - \alpha||_2^2 \qquad (7)$$

The total loss is weighted by a segmentation balancing coefficient λ_s:

$$\mathcal{L} = \mathcal{L}_r + \lambda_s \mathcal{L}_s \qquad (8)$$

IV. EXPERIMENTS

We construct our datasets **CarPatch3D** using three different sources: the KITTI multi-object tracking (KITTI-MOT) and object detection (KITTI-DET) tracks [26], and the DVM-Cars dataset [27]. The characteristics of each dataset are shown in Table I. Our work addresses three challenging tasks: single-view supervision for zero-shot learning, multi-view supervision for novel scene synthesis, and patch reconstruction with test-time optimization. Furthermore, we demonstrate Car-Studio's ability to perform controllable scene editing.

Baselines. We evaluate our proposed method against the following baselines: CodeNeRF [50], PixelNeRF [35], and AutoRF [5]. Note that CodeNeRF does not support zero-shot learning, and therefore, we only use it for the multi-view supervision few-shot learning task. Moreover, to ensure a fair comparison since our problem is category-specific, we convert PixelNeRF to the canonical coordinate system.

Test-time Optimization. For the auto-decoder model [50], we optimize the latent vectors directly in the test dataset. However, for the encoder-decoder model [5], [35] and our model, we first obtain the latent vectors using the encoder and then perform joint optimization of the latent vectors and decoder during the test-time optimization stage.

Runtime Environment Details. All experiments are conducted on an Nvidia RTX 3090 Ti graphics card. We optimize all NeRF models for 500,000 steps with a pixel sampler with a batch size of 3,072, which takes approximately two days to complete. Our pipeline is built upon NerfStudio [53]. We also reproduce the CodeNeRF [50], Canonical PixelNeRF-ResNet [35], Canonical PixelNeRF-MLP with an MLP backbone, and AutoRF [5] based on a community implementation [54] for reusing the data loader. We use the RAdam optimizer [55] with hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, lr $= 10^{-3}$,
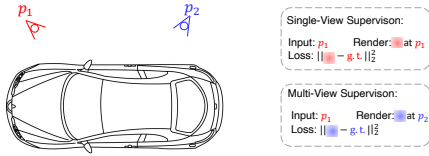
Fig. 3: Illustration of Single-View and Multi-View Supervision. Single-view supervision computes the loss by comparing the rendered image at pose p_1 to the ground truth image at the same pose. In multi-view supervision, the loss is computed by comparing the captured image at pose p_1 with the rendered image at pose p_2.

TABLE II: Performance comparison of different models for single-view supervision zero-shot learning. The best result is highlighted in red, and the second best result is highlighted in blue.

Method	PSNR↑ (dB)	SSIM↑	LPIPS↓
PixelNeRF-ResNet	11.61	0.1948	0.7375
PixelNeRF-MLP	17.87	0.5103	0.4014
AutoRF	20.75	0.5903	0.3466
Car-NeRF(w/o mask)	20.93	0.6020	0.3396
Car-NeRF(w/ mask)	20.23	0.6209	0.3284

and $\epsilon = 10^{-8}$. We use the exponential decay scheduler with $\text{lr}_{\text{final}} = 10^{-4}$ and $\text{step}_{\text{max}} = 200,000$. All other configurations follow the original setups.

Evaluation and Metrics. We assess the performance of the models in novel view synthesis. Figure 3 illustrates both single-view supervision and multi-view supervision approaches. To quantitatively evaluate the results, we employ three metrics: Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM) [56], and Learned Perceptual Image Patch Similarity (LPIPS) [57]. PSNR measures the noise between the rendered image and the original image, while SSIM evaluates the structural similarity between them. LPIPS utilizes a learned model to measure the perceptual similarity between the rendered and original images. These metrics offer objective measures for assessing the algorithm's performance.

