EVolSplat: Efficient Volume-based Gaussian Splatting for Urban View Synthesis

Sheng Miao[1], Jiaxin Huang[1], Dongfeng Bai[2], Xu Yan[2], Hongyu Zhou[1], Yue Wang[1], Bingbing Liu[2] Andreas Geiger[3,4], Yiyi Liao[1]✉

[1] Zhejiang University [2] Huawei Noah's Ark Lab [3] University of Tübingen [4] Tübingen AI Center

Project Page: https://xdimlab.github.io/EVolSplat/

Figure 1. **Illustration.** Left: Existing feed-forward 3DGS methods (e.g., MVSplat) predict per-pixel Gaussians with local cost volumes. When accumulating Gaussians from multiple local volumes in global coordinates, we observe inconsistencies in the accumulated Gaussians (e.g. car in the figure), leading to ghost artifacts in the rendering. In contrast, EVolSplat predicts 3DGS using a global volume, improving consistency and rendering quality. Right: Our method achieves real-time rendering while maintaining high NVS rendering quality on novel street scenes with lower memory consumption. The circle size indicates memory consumption during inference.

Abstract

Novel view synthesis of urban scenes is essential for autonomous driving-related applications. Existing NeRF and 3DGS-based methods show promising results in achieving photorealistic renderings but require slow, per-scene optimization. We introduce EVolSplat, an efficient 3D Gaussian Splatting model for urban scenes that works in a feed-forward manner. Unlike existing feed-forward, pixel-aligned 3DGS methods, which often suffer from issues like multi-view inconsistencies and duplicated content, our approach predicts 3D Gaussians across multiple frames within a unified volume using a 3D convolutional network. This is achieved by initializing 3D Gaussians with noisy depth predictions, and then refining their geometric properties in 3D space and predicting color based on 2D textures. Our model also handles distant views and the sky with a flexible hemisphere background model. This enables us to perform fast, feed-forward reconstruction while achieving real-time rendering. Experimental evaluations on the KITTI-360 and Waymo datasets show that our method achieves state-of-the-art quality compared to existing feed-forward 3DGS- and NeRF-based methods.

✉ Corresponding author.

1. Introduction

Novel view synthesis (NVS) of urban scenes is essential for autonomous driving applications [7, 24, 33]. Although NeRF-based approaches have shown promising results in urban environments [23, 26, 37, 48], their slow rendering speeds limit practical applications. Recently, newer methods have employed 3DGS for urban view synthesis [5, 16, 17, 45, 55, 57]. However, they still require approximately one hour of training per urban scene.

To reduce training time while maintaining rendering efficiency, recent methods have explored feed-foward 3DGS predictions [2, 6, 21, 28, 31, 32, 39]. These methods share a similar design, utilizing 2D CNN and transformer architectures to predict pixel-aligned 3D Gaussians from 2D reference images. Despite promising NVS results, they encounter challenges in street scenes: 1) These methods rely heavily on feature matching, constructing local frustums under each reference view to predict per-pixel Gaussians. However, driving datasets typically feature small parallax angles and texture-less regions, making depth prediction through feature matching difficult. 2) The pixel-aligned design can lead to duplicate and inconsistent Gaussians in overlapping image regions (e.g. the multi-layer surfaces),

resulting in ghosting artifacts and increased memory usage, as illustrated in Fig. 1.

In this paper, we propose Efficient Volume-based Gaussian Splatting (EVolSplat) for fast reconstruction and real-time rendering of street view images. Unlike existing methods that predict per-pixel aligned Gaussians, our approach directly predicts 3D Gaussians over multiple frames, where the geometry attributes are learned from 3D context and the color is predicted based on 2D texture information. Specifically, given a sequence of sparse images, we initialize our model by accumulating their monocular depth predictions into a noisy point cloud. This point cloud is then processed by a generalizable 3D-CNN, which decodes geometry attributes to transform the noisy point cloud into 3D Gaussians by predicting position offsets, scale, rotation, and opacity. While the 3D-CNN enables accurate geometry prediction, it struggles to capture high-frequency appearance details due to its inherent smoothness bias. To address this, we introduce an occlusion-aware, image-based rendering (IBR) module that predicts Gaussian colors from aggregated 2D features for regions within the volume's range. For feed-forward inference over unbounded scenes, we further model distant views and the sky with 3D Gaussians positioned on a distant hemisphere.

