READ: Large-Scale Neural Scene Rendering for Autonomous Driving

Zhuopeng Li[1], **Lu Li**[1], **Jianke Zhu**[1,2*]

[1]Zhejiang University, Zhejiang, China
[2] Alibaba-Zhejiang University Joint Institute of Frontier Technologies
{lizhuopeng, lu.lee, jkzhu}@zju.edu.cn

Abstract

With the development of advanced driver assistance systems (ADAS) and autonomous vehicles, conducting experiments in various scenarios becomes an urgent need. Although having been capable of synthesizing photo-realistic street scenes, conventional image-to-image translation methods cannot produce coherent scenes due to the lack of 3D information. In this paper, a large-scale neural rendering method is proposed to synthesize the autonomous driving scene (READ), which makes it possible to generate large-scale driving scenes in real time on a PC through a variety of sampling schemes. In order to effectively represent driving scenarios, we propose an $\omega - net$ rendering network to learn neural descriptors from sparse point clouds. Our model can not only synthesize photo-realistic driving scenes but also stitch and edit them. The promising experimental results show that our model performs well in large-scale driving scenarios.

Introduction

Synthesizing free-view photo-realistic images is an important task in computer vision and robotics. Especially, the synthetic large-scale street views are essential to a series of real-world applications, including autonomous driving (Kim et al. 2013), robot simulation (Dosovitskiy et al. 2017). As illustrated in Fig. 1, the objective of neural scene rendering is to synthesize the 3D scene from a moving camera, where the user can browse the street scenery from different views and conduct experiments through automatic driving simulation. In addition, it is able to generate multi-view images to provide data for robotic tasks.

With the development of autonomous driving, it is challenging to conduct experiments in various driving scenarios. Due to the complicated geographic locations, varying surroundings, and road conditions, it is usually difficult to simulate outdoor environments. Additionally, it is hard to model some unexpected traffic scenarios, such as car accidents, where the simulators can help to reduce the reality gap. However, the data generated by the widely used simulator like CARLA (Dosovitskiy et al. 2017) is far different from the real world using the conventional rendering pipeline.

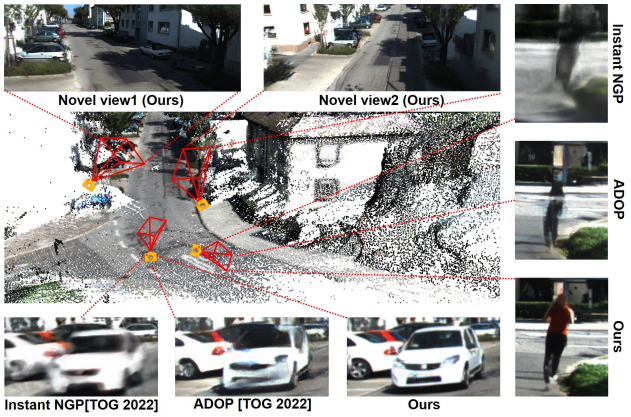

Figure 1: Given the input point clouds, our proposed approach synthesizes photo-realistic driving scenes from different views. Both Instant NGP and ADOP cannot deal with the regions with sparse point cloud, including sky, moving objects, and distant areas, where the synthesized novel views tend to be blurry with large artifacts.

The image-to-image translation-based methods (Gao et al. 2020; Tang, Bai, and Sebe 2020) synthesize the street views with semantic labels by learning the mapping between source images and targets. Despite generating the encouraging street scene, there exist some large artifacts and incoherent textures. Moreover, the synthesized image has only a single view that cannot provide the rich multi-view traffic conditions for autonomous vehicles. This hinders them from a large number of real-world applications.

Recently, Neural Radiance Field (NeRF) based methods (Zhang et al. 2021; Niemeyer and Geiger 2021; Mildenhall et al. 2020; Wang et al. 2021) achieve promising results in synthesizing the photo-realistic scenes with multi-view. As suggested in (Deng et al. 2022), they cannot produce reasonable results with only a few input views, which typically happens in the driving scenario with the objects appearing in only a few frames. Moreover, the NeRF-based methods mainly render either the interiors or objects. They have difficulty synthesizing the large-scale driving scenes with the complicated environment, where the large artifacts occur in the closed-up views and surroundings. To tackle this prob-

*corresponding authors

lem, NeRFW (Martin-Brualla et al. 2021) makes use of the additional depth and segmentation annotations to synthesize an outdoor building, which takes about two days with 8 GPUs. Such a long reconstruction time is mainly due to the unnecessary sampling of the vast spaces. To solve this problem, Instant NGP (Müller et al. 2022) proposes a multi-resolution hash encoding scheme to reduce the training time. However, there are still some noticeable artifacts, and the inference speed is slow.

