

S-NeRF++: Autonomous Driving Simulation via Neural Reconstruction and Generation

Yurui Chen Junge Zhang Ziyang Xie Wenye Li Feihu Zhang Jiachen Lu Li Zhang

Abstract—Autonomous driving simulation system plays a crucial role in enhancing self-driving data and simulating complex and rare traffic scenarios, ensuring navigation safety. However, traditional simulation systems, which often heavily rely on manual modeling and 2D image editing, struggled with scaling to extensive scenes and generating realistic simulation data. In this study, we present S-NeRF++, an innovative autonomous driving simulation system based on neural reconstruction. Trained on widely-used self-driving datasets such as nuScenes and Waymo, S-NeRF++ can generate a large number of realistic street scenes and foreground objects with high rendering quality as well as offering considerable flexibility in manipulation and simulation. Specifically, S-NeRF++ is an enhanced neural radiance field for synthesizing large-scale scenes and moving vehicles, with improved scene parameterization and camera pose learning. The system effectively utilizes noisy and sparse LiDAR data to refine training and address depth outliers, ensuring high quality reconstruction and novel-view rendering. It also provides a diverse foreground asset bank through reconstructing and generating different foreground vehicles to support comprehensive scenario creation. Moreover, we have developed an advanced foreground-background fusion pipeline that skillfully integrates illumination and shadow effects, further enhancing the realism of our simulations. With the high-quality simulated data provided by our S-NeRF++, we found the perception methods enjoy performance boost on several autonomous driving downstream tasks, which further demonstrate the effectiveness of our proposed simulator.

I. INTRODUCTION

The recent surge in street view data acquisition by autonomous vehicles has opened new avenues in the field of autonomous driving simulations. This data can be effectively employed to construct simulation systems that are not only capable of generating new datasets but also proficient in synthesizing corner case scenarios. Historically, this area has seen the introduction of various innovative methods, reflecting a significant advancement in autonomous driving technology. [1], [2] leverage virtual 3D modeling and rendering engines to establish comprehensive simulation systems. Another notable technique, as proposed by [3], involves the innovative use of image warping to generate simulations with novel perspectives. [4] focuses on reconstructing vehicle meshes from self-driving datasets and integrating them into videos based on depth information. [5] takes advantage of Generative Adversarial Networks (GANs) to enhance the realism of edited videos, aligning them more closely with given datasets. Despite their merits, these methods either demand

Li Zhang is the corresponding author (lizhangfd@fudan.edu.cn). Yurui Chen, Junge Zhang, Ziyang Xie, Wenye Li, Jiachen Lu and Li Zhang are with Fudan University. Feihu Zhang is with University of Oxford.

substantial computing resources or exhibit limitations in terms of simulation flexibility.

The recent emergence of Neural Radiance Fields (NeRFs) [6], a kind of 3D reconstruction algorithm which have shown impressive performance on photo-realistic novel view rendering, gives us a new way of thinking about construction simulation system. Original NeRF is usually designed for object-centric scenes and require camera views to be heavily overlapped (as shown in Figure 1(a)). However, urban driving data are often collected in the unbounded outdoor scenes (e.g. nuScenes [7] and Waymo [8] datasets). The camera placements of such data acquisition systems are usually in a panoramic settings without object-centric camera views (Figure 1(b)). This presents a challenge for NeRF in achieving high-quality reconstructions in these contexts.

We introduce a novel NeRF design, S-NeRF++, as the core of our driving simulation system for synthesizing both expansive background scenes and dynamic foreground vehicles. To address the challenge NeRF faces in accurately reconstructing large scenes, we leverage dense depth supervision to guide the NeRF training. The per-pixel dense depth maps are completed from the sparse LiDAR points using off-the-shelf depth completion model. In order to obtain more reliable depth supervision, we introduce to use learnable confidence map to evaluate the depth quality and thus ensuring more robust and stable optimization.

To construct an effective simulation system, it's essential to include manipulable foreground objects for dynamic and realistic scenario generation. [9], [10] solve this problem by decoupling the foreground reconstruction from training logs, enabling manipulation of foreground vehicles. However, their approach is limited to reusing vehicles within the same scene and does not consider natural composition of foreground objects and background scenes, which leads to unrealistic generation. To address this issue, we have developed a sophisticated foreground-background fusion algorithm. This solution enhances the realism of the generated virtual data and ensures that the lighting of the inserted foreground objects seamlessly adapts to various background environmental conditions, such as night and cloudy settings. Furthermore, our sampling strategy facilitates the automated placement of foreground objects, streamlining the process of integrating these elements into diverse scenarios without manual intervention. This approach significantly enhances the efficiency and flexibility of the simulation, allowing for rapid and diverse scene generation.

We proposed to use advanced generative algorithms to further enhance the variety of our foreground data. In order to ensure a comprehensive and diverse range of scenarios, we

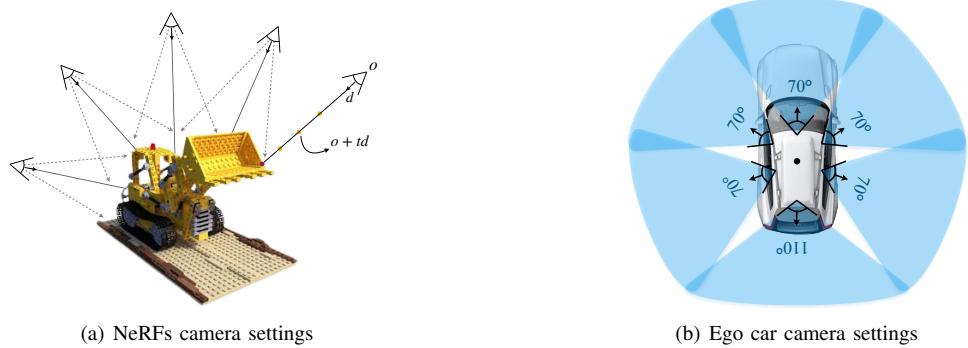


Fig. 1. Problem illustration. (a) Conventional NeRFs [6], [12] require object-centric camera views with large overlaps. (b) In the challenging large-scale outdoor driving scenes [7], [8]), the camera placements for data collection are usually in a panoramic view settings. Rays from different cameras barely intersect with others in the unbounded scenes. The overlapped field of view between adjacent cameras is too small to be effective for training the existing NeRF models.

leverage category-specific, fine-tuned Latent Diffusion Model (LDM) to generate various foreground assets from uncolored meshes, including but not limited to people, bicycles, and motorcycles, which are not present in the training dataset. Beyond just utilizing generative algorithms, we also broaden our foreground asset library by aggregating and reconstructing a variety of objects from multiple datasets. This comprehensive approach not only diversifies the types of foreground objects available but also enhances the overall richness and authenticity of our asset library.

Our preliminary work S-NeRF [11] has been published in ICLR 2023. We have extended our conference version as follows: (i) Enhancing the street-view reconstruction module which now can model dynamic objects together with the static background and is highly improved in rendering speed and quality. (ii) Constructing a foreground asset library through generative methods and reconstruction techniques. (iii) Developing a self-driving environment simulation system which can automatically generate novel and realistic simulation data by editing scenes and changing view perspective. (iv) The quality of data produced by our simulation system is verified by measuring the gap between real and simulation data to train downstream tasks in nuScenes and Waymo. Our simulation data with part of real data can outperform 10 times real data in image semantic segmentation, monocular 3D detection and multi-view 3D detection.

II. RELATED WORK

A. Data-driven Self-driving Simulation System

Data-driven self-driving Simulation System aims to simulate the driving environment or novel data for the self-driving tasks from the given sensor data collected by a vehicle driving in the urban. Different from the simulation platforms [1], [2] based on virtual 3D modeling and rendering engine. Data-driven simulation has less computation demand and less sim-to-real gap, making it a more efficient and realistic approach for training and testing autonomous driving systems. [3] use warping images to get novel view simulation. [4] reconstructs meshes of vehicle from the self-driving dataset and inserts the reconstructed cars into given videos through the depth, while it can not synthesis the novel view of the urban scene leading to the lack of diversity. [14] simulates the LiDAR data through

aggregating the LiDAR scans, build meshes and performing raycasting to generate point clouds. [5] uses GANs to edit videos more reasonable and similar to the given dataset. [9], [10] utilize NeRF-based methods decoupled the foreground from training logs to possess the ability of rendering scenes when manipulating foreground vehicles to some extent.

However, these methods frequently struggle to generate novel views of urban scenes, suffer from a lack of diversity in the generated data, and encounter difficulties in accurately rendering foreground vehicles when manipulating them within the scenes. These issues can hinder the effectiveness and realism of the simulation system.

B. 3D Reconstruction and Novel View Synthesis

3D reconstruction of objects or scenes through images has always been a popular but difficult research topic. Traditional reconstruction and novel view rendering [15] often rely on Structure-from-Motion (SfM), multi-view stereo and graphic rendering [16]. Recently, from differentiable rasterization to volume rendering, learning-based approaches [17], [18], [19] have been widely used in 3D scene and object reconstruction. They often use a deep neural network to extract the feature of the sensor inputs and learn various representations, such as voxels [20], [17], patches [21] and meshes [22], [4], to reconstruct the 3D geometry of the scene.

Recently, Neural Radiance Fields (NeRF) [6] has been widely used for 3D reconstruction. To further speed up training, various types of NeRFs [23], [24] have been proposed. Explicit grid occupancy is used in [25], [26] for more efficient query by raytracing, hash grid based method like [27] balance speed and storage.

a) Unbounded street-view NeRF: Many NeRFs have been proposed to address the challenges of large-scale outdoor scenes. NeRF in the wild [28] applies appearance and transient embeddings to solve the lighting changes and transient occlusions. Neural plenoptic sampling [29] proposes to use a Multi-Layer Perceptron (MLP) as an approximator to learn the plenoptic function and represent the light-field in NeRF. Mip-NeRF [12] develops a conical frustum encoding to better encode the scenes at a continuously-valued scale. Using Mip-NeRF as a base block, Block-NeRF [30] employs a block-combination strategy along with pose refinement, appearance,

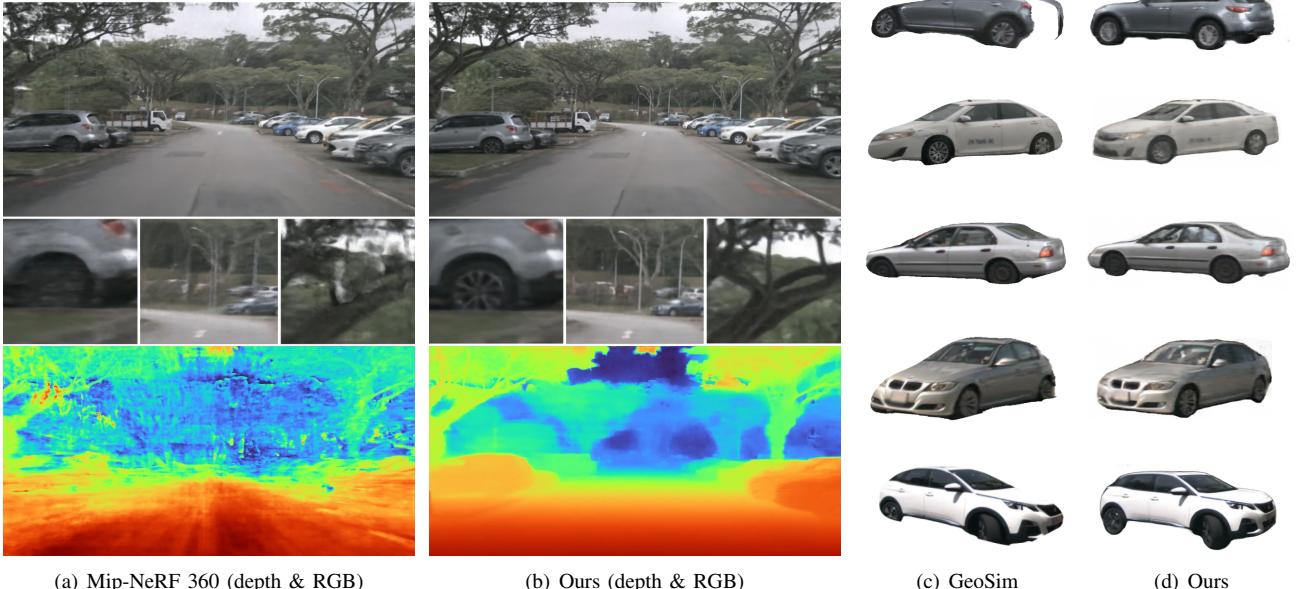


Fig. 2. Performance illustration in novel view rendering on a challenging nuScenes scene [7], (a) the state-of-the-art method [13] produces poor results with blurred texture details and plenty of depth errors, (b) our S-NeRF can achieve accurate depth maps and fine texture details with fewer artifacts. (d) Our method can also be used for the reconstruction of moving vehicles which is impossible for previous NeRFs. It can synthesize better novel views compared with the mesh method [4].

and exposure embedding on large-scale scenes. Mip-NeRF 360 [13] improves the Mip-NeRF for unbounded scenes by contracting the whole space into a bounded area to get a more representative position encoding. Zip-NeRF [31] utilize hash-grid encoding and contracting space with multi-sampling in conical frustum to achieve better synthesis quality and quicker rendering speed. For scenes with moving objects like vehicles in the road, some methods [32], [33] try to model moving objects decoupling from the static background by their 3D box.

b) *3D object reconstruction*: It is often difficult to obtain high-quality surface of the object with NeRFs expressed in volume density. Signed Distance Field (SDF) based method [34] who learn the absolute distance from a point to the object surface can reconstruct the object's surface more precisely, and allow us to get the mesh of the objects by MarchingCube [35]. Other method [4] use a learned deformable mesh by differentiable rasterization to make the construction. To model the non-rigid objects like human avatar, more prior knowledge is used like SMPL [36]. [37] uses NeRF with coordinate mapping from canonical space to the deformable space reconstruct the avatar from multi-camera videos. [38] allow the constructed human NeRF rendered in different poses without finetuning. [39] extracts the textured mesh by neural rendering method.

C. 3D Object Generation

3D object generation aims to get 3D representation by text-prompt or noises which does not need the ground truth training data like reconstruction methods.