Our contributions can be summarized as follows: 1) We introduce a novel feed-forward reconstruction method tailored to unbounded driving scenes, employing two distinct generalizable components for the foreground and background. It achieves efficient reconstruction and real-time rendering from sparse vehicle-mounted cameras. 2) We predict Gaussian geometric and appearance properties separately using a global volume-based representation and an occlusion-aware IBR module. These enhancements enable high-quality reconstruction of urban scenes, particularly in the presence of occlusions, while also reducing memory consumption. Experimental results show that our method outperforms other generalizable baselines on the KITTI-360 dataset and exhibits promising zero-short generalization abilities on the Waymo dataset.

2. Related Work

Novel View Synthesis for Driving Scenes: The rapid development of radiance field techniques[15, 24] have significantly advanced the development of novel view synthesis for driving scenes. These approach represent street scenes as implicit neural fields [8, 12, 26, 40, 41, 48] or anisotropic 3D Gaussians [5, 16, 17, 45, 55, 57], achieving expressive synthesis quality. A few works [26, 41, 48] leverage sparse vehicle-mounted LiDAR sensors to boost the training and learn a robust geometry. Other works also incorporate additional semantic and geometry cues as priors or supervision to enhance scene comprehension, including semantic understanding [11, 19, 40], geometric constraints [12, 17] and static-dynamic decomposition [47, 57]. However, these methods require expensive per-scene optimization. Our generalizable method, in contrast, aims to perform efficient reconstruction on novel street scenes.

Feed-Forward Scene Reconstruction: To perform generalizable reconstruction, researchers train neural networks across large-scale datasets to gain domain-specific prior knowledge. These methods [2, 6, 32, 52–54] have evolved to work with sparse image sets and can directly reconstruct scenes via fast feed-forward inference. Generalizable NeRF performs ray-based rendering and estimates the 3D radiance field of scenes, which enables superior cross-dataset generalization capabilities in both small [3, 38, 52] and large baseline [9, 43] settings. However, NeRF based methods are notoriously slow to render. Another line of work directly predicts pixel-aligned 3D Gaussians from sparse reference images [6, 7, 31, 32, 39, 54]. These methods either utilize an epipolar transformer [2, 39] or construct local cost volumes [6, 21, 53] for learning geometry. However, most of these methods focus on small-scale scenes and require overlaps in the input images, making them difficult to apply in driving scenes with large camera movements and small parallax. Our method predicts the global scene representation instead of per-pixel Gaussian by accumulating and refining depth information in 3D space. This improves global consistency and reduces ghosting artifacts when rendering new images from new viewpoints. In related work, Flash3D [31] and DrivingForward [36] also uses monocular depth predictions to initialize Gaussian locations but regresses pixel-aligned Gaussian parameters from a 2D feature map. In contrast, we downsample the initial primitives to reduce computation and learn Gaussian geometry in 3D, better incorporating neighbor information.

Point-based Novel View Synthesis: Point clouds capture the geometric structure of a scene, making them a popular representation for enhancing NVS quality. While points can be augmented with learnable descriptors [27, 30], they often lead to rendering holes. To address this, some methods extend point primitives to continuous 3D Gaussians [15], or use neural networks for post-processing to fill gaps [1, 10, 27]. Although these approaches are efficient for rendering complex geometry, they rely on per-scene optimization. In contrast, we propose a generalizable model based on point cloud data.

A few works [13, 46] pretrain an encoder-decoder architecture with skip connections to extract the spatial and local features of point clouds for downstream tasks like object detection and semantic segmentation, but do not consider unbounded street views. PointNeRF [44] and EDUS [23] learn implicit radiance fields from dense input point clouds, limiting their practical applicability due to slow rendering.