Unlike the NeRF-based methods that purely depend on per-scene fitting, the neural rendering approaches (Thies, Zollhöfer, and Nießner 2019) can be effectively initialized via neural textures, which are stored as maps on top of the 3D mesh proxy. Similarly, NPBG (Aliev et al. 2020) learns neural descriptors from a raw point cloud to encode the local geometry and appearance, which avoids sample rays in empty scene space by the classical point clouds reflecting the geometry of the scene in the real world. Moreover, ADOP (Rückert, Franke, and Stamminger 2022) improves NPBG by adding a differentiable camera model with a tone mapper, which introduces the formulation to better approximate the spatial gradient of pixel rasterization. In general, the point-based neural rendering method can synthesize a larger scene with fewer captured images by initializing the scene through a 3D point cloud. Although neural rendering-based methods can synthesize the photo-realistic novel views in both indoor and outdoor scenes, it is still very challenging to deal with the large-scale driving scenarios due to the limitations on model capacity, as well as constraints on memory and computation. Additionally, it is difficult to render the photo-realistic views with rich buildings, lanes, and road signs, where the sparse point cloud obtained from the few input images usually contains lots of holes.

In this paper, we propose an effective neural scene rendering approach, which makes it possible to synthesize the large-scale driving scenarios in real time through efficient Monte Carlo sampling, screening of point clouds, cube projection, and patch sampling. It is worth mentioning that our method synthesized large-scale driving scenarios within two days of training on a PC having two GPUs. This greatly reduces the computational cost so that large-scale scene rendering can be achieved on affordable hardware. For sparse point clouds, we fill in the missing areas of point clouds by multi-scale feature fusion. To synthesize photo-realistic driving scenes from sparse point clouds, we propose an $\omega - net$ network to filter neural descriptors through basic gate modules. Moreover, it fuses features of the same scale and different scales with various strategies. Through $\omega - net$, our model can not only synthesize the photo-realistic driving images but also edit and stitch scenes via neural descriptors. Importantly, we can update the specific areas and stitch them together with the original scene. Scene editing can be used to synthesize the diverse driving scene data from different views even for traffic emergencies.

The main contributions of this paper are summarized as: 1) Our proposed neural rendering engine (READ) enables us to build off a large-scale driving simulation environment that generates realistic data in real time for advanced driver assistance systems; 2) $\omega - net$ network is proposed to obtain a more realistic and detailed driving scenario, where cube projection and multiple sampling strategies are introduced to enable synthesize the large-scale driving scenes; 3) Experiments on the KITTI benchmark (Geiger, Lenz, and Urtasun 2012) and Brno Urban dataset (Ligocki, Jelinek, and Zalud 2020) show the good qualitative and quantitative results, where the driving scenes can be edited and stitched so as to synthesize larger and more diverse driving data.

Related Work

Image-to-image Translation

Many researchers (Gao et al. 2020; Tang, Bai, and Sebe 2020) employ image-to-image translation technique to synthesize photo-realistic street scenes. (Gao et al. 2020) propose an unsupervised GAN-based framework, which adaptively synthesizes images from segmentation labels by considering the specific attributes of the task from segmentation labels to image synthesis. As in (Gao et al. 2020), (Tang, Bai, and Sebe 2020) present a dual-attention GAN that synthesizes the photo-realistic and semantically consistent images with fine detail from input layouts without the additional training overhead. Although the image-to-image translation method can synthesize the realistic street scene, it still cannot guarantee coherence in the scene transformation. Moreover, it can only synthesize the scene from a single view, whose results are far different from the real scene in terms of texture details.

Novel View Synthesis

Neural Radiance Fields (Mildenhall et al. 2020) become a breakthrough for the novel view synthesis task, which is proposed to use a fully connected network of entire scenes optimized by differentiable volume rendering. Recently, lots of its variants are presented to render different objects, such as human (Pang et al. 2021), car (Niemeyer and Geiger 2021), interior scene (Wang et al. 2021), and building (Martin-Brualla et al. 2021). However, NeRF-based methods depend on per-scene fitting. It is hard to fit a large-scale driving scenario. NeRFW (Martin-Brualla et al. 2021) combines appearance embedding and decomposition of transient and static elements through uncertainty fields. Unfortunately, dynamic objects are ignored, which may lead to occlusions in the static scene. Moreover, synthesizing scenes as large as street view requires huge computing resources, which cannot be rendered in real time.

Scene Synthesis by Neural Rendering

NRW take a latent appearance vector and a semantic mask of the transient object's location as input, which render the scene points into the deep frame buffer and learn these initial render mappings to the real photo. This requires a lot of semantic annotation and ignores the transient objects. By combining the traditional graphics pipelines with learnable components, (Thies, Zollhöfer, and Nießner 2019) introduce a new image composition paradigm, named Deferred Neural Rendering, where feature mapping of the target image is learned from UV-map through neural texture. Despite the