2D image generation has been well developed. Generative adversarial networks (GANs) [40], [41] and variety of variational autoencoders (VAEs) [42] have synthesized high-quality images partly because of the massive 2D training

data. Recently, Latent Diffusion Models (LDM) [43], [44] show strong generative ability for the images from different conditions.

The key of 3D generation is how to make the representation consistency with the view poses. It is hard to generate the 3D models directly for the lack of 3D ground truth data. Many work generate 3d models by lifting the 2D image generation models' knowledge to 3D representation like NeRF through optimization process. [45] use the frozen discriminator in the pre-trained GAN to make the renderings' distribution close the GAN. [46], [47] apply Score Distillation Sampling (SDS) who use LDMs to realize text-to-3D. [48] texture for a given mesh through inpainting the bare mesh patch-by-patch.

III. SIMULATION SYSTEM DEPLOYMENT

Our goal is to create a self-driving simulation system that can produce photo-realistic street views. This system will also allow for the manipulation of foreground objects, enabling object insertion, movement, or alteration, by utilizing a comprehensive autonomous driving dataset. This dataset encompasses a wide array of data, including images, precise camera positions, synchronized LiDAR point clouds. These elements are crucial for accurately reconstructing urban street scenes.

In this section, we first introduce how we construct our simulation asset bank by reconstructing and generating diverse background scenes and foreground objects using common self-driving dataset. We further propose a automatic foreground object insertion strategy to simplify the manipulation of the foreground objects as well as ensuring the geometric consistency when rendering novel views. Additionally, we employ a learning-based image refinement module to enhance lighting consistency and overall realism of the generated scenes. This

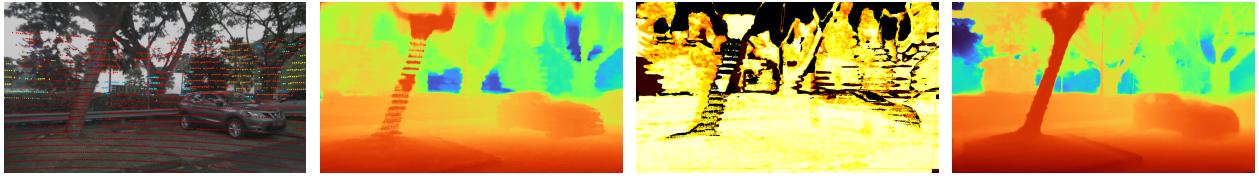


Fig. 3. Our depth supervision and rendering pipeline.

approach is designed to minimize the simulation-to-reality gap, facilitating easier application in real-world conditions. Our comprehensive strategy enables the creation of a versatile and adaptable simulated environment, crucial for autonomous driving research and capable of accommodating a wide range of environmental conditions.

A. Preliminary

Neural Radiance Field (NeRF) represents a scene as a continuous radiance field and learns a mapping function $f : (\mathbf{x}, \theta) \rightarrow (\mathbf{c}, \sigma)$. It takes the 3D position $\mathbf{x}_i \in \mathbb{R}^3$ and the viewing direction θ_i as input and outputs the corresponding color \mathbf{c}_i with its differential density σ_i . The mapping function is realized by two successive multi-layer perceptrons (MLPs).

NeRF uses the volume rendering to render image pixels. For each 3D point in the space, its color can be rendered through the camera ray $\mathbf{r}(t) = \mathbf{o} + td$ with N stratified sampled bins between the near and far bounds of the distance. The output color is rendered as:

$$\hat{\mathbf{I}}(\mathbf{r}) = \sum_{i=1}^N T_i (1 - e^{-\sigma_i \delta_i}) \mathbf{c}_i, \quad T_i = \exp \left(- \sum_{j=1}^{i-1} \sigma_j \delta_j \right) \quad (1)$$

where \mathbf{o} is the origin of the ray, T_i is the accumulated transmittance along the ray, \mathbf{c}_i and σ_i are the corresponding color and density at the sampled point t_i . $\delta_j = t_{j+1} - t_j$ refers to the distance between the adjacent point samples.

B. Data Preparation

Reconstruction tasks using existing autonomous driving datasets can be challenging due to their complex nature, as these datasets are typically not tailored for reconstruction purposes. In this section, we address this challenge by introducing an advanced pipeline designed to reconstruct street views more effectively using the available data. This pipeline incorporates a decoupled camera pose preprocessing step for reconstructing the foreground vehicles and background scenes, a confidence-based dense depth supervision mechanism aimed at enhancing the overall geometric accuracy and a semantic distillation techniques to integrate semantic information into the reconstruction process for a more effective manipulation of the scene.

1) *Camera Pose Processing*: In previous NeRF implementations, Structure from Motion (SfM [49]) techniques often struggle to accurately reconstruct camera poses for self-driving data. This difficulty arises due to limited overlap in camera views, as depicted in Figure 1(b). Additionally, the

presence of large numbers of moving objects within the scene further complicates the feature matching process, reducing the success rate of SfM. To address this issue, we propose two distinct methods designed to separately reconstruct camera poses for the static background and the moving vehicles in the foreground.

a) *Background scenes*: For the static background, we use the camera parameters achieved by sensor-fusion SLAM and IMU of the self-driving cars [7], [8] and further reduce the inconsistency between multi-cameras with a learning-based pose refinement network. We follow Wang *et al.* [50] to implicitly learn a pair of refinement offsets $\Delta P = (\Delta R, \Delta T)$ for each original camera pose $P = [R, T]$, where $R \in SO(3)$ and $T \in \mathbb{R}^3$. The pose refine network can help us to ameliorate the error introduced in the SLAM algorithm, thereby enhancing the robustness of our system.

b) *Moving vehicles*: The method previously described is well-suited for static backgrounds. However, estimating camera poses for moving objects presents unique challenges due to the complex dynamics involving both the ego vehicle and the target objects. As illustrated in Figure 4, we compute the relative position \hat{P} between the camera and the target object. We use the center of the target object as the origin of the coordinate system. In the experiments, we use the 3D detectors (*e.g.* [51]) to detect the 3D bounding box and the center of the target object.

The ego car's center is the initial origin of our coordinate system, where P_b denotes the position of the target object, and P_i represents the position of camera i , with 5~6 cameras installed on the ego car. To simplify the complexity involved in tracking and estimating the poses of moving objects relative to the camera, as illustrated in Figure 3, we transform the coordinate system to center it around the target object and compute the relative camera pose to the object center.

$$\hat{P}_i = (P_i P_b^{-1})^{-1} = P_b P_i^{-1}, \quad P^{-1} = \begin{bmatrix} R^T & -R^T T \\ \mathbf{0}^T & 1 \end{bmatrix}. \quad (2)$$

Here, $P = \begin{bmatrix} R & T \\ \mathbf{0}^T & 1 \end{bmatrix}$ represents the old position of the camera or the target object. \hat{P}_i is the new relative camera position.

2) *Depth*: To improve the reconstruction quality of the scene, we proposed to use dense depth supervision to guided the model training. We first project the LiDAR point clouds onto the camera's image plane to acquire sparse depth information and then propagate these sparse depths to a dense depth map to supervise the NeRF training. We further create a confidence map to manage and mitigate depth outliers, which

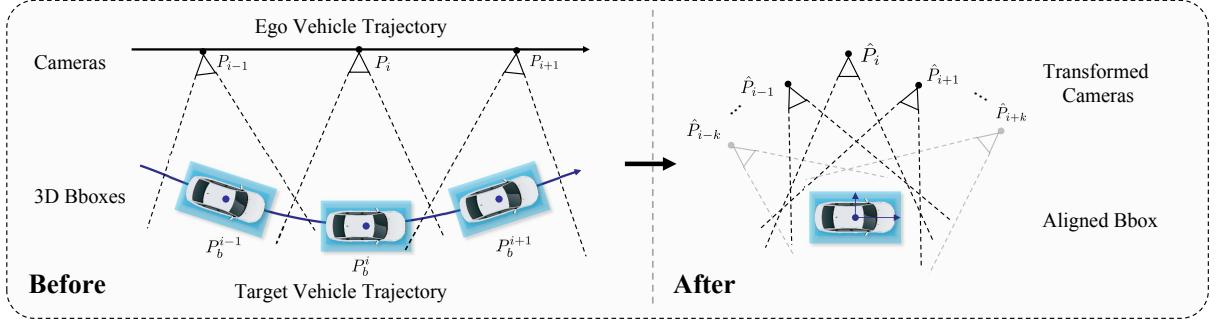


Fig. 4. Illustration of our camera transformation process for moving vehicles. During the data collection, the ego car (camera) is moving and the target car (object) is also moving. The virtual camera system treats the target car (moving object) as static and then compute the relative camera poses for the ego car's camera. These relative camera poses can be estimated through the 3D object detectors. After the transformation, only the camera is moving which is favorable in training NeRFs.

serves as a critical component in ensuring the accuracy and reliability of our depth data. Our depth confidence is defined as a learnable combination of the **perception confidence** and the **geometry confidence**. During the training, the confidence maps are jointly optimized with the color rendering for each input view.

We use NLSPN [52] to propagate the sparse depth to surrounding pixels. Although NLSPN performs well with 64-channel LiDAR data (*e.g.* KITTI [53], [54] dataset), it doesn't generate good results with nuScenes' 32-channel LiDAR data, which are too sparse for the depth completion network. As a result, we accumulate neighbour LiDAR frames to get much denser depths for NLSPN. These accumulated LiDAR data, however, contain a great quantity of outliers due to the moving objects, ill poses and occlusions, which give wrong depth supervisions. To address this problem, we design a robust confidence measurement that can be jointly optimized while training our NeRF as illustrated in Figure 3.

Our depth confidence is defined as a learnable combination of the **perception confidence** and the **geometry confidence**. During the training, the confidence maps are jointly optimized with the color rendering for each input view.

Perception confidence: To measure the accuracy of the depths and locate the outliers, we first use an warping operation ψ to reproject pixels $\mathbf{X} = (x, y, d)$ from the source images I_s to the target image I_t . Let P_s, P_t be the source and the target camera parameters, $d \in \mathcal{D}_s$ as the depth from the source view. The warping operation can be represented as:

$$\mathbf{X}_t = \psi(\psi^{-1}(\mathbf{X}_s, P_s), P_t) \quad (3)$$

ψ represents the warping function that maps 3D points to the camera plane and ψ^{-1} refers to the inverse operation from 2D to 3D points. Since the warping process relies on depth maps \mathcal{D}_s , the depth outliers can be located by comparing the source image and the inverse warping one. We introduce the RGB, SSIM [55] and the pre-trained VGG feature [56] similarities to measure the perception confidence in the pixel, patch structure, and feature levels:

$$\begin{aligned} C_{\text{rgb}} &= 1 - |\mathcal{I}_s - \hat{\mathcal{I}}_s|, \quad C_{\text{ssim}} = \text{SSIM}(\mathcal{I}_s, \hat{\mathcal{I}}_s), \\ C_{\text{vgg}} &= 1 - ||\mathcal{F}_s - \hat{\mathcal{F}}_s||. \end{aligned} \quad (4)$$

Where $\hat{\mathcal{I}}_s = \mathcal{I}_t(\mathbf{X}_t)$ is the warped RGB image, and the $\hat{\mathcal{F}}_s = \mathcal{F}_t(\mathbf{X}_t)$ refers to the feature reprojection. The receptive fields of these confidence maps increase from pixels and local patches to non-local regions. This gradual expansion helps in building strong and reliable measurements of confidence.

Geometry confidence: We further impose a geometry constrain to measure the geometry consistency of the depths and flows across different views. Given a pixel $\mathbf{X}_s = (x_s, y_s, d_s)$ on the depth image \mathcal{D}_s we project it to a set of target views using (3). The coordinates of the projected pixel $\mathbf{X}_t = (x_t, y_t, d_t)$ are then used to measure the geometry consistency. For the projected depth d_t , we compute its consistency with the original target view's depth $\hat{d}_t = D_t(x_t, y_t)$:

$$C_{\text{depth}} = \gamma(|d_t - \hat{d}_t|)/d_s, \quad \gamma(x) = \begin{cases} 0, & \text{if } x \geq \tau, \\ 1 - x/\tau, & \text{otherwise.} \end{cases} \quad (5)$$

For the flow consistency, we apply off-the-shelf optical flow prediction model [57] to compute the pixel's motions from the source image to the adjacent target views $f_{s \rightarrow t}$. This approach could further reduce inconsistency by aligning the pixel movements across different frames, ensuring a more coherent and seamless transition of depth. The flow consistency is then formulated as:

$$C_{\text{flow}} = \frac{\gamma \|\Delta_{x,y} - f_{s \rightarrow t}(x_s, y_s)\|}{\|\Delta_{x,y}\|}, \quad \Delta_{x,y} = (x_t - x_s, y_t - y_s). \quad (6)$$

Where τ is a threshold in γ to identify the outliers through the depth and flow consistencies.

Computation of confidence: Figure 5 shows the process of how we generate the confidence maps. We compute the depth confidence by reprojection the depth maps to other views and measure the confidence level based on the reprojection and geometry consistency. Specifically, given the LiDAR positions of a set of consecutive frames $[P_{t-1}, P_t, P_{t+1}]$ at different time $[t-1, t, t+1]$, we project the 3D LiDAR points at time $t-1$ and $t+1$ to t by ΔP between P_t and P_{t-1}/P_{t+1} . This is similar to the mapping function ψ in Eq. (3). Most outliers of the moving objects and other regions can be removed by the consistency check using the optical flow and the depth projection (Eq.(5))

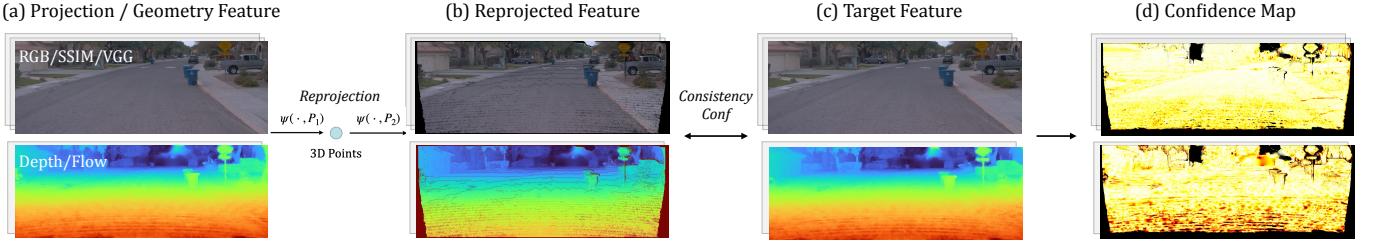


Fig. 5. Illustration of the confidence computation process.