A. Single-view Supervision for Zero-shot Learning

In Figure 4, we present qualitative comparisons with the baselines. In this experiment, all testing instances are unseen in the training set, and no test-time optimization is performed. The experiment uses a uniform 9:1 train-test split on all valid KITTI-DET-derived patches. We observe that the default architecture of PixelNeRF-ResNet degenerates to an empty output. Incorporating the segmentation loss $\mathcal{L}_s$ into our approach results in sharper contours compared to the baselines. Additionally, our architecture design reduces the presence of floaters around the target car.

Table II presents the quantitative results of single-view supervision zero-shot novel view synthesis. PixelNeRF [35] uses a ResNet-based backbone by default. However, during the training process, PixelNeRF-ResNet encountered degradation. As a result, we also compare the performance of an MLP-based version of PixelNeRF. The results show that Car-NeRF with mask segmentation loss achieves the best SSIM and LPIPS performance, while Car-NeRF without mask segmentation loss performs best in terms of PSNR.

The quantitative results suggest that incorporating mask segmentation loss can improve the quality of rendered images in

TABLE III: Comparison of performance across models trained with multi-view supervision and single-view supervision. The best result is highlighted in red, and the second best result is highlighted in blue.

Method	PSNR↑ (dB)	SSIM↑	LPIPS↓
PixelNeRF-MLP	17.16	0.4715	0.4343
AutoRF	17.84	0.5212	0.3852
CodeNeRF	19.01	0.5737	0.3299
Car-NeRF(multi-view)	17.40	0.4965	0.4075
Car-NeRF(single-view)	20.12	0.6251	0.2825

terms of structural similarity and perceptual quality. This finding is consistent with the intuition of using mask segmentation to guide the training of neural rendering models. However, it is important to note that Car-NeRF without mask segmentation loss performs better in terms of peak signal-to-noise ratio. This indicates that relying solely on PSNR may not provide a comprehensive evaluation of image quality. Therefore, it may be necessary to consider various metrics, such as structural similarity and perceptual quality, in addition to PSNR, to obtain a more complete assessment of the performance of neural rendering models.

B. Muti-view Supervision Novel View Synthesis

In this experiment, we compared our method with other approaches to assess the effectiveness of single-view supervision versus multi-view supervision. While the baselines in Section IV-A can obtain latent vectors by inputting an arbitrary image to the encoder, the auto-decoder architecture does not have this ability. To provide a fair comparison with these approaches, we use a more comprehensive dataset with instance labels across frames, which requires additional labor but provides more ideal evaluation conditions for multi-view supervision approaches.

We train the models on patches derived from the KITTI-MOT dataset, using the cross-frame instance label as the ground truth for multi-view supervision. To ensure a fair comparison, we only consider instances that have multiple images available for training. This is because CodeNeRF [50] uses global latent vectors as input, and in the case where an instance is not seen in the training set, using an average initialization without any instance-specific information may introduce potential unfairness to CodeNeRF. Similar to the experiment in Section IV-A, we do not apply any test-time optimization to the models in this experiment.