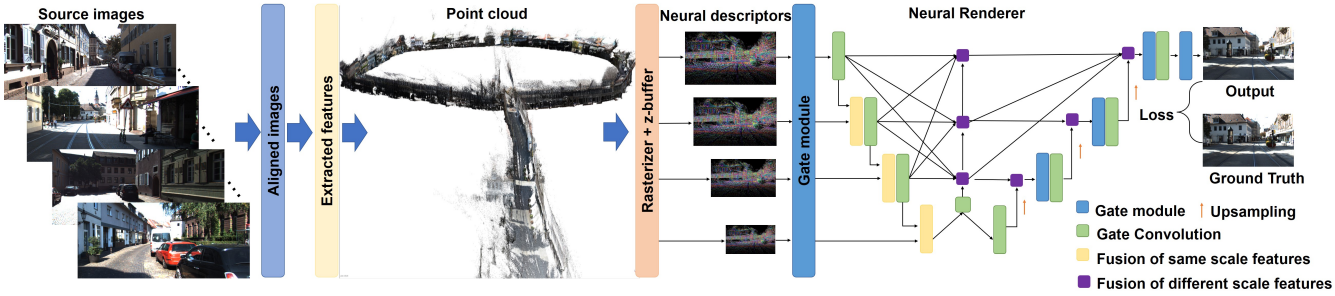

Figure 2: Overview of our proposed large-scale neural scene Rendering (READ) for Autonomous Driving. The input image is firstly aligned, and then the point cloud is obtained by matching feature points and dense construction. We rasterize points at several resolutions. Given the point cloud P, the learnable neural descriptor D, and the camera parameter C, our presented $\omega - net$ rendering network synthesizes the realistic driving scenes by filtering neural descriptors learned from the data and fusing the features from the same scale and different scales.

promising results, it is time-consuming to obtain explicit surfaces of good quality from the point cloud.

The point-based neural rendering method employs point clouds as input to learn the scene representation. NPBG (Aliev et al. 2020) encodes the local geometric shapes and appearance by learning neural descriptors, which synthesizes high-quality novel indoor views from point clouds. TRANSPR (Kolos, Sevastopolsky, and Lempitsky 2020) extends NPBG by augmenting point descriptors with alpha values and replacing Z-buffer rasterization with ray marching, it is able to synthesize semi-transparent parts of the scene. ADOP (Rückert, Franke, and Stamminger 2022) proposes a point-based differentiable neural rendering method, where the parameters of all scenarios are optimized by designing the stages of the differentiable pipeline.

Large-Scale Neural Scene Rendering

Our proposed Neural Scene Rendering approach aims to synthesize the photo-realistic images from an arbitrary camera viewpoint by representing the driving scenes with point clouds. In this section, we first outline our proposed method. Secondly, cube projection is presented in the rasterization process. Thirdly, multiple sampling strategies of sparse point clouds are proposed to reduce the computational cost for large-scale driving scenes. Finally, $\omega - net$ is proposed to represent driving scenes with sparse point clouds and synthesize realistic driving scenarios, and the driving scenes are edited and stitched to provide synthetic data for larger and richer driving scenes.

Overview

Given a set of input images for a driving scene and the point cloud $P = \{p_1, p_2, \ldots, p_N\}$ with known camera parameters, our framework is capable of synthesizing the photo-realistic driving scenes from multiple views, as well as stitching and editing driving scenes. To this end, we propose an end-to-end large-scale neural scene rendering that synthesizes realistic images from sparse point clouds. Our framework is divided into the following three parts: rasterization, sampling with the sparse point cloud, and rendering network $\omega - net$. The overview of our proposed framework is illustrated in Fig. 2.

Sparse 3D point cloud P_i can be obtained through the classic structure-from-motion and multiview stereo pipelines, such as Agisoft Metashape (Agisoft 2019). Each point i is located at p_i, which is associated with a neural descriptor vector encoding the local scene content. As in (Aliev et al. 2020), each input 3D point in P contains the position $p_i = \{x_i, y_i, z_i\}$, whose appearance feature is extracted by mapping the RGB value of image pixel to its corresponding 3D space. Neural descriptors $D_i = \{d_1, d_2, \ldots, d_N\}$ are calculated from the input point cloud, namely latent vectors representing local geometry and photometric properties. We update these features by propagating gradient to the input so that the features of the neural descriptor can be automatically learned from data.

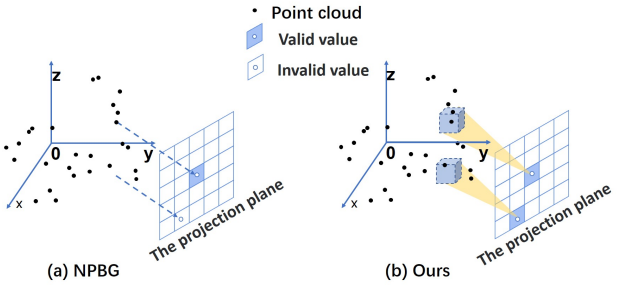

Figure 3: Comparison of point cloud projection methods. (a) NPBG project 3D point clouds directly onto the plane. (b) The cube projection method is proposed to represent the 3D point cloud region corresponding to the neural descriptor.

Rasterization

In the rasterization phase, images of size $W \times H$ are captured by pinhole camera C. We construct a pyramid of rasterized raw images $\{S_t\}_{t=1}^{T}$ ($T = 4$ in all our experiments, the spatial size of S_t is $\frac{W}{2^t} \times \frac{H}{2^t}$), which is to assign the points that pass the depth test to the neural descriptor. Then, it is projected onto the pixel under the full projection transformation of the camera. Essentially, the neural descriptors feature D_i encodes the local 3D scene content around p_i. The rendering network represents a local 3D function g that outputs the

specific neural descriptor $g(i, x)$ at x, modeled by the neural point in its local frame.

Screen out occluded point clouds To avoid updating the descriptors of the occluded points, we approximate the visibility of each point. We use the nearest rasterization scheme by constructing a Z-buffer, reserving only the points with the lowest Z-value at the pixel position. The computational cost of calculating the occluded point cloud is reduced so that the training efficiency is greatly improved.