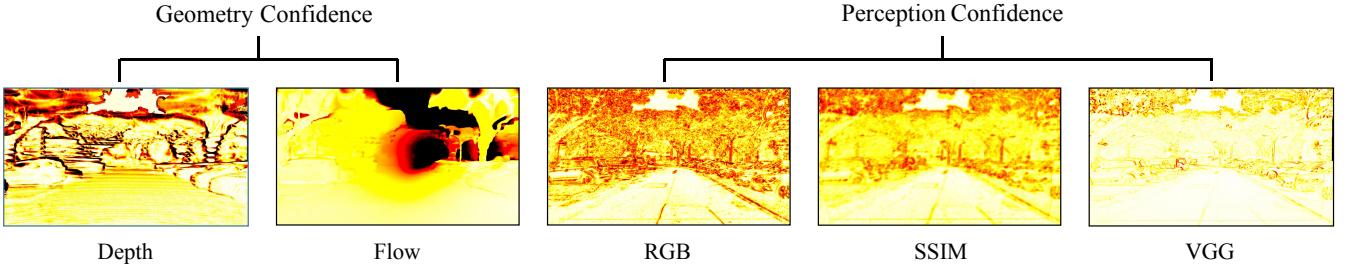


Fig. 6. Visualization of each confidence component. Brighter regions indicate higher confidence. Geometry confidences (flow and depth) represents the geometry consistency as computed in Fig. 2(a.1). The perception confidence measure the photometric, local structure and feature consistency as computed in Fig. 2(a.2).

and (6)). The rest outliers can also be handled by the proposed confidence-guided learning. In Figure 6, we visualize different confidence components. Depth and optical flow confidence maps focus on geometry consistency between adjacent frames, while RGB, SSIM and VGG confidence maps compute the photometric consistency from pixel, structure and feature level.

Learnable confidence combination: To compute the final confidence map, we assign learnable weights ω for each individual confidence metric and jointly optimize them during the training. The final confidence map can be learned as $\hat{C} = \sum_i \omega_i C_i$, where $\sum_i \omega_i = 1$. The $i \in \{rgb, ssim, vgg, depth, flow\}$ represents the optional confidence metrics. The learnable weights ω adapt the model to automatically focus on correct confidence. One thing to note that we predict a sky map and set the depth of the sky at a sufficiently large value.

3) Semantic Labels: The semantic labels are crucial to the scene understanding in autonomous driving environments, serving for the subsequent decision-making and control. Here, we first consider the most difficult cases (*e.g.* nuScenes scenes) where there is no any annotated label from the collected driving data (images and LiDAR points). Given unlabeled real images collected from multiple cameras of the self-driving cars, we train a SegFormer [58] model, on the mixture of other datasets including Cityscapes [59], Mapillary [60], BDD [61], IDD [62] to compute weak labels that serve as inputs to the NeRF reconstruction model. To achieve better cross-dataset generalization of the SegFormer and avoid conflicts in the label definition, we utilize the learning settings and label merging strategy in the [63].

Considering that the generated weak labels may have many outliers, to avoid the influence of the outliers and generate

more accurate 3D point labels, we take full use of multi-view geometric and video spacial temporal consistency in our NeRF reconstruction.

In some other cases, when there is a small number of labeled images or LiDAR frames, we can also leverage the existing ground-truth labels for more robust label generation. For example, in the nuScenes dataset, a small part of the LiDAR frames (about 1/10) was labeled with semantic annotations. We take the sparse 3D point labels along with the weak 2D image labels to learn more accurate semantic radiance fields.

C. Representation of Street Scenes

Recent advances in neural rendering techniques have enabled efficient and high-quality scene reconstruction through versatile representation. Mip-NeRF 360 [13] is a MLP based network for unbounded scene representation using frustum-encoding largely improve the rendering quality. Zip-NeRF [31] ingeniously integrates hash-grid and frustum-encoding methods to enhance scene reconstruction techniques. Drawing upon the strengths of each approach, it offers the high efficiency of the hash-grid method for representation, while leveraging the frustum-encoding method's excellent anti-aliasing capabilities under multi-resolution observation. This hybrid approach has proven successful in delivering more accurate and efficient scene reconstructions. In our simulation pipeline, we implement two types of backbone networks: an MLP based (Mip-NeRF 360) and a hash grid based (Zip-NeRF) to represent the reconstructed scenes. Our experiments demonstrate the efficacy of our method with both MLP-based and grid-based backbones. We enhance the MLP decoder to make it not only

outputs the density σ and RGB value \mathbf{c} , but also outputs the semantic logits \mathbf{s} , enabling it to predict semantic labels.

However, models designed for static scenes are less effective in outdoor environments with dynamic elements like cars, trucks, buses, and pedestrians. These moving objects can hinder the learning process of scene representation. Inspired by [33], [32], we model the dynamic objects as individual instances and integrate the learning process of foreground and background during the scene reconstruction. This approach allows for the comprehensive reconstruction of scenes, incorporating both moving and static elements. Figure 7 illustrates the combination process of moving objects with the static background in our reconstruction method.

We describe i^{th} object as a series of time-dependent 3d bounding box with 6DoF parameters $[\mathbf{R}_{i,t}, \mathbf{T}_{i,t}]$ in the background world and a size $dim_i = [w_i, h_i, l_i]^T$. We track all the objects in a training scene, and apply each object an instance vector \mathbf{v}_i . We can easily check if one ray is intersected with a certain object box at time t , and for each sampled point on the ray, we decode its color and density by two branches. If the point is in i^{th} object box, we transform the point position \mathbf{x} and ray direction \mathbf{d} to the uniformed object box coordinate system from the background world coordinate system by

$$\mathbf{x}_o = \mathbf{R}_{i,t}^\top (\mathbf{x} - \mathbf{T}_{i,t}) \cdot \frac{1}{dim_i}, \quad \mathbf{d}_o = \mathbf{R}_{i,t}^\top \mathbf{d}. \quad (7)$$

We then query a category-specific INGP [27] whose weights are shared by the objects in the same class to get the feature and after concatenated its instant vector \mathbf{v}_i , we decode the color, semantic probability and density by a instance MLP. While if the point is not in any object box, we get the attributes through the background Zip-NeRF based network. We then aggregate all the points along the ray for volume rendering to get the final integral results.

The overview of our reconstruction method is shown in Figure 8. We first propagate the sparse LiDAR points into a dense depth map and compute the geometry and the perception confidence maps. We use learnable combination to achieve the final confidence maps to reduce the influence of depth outliers.

Representing the urban scenes in two brunch way makes our method more flexible to the reconstruction of street view. In order to train NeRF with a limited number of views (e.g. 2 ~ 6 image views), we compute the dense depth maps for the moving cars as an extra supervision. We follow GeoSim [4] to reconstruct coarse mesh from multi-view images and the sparse LiDAR points. After that, a differentiable neural renderer [64] is used to render the corresponding depth map with the camera parameter (Section III-B1b). The backgrounds are masked during the training by an instance segmentation network [65].

D. Reconstruction Learning

In this part, due to the extreme complication of the driving scenes, we demonstrate all the effective losses including the supervision and regularization for the training process of reconstruction. The overall loss, represented as \mathcal{L} , is formulated as follows:

$$\mathcal{L} = \mathcal{L}_{sup} + \mathcal{L}_{reg}. \quad (8)$$

The supervision loss, denoted as \mathcal{L}_{sup} , can be expressed as:

$$\mathcal{L}_{sup} = \mathcal{L}_{rgb} + \lambda_d \mathcal{L}_{depth} + \lambda_L \mathcal{L}_{pc} + \lambda_s \mathcal{L}_{sem}. \quad (9)$$

$$\mathcal{L}_{rgb} = |\hat{\mathbf{I}} - \mathbf{I}|, \quad \mathcal{L}_{depth} = \hat{\mathcal{C}}|\hat{\mathcal{D}} - \mathcal{D}|, \quad (10)$$

$$\mathcal{L}_{pc} = \sum_{\mathbf{r} \in \mathcal{R}_L} |\hat{\mathcal{D}}(\mathbf{r}) - \mathcal{D}_L(\mathbf{r})|. \quad (11)$$

In our loss expression, we simplify the notation by omitting the summation over rays $\sum_{\mathcal{R}}(\cdot)$. Although, in practice, we sample $|\mathcal{R}|$ rays ays in each training step. We employ an RGB L1 loss, denoted as \mathcal{L}_{rgb} , and a disparity L1 loss with masking, represented as \mathcal{L}_{depth} . In this context, $\hat{\mathcal{C}}$ denotes the weights of learnable confidence map, $\hat{\mathcal{D}}$ refers to the disparity map rendered by our reconstruction network, \mathcal{D} indicates the inversion of dense depth map propagated from the sparse LiDAR data.

Furthermore, we use sparse point clouds \mathcal{R}_L as the ground truth depth of a ray \mathbf{r} captured by the LiDAR sensor. To enhance geometric accuracy, we include a disparity loss term, \mathcal{L}_{pc} , which aligns with the sparse point cloud data. This approach helps in refining the geometry in our reconstructions.

For our reconstruction backbone outputs semantic label logits \mathbf{s}_i , we can calculate the semantic label of a ray by following Eq. 1:

$$\hat{\mathbf{S}}(\mathbf{r}) = \sum_{i=1}^N T_i (1 - e^{-\sigma_i \delta_i}) \text{Softmax}(\mathbf{s}_i). \quad (12)$$

The semantic loss can be calculated as the cross entropy loss of the rendered semantic map $\hat{\mathbf{S}}$ and the supervision semantic label \mathbf{S} :

$$\mathcal{L}_{sem} = \text{CrossEntropy}(\hat{\mathbf{S}}, \mathbf{S}) \quad (13)$$

It's important to note that the σ_i is stopped-gradient with respect to the reconstruction network when we calculate \mathcal{L}_{sem} . This approach is adopted to prevent the influence of potentially inaccurate semantic information on the geometric aspects of the model. By doing so, we ensure that the geometry remains unaffected by any errors in the semantic data.

Regularization loss \mathcal{L}_{reg} is consist of distortion loss \mathcal{L}_{dt} , anti-aliased interlevel loss \mathcal{L}_{prop} and edge-aware smoothness loss \mathcal{L}_{smo} with weights λ_{dt} , λ_{prop} and λ_{smo} . We follow [31] to set up \mathcal{L}_{dt} and \mathcal{L}_{prop} , and \mathcal{L}_{smo} is implemented as

$$\mathcal{L}_{smo} = |\partial_x \hat{\mathcal{D}}| \exp^{-|\partial_x \hat{\mathcal{D}}|} + |\partial_y \hat{\mathcal{D}}| \exp^{-|\partial_y \hat{\mathcal{D}}|}. \quad (14)$$

The whole regularization term can be written as:

$$\mathcal{L}_{reg} = \lambda_{dt} \mathcal{L}_{dt} + \lambda_{prop} \mathcal{L}_{prop} + \lambda_{smo} \mathcal{L}_{smo}. \quad (15)$$

E. Foreground Augmentation

Foreground object manipulation plays a crucial role in the system of driving scene simulation, which requires an instance-aware representation for 3D objects, considering that the actions of foreground objects are critical for the driving scenario. III-C introduce our method to reconstruct moving vehicles through street view data, which will be part of our foreground bank. Reasonable use of various open source data sets and generative models on the network can greatly increase the types and quantities of foreground objects.

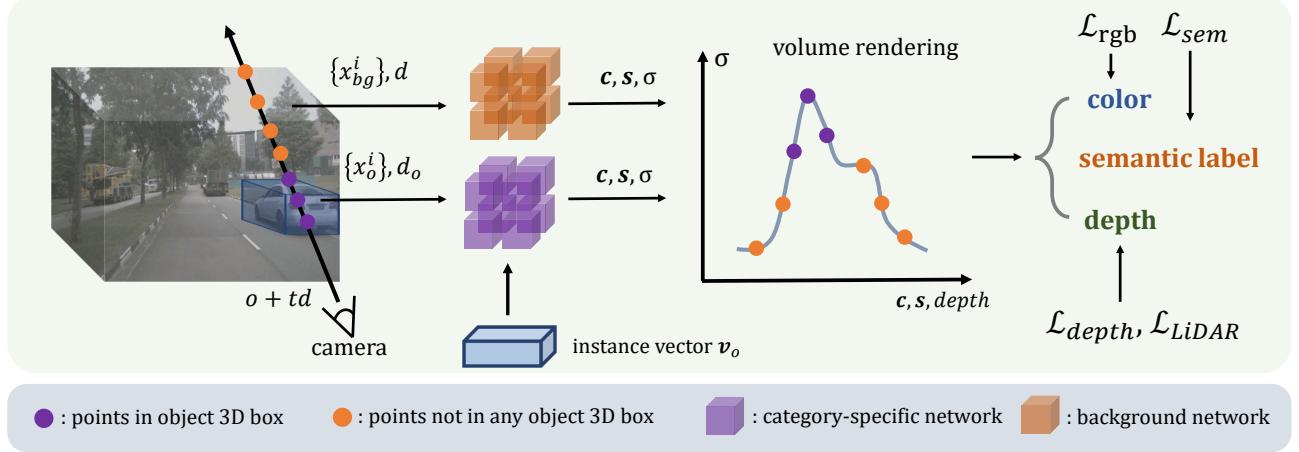


Fig. 7. For points on a ray, we deicode their attributes by two brunches. For points in a certain object 3D box, we use a category-specific network to get the color \mathbf{c} , density σ and semantic logit \mathbf{s} . For points not in any object 3D box, we decode their attributes through the background network.