The comparison results are presented in Table III, which demonstrate that our Car-NeRF model achieves competitive performance with PixelNeRF-MLP [35] and AutoRF [5] in the multi-view supervision setting. Notably, our method that employs only single-view supervision outperforms all of the multi-view supervision methods. This observation could be attributed to the fact that the rough estimation of camera parameters introduces more inconsistent noise to the multi-view supervision framework. In contrast, using single-view supervision avoids this inconsistency by relying on a single viewpoint for training. Furthermore, CodeNeRF outperforms the encoder-decoder architecture approaches (all methods except CodeNeRF in this experiment) in the multi-view setting.
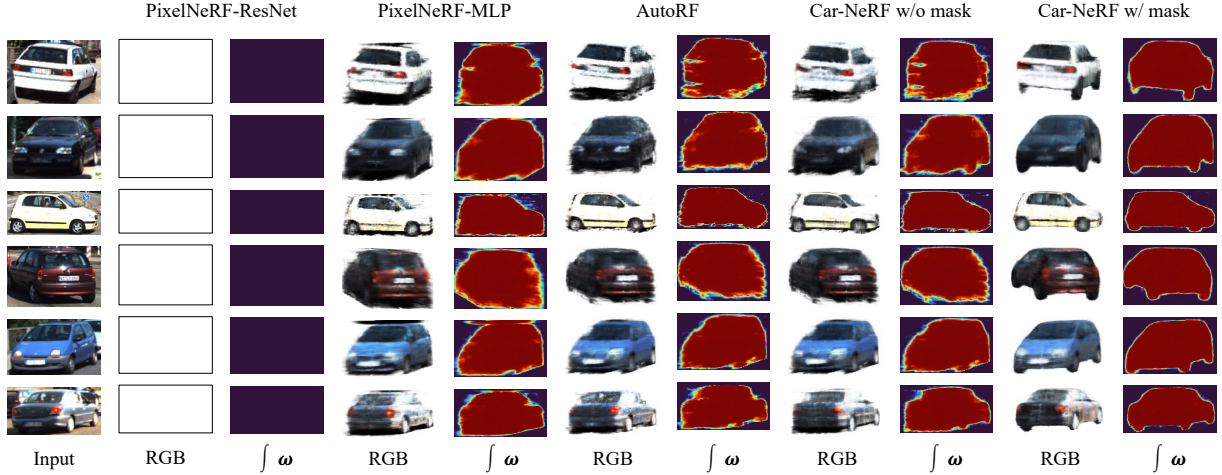
Fig. 4: Comparison of RGB and accumulated weights $\int \omega$ rendering, using a zero-shot learning paradigm with all instances unseen in the training set. The figure depicting accumulated density uses a Turbo color scheme, where dark blue represents low values (down to 0) and dark red represents high values (up to 1).
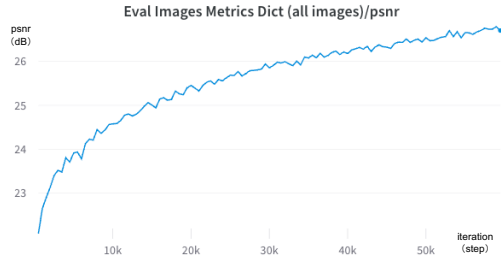

Fig. 5: Learning curve for test-time optimization

Fig. 6: Qualitative ablation study for test-time optimization. The third row shows instances with test-time optimization, achieving higher image quality compared to the second row without test-time optimization. Details on project page.

This may be because encoder-decoder models use a global encoder to extract the latent vectors, which are optimized across all instances, while auto-decoder models optimize the latent vectors instance-specifically. Therefore, we further investigate the potential benefits of instance-specific optimization.

C. Foreground Reconstruction with Test-Time Optimization

To further leverage the advantages of independently optimizing the latent vector, akin to auto-decoder approaches, we implement a test-time optimization on the latent vector decoupled from the encoder. In the first stage, we train the Car-NeRF model using the KITTI-DET-derived dataset. We then detach the latent code from the encoder and proceed to
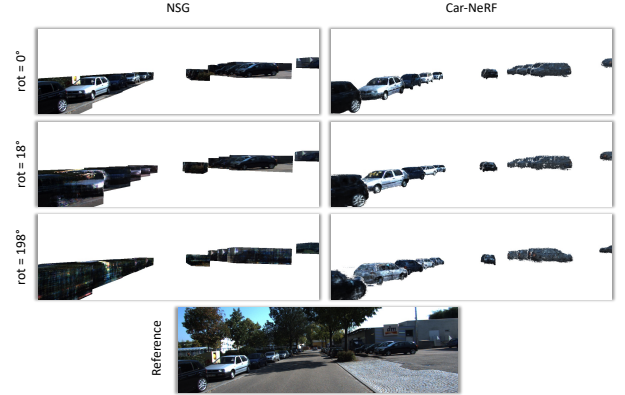