The cube projection Point-based rendering methods, such as NPBG and ADOP, rendering is performed by rasterizing each world point p_i into a square where the color value of a pixel coordinates I_i on the image space is only determined by a single ray. The world point p_i is directly projected onto the image plane as a neural descriptor, which may lead to many invalid values in the descriptor due to the holes in the sparse point cloud. We consider a cube region to represent the world point p_i. As shown in Fig. 3, this greatly reduces the invalid region of the neural descriptor.

We divide the world space into regions with N voxels. For pixel coordinates I_i on the image space, its color $\hat{I}(I_i)$ is determined by the following:

$$\hat{I}(I_i) = \frac{1}{N} \sum_{i=1}^{N} \hat{I}(p_i) \tag{1}$$

where $\hat{I}(p_i)$ is the color value of world point p_i.

Sampling with Sparse Point Cloud

Synthesizing driving scenes with thousands of meters requires enormous computational power. Therefore, the key idea of our proposed approach is to reduce memory usage and improve training efficiency. Instead of fitting each scene separately, we employ the point clouds to initialize the geometry of the real world scene.

Monte Carlo sampling Due to the different distribution of point clouds in the scene, there are abundant points in the area with obvious features. For the area of the sky or dynamic objects, there are fewer corresponding regional point clouds due to the lack of obvious features or fewer feature points. To train effectively, we propose a training strategy for the Monte Carlo method (Shapiro 2003) to sample a large amount of driving scene data. For the image set S_e in the training phase e, $S_e^* \leftarrow \arg \mathrm{Top}_n Q(I_e)$. $Q(I_e)$ is the synthetic quality of image I_e, which is calculated by perceptual loss. We employ the Top_n samples with the worst performance at each phase as training data. Through sampling, the model strengthens to learn the sparse region of the point cloud. Thus, the overall training time is greatly reduced.

Patch sampling Image resolution also plays a very important role in memory consumption. To this end, we randomly divide the whole image into multiple patches through a sampling strategy, which can select the random patches with the size of $w \times h$ according to the available GPU memory. It is worthy mentioning that the proportion of pixels in patch $w \times h$, to the whole image $W \times H$ is less than 15% in our task. Given the intrinsic matrix K as below

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

where f_x and f_y represent the focal lengths of the x and y axes, respectively. (c_x, c_y) is the position of the principal point with respect to the image plane.

For each image I in S_e, the patch set S_p is obtained by the following strategy to ensure that all areas can be trained:

$$S_p(x, y) \leftarrow \begin{cases} (\alpha \cdot f_x, \alpha \cdot f_y) \\ (c_x + \Delta x, c_y + \Delta y) \end{cases} \tag{2}$$

where α is zoom ratio. It shifts the patch $(\Delta x, \Delta y)$ to enhance the synthetic quality of the scene from different views.

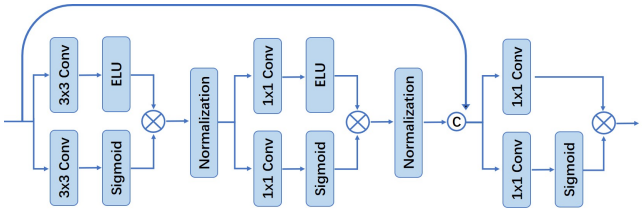

Figure 4: Basic gate module. Neural descriptors learned from sparse point clouds can effectively screen out invalid values.

ω-net

The point clouds, especially those from external reconstruction methods (e.g., Metashape (Agisoft 2019)), often have holes and outliers that degrade the rendering quality. Motivated by MIMO-UNet (Cho et al. 2021), rendering network $\omega - net$ is proposed to synthesize the novel view from sparse point clouds, which consists of three parts.

Given the sparse point cloud P, the purpose of the rendering network is to learn the reliable neural descriptors to represent scenes. However, neural descriptors learned from point clouds still have holes. To deal with this problem, we design a basic gate module to filter the neural descriptors of different scales, as shown in Fig. 4.

By taking into consideration the efficiency, we firstly employ 3×3 convolution layers $CONV$ to extract the feature of neural descriptor D_i. A mask is learned by the sigmoid function to filter the invalid values in the neural descriptor. The output is the value range of (0,1), which represents the importance of features in the neural descriptor. To improve the learning efficiency, we employ the ELU activation function for the neural descriptor. $\otimes$ denotes the element-wise multiplication. We concatenate (©) the initial feature with the filtered one as a new feature. Finally, we use an additional 1×1 convolution layer to further refine the concatenated features. In addition, Gate convolution (Yu et al. 2019) is introduced to re-filter the fused features.