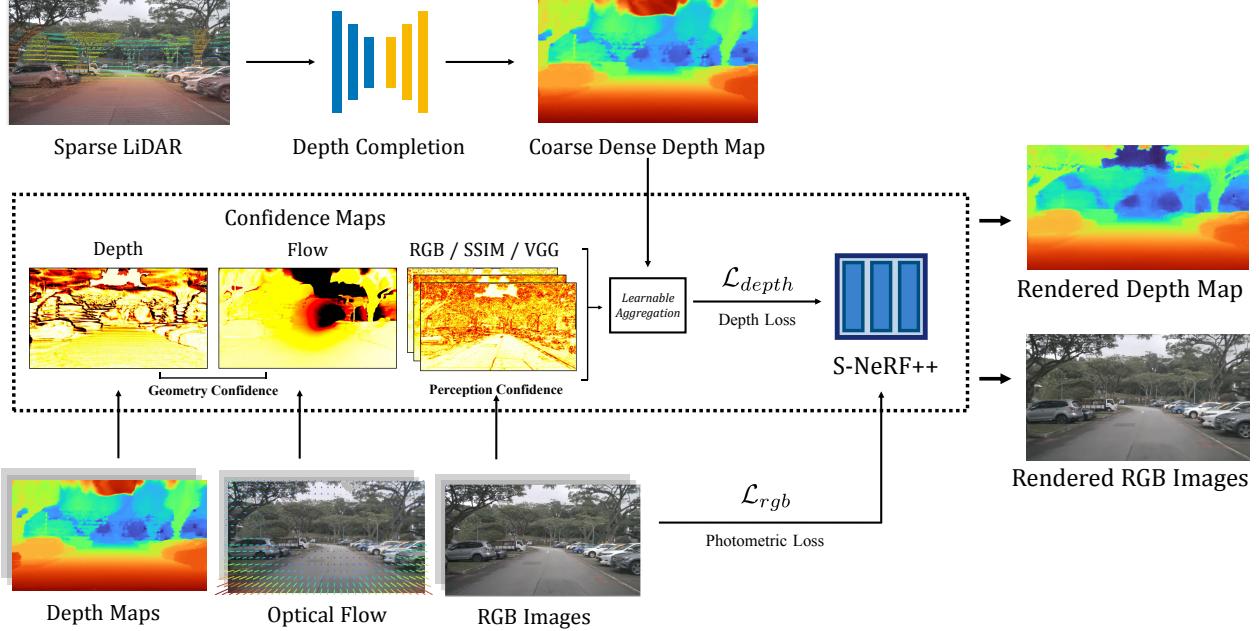


Fig. 8. Overview of our S-NeRF++ reconstruction framework supervised by generated dense depth maps with learnable confidence maps.

Given a camera matrix \mathbf{P} , we render the corresponding foreground image \mathbf{I} and mask \mathbf{M} for synthesizing simulation data. Additionally, the corresponding mesh of object is needed for generating 3D annotations and LiDAR simulation. To accomplish this, we apply two strategies: the **generation-based** simulation and the **reconstruction-based** simulation. The generation-based approach utilizes a pre-trained diffusion model to guide the generation process, while the reconstruction approach leverages neural rendering based method to represent foreground objects and reconstruct meshes.

a) *Generation-based Approach:* Diffusion models are latent variable models of the form $p_\theta(\mathbf{x}_0) = \int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T}$. $\mathbf{x}_{1:T}$ are the latent variables following the posterior distribution of a Markov chain with Gaussian transition. The posterior

$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1})$ is called forward process. The learned distribution $p_\theta(\mathbf{x}_{0:T}) = p_\theta(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is also approximated as a Gaussian transition Markov chain which is known as reverse process. After we maximize the ELBO of $p_\theta(\mathbf{x}_0)$, we can sample \mathbf{x}_0 from a normal noise by the reverse process. A **latent diffusion model** is a diffusion model not direct calculate the distribution of \mathbf{x}_0 but its latent codes \mathbf{z}_0 . It utilizes a autoencoder's encoder to map \mathbf{x}_0 to the latent space \mathbf{z}_0 , and use its decoder to decode the latent codes \mathbf{z}_0 to $\hat{\mathbf{x}}_0$. Recently, StableDiffusion [43] and other LDM show powerful 2D image generation capabilities. However, it is hard to utilize 2D LDM knowledge for better 3D generation. Dreamfusion [46] raises a method that build the bridge of 3D and 2D image named **score distillation sampling**. It aims to

minimize the KL divergence of $q(\mathbf{z}_t|g(\phi), y, t)$ and $p_\theta(\mathbf{z}_t|y, t)$. It has a simplified calculation version of the grad of ϕ :

$$\nabla_\phi \mathcal{L}_{\text{SDS}}(\theta, \mathbf{z} = g(\phi)) = \mathbb{E}_{t, \epsilon} \left[w(t) (\hat{\epsilon}_\theta(\mathbf{z}_t; y, t) - \epsilon) \frac{\partial \mathbf{z}}{\partial \phi} \right].$$

Where $\hat{\epsilon}_\theta$ is the fixed pre-trained LDM U-net module, y is the condition of LDM. In 3D generation, $g(\phi)$ can be the differentiable rendering process of an object. The LDM and SDS make us to manipulate the texture of a mesh or directly generate 3D models in different representation.

Given a 3D mesh with its category, we use input text to control the texture of the mesh represented as an atlas through a pre-computed UV mapping. [48] proposed a pipeline of painting texture through pre-trained depth conditioned diffusion model and inpainting diffusion model, while the generated textures are always mismatched and lack of realism, especially painting textures for cars and bicycles, which are highly concerned objects in autonomous driving. To get high-fidelity 3D objects for our foreground bank, we fine-tuned the diffusion model with category-specific data to learn the data distribution prior better and then guide the generation processing. To generate texture with ensured quality we further refine the UV map through Score Distillation Sampling process.

Due to the inherent flexibility of non-rigid human bodies, we implement skinning techniques for avatar meshes, allowing us to create diverse postures either through a text-guided generative method or pre-programmed motions. This enables us to render humans in various poses using the skinned, textured mesh along with specified pose parameters.

b) Reconstruction-based Approach: Unlike the former which utilizes pre-constructed meshes, this approach reconstruct meshes and render novel images of existing objects via a NeRF-like method. Recently, methods based on signed distance function including NeuS [34] and VolSDF [66] are preferred for mesh reconstruction and novel rendering as they combine SDF (Sign Distance Function, from which can be extracted mesh easily) and NeRF (which performs high qualities in novel rendering) together. Here, we use NeuS [34] with the accelerating technique of hash encoding presented in Instant-NGP [27] as the reconstruction method. Given multi-view object-centered images of an object, we utilize COLMAP [49] to extract camera matrices and fit the object's SDF through NeuS [34]. Then we can extract mesh from the reconstructed SDF via Marching Cubes algorithm [35] and render images based on NeuS [34] render equation. Foreground masks can be derived from NeuS rendering and mesh rasterization. Compared with generation approach, reconstruction approach costs more computation but provides more realistic representing of foreground objects.

It should be noted that we have the capability to render both the RGB and the foreground mask using volume rendering. Alternatively, we can extract their textured meshes and achieve renderings through rasterization, especially since the foreground is reconstructed using a SDF-based method.

For human reconstruction, we follow another neural rendering based approach SelRecon [39] which utilizes the prior knowledge of the human body as much as possible to recon-

struct the mesh and texture of the avatar from the video with a person in motion.

F. Composition of Background and Foreground

After reconstructing background scenes and object-center foreground objects, we propose a straight-forward method to synthesize simulated data, including RGB images, dense depth map, semantic labels and 3D detection annotations. The overview of our simulation system is in Figure 10. First, we construct a BEV-aware ground plan based on rendered background depth maps and semantic labels to determine the placement of foreground objects. Given the computed render poses, the corresponding foreground RGB figures are rendered with masks and aligned meshes. Foreground objects are then merged in background scenes producing combined RGB figures and annotations. At last, boundary inpainting, illumination handling and shadow generating are performed to make synthetic figures more realistic.

1) Foreground Placement based on 2D Ground Plan: To place foreground objects on sensible regions, we leverage rendered background data to build a rough BEV-aware ground plan with semantic annotation and randomly choose placement on available regions. Given rendered dense depth maps \mathbf{D}_i , semantic maps \mathbf{S}_i and camera matrices \mathbf{P}_i , a 3D semantic point cloud \mathbf{C} can be constructed by inverse projecting 2D pixel-level semantic label into 3D space:

$$\mathbf{C} = \bigcup_{i=1}^K \pi^{-1}(\mathbf{S}_i, \mathbf{D}_i, \mathbf{P}_i) = \{(\mathbf{x}_j, s_j)\}_{j=1}^N \quad (16)$$

where π^{-1} means inverse projection, $\mathbf{x}_j \in \mathbb{R}^3$ and $s_j \in \mathcal{S}$ represents the coordinates and semantic label of point j .

Point cloud \mathbf{C} represents reconstructed background scenes with semantic information, but it is not suitable for placement planning for its huge storage cost (the number of points N can be huge) and redundancy (points are too compact). Therefore, we first randomly drop a large proportion of points and modify the point cloud to a grid based ground plan. To be more detailed, we set $H \times W$ 2D ground grids and project semantic point cloud \mathbf{C} vertically to determine the semantic label of each grid, providing a semantic ground plan \mathbf{G}_s . Similarly, we construct another ground plan \mathbf{G}_h with the same size to record the height of each grid. Each plan can be represented as a $H \times W$ array (i.e. a matrix). To determine placement, we randomly choose grids on available regions in \mathbf{G}_s and query corresponding height values in \mathbf{G}_h . The available regions can be defined by user. Fig 9 illustrates the placement sampling procedure.

2) Image Synthesis: After determining the placement, we derive camera pose to render the foreground object, giving RGB figure \mathbf{I} , foreground mask \mathbf{M} , with a mesh which has been reconstructed in former parts. The composition of background and foreground can be approached through an occlusion-aware pasting, and various annotations are extracted from mesh and mask. Explicitly, we first place mesh under camera coordinate to align with the figure \mathbf{I} , then apply a rasterization to compute depth map of the foreground object \mathbf{D}_f . Based on \mathbf{D}_f and background depth map \mathbf{D}_g , we can

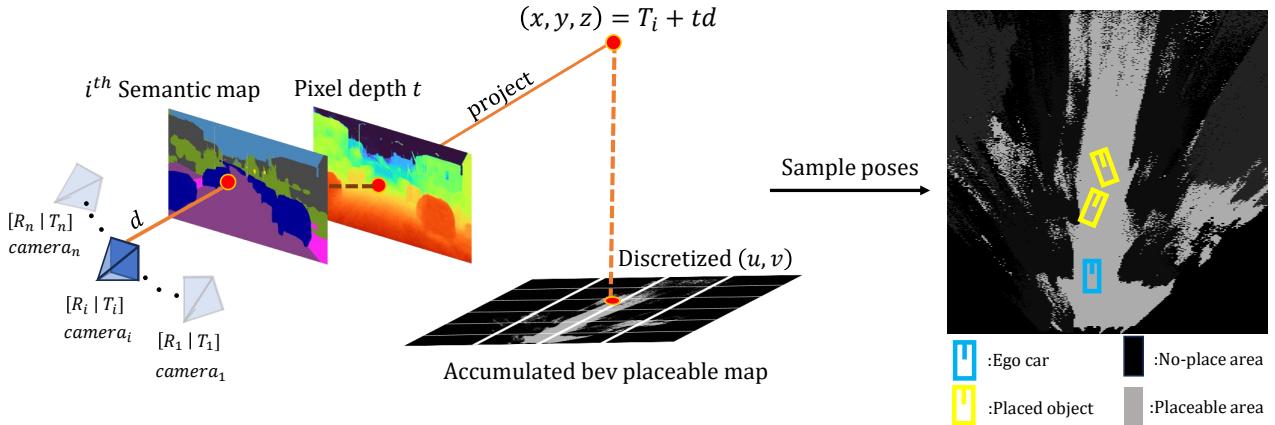


Fig. 9. The procedure of sampling foreground object's placement. We project 3D semantic point cloud from rendered semantic and depth maps to explicitly represent the reconstructed scene. Then we vertically project it to 2D ground plans, recording semantic labels and height of each grid. Finally, we randomly sample placement positions in valid area (The grey part in ground plan above represents the placeable area).

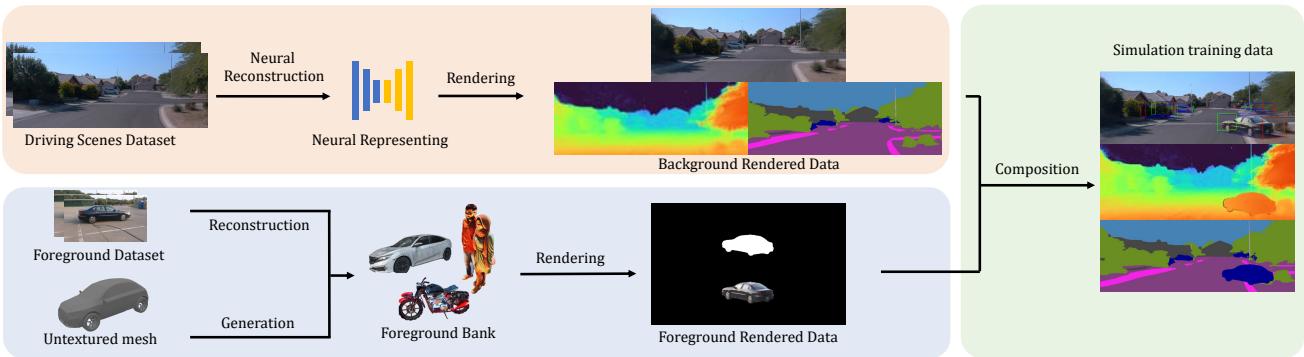


Fig. 10. The whole pipeline of our simulation system. We construct our background representation and foreground bank separately. Finally we can output our simulation data with any combination of foreground and background by fusing the renderings.

derive occlusion mask \hat{M} by comparing two depth value of each pixel, i.e. $\hat{M} = M \cdot (\mathbf{D}_f < \mathbf{D}_g)$. 2D synthetic data including RGB figures, semantic label and depth maps are composed through pasting corresponding data with respect to mask \hat{M} ; 3D annotations like bounding boxes can be extracted from the aligned mesh.

It is noted that \mathbf{D}_f can be also computed through the NeRF-based methods in render stage. We choose the mesh-based approach to be consistent with the 3D annotations in detection task.

3) *Figures Refinement*: Due to the boundary's inconsistency and illumination difference, the inserted foreground objects might look conspicuous and weird in color. To make it more realistic, we follow GeoSim [4] to apply an inpainting-based method to first “soften” boundary of inserted object and then adapt lighting. We use Lama [67] as our inpainting network architecture. First we extract the foreground boundary M_b from foreground object's mask \hat{M} and mask the inconsistent boundary in original synthetic image I , getting a masked image $I_m = I \cdot (1 - M_b)$. Network takes boundary mask M_b and figure I_m to inpaint the masked regions in I_m , providing a more natural synthetic image I' . For adapting illumination, network takes \hat{M} and figure I' as inputs, and outputs the light-balanced figure I'' . Note that both of two stages share the same

inpainting network architecture with the only difference in the weight of parameters and the network input.