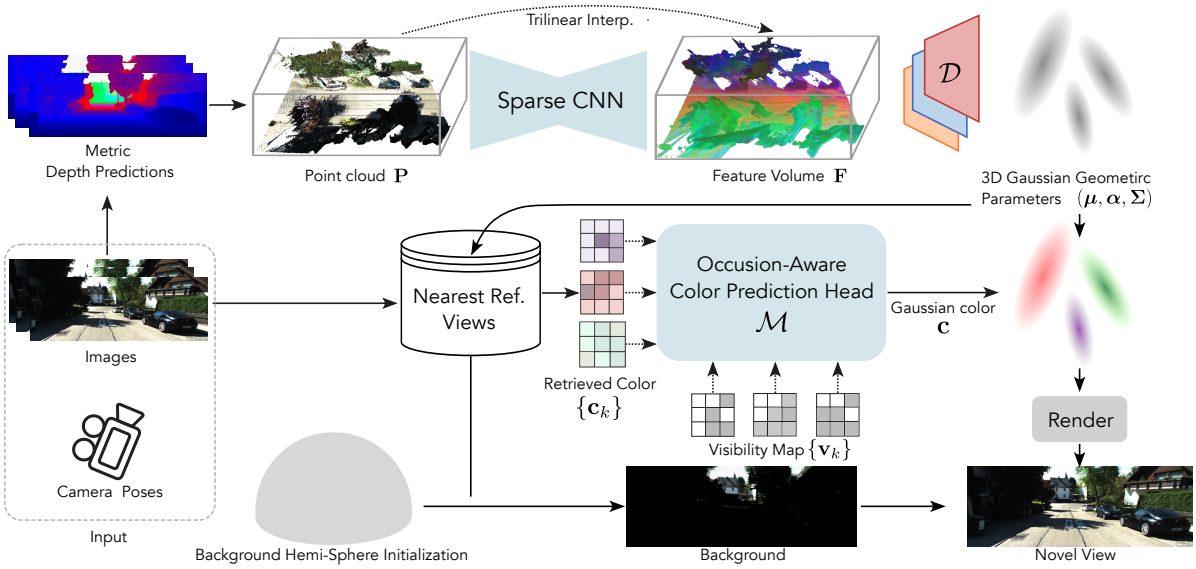


Figure 2. **Method**. EVolSplat learns to predict 3D Gaussians of urban scenes in a feed-forward manner. Given a set of posed images $\{I_n\}_{i=1}^{N}$, we first leverage off-the-shelf metric depth estimators to provide depth estimations $\{D_n\}_{n=1}^{N}$. The depth maps are unprojected and accumulated into a global point cloud $\mathbf{P}$, which is fed into a sparse 3D CNN for extracting a feature volume $\mathbf{F}$. We leverage the 3D context of $\mathbf{F}$ to predict the geometry attributes of 3D Gaussians, including their center $\boldsymbol{\mu}$, opacity $\boldsymbol{\alpha}$, and covariance $\boldsymbol{\Sigma}$. Furthermore, we project the 3D Gaussians to the nearest reference views to retrieve 2D context, including color window $\{\mathbf{c}_k\}_{k=1}^{K}$ and visibility maps $\{\mathbf{v}_k\}_{k=1}^{K}$ to decode their color. To model far regions, we propose a generalizable hemisphere Gaussian model, where the geometry is fixed and the color is predicted in a similar manner as the foreground volume.

Our method achieves real-time rendering by feed-forward prediction of Gaussian primitives.

3. Method

Our goal is to learn a feed-forward 3DGS model for efficient street scene reconstruction from sparse input views, with optional fine-tuning for further improvement. An overview of EVolSplat is provided in Fig. 2. We decompose the entire street scene into foreground and background, and represent them via two independent GS prediction models $\mathcal{G}_{fg}$ and $\mathcal{G}_{bg}$. The term **foreground** denotes the region within a predefined volume that encompasses the vehicle's trajectory, while **background** refers to the distant scenery and sky outside this volume.