Fig. 7: Unseen view synthesis comparison between NSG [10] and our method. The left column shows NSG's results, while the right column shows our results. Our method performs better than NSG in generating sharp and realistic images, even when a large rotation angle is applied. Specifically, while the images generated by NSG become blurry when an 18-degree rotation angle is applied, our method is still able to generate better quality images. When a larger rotation angle is applied, NSG fails to generate an image of the vehicle, whereas our method can still synthesize unseen views with higher fidelity. Details on project page.

the second stage of training. In the second stage, we perform a test-time optimization on the Car-NeRF's field and the latent codes using the KITTI-MOT-derived dataset. Figure 5 shows the quantitative learning curves for test-time optimization. Figure 6 presents qualitative comparisons between the results with and without test-time optimization. We also compare our approach with a no-prior method, taking NSG [10] as an example in Figure 7. These comparisons demonstrate that our Car-NeRF model is scalable for test-time optimization while retaining the ability to synthesize unseen views.

D. Applications for Autonomous Driving

Our new pipeline enables foreground editing in an autonomous driving simulator, including instance insertion, spatial transformation, deletion, and replacement of instances, as demonstrated in Figure 8. This capability is similar to NSG
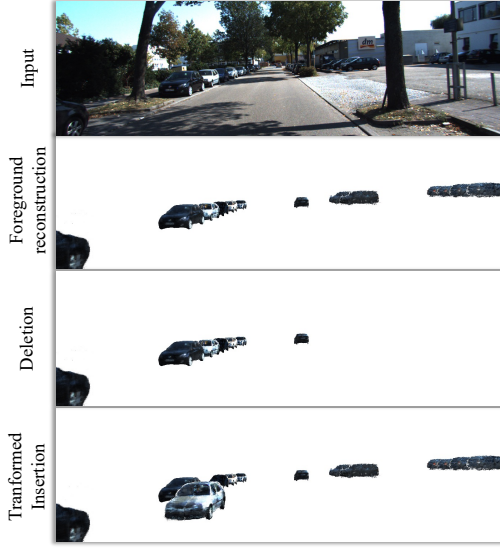
Fig. 8: Autonomous driving urban scene car instances editing showcases. Details on project page.
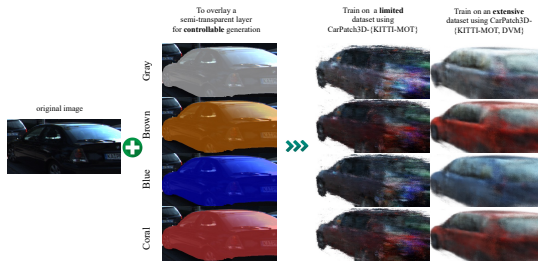

Fig. 9: Demonstration of **controllable** appearance editing using various synthesis inputs for our model.

[10], but our approach outperforms it in terms of handling unseen views, as illustrated in Figure 7.

Furthermore, our model's ability to use larger datasets enables new applications. We can now controllably and separately edit the appearance of objects without modifying their geometry features. This is demonstrated in Figure 9, where a transparent color mask is applied to the original patch to achieve the desired appearance. The figure shows that our model enables controllable appearance editing, and this ability improves as the dataset scale increases.

V. CONCLUSION

In this paper, we presented Car-Studio, a pipeline for learning car's neural radiance fields from in-the-wild 2D images. Our pipeline includes a data processing component and a neural radiance field component.

We collected CarPatch3D, a dataset that provides multi-perspective images of cars by processing images from existing datasets with different design purposes to address the gap in existing autonomous driving datasets that often provide front or rear views of cars.

Our designed Car-NeRF, a canonical, cone-tracing neural radiance field with extra segmentation loss, enables our model to learn from single-view supervision, anti-aliasing, and sharp

contours, leading to competitive performance in rendering realistic images of car objects.