The inserted object looks floating on the ground as the lack of shadows. Note that we have the aligned object's mesh, so one can use conventional methods or existing applications (e.g. Blender) to render the corresponding shadows. To speed up the simulating, we set the environment light direction and project mesh's vertices to the ground by light direction to derive a coarse mask of shadows M_s . As vertices of mesh are discrete, we perform a morphological closing operation to fill the holes in M_s . At last, a Gaussian blur is performed to make shadow more realistic. The procedure of refining synthetic images is showed in fig 11.

IV. EXPERIMENTS

In this section, we will provide a detailed explanation of how neural reconstruction is applied to existing autonomous driving datasets, including specific training details. We then introduce our foreground asset library and describe the simulation process on our autonomous driving platform, concluding with the validation of our simulation data through training on downstream tasks.

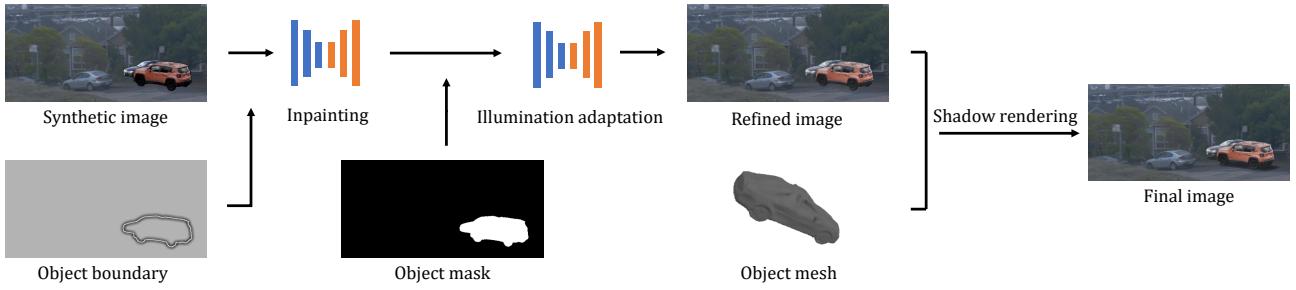


Fig. 11. The procedure of the refinement of synthetic images. We leverage a neural network (with different parameters) to inpaint the boundary of inserted object and balance the illumination. Lastly the aligned mesh is used to render object's shadow, making synthetic image more realistic.

A. Implementation Details

1) *Dataset*: We reconstruct the urban scenes on two datasets, **nuScenes** [7] and **Waymo Open Dataset** [8]. **nuScenes** dataset contains 1000 scenes collected from different city blocks. Each scene consists of around 1000 images in six views that cover the 360° field of view (captured by the front, right-front, right-back, back, left-back and left-front cameras). 32-channel LiDAR data are collected at 20 Hz. Human annotations are given in key LiDAR frames (one frame is labeled in every 10 frames). **Waymo Open Dataset** includes data from 1,000 driving segments. Each driving segment contains 20 seconds of continuous driving. Each segment has the data captured by five high-resolution 64-beam lidars and five front-and-side-facing cameras. Every lidar frames and images are with vehicles, pedestrians, cyclists, and signage carefully labeled.

2) *Street Scenes Reconstruction Settings*: Due to the limitation of storage and NeRF's expression ability, we divided each scene sequence into several scene fragments for reconstruction. More specifically, each scene segment contains about 50 successive video frames, each of which has its corresponding 360-degree LiDAR data, multi-view camera data and foreground object labels. We select 100 scene segments each on Waymo and 60 segments on nuScenes to train NeRFs for the background reconstruction.

We implement our method in **MLP based and hash grid based** way. We set the $\lambda_{sem} = 4e - 3$, $\lambda_d = 0.2$, $\lambda_L = 0$, $\lambda_{smo} = 0.01$, $\lambda_{dt} = 0.01$, $\lambda_{prop} = 1$ in the MLP based model implementation, and set the $\lambda_{sem} = 4e - 3$, $\lambda_d = 0.3$, $\lambda_L = 0.3$, $\lambda_{smo} = 1e - 3$, $\lambda_{dt} = 0.01$, $\lambda_{prop} = 1$ in the hash grid based model implementation for both Waymo and nuScenes training experiment.

3) *Foreground Bank Settings*: Most of our foreground bank comes from vehicle reconstructed in nuScenes. This collection is significantly expanded by including foreground objects reconstructed from our collected 360-degree surround, object-centered dataset, and further enriched by assets created using our proposed generation method.

a) *Foreground Objects Reconstructed from nuScenes*: We use a MLP based NeRF model to represent the vehicle which contains four layers with 256 hidden units in the MLP. The depth and smooth loss weights λ_d and λ_{smo} are set to 1 and 0.15 respectively. We sample 64 times along each ray during the RGB and depth rendering. We train our S-

NeRF++ for 30k iterations using Adam optimizer with 5^{-4} as the learning rate and 1024 as the batch size. The training takes about 2 hours for each vehicle on a single RTX3090 GPU. For each vehicle, there are around 2~8 views used for training and 1~3 views for testing.

b) *Foreground Objects Reconstructed from collected Datasets*: We collected more than 50 360-degree object videos which contains cars, bicycles, motorcycles and traffic cone. Most data are collected from the Co3D dataset [68], and some are self-shot videos. Then we run COLMAP [49] for sparse point clouds reconstruction and abnormal the failure cases. To model the foreground objects separated from the background, we apply a segmentation method to get the foreground mask.

We reconstruct human avatar from People-snap-shot dataset. For simplicity, we use the official checkpoint from SelfR-con [39]. We then give the meshes T-pose bone binding on MAXIMO website, and save a series of pose to a series transformed meshes in order to render different poses of the human avatar.

c) *Generative Foreground Objects*: We take objects mesh from diverse Internet resources, most of them are lack of satisfying textures. Our generative foreground bank include bicycles, motorcycles, cars, bus and human avatar.

For each object category, we collect thousands of images of the corresponding category and using a control-net to fine-tune the StableDiffusion [43] to get the category-specific text-condition diffusion model. We use TEXTure [48] to get the initial texture of a mesh. Then we refine the texture by SDS in latent space. We use a normal form 'a {object} in {color}' as text insertion of the diffusion model.

For human avatar texturing, we directly use the checkpoint of StableDiffusion to apply SDS to train the texture in latent space. Then we decode the latent texture map by the VAE decoder of the LDM to get a high resolution RGB texture.

4) *Figure Refinement Module Implementation*: In this section, we futher explain the detail of our figure refinement network. We modified the network architecture of LaMa [67] and finetuned it using the training datasets from Waymo and nuScenes. To construct the inpainting training dataset. We first generate an initial boundary mask based on the mask of moving vehicles. Then for each training step, we randomly expand the boundary for better overall performance. We then transfer the inpainting dataset to our relighting dataset by using different color-jitter to the mask of vehicle and

TABLE I
OUR METHOD IN MLP BASED AND HASH GRID BASED BACKBONE QUANTITATIVELY OUTPERFORMS STATE-OF-THE-ART METHODS [13], [31]. METHODS ARE TRAINED ON TWO WAYMO STATIC SCENES. AVERAGE PSNR, SSIM AND LPIPS ARE REPORTED.

Large-scale Static Scenes Synthesis on Waymo			
Methods	PSNR↑	SSIM↑	LPIPS↓
Urban-NeRF [69]	17.80	0.494	0.701
Mip-NeRF 360 [13]	22.10	0.724	0.445
Our S-NeRF++ (MLP)	23.60	0.743	0.422
Zip-NeRF [31]	25.05	0.799	0.230
Our S-NeRF++ (grid)	25.78	0.810	0.224

TABLE II
OUR METHOD QUANTITATIVELY OUTPERFORMS STATE-OF-THE-ART METHODS. METHODS ARE TESTED ON FOUR NUSCENES SEQUENCES WITH DYNAMIC VEHICLES. AVERAGE PSNR, SSIM AND LPIPS ARE REPORTED.

Large-scale Dynamic Scenes Synthesis on NuScenes			
Methods	PSNR↑	SSIM↑	LPIPS↓
Zip-NeRF [31]	25.99	0.807	0.187
Our S-NeRF++ (grid)	27.10	0.827	0.170

rest background. This adaptation is designed to enable our relighting network to learn and adapt to changes in light conditions, ensuring that the recolored vehicles harmonize with the background scene.

B. Experiment Results

1) *Renderings of Street view in nuScenes and Waymo:* Here we demonstrate the performance of our method by comparing it with the state-of-the-art MLP based and hash grid based methods Mip-NeRF 360 [13] and Zip-NeRF [31]. We expand the hidden units of the MLP to 1024 in Mip-NeRF 360 (the same as our MLP based S-NeRF++ in MLP backbone) and use a community reimplementation for Zip-NeRF.

We test three nuScenes sequences with dynamic vehicle and two Waymo sequences with only static background. nuScenes visualizations are in Figure 14, and Waymo sequences renderings are shown in Figure 15. We report the evaluation results in Table II and I. Extensive experiments show that our methods, based on MLP and hash grid, surpass Mip-NeRF 360 and Zip-NeRF across all three evaluation metrics. We observe significant improvements including a 6.7% increase in PSNR, a 3% rise in SSIM, and a 5% reduction in LPIPS compared with the Mip-NeRF 360 baseline which uses MLP. Additionally, our method demonstrates strong performance in handling moving vehicles within the scene, achieving better performance in nuScenes dynamic sequences.

2) *Novel View Synthesis for Foreground Vehicles Reconstructed in nuScenes:* In this section, we present our evaluation results for foreground vehicles. We compare our method with the latest non-NeRF car reconstruction method [4]. Note that existing NeRFs [6], [70] cannot be used to reconstruct the moving vehicles. To compare with the NeRF baseline [6], we also test our method on the static cars. Since COLMAP [49] fails in reconstructing the camera parameters here, the same

camera poses used by our S-NeRF++ are applied to the NeRF baseline for comparisons.

a) *Static vehicles:* Figure 16 and Table III show quantitative and visualized comparisons between our method and others in novel view synthesis. Optimizing NeRF baseline on a few (4~7) image views leads to severe blurs and artifacts. GeoSim produces texture holes when warping textures for novel view rendering. The shapes of the cars are also broken due to the inaccurate meshes. In contrast, our S-NeRF++ shows more fine texture details and accurate object shapes. It can improve the PSNR and SSIM by 45~65% compared with the NeRF and GeoSim baselines.

b) *Moving vehicles:* As compared in Figure 2(c)~2(d) and Table III, novel view synthesis by GeoSim [4] is sensitive to mesh errors that will make part of the texture missing or distorted during novel view warping and rendering. In contrast, our S-NeRF++ provides larger ranges for novel view rendering than geosim (see supplementary) and generates better synthesis results. S-NeRF++ can also simulate the lighting changing for different viewing directions, which is impossible for GeoSim. Furthermore, S-NeRF++ does not heavily rely on accurate mesh priors to render photorealistic views. The confidence-based design enables S-NeRF++ to eliminate the geometry inconsistency caused by depth outliers. S-NeRF++ surpasses the latest mesh based method [4] by 45% in PSNR and 18% in SSIM.

C. Simulation Evaluation

In order to verify the reliability of the data generated by our simulation system, we use the simulation data generated by the system to train the downstream task models. We use S-NeRF++ in hash grid based backbone. For most tasks, model trained by the pure simulation data already achieve some good results. Even more exciting, models trained by simulation data and fine-tuned with a small amount of real data can exceed models trained by 10 times real data used in finetuning. We show some nuScenes 6-camera simulation data in Figure 17 and front-and-side-facing simulation data on Waymo in Figure 18.

To generate the best background renderings, we sample rendering poses near the training cameras' poses. For every sample, we first sample two adjacent frames, and randomly interpolate their camera poses to get a base rendering pose P_0 . Then we apply a position and rotation disturbance to P_0 . We uniformly sample the position disturbance in $(-\Delta s, \Delta s)$, the elevation angle disturbance in $(-\Delta \theta, \Delta \theta)$, and the horizontal angle disturbance in $(-\Delta \phi, \Delta \phi)$.

a) *RGB Semantic Segmentation:* Due to the absence of semantic labels in the Waymo dataset, we use pseudo semantic labels to train our semantic branch. We use a pre-trained segmentation model Segformer [58] to predict the semantic label of each image frame. We sample 400 ground-truth semantic labels in the 100 sequences not seen by simulation system to serve as our test set, while using the remaining 13k ground-truth labels as our baseline training set. Then we randomly sample tenth of the training set to create a subset for fine-tuning. Around 20k frames of simulation data are used