Fusing features at different scales The lack of topology information in the point cloud leads to holes and bleeding. Given the feature F_a of the neural descriptor D_i with holes,

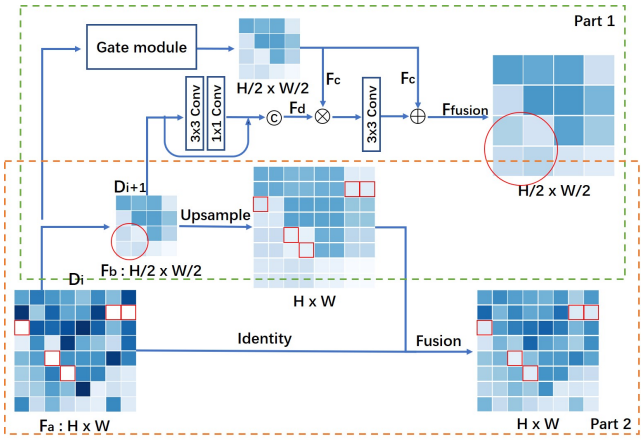

Figure 5: Feature fusion module. Part 1 fuses the features at the same scale, which takes advantage of the complementary information at same scale. Part 2 learns missing points in neural descriptors by fusing the features at different scales.

as shown in the red box in Fig. 5. Although D_i has a higher resolution with fine details, it still suffers from larger surface bleeding. For feature blocks in F_a with no values, rough values can be obtained after average pooling, so D_{i+1} has a low resolution to reduce surface bleeding. Therefore, we suggest $\omega - net$ to fuse multi-scale features to fill the hole in sparse point clouds. In our model, we use neural descriptors of four scales in order to achieve the trade-offs between efficiency and accuracy. Through fusing features at different scales, as shown in Fig. 2, our proposed model learns the missing points in the sparse point cloud so as to synthesize a realistic novel view of the sky, distant objects, etc.

Fusing features at the same scale In our presented method, the feature F_b of neural descriptor D_{i+i} is obtained from D_i by an average pooling operation. The downsampled feature F_d of neural descriptor D_{i+i} concatenates itself with the last 1×1 layer feature, which is used for detail enhancement to retain the lost information. At the same time, the feature F_c with the size of $\frac{H}{2} \times \frac{W}{2}$ is obtained by the neural descriptor D_i of the gate module using the gate convolution. The fusion of F_c and F_d features at the same scale can make use of complementary information among features, where the fusion feature is $F_{fusion} = F_c + F_d \otimes F_c$, as shown in the red circle of Fig. 5.

Scene Editing and Stitching

As our proposed model learns neural descriptors from point clouds to represent scenes, the scene can be edited by changing the trained neural descriptors without retraining. For example, $\{(x, y, z) \mid x \in (x_{min}, x_{max}), y \in (y_{min}, y_{max}), z \in (z_{min}, z_{max})\}$ represents the range of a car in the point clouds. Through back propagation, we employ a rendering network with learnable parameter θ to project all the neural descriptors D onto the RGB image.

$$I_{x,y} = \psi_{\text{pr}}\left(P, D, C, \theta\right) \tag{3}$$

where $I_{x,y}$ is the projected 2D image, and ψ_{pr} denotes the projection and rasterization process. We synthesize the novel view of car $I_{x',y'} = \psi_{\text{pr}}\left(P', D', C, \theta\right)$ via changing its position at $p' = (x', y', z')$. By taking advantage of scene editing, we can not only move objects in the scene, but also remove dynamic objects. This enables our approach to obtain more diverse driving scenes, as shown in Fig. 7.

To account for the large-scale driving scene, we propose a scene stitching method that is able to concatenate multiple scenes and update a block locally. For coordinates at the boundary of scene 1 (x_1, y_1, z_1), the boundary coordinates of scene 2 (x_2, y_2, z_2) need to be stitched. We firstly rotate the point clouds (P_1, P_2) of the two scenes so that they are aligned on a coordinate system at the boundary. The feature descriptors (D_1, D_2) represent the texture of their scenes after being trained by our rendering network. Then, D_1 and D_2 are stitched at the boundary to update the scene. The new scene is $concat(D_1, D_2)$.

Loss function Perceptual loss (Johnson, Alahi, and Fei-Fei 2016), also known as VGG loss, can effectively reflect the image quality of perception more than other loss functions. Thus, we employ the perceptual loss function to prevent smoothing the high-frequency details while encouraging color preservation. Specifically, we compute the perceptual loss between the synthetic novel view and ground truth image I_{GT}, which is calculated by a pretrained VGG layer Φ_l as follows:

$$L\left(D, \theta\right) = \sum_{j=1}^{J} \left(\Phi_j\left(I_{GT}\right) - \Phi_j\left(R_\theta\left(P, D, C\right)\right)\right), \tag{4}$$

where j denotes the cropped patches. Given point cloud P and camera parameters C, our driving scene renderer R_θ learns the neural descriptors D and network parameters θ.

Experiment

To fairly compare the qualitative and quantitative results of various methods, we conduct the experiments on Nvidia GeForce RTX 2080 GPU and evaluate our proposed approach on the two datasets for autonomous driving. To reduce memory consumption on loading the point cloud of the entire large-scale scene, all comparison methods use the sparse point cloud optimized by our method as input.

Datasets

KITTI Dataset (Geiger, Lenz, and Urtasun 2012): KITTI is a large dataset of real driving scenarios, which contains rich scenes. We mainly conducted experiments in three different scenarios, namely Residential, Road and City, covering 3724, 996, and 1335 meters, respectively. Due to the overlapping parts, we finally adopted 3560 frames, 819 frames, and 1584 frames in Residential, Road, and City scenes as the experimental data. We evaluated every 10 frames (e.g., frame 0, 10, 20...) by following the training and testing split of (Aliev et al. 2020; Rückert, Franke, and Stamminger 2022). The rest image frames are used for training. To demonstrate the effectiveness of our method, we conducted the more challenging experiment by discarding 5 test frames before and after every 100 frames as the new testing data, results are given in the supplementary materials.