We begin with a set of sparse images and exploit an off-the-shelf depth model to initialize global primitives (Section 3.2), which are transformed into an encoding volume to predict the Gaussian geometry and appearance parameters (Section 3.3). We further train a generalizable hemisphere model to represent the background (Section 3.4). Finally, the foreground and background are composed to reconstruct the full image (Section 3.5).

3.1. Preliminary

3D Gaussian Splatting [15] explicitly parameterizes the 3D radiance field of the underlying scene as a collection of 3D Gaussians pritmitives $\mathcal{G} = \{(\boldsymbol{\mu}_i, \boldsymbol{\alpha}_i, \boldsymbol{\Sigma}_i, \boldsymbol{c}_i)\}_{i=1}^{G}$, with attributes: a mean position $\boldsymbol{\mu_i}$, an opacity $\boldsymbol{\alpha_i}$, a covariance matrix $\boldsymbol{\Sigma_i}$ and view-dependent color $\boldsymbol{c_i}$ (computed by spherical harmonics). This efficient representation avoids expensive volumetric sampling and enables high-speed rendering. To render the image from a particular viewpoint, 3DGS employs the tile-based rasterizer for Gaussian splats to pre-sort and blend the ordered primitives using differentiable volumetric rendering:

$$\boldsymbol{c} = \sum_{i \in G} \boldsymbol{c}_i \boldsymbol{\alpha}_i \prod_{j=1}^{i-1} (1 - \boldsymbol{\alpha}_j) \tag{1}$$

3.2. Global Point Cloud Initialization

Depth Estimation: We integrate depth priors into the vanilla 3DGS framework to model the foreground. Given N input images $\{I_n\}_{n=1}^{N}$ for a single scene, we leverage a pre-trained depth model to predict metric depth maps $\{D_n\}_{n=1}^{N}$ for each frame (e.g. Metric3D [51], Unidepth [25]). Inspired by [8, 23], we also apply a depth consistency check to remove inconsistent depth values, and use 3D filters to remove clear floating artifacts in the unproject point clouds. The obtained metric depth information enables us to unproject per-frame monocular depth predictions to global world coordinates. We provide the depth consistency check details in the Supplementary material.

Global Point Cloud Construction: We accumulate the predicted metric depth maps $\{D_n\}_{n=1}^{N}$ into a global point

cloud. Specifically, we lift the RGB images $\{I_n\}_{n=1}^N$ into 3D space and accumulate them to construct a global scene point cloud $\mathcal{P} \in \mathbb{R}^{N_p \times 3}$ in world coordinates with the calibrated intrinsic matrices $\{K_n\}_{n=1}^N$ and extrinsic matrices$\{T_n = R_n \mid t_n\}_{n=1}^N$:

$$\mathcal{P} = \bigcup^{N} \pi^{-1}\left(I_n, D_n, \mathbf{T}_n, \mathbf{K}_n\right) \tag{2}$$

where π^{-1} denotes pixel unprojection. Previous pixel-aligned 3DGS methods will inevitably generate redundant and inconsistent primitives in overlapping region. To address this problem, we apply a uniform filter for the raw points $\mathcal{P}$ in 3D space to delete duplicate primitives while keeping the scene structure. Note that the processed primitives still contain noise, so we apply a statistical filter to remove floaters and explain how to further refine their locations in the next section.

3.3. Volume-Based Gaussian Generation

Construct Neural Volume: We draw inspiration from [4, 13, 14] and train a generalizable 3D latent neural volume for modeling prior knowledge across scenes. More specifically, We construct the sparse tensor from initialized noisy primitives $\mathcal{P}$ and employ an efficient sparse U-Net style 3DCNN ψ^{3D} to aggregate the neural feature volume $\mathbf{F} \in \mathbb{R}^{H \times W \times D \times C}$, where $H \times W \times D$ denotes the spatial resolution and C denotes the feature dimension. The sparse 3D ConvNet follows an encoder-decoder structure with a bottleneck of low spatial dimension. We additionally add skip connections to keep both local and global information.

$$\mathbf{F} = \psi^{\text{3D}}(\mathcal{P}). \tag{3}$$

Different from other feed-forward GS works using a 2D image-to-image neural network to encode the scene, this 3D latent representation helps integrate local and global information in different network layers for faithful reconstructions. Note that the output feature volume $\mathbf{F}$ is aligned with the world coordinate system.