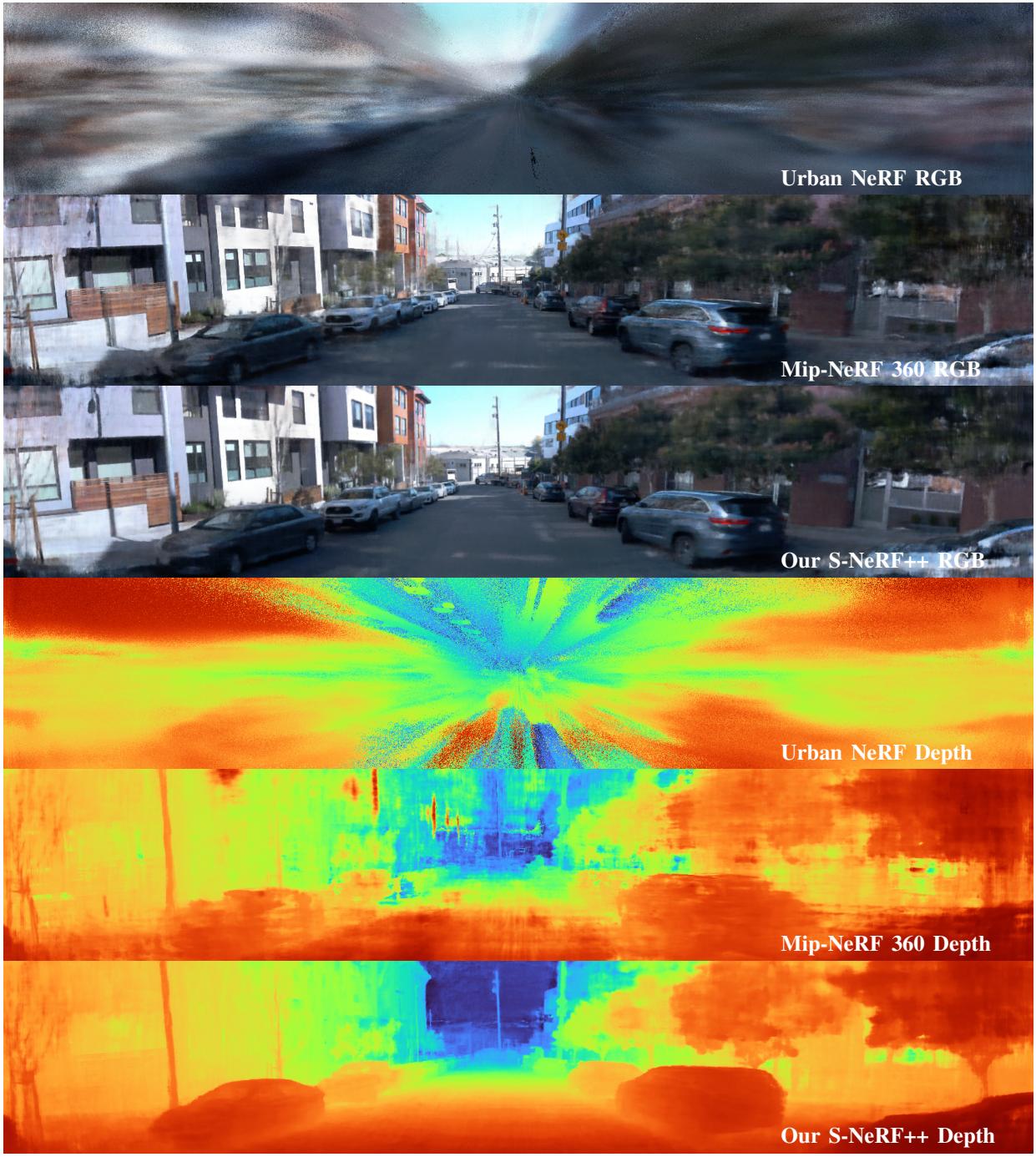
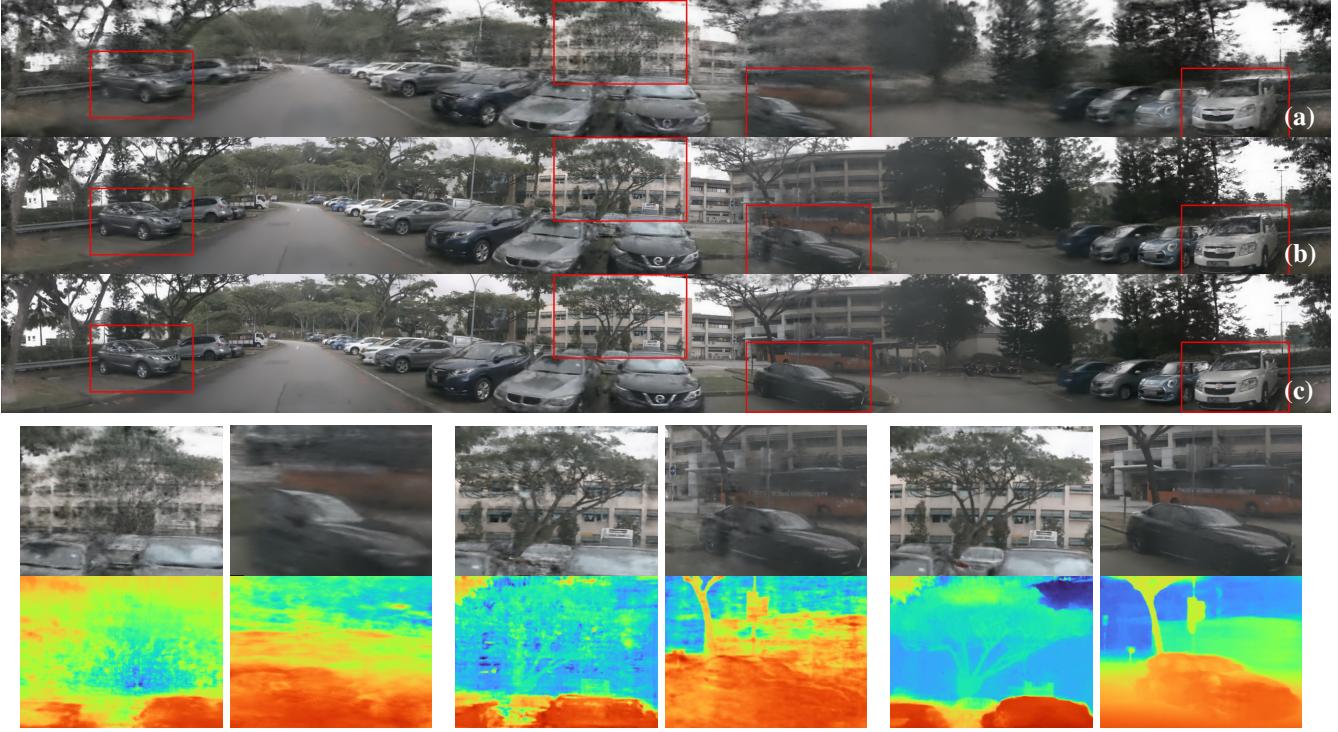


Fig. 12. We render more 180 degree panorama views for visual comparisons. Scenes are from the Waymo datasets. We use our MLP based backbone S-NeRF++.

to train segmentation models. Each frame is randomly fused by 2 foreground objects including pedestrian, car, bicycle and motorcycle. We set $\Delta s = [3, 3, 3]$, $\Delta\theta = 5^\circ$, $\Delta\phi = 20^\circ$. In Table VI, we evaluate two segmentation models based on the mIoU metrics across 19 classes. Results show that models trained with our simulation-augmented data outperform those using an amount of real data that is ten times larger. This not only demonstrates the high quality of our simulated data but also highlights its potential as a valuable tool for improving the efficiency and accuracy of model training in various applications, particularly in scenarios where collecting large

volumes of real-world data is challenging or impractical.

b) Monocular 3D Vehicle Detection: For the monocular 3D detection task, we conduct our experiments on the Waymo dataset. We select 10k real front view image frames from the 100 video sequences, which contain the 100 segment our system used as the baseline training set. From this collection, we randomly chose 1,000 frames to create a fine-tuning set. We then select 400 frames not seen by simulation system or included in baseline training set as our test set. We disturb the front camera view of training views and we set $\Delta s = [3, 3, 3]$, $\Delta\theta = 5^\circ$, $\Delta\phi = 60^\circ$. Each simulation frame



(a) Urban NeRF

(b) Mip-NeRF 360

(c) Ours

Fig. 13. We render the 360 degree panoramas (in a resolution of 8000×800) for comparisons (*zoom in to see details*). Significant improvements are highlighted by red rectangles and cropped patches are shown to highlight details. We compare the MLP based methods with our MLP based backbone S-NeRF.



Fig. 14. nuScenes sequences with moving vehicles. Our method in hash grid based backbone compares with the state-of-art hash grid based method Zip-NeRF. Our method can modify these vehicles and render images of better quality.

incorporated two fused cars. For detection, we use PGD [72] as the detector and use 0.5 IoU as the threshold of the positive samples. The evaluation results are shown in Table IV. Our method surpassed the performance of the those training on 1k real dataset and outperformed models training with 10k real data after further fine-tuning.

c) Multi-view 3D Vehicle Detection: We train the multi-view 3D vehicle Detection on nuScenes dataset. We use the 60 nuScenes segments trained in our simulation platform to generate simulation data. We sample 5k key frames in nuScenes as baseline real data training set and 500 of it as fine-tuning set. We select 200 frames from the 60 full nuScenes data sequences but not seen by our simulation platform as the test set. Disturbance is set to $\Delta s = [1, 1, 1]$, $\Delta\theta = 0^\circ$, $\Delta\phi = 20^\circ$. We generate 12k simulated frames. Each frame contains front,

front-left, front-right, back, back-left, back-right view with the same camera settings as in nuScenes. We insert 2 cars in each frame. Some foreground will appear across different views, and our simulation ensure the view consistency. The multi-view detection results are shown in Table V.

D. Ablation Study

1) In Terms of Reconstruction Components: In this section, we present a series of ablation studies for our MLP-based S-NeRF++ model. These studies were conducted using two street-view scenes from the nuScenes dataset, along with three foreground vehicles that were also constructed using data from nuScenes. Ablations include using different loss balance weights, confidence components, depth qualities and scene

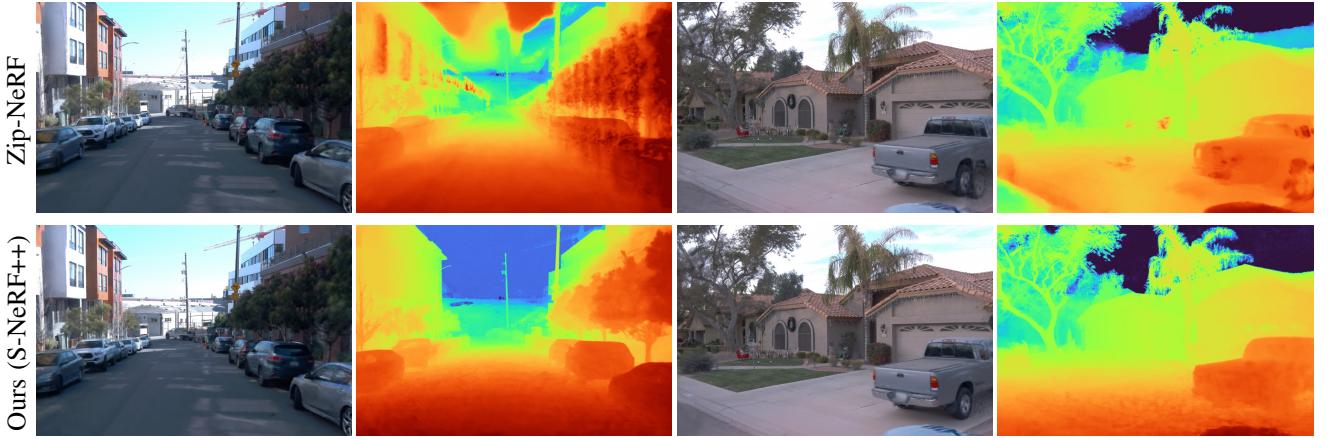


Fig. 15. Qualitative results of novel view synthesis on Waymo. Our method in hash grid based backbone compares with the state-of-art hash grid based method Zip-NeRF.

TABLE III

NOVEL VIEW SYNTHESIS RESULTS ON FOREGROUND CARS. WE COMPARE OUR METHOD WITH THE NeRF AND GEOsim BASELINES. SINCE COLMAP FAILS ON FOREGROUND VEHICLES, WE APPLY OUR CAMERA PARAMETERS TO THE NeRF BASELINE WHEN TRAINING THE STATIC VEHICLES. WE REPORT THE QUANTITATIVE RESULTS ON PSNR, SSIM [55] (HIGHER IS BETTER) AND LPIPS [71] (LOWER IS BETTER).

Methods	Static Vehicles			Moving Vehicles		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
NeRF [6]	11.78	0.539	0.444	-	-	-
GeoSim [4]	11.58	0.602	0.367	12.24	0.623	0.322
Ours	18.81	0.785	0.194	18.00	0.736	0.226

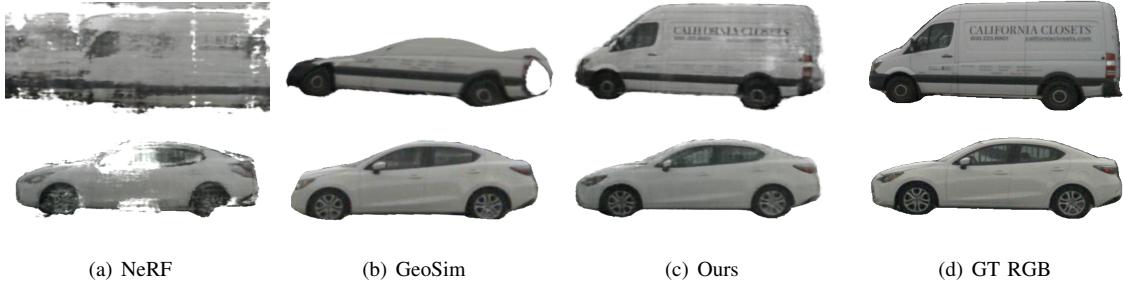


Fig. 16. Novel-view synthesis results for static foreground vehicles. Results are reconstructed from 4~7 views. Our method outperforms others [4], [6] with more texture details and accurate shapes.



Fig. 17. Some NuScenes simulation data, each scene is inserted two cars from foreground bank.



Fig. 18. Some Waymo simulation data, each image is inserted two objects from the foreground bank.

TABLE IV
MONOCULAR 3D VEHICLE DETECTION EVALUATION ON WAYMO

PGD [72] Training Results on Waymo	
Data	Car 3D AP ₅₀
Real 1k	5.0
Real 10k	11.6
Sim	6.3
Sim + Real 1k	12.3

TABLE VI
19 CLASS SEMANTIC SEGMENTATION EVALUATION ON WAYMO DATASET.

Semantic Segmentation Results on Waymo (mAP)		
Data	Segformer [58]	DeeplabV3 [74]
Real 1.3k	68.8	64.1
Real 13k	74.3	67.7
Sim	70.8	65.4
Sim + Real 1.3k	76.0	68.5

TABLE V
MULTI-VIEW 3D VEHICLE DETECTION EVALUATION ON NUSCENES.