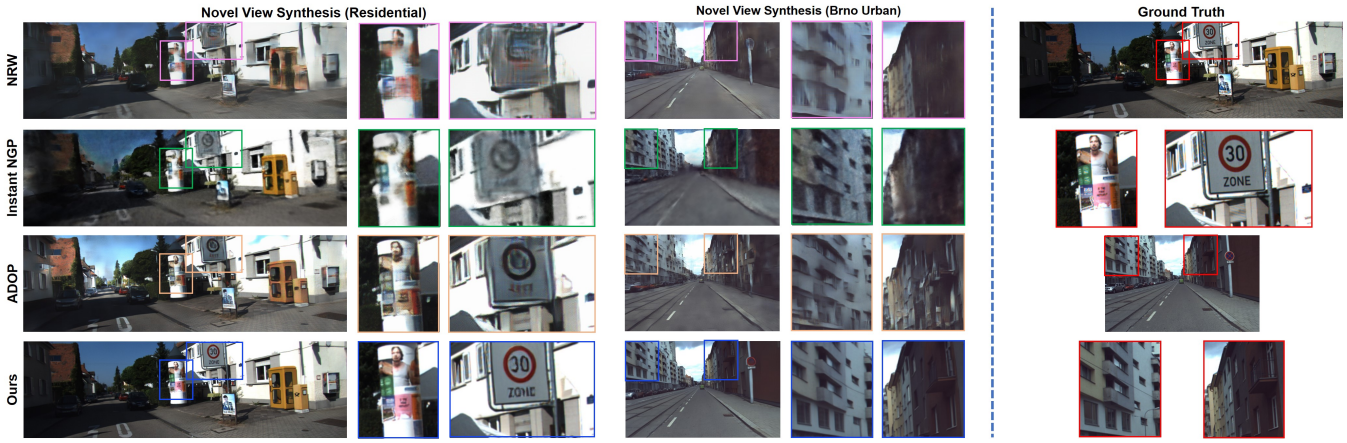

Figure 6: Comparative results of novel view synthesis on the Residential scenes from the KITTI benchmark, and a multiple view scene from the Brno Urban dataset. Comparing to NRW (Meshry et al. 2019), Instant NGP (Müller et al. 2022) and ADOP (Rückert, Franke, and Stamminger 2022), our approach performs the best in cases of vehicles, sky, buildings and road signs. Please zoom in for more details.

Brno Urban Dataset (Ligocki, Jelinek, and Zalud 2020): Compared to KITTI's single-view trajectory, the Brno Urban Dataset contains four views, including left, right, left-front and right-front. In this paper, we use 1641 frames of driving images in our experiments, using the same evaluation criteria as the KITTI Dataset.

Evaluation

Since point clouds from the off-the-shelf methods like MetaShape (Agisoft 2019) often contain holes and outliers, the quality of rendering is usually degraded. Moreover, taking the sparse point cloud as input brings a great challenge to scene synthesis.

To demonstrate the efficacy of our method, we compare it against the image-to-image translation, neural radiance field and neural point-based approaches, including DAGAN (Tang, Bai, and Sebe 2020), NRW (Meshry et al. 2019), NPBG (Aliev et al. 2020), Instant NGP (Müller et al. 2022) and ADOP (Rückert, Franke, and Stamminger 2022), which have achieved promising results in scene synthesis. Followed by the above methods, we employ Peak Signal-to-Noise Ratio (PSNR), Learned Perceptual Image Patch Similarity (LPIPS), Structural Similarity (SSIM), and VGG loss (VGG) as the evaluation metrics.

Tang et al. propose a Novel Dual Attention GAN (DAGAN) algorithm, which can effectively model semantic attention at spatial and channel dimensions. NRW renders the points into the deep frame buffer and learns the mapping from the initial rendering to the actual photo. Instant NGP reconstructs a volumetric radiance field using Hash Encoding. These methods do not work well in the case of detailed textures, as shown in Fig. 6. NPBG (Aliev et al. 2020) and ADOP (Rückert, Franke, and Stamminger 2022) are point-based rendering methods that render realistic textures in real time. However, they have difficulty in dealing with very sparse input, which tends to synthesize the blurred images in the area of point clouds with holes. In addition, ADOP has complicated scene parameters of pipeline having a two-stage rendering network, per image exposure and per image white balance needs to be manually adjusted.

In our proposed approach, we initialize the driving scene through a point cloud in four datasets, which use neural descriptors to encode the geometry and appearance. At each iteration, we sample ten target patches with the size of 256×256 in the KITTI Dataset. Due to the high resolution of images in Brno Urban Dataset, the patch with the size of 336×336 is used for training. In Monte Carlo sampling, we set the sampling ratio to 80%. Table 1 shows the results of our method. It can be seen that our proposed approach is significantly better than the previous methods on all metrics.