Gaussian Geometry Parameters Prediction: We decode the geometry attributes of N_p 3D Gaussians based on the feature volume $\mathbf{F}$, taking the scene point cloud $\mathcal{P}$ as the initial Gaussian centers, i.e., $\mu_i^{init} = \mathbf{p}_i \in \mathcal{P}$. For each μ_i^{init}, we query a latent feature from $\mathbf{F}$ using trilinear interpolation, and map it to a position offset $\Delta\mu_i$, opacity $\boldsymbol{\alpha}_i$, and covariance matrix $\boldsymbol{\Sigma}_i$ based on three independent MLP heads, respectively.

Considering that the network is supposed to gradually move the 3D Gaussians to the correct locations over the training process, we aim to use the updated locations $\mu_i + \Delta\mu_i$ to query the feature volume to decode $\boldsymbol{\alpha}_i$ and $\boldsymbol{\Sigma}_i$. This encourages the network to predict $\boldsymbol{\alpha}_i$ and $\boldsymbol{\Sigma}_i$ at a more accurate position, i.e., near the object surface. However, we don't have a good estimation of the $\Delta\mu_i$ before training. Therefore, we build an updating rule, with which $\Delta\mu_i$ converges at the end of the training. Specifically, we keep track of the network's prediction of $\Delta\mu_i$ from the last iteration as $\Delta\mu_i^{prev}$. For each iteration, we query $\mathbf{F}$ at $\mu_i^{init} + \Delta_i^{prev}$, obtaining a feature vector $\mathbf{f}_i$:

$$\mathbf{f}_i = \mathbf{F}(\mu_i^{init} + \Delta\mu_i^{prev}) \tag{4}$$

$$\boldsymbol{\alpha}_i = \mathcal{D}_{opa}(\mathbf{f}_i), \quad \boldsymbol{\Sigma}_i = \mathcal{D}_{cov}(\mathbf{f}_i) \tag{5}$$

$$\Delta\mu_i = \mathit{Tanh}(\mathcal{D}_{pos}(\mathbf{f}_i)) \cdot v_{\text{size}}, \quad \mu_i = \mu_{init} + \Delta\mu_i \tag{6}$$

where $\mathcal{D}_{opa}$, $\mathcal{D}_{cov}$ and $\mathcal{D}_{pos}$ denote MLP heads, and μ_i, $\boldsymbol{\alpha}_i$ and $\boldsymbol{\Sigma}_i$ are subsequently used for rendering. Here, the output of $\mathcal{D}_{pos}$ is passed through a *Tanh()* activation function, and then scaled by the voxel size v_{size} to yield the final adjustment. Note that this leads to a recursive formulation of $\Delta\mu_i$ where its value depends on the output of the previous estimations. We show that $\Delta\mu_i$ is stabilized to a stationary point at infinite training iterations in the supplementary. During inference, we initialize $\Delta\mu_i^{prev} = 0$, recursively execute Eq. (4) and Eq. (6), and use the converged point to query $\boldsymbol{\alpha}_i$ and $\boldsymbol{\Sigma}_i$. We empirically find that it is sufficient to only execute the recursion twice to obtain a converged $\Delta\mu_i$.

As shown in our ablation experiment, the inclusion of offset allows our method to compensate for noisy positions and enhances the representational capacity.

Occlusion-Aware IBR-based Color Prediction: Our foreground volume-based representation downsamples the initial primitives, which risks losing high-frequency appearance details. To trade off memory consumption and model high-detail appearance, we introduce the image-based rendering technique to learn Gaussian appearance. However, two challenges arise when applying IBR to urban scenes: 1) coarse Gaussian centers initialization results in inaccurate 2D projection, and 2) the retrieved color is inconsistent for a Gaussian due to occlusions, as illustrated in Fig. 3. To address these issues, we extend the projection location into a local sample window and leverage the depth priors to eliminate the invisibility projection.