DETR3D[73] Training Results on NuScenes	
Data	Car 3D AP _{bus}
Real 500	21.2
Real 5k	43.5
Sim	33.9
Sim + Real 500	45.7

parameterizations. For these experiments, we trained the S-NeRF++ model for 30k iterations on a RTX3090 GPU.

a) *Loss balance weights*: As shown in Table XIV and XV, we evaluate the performance of our S-NeRF++ under varying loss balance weights. Our S-NeRF++ exhibits robust performance to those weights variation. Using larger or smaller λ_d or λ_{smo} just slightly reduces the PSNR and SSIM by 0.2~1% on the background novel view rendering. [0.1, 0.4] (for λ_d) and (0.005, 0.02) (for λ_{smo}) are the reasonable range

for our loss balance weights for training street background views. In the case of foreground vehicles, the recommended ranges are [0.5, 2] (for λ_d) and (0.075, 0.015) (for λ_{smo})

b) *Confidence components*: We also evaluate the effectiveness of S-NeRF++ by incorporating various confidence components (geometry and perception confidence) for learning our final confidence map. As shown in Table XVI, when we remove the perception confidence module, PSNR slightly dropped by 0.4%. And when we train S-NeRF++ without geometry confidence, the PSNR and SSIM is about 0.7% lower. We also test the effects of the threshold τ used in the geometry confidence (Eq. (7) of the paper). We find that the geometry confidence is not sensitive to the threshold τ . [10%, 40%] is a reasonable range for the threshold τ .

For the foreground vehicles, we only use RGB and SSIM confidences. This is because the depth map used in training vehicles are relatively better than the backgrounds. Thus, we do not need strong geometry confidences. We report these ablation results in Table XVII by using only RGB confidence or SSIM confidence. We find that using only RGB or SSIM

TABLE VII
ABLATIONS ON FOREGROUND INSERTION.

Task	Method	Indicator	w/ Insertion	w/o Insertion
Segmentation	Segformer	MIoU	70.8	69.1
Mono Detection	PGD	Car AP ₅₀	6.27	4.38
Mult-view Detection	DETR3D	Car AP _{nus}	33.9	26.5

TABLE VIII
ABLATION STUDY ON CONFIDENCE COMPONENTS.

Components of confidence			Metrics			
Rerojection confidence			Geometry confidence		Metrics	
rgb	ssim	vgg	depth	flow	PSNR	Depth error rate
✓					23.15	30.70%
	✓				23.83	26.85%
		✓			23.97	26.73%
			✓		23.94	26.25%
				✓	23.95	25.50%
				✓	24.03	24.92%
✓	✓	✓	✓	✓	23.95	24.59%
✓	✓	✓	✓	✓	23.88	26.75%
✓	✓	✓	✓	✓	24.05	24.43%

TABLE IX

WE STUDY THE EFFECTS WHEN TRAINING OUR S-NeRF WITH DEPTH MAP IN DIFFERENT QUALITIES. WE ADD RANDOM GAUSSIAN NOISE TO THE ORIGINAL INPUT DEPTH MAPS. THE STRENGTH OF THE NOISE IS MEASURED AS PSNR AND ERROR RATES. ERROR RATES REPRESENT HOW MANY OUTLIERS ARE INTRODUCED BY THE NOISE. OUR S-NeRF IS ROBUST TO DEPTH NOISES.

Depth noise (PSNR)	Depth error rate (%)	PSNR↑	SSIM↑	LPIPS↓
>50	9.5	24.07	0.772	0.384
>50	15.2	24.04	0.771	0.384
50	26.5	24.01	0.771	0.384
35	85	23.97	0.770	0.386
25	95	23.95	0.767	0.391
raw depth	0	24.05	0.771	0.384

confidence could achieve a little better PSNR ($1\sim 2\% \uparrow$), but a relative worse LPLPS ($6.6 \sim 11\% \downarrow$). Taken all these three evaluation metrics into consideration, using both RGB and SSIM confidence gives a better performance in training our S-NeRF++. These findings underscore the importance of both geometry and perception confidence in optimizing the performance and improving the robustness of our S-NeRF++ model.

We also report depth error rate in Table **VIII**. For the accurate depths, it predicts high confidence to encourage the NeRF geometry to be consistent with the LiDAR signal.

c) *Depth quality*: As shown in Table **IX**, we further investigate the impact of employing depth maps of varying quality for training. To simulate depth errors of different magnitudes, we add random Gaussian Noise to the original depth map inputs. The intensity of the noise, indicative of depth quality, is measured by PSNR and error rates compared

with the original depth inputs. Here, 'error rate' refers to the proportion of outliers induced by the noise. We set a *threshold* = 1 to calculate the rate of outliers compared to the original inputs. Our method demonstrate consistent rendering qualities when training with light noises, indicating that such light noise does not significantly impact the performance of our S-NeRF++. When training with high noise levels, S-NeRF++ still maintains comparable performance; the PSNR and SSIM values exhibit only a slight decline, ranging from 0.05~0.5%. This modest reduction highlights the robustness of S-NeRF++ in handling depth inaccuracies. Our confidence strategy effectively locates and accurately measures depth outliers, mitigating the adverse effects of depth noise during the training of S-NeRF++.

Additionally, we also test the performance of our S-NeRF++ when only using sparse depth for supervision (Table **XI**). It performs better than our RGB-only S-NeRF++ (improv-

TABLE X

ABLATIONS ON DIFFERENT SETTINGS. FOR BACKGROUND SCENES, WE USE TWO NUSCENES SEQUENCES FOR EVALUATIONS. FOR MOVING VEHICLES, FOUR CARS ARE TRAINED UNDER DIFFERENT SETTINGS.

	Background Scenes (background)			Moving Vehicles		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Mip-NeRF or GeoSim	15.55	0.533	0.47	13.00	0.651	0.280
Our RGB	24.41	0.790	0.230	15.26	0.718	0.282
w/o depth confidence	24.30	0.775	0.279	18.09	0.748	0.206
w/o smooth loss	25.06	0.804	0.231	19.62	0.796	0.159
Full Settings	25.01	0.805	0.232	19.64	0.803	0.136

TABLE XI

ABLATIONS ON DEPTH SUPERVISION FOR NOVEL STREET-VIEW RENDERING (BACKGROUND).

Depth Settings	PSNR↑	SSIM↑	LPIPS↓
No Depth	22.45	0.742	0.433
Sparse	23.65	0.756	0.415
Dense w/o confidence	23.15	0.757	0.424
Dense w/ confidence	24.05	0.771	0.384

TABLE XII

COMPARISONS OF NERF-, BARF AND SCNERF IN TRAINING.

Method	PSNR	SSIM	LPIPS
BARF [75]	22.23	0.751	0.451
BARF w/o initialization	12.16	0.527	0.634
SCNeRF [76]	25.15	0.784	0.377
SCNeRF w/o initialization	14.24	0.578	0.583
Ours	25.68	0.788	0.375

ing the PSNR by 5%). However, when compared to our default configuration that utilizes dense depth maps and the proposed confidence metric, there is a significant decrease in performance: a 2% reduction in PSNR and 8% lower in LPIPS than our default settings. This indicates that The enhanced detail and data richness in dense maps contribute significantly to the improved accuracy and realism of the rendered images, underlining the importance of incorporate dense depth supervision in our training process.

We conducted additional tests on our S-NeRF++ by incorporating two alternative depth completion methods [77], [78] (Table XIII). Intriguingly, even when applying a traditional depth completion method from [77], which demonstrates a slightly lesser performance, our S-NeRF++ still manages to maintain a almost identical rendering quality (24.41→24.40). This result indicates that our method's efficacy is not tied to a specific depth completion model. Thanks to our confidence-guided depth supervision framework, S-NeRF++ can seamlessly integrate various depth completion models. This adaptability broadens the potential applications of S-NeRF++, allowing it to function effectively across a range of depth data qualities.

d) Scene and ray parameterization: Our investigation also encompassed the impact of varying the radius in the scene parameterization function, as described in Eq. 7 of the paper. The radius r lays a crucial role in encapsulating the entirety of large-scale scenes within a defined range. It enables different distance mapping functions for close and distant points, allowing our S-NeRF++ to preserve finer details in objects that are nearer. In Table XIX, we report the performance of our S-NeRF++ when using different radius for scene parameterization. We find that choosing r in 3~10m could produce a better results than the original setting in Mip-NeRF 360 [13]. This improvement is attributed to the fact that the street views in datasets like nuScenes and Waymo often extend over considerable distances (>200m). The original configuration in Mip-NeRF 360 [13], which does not include an adjustable radius for scene parameterization, is less suited for these types of expansive scenes. Our adjustable radius parameter in S-NeRF++ therefore provides a significant advantage, allowing for more precise and effective parameterization of wide-ranging scenes, which is critical for accurately capturing and rendering the complexities inherent in large-scale environments.

e) Influence of quality of 3D detection results: As shown in Table XX, the quality of 3D object detector does not significantly influence the rendering quality. This is because we apply pose refinement technique to improve the inaccurate initial results of the virtual camera pose estimation. The pose refinement is guided by depth supervision and visual multi-view constraints during training. To further verify our system's robustness against inaccurate 3D bounding boxes, we conducted tests using bounding boxes of varying quality levels (mIoU: 0.79, 0.67 and 0.55). These were then compared against the ground truth bounding boxes, which were obtained from the dataset labels. Our findings reveal that the degradation in PSNR relative to ground truth bounding boxes is minimal. Specifically, with bounding boxes at an mIoU of 0.79 (using the default detector), the PSNR experienced a slight decrease of only 0.23, SSIM drops by 0.04, and LPIPS remained unchanged. Even with lower-quality detections, our S-NeRF++ still delivered satisfactory results. (E.g. 0.49 ↓when the mIoU is 0.67 and 0.97 ↓for a poor mIoU of 0.55). This outcome demonstrates that bounding boxes with an IoU above 0.5 are sufficiently accurate for training our S-NeRF++, a benchmark achievable by numerous existing 3D

TABLE XIII
EFFECTS OF DIFFERENT DEPTH COMPLETION METHODS

Method	Mean absolute error/[mm] (KITTI)	Rank on KITTI leaderboard	PSNR	SSIM	LPIPS
Traditional method [77]	302.60	116	24.40	0.782	0.344
[78]	215.02	61	24.50	0.785	0.344
Our default NLSPN	199.59	40	24.41	0.783	0.345

TABLE XIV
ABLATIONS ON LOSS BALANCE WEIGHTS λ_d , λ_{smo} FOR BACKGROUND STREET VIEWS.

λ_d	λ_{smo}	PSNR↑	SSIM↑	LPIPS↓
0.1	0.01	24.00	0.770	0.385
0.4	0.01	23.85	0.767	0.397
0.2	0.005	23.95	0.769	0.382
0.2	0.02	23.98	0.771	0.390
0.2	0.05	23.78	0.767	0.406
0.2	0.01	24.05	0.771	0.384

TABLE XV
ABLATIONS ON LOSS BALANCE WEIGHTS λ_d , λ_{smo} FOR FOREGROUND VEHICLES.

λ_d	λ_{smo}	PSNR↑	SSIM↑	LPIPS↓
0.5	0.15	19.56	0.794	0.149
2	0.15	19.67	0.802	0.149
1	0.075	19.85	0.803	0.147
1	0.3	19.47	0.789	0.147
1	0.15	19.64	0.803	0.136

object detectors. Hence, our system proves to be robust and versatile, effectively handling a range of detection qualities without significant compromises in performance.

f) Different pose refinements: In our exploration of various pose refinement strategies, we experimented with different models such as NeRF– [50], BARF [75] and SCNeRF [76]. As shown in Table XII, We find that NeRF– helps achieve the best quality when training on the self-driving dataset. This superior performance can be attributed to the straightforward nature of NeRF–. It directly adjusts translation and rotation parameters, with these shifts being relatively simpler to learn under depth supervision. Additionally, the efficacy of NeRF– may also be linked to its proficiency in situations where there is limited overlap in the field of view between different cameras, a scenario depicted in Figure 1 of our paper.

2) In Terms of Simulation Components: In our final analysis, we examined the influence of different elements within our simulation system on the quality of the resulting simulation data. This experiment aimed to identify how each element contributes to the overall fidelity and effectiveness of the simulated environment.

a) Fusion adaption module: Our focus here was to assess the enhancement brought by the fusion refinement

TABLE XVI
ABLATIONS ON CONFIDENCE SETTINGS FOR NOVEL STREET-VIEW RENDERING (BACKGROUND).

Reprojection	Geometry	PSNR↑	SSIM↑	LPIPS↓
Yes	No	23.88	0.767	0.385
No	$\tau = 20\%$	23.95	0.770	0.385
Yes	$\tau = 10\%$	23.93	0.771	0.385
Yes	$\tau = 40\%$	23.97	0.771	0.384
Yes	$\tau = 20\%$	24.05	0.771	0.384

TABLE XVII
ABLATIONS ON CONFIDENCE COMPONENTS (RGB AND SSIM) FOR FOREGROUND VEHICLES.

RGB	SSIM	PSNR↑	SSIM↑	LPIPS↓
Yes	No	19.79	0.803	0.152
No	Yes	19.97	0.807	0.145
Yes	Yes	19.64	0.803	0.136

module, which includes inpainting, relighting, and shading, on the realism of simulation data. We generate 10k simulation data from Waymo scenes inserted by 2 foreground cars, and calculate Fréchet inception distances (FID [79]) before fusion refinement and after refinement in Waymo dataset. Results are shown in Table XVIII, images processed by our fusion adaption module achieve better reality and closer to the original dataset.

b) Foreground insertion: Additionally, we investigated the influence of inserting foreground objects on downstream training tasks. We show the training results with simulation data without the foreground objects in Table VII. Across all tasks, the inclusion of foreground objects demonstrated varied levels of improvement, enriching the diversity of the simulation data and potentially enhancing the training efficacy.

V. CONCLUSION

In conclusion, the S-NeRF++ autonomous driving simulation system represents a significant advancement in the field of self-driving data enhancement and traffic scenario simulation. Our system harnesses the power of neural reconstruction to faithfully reconstruct 3D street scenes from autonomous driving datasets, while also rendering visually compelling novel views. The integration of sophisticated generation techniques, rooted in the Latent Diffusion Model, has substantially broadened the variety and quality of both the

TABLE XVIII
ABLATIONS ON REFINEMENT MODULE ON SIMULATION QUALITY.

	w/ refinement	w/o refinement
FID↓	68.9	75.2

TABLE XIX
ABLATIONS ON RADIUS r IN OUR SCENE PARAMETERIZATION FUNCTION (EQ. (14) OF THE PAPER) FOR NOVEL STREET-VIEW RENDERING.

r	PSNR↑	SSIM↑	LPIPS↓
1m	23.95	0.769	0.384
5m	24.07	0.771	0.385
10m	24.05	0.771	0.384
3m	24.05	0.771	0.384

foreground asset bank and the simulation data. Additionally, the implementation of a meticulous foreground-background fusion pipeline addresses the challenges of illumination and shadow processing effectively. The efficacy of S-NeRF++ is further validated through its application in training diverse downstream tasks on NuScenes and Waymo datasets, where it consistently produces high-quality and reliable simulation data. This underscores its potential as a valuable tool in the ongoing development of autonomous driving technologies.