Moreover, the trained Car-NeRF model can serve as a component for an editable autonomous driving simulator. As the scale of CarPatch3D expands, we anticipate that our model's performance in completely invisible perspective synthesis and appearance editing will continue to improve.

In our future research, we will undertake a deeper investigation into the impact of dynamic lighting conditions and reflective surfaces within the environment. Moreover, we will focus on refining the controllable editing effects on the car's surface to enhance its realism and visual fidelity.

REFERENCES

[1] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.

[2] D. Park, R. Ambrus, V. C. Guizilini, J. Li, and A. Gaidon, "Is pseudo-lidar needed for monocular 3d object detection?" *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3122–3132, 2021.

[3] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. B. Girshick, "Segment anything," *ArXiv*, vol. abs/2304.02643, 2023.

[4] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *ArXiv*, vol. abs/2003.08934, 2020.

[5] N. Müller, A. Simonelli, L. Porzi, S. R. Bulò, M. Nießner, and P. Kontschieder, "Autorf: Learning 3d object radiance fields from single view observations," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3961–3970, 2022.

[6] Z. Xie, J. Zhang, W. Li, F. Zhang, and L. Zhang, "S-nerf: Neural radiance fields for street views," *ArXiv*, vol. abs/2303.00749, 2023.

[7] A. Kundu, K. Genova, X. Yin, A. Fathi, C. Pantofaru, L. J. Guibas, A. Tagliasacchi, F. Dellaert, and T. A. Funkhouser, "Panoptic neural fields: A semantic object-aware neural scene representation," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12 861–12 871, 2022.

[8] H. Turki, J. Y. Zhang, F. Ferroni, and D. Ramanan, "Suds: Scalable urban dynamic scenes," *ArXiv*, vol. abs/2303.14536, 2023.

[9] Y. Xu, M. Chai, Z. Shi, S. Peng, I. Skorokhodov, A. Siarohin, C. Yang, Y. Shen, H.-Y. Lee, B. Zhou, and S. Tulyakov, "Discoscene: Spatially disentangled generative radiance fields for controllable 3d-aware scene synthesis," *ArXiv*, vol. abs/2212.11984, 2022.

[10] J. Ost, F. Mannan, N. Thuerey, J. Knodt, and F. Heide, "Neural scene graphs for dynamic scenes," *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2855–2864, 2020.

[11] B. Shen, X. Yan, C. Qi, M. Najibi, B. Deng, L. J. Guibas, Y. Zhou, and D. Anguelov, "Gina-3d: Learning to generate implicit neural assets in the wild," *ArXiv*, vol. abs/2304.02163, 2023.

[12] Y. Chen, F. Rong, S. Duggal, S. Wang, X. Yan, S. Manivasagam, S. Xue, E. Yumer, and R. Urtasun, "Geosim: Realistic video simulation via geometry-aware composition for self-driving," *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7226–7236, 2021.

[13] J. Y. Zhang, G. Yang, S. Tulsiani, and D. Ramanan, "Ners: Neural reflectance surfaces for sparse-view 3d reconstruction in the wild," in *Neural Information Processing Systems*, 2021.

[14] J. Guo, N. Deng, X. Li, Y. Bai, B. Shi, C. Wang, C. Ding, D. Wang, and Y. Li, "Streetsurf: Extending multi-view implicit surface reconstruction to street views," *ArXiv*, vol. abs/2306.04988, 2023.

[15] M. Tancik, V. Casser, X. Yan, S. Pradhan, B. Mildenhall, P. P. Srinivasan, J. T. Barron, and H. Kretzschmar, "Block-nerf: Scalable large scene neural view synthesis," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8238–8248, 2022.

[16] Z. Wang, T. Shen, J. Gao, S. Y. Huang, J. Munkberg, J. Hasselgren, Z. Gojcic, W. Chen, and S. Fidler, "Neural fields meet explicit geometric representation for inverse rendering of urban scenes," *ArXiv*, vol. abs/2304.03266, 2023.