Ablation Study

In the ablation experiment, we examine the effectiveness of each module, and more results are given in the supplementary materials. For a fair comparison, we have added the sampling strategy to NPBG as our baseline, namely sampling NPBG, as shown in the first row of Table 2. Then, we gradually add each module to our proposed approach and evaluate them in the KITTI Road scenario. In contrast to sampling NPBG, our proposed gate module can effectively filter the invalid values in neural descriptors, which obtains significant improvement in PSNR, SSIM, and other metrics. By fusing the features at the same scale (Same), the texture of the scene is enhanced with fine details. The fusion of different scale features modules (Differ) and the cube projection(Cube) can effectively fill the value of neural descriptors close to zero. All metrics are greatly improved over the baseline. We also study the influence of different loss functions. It can be observed that combining L_1, PSNR Loss L_{PSNR} and VGG Loss can improve the SSIM index slightly.

Rendering Speed

Rendering speed is an important factor in the inference stage, which facilitates real time feedback in the ADAS. We evaluated the rendering time of the popular methods for novel view synthesis, namely Neural Radiance Fields

	KITTI Residential			KITTI Road			KITTI City			Brno Urban		
	VGG↓	PSNR↑	LPIPS↓	VGG↓	PSNR↑	LPIPS↓	VGG↓	PSNR↑	LPIPS↓	VGG↓	PSNR↑	LPIPS↓
DAGAN	1031.2	14.27	0.3800	847.2	16.84	0.2916	1128.8	13.40	0.3971	657.6	19.08	0.2445
NRW	767.4	18.43	0.3197	748.0	18.58	0.2809	823.7	18.02	0.3102	619.6	19.74	0.3125
NPBG	621.2	19.32	0.2584	597.3	20.25	0.2517	632.8	19.58	0.2480	531.6	20.30	0.2705
Instant NGP	1081.8	15.26	0.5632	1044.0	15.40	0.6056	1105.3	15.46	0.5626	789.3	18.21	0.4857
ADOP	610.8	19.07	0.2116	577.7	19.67	0.2150	560.9	20.08	0.1825	520.6	20.83	0.2189
READ (Ours)	**444.1**	**22.30**	**0.1710**	**359.7**	**24.43**	**0.1351**	**385.5**	**23.58**	**0.1293**	**341.1**	**24.85**	**0.1513**

Table 1: Quantitative evaluation of novel view synthesis on the KITTI and Brno Urban Dataset.

Gate	Same	Differ	Cube	L_1	L_{PSNR}	PSNR↑	LPIPS ↓	SSIM↑
						20.63	0.2359	0.6109
✓						22.29	0.1893	0.6883
✓	✓					23.76	0.1477	0.7205
✓	✓	✓				24.29	0.1465	0.7402
✓	✓	✓	✓			**24.43**	**0.1351**	0.7417
✓	✓	✓		✓		24.16	0.1865	0.7487
✓	✓	✓			✓	23.96	0.1506	0.7325
✓	✓	✓		✓	✓	24.19	0.1863	**0.7490**

Table 2: Ablation study of our method on KITTI road dataset.

Method	VGG↓	PSNR↑	LPIPS↓	SSIM↑
READ	444.1	22.30	0.1710	0.7314
READ w/ stitching	427.3	22.61	0.1612	0.7405

Table 3: Comparisons of scene stitching on KITTI dataset.

(Instant NGP, PyTorch version) and point-based Rendering (ADOP). The rendering resolution is the same as the training images: 1216×368 for the KITTI road scene. As shown in Table 4. Our method is ahead of other methods in rendering quality and time. In particular, The rendering speed advantage of READ over Instant NGP is large (≈ 22x).

Driving Scene Editing and Stitching

Editing the driving scenarios not only provides more synthetic data for ADAS but also simulates the rare traffic conditions in daily life, i.e., a car going the wrong way is about to crash. Moreover, our proposed approach can remove the dynamic objects in the scene so that data collection staff do not need to worry about the impact of complex traffic and vehicles on the restoration of the real scene. This provides convenience for data collection. Additionally, the panoramic view can be synthesized through our method. The larger field-of-view provides more street view information for ADAS, which makes it easier to observe the surrounding environment and deal with emergencies in a timely manner, as shown in Fig. 7. More results are presented in the supplementary materials. By taking advantage of scene stitching, our model is able to synthesize the larger driving scenes and update local areas with obvious changes in road conditions. This can divide the large-scale scene into small parts for efficient training in parallel. As shown in Table 3, it can be seen

Method	PSNR↑	FPS↑
Instant NGP	15.40	0.82
ADOP	19.67	12.02
READ(Ours)	24.43	18.09

Table 4: Comparisons the speeds on KITTI road dataset.

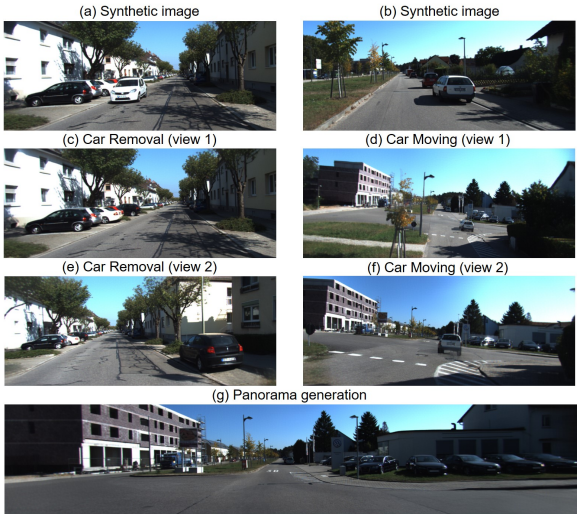

Figure 7: Example results of scene editing. We can move and remove the cars in different views.

that decomposing large scenes into small parts achieves better results.