REFERENCES

- [1] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “Carla: An open urban driving simulator,” in *Conference on robot learning*, 2017. [1](#) [2](#)
- [2] S. Shah, D. Dey, C. Lovett, and A. Kapoor, “Airsim: High-fidelity visual and physical simulation for autonomous vehicles,” in *Field and Service Robotics*, 2018. [1](#) [2](#)
- [3] W. Li, C. Pan, R. Zhang, J. Ren, Y. Ma, J. Fang, F. Yan, Q. Geng, X. Huang, H. Gong *et al.*, “Aads: Augmented autonomous driving simulation using data-driven algorithms,” *Science robotics*, 2019. [1](#) [2](#)
- [4] 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,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2021. [1](#) [2](#) [3](#) [7](#) [10](#) [12](#) [15](#)
- [5] S. W. Kim, J. Philion, A. Torralba, and S. Fidler, “Drivegan: Towards a controllable high-quality neural simulation,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2021. [1](#) [2](#)
- [6] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. Barron, R. Ramamoorthi, and R. Ng, “Nerf: Representing scenes as neural radiance fields for view synthesis,” in *European Conference on Computer Vision*, 2020. [1](#) [2](#) [12](#), [15](#)
- [7] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, “nuscenes: A multimodal dataset for autonomous driving,” *arXiv preprint*, 2019. [1](#) [2](#) [3](#) [4](#), [11](#)
- [8] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Choi, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, “Scalability in perception for autonomous driving: Waymo open dataset,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020. [1](#) [2](#), [4](#), [11](#)
- [9] Z. Yang, Y. Chen, J. Wang, S. Manivasagam, W.-C. Ma, A. J. Yang, and R. Urtasun, “Unisim: A neural closed-loop sensor simulator,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2023. [1](#) [2](#)
- [10] Z. Wu, T. Liu, L. Luo, Z. Zhong, J. Chen, H. Xiao, C. Hou, H. Lou, Y. Chen, R. Yang *et al.*, “Mars: An instance-aware, modular and realistic simulator for autonomous driving,” *arXiv preprint*, 2023. [1](#) [2](#)
- [11] Z. Xie, J. Zhang, W. Li, F. Zhang, and L. Zhang, “S-nerf: Neural radiance fields for street views,” in *International Conference on Learning Representations*, 2022. [2](#)
- [12] 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,” in *IEEE International Conference on Computer Vision*, 2021. [2](#)
- [13] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, “Mip-nerf 360: Unbounded anti-aliased neural radiance fields,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2022. [3](#), [6](#), [12](#), [18](#)
- [14] S. Manivasagam, S. Wang, K. Wong, W. Zeng, M. Sazanovich, S. Tan, B. Yang, W.-C. Ma, and R. Urtasun, “Lidarsim: Realistic lidar simulation by leveraging the real world,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020. [2](#)
- [15] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski, “Building rome in a day,” *Communications of the ACM*, 2011. [2](#)
- [16] F. Losasso and H. Hoppe, “Geometry clipmaps: terrain rendering using nested regular grids,” in *ACM Siggraph*, 2004. [2](#)
- [17] V. Sitzmann, J. Thies, F. Heide, M. Nießner, G. Wetzstein, and M. Zollhöfer, “DeepVoxels: Learning persistent 3d feature embeddings,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. [2](#)
- [18] Q. Xu, W. Wang, D. Ceylan, R. Mech, and U. Neumann, “DISN: deep implicit surface network for high-quality single-view 3d reconstruction,” in *Advances in Neural Information Processing Systems*, 2019. [2](#)
- [19] F. Engelmann, K. Rematas, B. Leibe, and V. Ferrari, “From points to multi-object 3d reconstruction,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2021. [2](#)
- [20] A. Kar, C. Häne, and J. Malik, “Learning a multi-view stereo machine,” in *Advances in Neural Information Processing Systems*, 2017. [2](#)
- [21] T. Groueix, M. Fisher, V. G. Kim, B. Russell, and M. Aubry, “AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. [2](#)
- [22] 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. [2](#)
- [23] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa, “Plenoctrees for real-time rendering of neural radiance fields,” in *IEEE International Conference on Computer Vision*, 2021. [2](#)
- [24] D. Rebain, W. Jiang, S. Yazdani, K. Li, K. M. Yi, and A. Tagliasacchi, “Derf: Decomposed radiance fields,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2021. [2](#)
- [25] C. Sun, M. Sun, and H.-T. Chen, “Direct voxel grid optimization: Superfast convergence for radiance fields reconstruction,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2022. [2](#)
- [26] L. Liu, J. Gu, K. Zaw Lin, T.-S. Chua, and C. Theobalt, “Neural sparse voxel fields,” in *Advances in Neural Information Processing Systems*, 2020. [2](#)
- [27] T. Müller, A. Evans, C. Schied, and A. Keller, “Instant neural graphics primitives with a multiresolution hash encoding,” 2022. [2](#), [7](#), [9](#)
- [28] R. Martin-Brualla, N. Radwan, M. S. M. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, “Nerf in the wild: Neural radiance fields for unconstrained photo collections,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2021. [2](#)
- [29] J. Li, Y. Shi, and H. Li, “Neural plenoptic sampling: Capture light-field from imaginary eyes,” in *ACCV*, 2021. [2](#)
- [30] M. Tancik, V. Casser, X. Yan, S. Pradhan, B. Mildenhall, P. Srinivasan, J. T. Barron, and H. Kretzschmar, “Block-NeRF: Scalable large scene neural view synthesis,” *arXiv preprint*, 2022. [2](#)

TABLE XX
PERFORMANCE OF OUR S-NERF++ BY USING 3D BOUNDING BOXES IN DIFFERENT QUALITIES (MIOU).

Bounding box types	mIoU	PSNR	SSIM	LPIPS
GT bounding box	1.0	23.50	0.862	0.111
CenterPoints [51]	0.787	23.27	0.858	0.111
poor bounding box	0.672	23.01	0.845	0.126
poor bounding box	0.554	22.53	0.841	0.129

- [31] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, “Zip-nerf: Anti-aliased grid-based neural radiance fields,” *arXiv preprint*, 2023. 3, 6, 7, 12
- [32] A. Kundu, K. Genova, X. Yin, A. Fathi, C. Pantofaru, L. J. Guibas, A. Tagliasacchi, F. Dellaert, and T. Funkhouser, “Panoptic neural fields: A semantic object-aware neural scene representation,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 3, 7
- [33] J. Ost, F. Mannan, N. Thurey, J. Knodt, and F. Heide, “Neural scene graphs for dynamic scenes,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 3, 7
- [34] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang, “Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction,” in *Advances in Neural Information Processing Systems*, 2021. 3, 9
- [35] W. E. Lorensen and H. E. Cline, “Marching cubes: A high resolution 3d surface construction algorithm,” *ACM SIGGRAPH Computer Graphics*, 1987. 3, 9
- [36] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, “Smpl,” *ACM Transactions on Graphics*, 2015. 3
- [37] S. Peng, Y. Zhang, Y. Xu, Q. Wang, Q. Shuai, H. Bao, and X. Zhou, “Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 3
- [38] J. Chen, Y. Zhang, D. Kang, X. Zhe, L. Bao, X. Jia, and H. Lu, “Animatable neural radiance fields from monocular rgb videos,” *arXiv preprint*, 2021. 3
- [39] B. Jiang, Y. Hong, H. Bao, and J. Zhang, “Selfrecon: Self reconstruction your digital avatar from monocular video,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 3, 9, 11
- [40] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 3
- [41] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 3
- [42] A. Van Den Oord, O. Vinyals *et al.*, “Neural discrete representation learning,” in *Advances in Neural Information Processing Systems*, 2017. 3
- [43] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, “High-resolution image synthesis with latent diffusion models,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 3, 8, 11
- [44] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, “Zero-shot text-to-image generation,” in *International Conference on Machine Learning*, 2021. 3
- [45] E. R. Chan, C. Z. Lin, M. A. Chan, K. Nagano, B. Pan, S. De Mello, O. Gallo, L. J. Guibas, J. Tremblay, S. Khamis *et al.*, “Efficient geometry-aware 3d generative adversarial networks,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 3
- [46] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall, “Dreamfusion: Text-to-3d using 2d diffusion,” *arXiv preprint*, 2022. 3, 8
- [47] C.-H. Lin, J. Gao, L. Tang, T. Takikawa, X. Zeng, X. Huang, K. Kreis, S. Fidler, M.-Y. Liu, and T.-Y. Lin, “Magic3d: High-resolution text-to-3d content creation,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2023. 3
- [48] E. Richardson, G. Metzer, Y. Alaluf, R. Giryes, and D. Cohen-Or, “Texture: Text-guided texturing of 3d shapes,” *arXiv preprint*, 2023. 3, 9, 11
- [49] J. L. Schönberger and J.-M. Frahm, “Structure-from-motion revisited,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 4, 9, 11, 12
- [50] Z. Wang, S. Wu, W. Xie, M. Chen, and V. A. Prisacariu, “NeRF—: Neural radiance fields without known camera parameters,” *arXiv preprint*, 2021. 4, 19
- [51] T. Yin, X. Zhou, and P. Krähenbühl, “Center-based 3d object detection and tracking,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 4, 20
- [52] J. Park, K. Joo, Z. Hu, C.-K. Liu, and I. S. Kweon, “Non-local spatial propagation network for depth completion,” in *European Conference on Computer Vision*, 2020. 5
- [53] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012. 5
- [54] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The kitti dataset,” *IJRR*, 2013. 5
- [55] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE Transactions on Image Processing*, 2004. 5, 15
- [56] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *International Conference on Learning Representations*, 2015. 5
- [57] F. Zhang, O. J. Woodford, V. A. Prisacariu, and P. H. Torr, “Separable flow: Learning motion cost volumes for optical flow estimation,” in *IEEE International Conference on Computer Vision*, 2021. 5
- [58] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, “Segformer: Simple and efficient design for semantic segmentation with transformers,” *Advances in Neural Information Processing Systems*, 2021. 6, 12, 16
- [59] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The cityscapes dataset for semantic urban scene understanding,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 6
- [60] G. Neuhold, T. Ollmann, S. Rota Bulo, and P. Kontschieder, “The mapillary vistas dataset for semantic understanding of street scenes,” in *IEEE International Conference on Computer Vision*, 2017. 6
- [61] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, “Bdd100k: A diverse driving dataset for heterogeneous multitask learning,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 6
- [62] G. Varma, A. Subramanian, A. Namboodiri, M. Chandraker, and C. Jawahar, “Idd: A dataset for exploring problems of autonomous navigation in unconstrained environments,” in *IEEE Winter Conference on Applications of Computer Vision*, 2019. 6
- [63] J. Lambert, Z. Liu, O. Sener, J. Hays, and V. Koltun, “Mseg: A composite dataset for multi-domain semantic segmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 6
- [64] S. Liu, T. Li, W. Chen, and H. Li, “Soft rasterizer: A differentiable renderer for image-based 3d reasoning,” in *IEEE International Conference on Computer Vision*, 2019. 7
- [65] X. Wang, R. Zhang, T. Kong, L. Li, and C. Shen, “Solov2: Dynamic and fast instance segmentation,” in *Advances in Neural Information Processing Systems*, 2020. 7
- [66] L. Yariv, J. Gu, Y. Kasten, and Y. Lipman, “Volume rendering of neural implicit surfaces,” in *Advances in Neural Information Processing Systems*, 2021. 9
- [67] R. Suvorov, E. Logacheva, A. Mashikhin, A. Remizova, A. Ashukha, A. Silvestrov, N. Kong, H. Goka, K. Park, and V. Lempitsky, “Resolution-robust large mask inpainting with fourier convolutions,” in *IEEE Winter Conference on Applications of Computer Vision*, 2022. 10, 11
- [68] J. Reizenstein, R. Shapovalov, P. Henzler, L. Sbordone, P. Labatut, and D. Novotny, “Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction,” in *IEEE International Conference on Computer Vision*, 2021. 11
- [69] K. Rematas, A. Liu, P. P. Srinivasan, J. T. Barron, A. Tagliasacchi, T. Funkhouser, and V. Ferrari, “Urban radiance fields,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 12
- [70] K. Deng, A. Liu, J.-Y. Zhu, and D. Ramanan, “Depth-supervised NeRF: Fewer views and faster training for free,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 12
- [71] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The unreasonable effectiveness of deep features as a perceptual metric,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 15
- [72] T. Wang, Z. Xinge, J. Pang, and D. Lin, “Probabilistic and geometric depth: Detecting objects in perspective,” in *Conference on Robot Learning*, 2022. 14, 16
- [73] Y. Wang, V. C. Guizilini, T. Zhang, Y. Wang, H. Zhao, and J. Solomon, “Detr3d: 3d object detection from multi-view images via 3d-to-2d queries,” in *Conference on Robot Learning*, 2022. 16
- [74] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, “Rethinking atrous convolution for semantic image segmentation,” *arXiv preprint*, 2017. 16
- [75] C.-H. Lin, W.-C. Ma, A. Torralba, and S. Lucey, “Barf: Bundle-adjusting neural radiance fields,” in *IEEE International Conference on Computer Vision*, 2021. 18, 19
- [76] Y. Jeong, S. Ahn, C. Choy, A. Anandkumar, M. Cho, and J. Park, “Self-calibrating neural radiance fields,” in *IEEE International Conference on Computer Vision*, 2021. 18, 19
- [77] J. Ku, A. Harakeh, and S. L. Waslander, “In defense of classical image processing: Fast depth completion on the cpu,” in *2018 15th Conference on Computer and Robot Vision (CRV)*. IEEE, 2018. 18, 19

- [78] W. Van Gansbeke, D. Neven, B. De Brabandere, and L. Van Gool, “Sparse and noisy lidar completion with rgb guidance and uncertainty,” in *2019 16th international conference on machine vision applications (MVA)*. IEEE, 2019. [18](#), [19](#)
- [79] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “Gans trained by a two time-scale update rule converge to a local nash equilibrium,” in *Advances in Neural Information Processing Systems*, 2017. [19](#)