[17] Y. Yang, Y. Yang, H. Guo, R. Xiong, Y. Wang, and Y. Liao, "Urban-giraffe: Representing urban scenes as compositional generative neural feature fields," *ArXiv*, vol. abs/2303.14167, 2023.

[18] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *International Symposium on Field and Service Robotics*, 2017.

[19] A. Dosovitskiy, G. Ros, F. Codevilla, A. M. López, and V. Koltun, "Carla: An open urban driving simulator," *ArXiv*, vol. abs/1711.03938, 2017.

[20] "Generating ai-based potential accident scenarios for autonomous vehicles," https://developer.nvidia.com/blog/generating-ai-based-accident-scenarios-for-autonomous-vehicles/.

[21] A. Kadian, J. Truong, A. Gokaslan, A. Clegg, E. Wijmans, S. Lee, M. Savva, S. Chernova, and D. Batra, "Sim2real predictivity: Does evaluation in simulation predict real-world performance?" *IEEE Robotics and Automation Letters*, vol. 5, pp. 6670–6677, 2019.

[22] P. Cai, H. Wang, H. Huang, Y. Liu, and M. Liu, "Vision-based autonomous car racing using deep imitative reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 6, pp. 7262–7269, 2021.

[23] J. Vertens and W. Burgard, "Realistic real-time simulation of rgb and depth sensors for dynamic scenarios using augmented image based rendering," *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8181–8188, 2022.

[24] E. Heiden, D. Millard, E. Coumans, Y. Sheng, and G. S. Sukhatme, "Neuralsim: Augmenting differentiable simulators with neural networks," *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9474–9481, 2020.

[25] M. Adamkiewicz, T. Chen, A. Caccavale, R. Gardner, P. Culbertson, J. Bohg, and M. Schwager, "Vision-only robot navigation in a neural radiance world," *IEEE Robotics and Automation Letters*, vol. PP, pp. 1–1, 2021.

[26] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3354–3361, 2012.

[27] J. Huang, B. Chen, L. Luo, S. Yue, and I. Ounis, "Dvm-car: A large-scale automotive dataset for visual marketing research and applications," *2022 IEEE International Conference on Big Data (Big Data)*, pp. 4140–4147, 2021.

[28] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, "Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields," *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5835–5844, 2021.

[29] C. Gao, A. Saraf, J. Kopf, and J.-B. Huang, "Dynamic view synthesis from dynamic monocular video," *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5692–5701, 2021.

[30] J. Gu, A. Trevithick, K.-E. Lin, J. M. Susskind, C. Theobalt, L. Liu, and R. Ramamoorthi, "Nerfdiff: Single-image view synthesis with nerf-guided distillation from 3d-aware diffusion," *ArXiv*, vol. abs/2302.10109, 2023.

[31] D. Watson, W. Chan, R. Martin-Brualla, J. Ho, A. Tagliasacchi, and M. Norouzi, "Novel view synthesis with diffusion models," *ArXiv*, vol. abs/2210.04628, 2022.

[32] R. Liu, R. Wu, B. V. Hoorick, P. Tokmakov, S. Zakharov, and C. Vondrick, "Zero-1-to-3: Zero-shot one image to 3d object," *ArXiv*, vol. abs/2303.11328, 2023.

[33] X. Liu, S. hong Kao, J. Chen, Y.-W. Tai, and C.-K. Tang, "Deceptive-nerf: Enhancing nerf reconstruction using pseudo-observations from diffusion models," *ArXiv*, vol. abs/2305.15171, 2023.

[34] G. Qian, J. Mai, A. Hamdi, J. Ren, A. Siarohin, B. Li, H.-Y. Lee, I. Skorokhodov, P. Wonka, S. Tulyakov, and B. Ghanem, "Magic123: One image to high-quality 3d object generation using both 2d and 3d diffusion priors," *ArXiv*, vol. abs/2306.17843, 2023.

[35] A. Yu, V. Ye, M. Tancik, and A. Kanazawa, "pixelnerf: Neural radiance fields from one or few images," *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4576–4585, 2020.