Specially, for one Gaussian primitive $\mathcal{G}_i$, we project its center μ_i to the nearby K reference frames and sample the local color $\left\{\mathbf{c}_{ik} \in \mathbb{R}^{W \times W \times 3}\right\}_{k=1}^K$ with a predefined $W \times W$ window centered at the 2D projection of the 3D Gaussian. This strategy allows for integrating additional 2D texture information, enhancing the model's robustness against noisy 3D Gaussian locations. We experimentally observe that $W = 3$ is an effective choice, as demonstrated in the ablation study.

To perform an occlusion check, we bilinearly retrieve the monocular depth D_r as pseudo ground truth and estimate the visibility maps $\{\mathbf{v}_{ik} \in \mathbb{R}^{W \times W}\}$ based on the difference between the projected depth and D_r. Specifically, let δ_{ik} denote the distance between one Gaussian center μ_i and

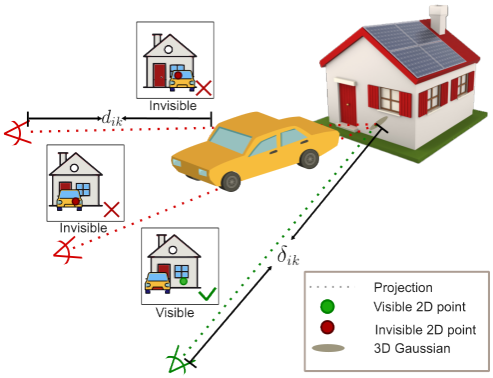


Figure 3. **Occlusion Illustration**. One Gaussian in 3D space may retrieve inaccurate color information from 2D reference images due to occlusions. EVolSplat comprises geometric priors to reduce the impact of invisible colors to enhance rendering quality.

the projected camera center $\mathbf{t}_k$, i.e., $\delta_{ik} = \|\mu_i - \mathbf{t}_k\|_2$. We estimate the visibility based on the depth window $\mathbf{d}_{ik} \in \mathbb{R}^{W\times W}$ corresponding to $\mathbf{c}_{ik}$, where $\mathbf{v}_{ik} = (\delta_{ik} - \mathbf{d}_{ik})/\delta_{ik}$. With the powerful input depth priors as guidance, our method effectively distinguishes the occlusion-caused feature, allowing it to focus on the visible views. Instead of removing invisible projections that require a manually determined threshold, we propose an aggregation module $\mathcal{M}$ to derive the Gaussian color $\mathbf{c}_i$ by combining the local window colors with their corresponding visibility terms. We implement $\mathcal{M}$ as a three-layer MLP of width 64 and adopt the spherical harmonics coefficients (SH) to represent the color. Inspired by [22], we further take as input the distance and relative direction between Gaussian and target camera to make our model structure-aware and facilitate better image synthesis quality.

$$\mathbf{c}_i = \mathcal{M}\left(\{\mathbf{c}_{ik}, \mathbf{v}_{ik}, \delta_{ik}, \overrightarrow{\theta}_{ik}\}_{k=1}^{K}\right), \overrightarrow{\theta}_{ik} = \frac{\mu_i - \mathbf{t}_k}{\|\mu_i - \mathbf{t}_k\|_2} \tag{7}$$

3.4. Generalizable Background Model

In this section, we introduce our generalizable background design tailored for driving scenes to model infinite sky and distant landscapes (often 100 meters away). Obviously, the volume-based representation covers a limited street scene and is insufficient for the unbounded region. Previous work [5, 26, 42, 49] construct environment map conditioned on input direction for a special scene, hence struggle to handle multiple scenes simultaneously.