Conclusion

This paper proposed an efficient neural scene rendering approach to autonomous driving, which makes it possible to synthesize large-scale scenarios on a PC through a variety of sampling schemes and cube projection. We presented an $\omega - net$ rendering network to filter the neural descriptors through basic gate modules, which fused features at the same scale and different scales with different strategies. Our proposed approach not only synthesized the photo-realistic views in real time but also edited and stitched the driving scenes. This can generate various photo-realistic images to train and evaluate the autonomous driving system. The encouraging experimental results showed that our proposed approach significantly outperforms the alternative methods both qualitatively and quantitatively.

Acknowledgements

This work is supported by National Natural Science Foundation of China under Grants (61831015).

References

Agisoft. 2019. Metashape software. *retrieved 20.05.2019*.

Aliev, K.-A.; Sevastopolsky, A.; Kolos, M.; Ulyanov, D.; and Lempitsky, V. 2020. Neural point-based graphics. In *European conference on computer vision*, 696–712. Springer.

Cho, S.-J.; Ji, S.-W.; Hong, J.-P.; Jung, S.-W.; and Ko, S.-J. 2021. Rethinking coarse-to-fine approach in single image deblurring. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4641–4650.

Deng, K.; Liu, A.; Zhu, J.-Y.; and Ramanan, D. 2022. Depth-supervised NeRF: Fewer Views and Faster Training for Free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; and Koltun, V. 2017. CARLA: An open urban driving simulator. In *Conference on robot learning*. PMLR.

Gao, L.; Zhu, J.; Song, J.; Zheng, F.; and Shen, H. T. 2020. Lab2pix: Label-adaptive generative adversarial network for unsupervised image synthesis. In *Proceedings of the ACM International Conference on Multimedia*, 3734–3742.

Geiger, A.; Lenz, P.; and Urtasun, R. 2012. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3354–3361. IEEE.

Johnson, J.; Alahi, A.; and Fei-Fei, L. 2016. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, 694–711. Springer.

Kim, J.; Kim, H.; Lakshmanan, K.; and Rajkumar, R. 2013. Parallel scheduling for cyber-physical systems: Analysis and case study on a self-driving car. In *Proceedings of the ACM/IEEE international conference on cyber-physical systems*, 31–40.

Kolos, M.; Sevastopolsky, A.; and Lempitsky, V. 2020. TRANSPR: Transparency ray-accumulating neural 3D scene point renderer. In *International Conference on 3D Vision*, 1167–1175. IEEE.

Ligocki, A.; Jelinek, A.; and Zalud, L. 2020. Brno Urban Dataset-The New Data for Self-Driving Agents and Mapping Tasks. In *IEEE International Conference on Robotics and Automation*, 3284–3290. IEEE.

Martin-Brualla, R.; Radwan, N.; Sajjadi, M. S.; Barron, J. T.; Dosovitskiy, A.; and Duckworth, D. 2021. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7210–7219.

Meshry, M.; Goldman, D. B.; Khamis, S.; Hoppe, H.; Pandey, R.; Snavely, N.; and Martin-Brualla, R. 2019. Neural rerendering in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6878–6887.

Mildenhall, B.; Srinivasan, P. P.; Tancik, M.; Barron, J. T.; Ramamoorthi, R.; and Ng, R. 2020. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, 405–421. Springer.

Müller, T.; Evans, A.; Schied, C.; and Keller, A. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4).

Niemeyer, M.; and Geiger, A. 2021. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11453–11464.

Pang, A.; Chen, X.; Luo, H.; Wu, M.; Yu, J.; and Xu, L. 2021. Few-shot Neural Human Performance Rendering from Sparse RGBD Videos. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, 938–944. International Joint Conferences on Artificial Intelligence Organization.

Rückert, D.; Franke, L.; and Stamminger, M. 2022. Adop: Approximate differentiable one-pixel point rendering. *ACM Transactions on Graphics*, 41(4).

Shapiro, A. 2003. Monte Carlo sampling methods. *Handbooks in operations research and management science*, 10: 353–425.

Tang, H.; Bai, S.; and Sebe, N. 2020. Dual attention gans for semantic image synthesis. In *Proceedings of the ACM International Conference on Multimedia*, 1994–2002.

Thies, J.; Zollhöfer, M.; and Nießner, M. 2019. Deferred neural rendering: Image synthesis using neural textures. *ACM Transactions on Graphics*, 38(4): 1–12.

Wang, Q.; Wang, Z.; Genova, K.; Srinivasan, P. P.; Zhou, H.; Barron, J. T.; Martin-Brualla, R.; Snavely, N.; and Funkhouser, T. 2021. Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4690–4699.

Yu, J.; Lin, Z.; Yang, J.; Shen, X.; Lu, X.; and Huang, T. S. 2019. Free-form image inpainting with gated convolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4471–4480.

Zhang, J.; Yang, G.; Tulsiani, S.; and Ramanan, D. 2021. NeRS: Neural Reflectance Surfaces for Sparse-view 3D Reconstruction in the Wild. *Advances in Neural Information Processing Systems*, 34.