[36] A. Kanazawa, S. Tulsiani, A. A. Efros, and J. Malik, "Learning category-specific mesh reconstruction from image collections," *ArXiv*, vol. abs/1803.07549, 2018.

[37] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang, "Pixel2mesh: Generating 3d mesh models from single rgb images," in *European Conference on Computer Vision*, 2018.

[38] P. Henderson, V. Tsiminaki, and C. H. Lampert, "Leveraging 2d data to learn textured 3d mesh generation," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7495–7504, 2020.

[39] J. Munkberg, J. Hasselgren, T. Shen, J. Gao, W. Chen, A. Evans, T. Müller, and S. Fidler, "Extracting triangular 3d models, materials, and lighting from images," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8270–8280, 2021.

[40] S. Wu, C. Rupprecht, and A. Vedaldi, "Unsupervised learning of probably symmetric deformable 3d objects from images in the wild," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–10, 2019.

[41] G. Gkioxari, J. Malik, and J. Johnson, "Mesh r-cnn," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9784–9794, 2019.

[42] S. Saito, Z. Huang, R. Natsume, S. Morishima, A. Kanazawa, and H. Li, "Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2304–2314, 2019.

[43] A. Kundu, Y. Li, and J. M. Rehg, "3d-rcnn: Instance-level 3d object reconstruction via render-and-compare," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3559–3568, 2018.

[44] J. Wang, S. Manivasagam, Y. Chen, Z. Yang, I. A. Bârsan, A. Yang, W.-C. Ma, and R. Urtasun, "Cadsim: Robust and scalable in-the-wild 3d reconstruction for controllable sensor simulation," in *Conference on Robot Learning*, 2022.

[45] S. Zakharov, R. Ambrus, V. C. Guizilini, D. Park, W. Kehl, F. Durand, J. B. Tenenbaum, V. Sitzmann, J. Wu, and A. Gaidon, "Single-shot scene reconstruction," in *Conference on Robot Learning*, 2021.

[46] B. Yang, Y. Zhang, Y. Xu, Y. Li, H. Zhou, H. Bao, G. Zhang, and Z. Cui, "Learning object-compositional neural radiance field for editable scene rendering," *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 13 759–13 768, 2021.

[47] A. Chen, Z. Xu, F. Zhao, X. Zhang, F. Xiang, J. Yu, and H. Su, "Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo," *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14 104–14 113, 2021.

[48] K.-E. Lin, Y.-C. Lin, W.-S. Lai, T.-Y. Lin, Y. Shih, and R. Ramamoorthi, "Vision transformer for nerf-based view synthesis from a single input image," *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 806–815, 2022.

[49] M. S. M. Sajjadi, H. Meyer, E. Pot, U. M. Bergmann, K. Greff, N. Radwan, S. Vora, M. Lucic, D. Duckworth, A. Dosovitskiy, J. Uszkoreit, T. A. Funkhouser, and A. Tagliasacchi, "Scene representation transformer: Geometry-free novel view synthesis through set-latent scene representations," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6219–6228, 2021.

[50] W. J. Jang and L. de Agapito, "Codenerf: Disentangled neural radiance fields for object categories," *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12 929–12 938, 2021.

[51] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015.

[52] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[53] M. Tancik, E. Weber, E. Ng, R. Li, B. Yi, J. Kerr, T. Wang, A. Kristoffersen, J. Austin, K. Salahi, A. Ahuja, D. McAllister, and A. Kanazawa, "Nerfstudio: A modular framework for neural radiance field development," *ArXiv*, vol. abs/2302.04264, 2023.

[54] "Autorf-pytorch," https://github.com/skyhehe123/AutoRF-pytorch.

[55] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," *ArXiv*, vol. abs/1908.03265, 2019.

[56] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, vol. 2, pp. 1398–1402 Vol.2, 2003.

[57] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 2018.