Given that the background region is far away, we propose a generalizable hemisphere Gaussian model to represent the background. We uniform sampling points over the hemisphere and project all these points to reference images to query the 2D color $\{\mathbf{c}_k\}_{k=1}^{K}$ on the reference image. This background hemisphere has a fixed radius r_{bg} that moves along with the camera motion such that the relative distance is fixed wrt. the target view. To convert each point on this hemisphere into Gaussian primitives $\mathcal{G}_{bg}$, we employ a lightweight two-layer MLP $\mathcal{M}_{bg}$ to learn its color and scale from $\{\mathbf{c}_k\}_{k=1}^{K}$.

$$\mathbf{c}_{bg}, \mathbf{s}_{bg} = \mathcal{M}_{bg}(\{\mathbf{c}_k\}_{k=1}^{K}) \tag{8}$$

Note that we use fixed parameters for its rotation and opacity, see supplementary for more details. This simple strategy allows our model to reconstruct the full image via efficient 3DGS rasterization while keeping the desirable background rendering.

3.5. Training & Fine-tuning

Training Loss: We jointly train our scene representation, foreground volume, and hemisphere background models. We apply L1 and SSIM losses between rendered and observed images for supervision of RGB rendering.

$$\mathcal{L}_{\text{rgb}} = (1 - \lambda_r)\,\mathcal{L}_1 + \lambda_r \mathcal{L}_{\text{ssim}} \tag{9}$$

Following[12, 45], we additionally adopt entropy regularization loss on the foreground accumulated alpha value $\mathbf{O}_{\text{fg}}$ to encourage opaque rendering.

$$\mathcal{L}_{\text{entropy}} = -\sum\left(\mathbf{O}_{\text{fg}} \log \mathbf{O}_{\text{fg}} + (1 - \mathbf{O}_{\text{fg}}) \log\left(1 - \mathbf{O}_{\text{fg}}\right)\right) \tag{10}$$

The total loss can be summarized as follows:

$$\mathcal{L} = (1 - \lambda_r)\,\mathcal{L}_1 + \lambda_r \mathcal{L}_{\text{ssim}} + \lambda_e \mathcal{L}_{\text{entropy}} \tag{11}$$

Fine-tuning: Once fine-tuning is applied, our EVolSplat can achieve photorealism on par with or surpassing other methods, leveraging powerful pretrained weights. Specifically, we first predicts a set of 3D Gaussian primitives for initialization via a direct feed-forward process, where both the geometry and color attributes are generated. The fine-tuning process follows the vanilla 3DGS [15], where we optimize all geometry and color attributes directly. We also enable the growing and pruning of 3D primitives during fine-tuning. During fine-tuning, the number of Gaussians is significantly reduced, as the initial feed-forward Gaussians are redundant. As a result, our model maintains high fidelity while reducing memory consumption. We demonstrate that the pretrained priors accelerate network training and enable faster convergence compared to other test-time optimization methods.

Implementation Details: We train EVolSplat on multiple scenes using Adam optimizers [18] with learning rate 1×10^{-3}. We use torchsparse [35] as the implementation of 3D sparse convolution and choose gsplat [50] as our Gaussian rasterization library. We set the SH degree to 1 for simplicity, following StreetGaussian [45]. During training, we set $\lambda_r = 0.2$, $\lambda_e = 0.1$ as loss weights in our work. For each iteration, our model randomly selects a single image from a random scene as supervision.

Method	KITTI-360			Waymo			FPS↑	Mem.(GB)↓
	PSNR(dB)↑	SSIM↑	LPIPS↓	PSNR(dB)↓	SSIM↑	LPIPS↓		
MVSNeRF [3]	18.44	0.638	0.317	17.86	0.595	0.433	0.025	12.03
MuRF [43]	22.77	0.780	0.229	23.33	0.770	0.269	0.31	26.45
EDUS [23]	22.13	0.745	**0.178**	23.18	0.745	**0.164**	0.14	**5.75**
MVSplat [6]	21.22	0.695	0.268	21.33	0.665	0.308	81.58	16.14
PixelSplat [2]	19.41	0.584	0.357	16.65	0.541	0.579	70.46	26.75
Ours	**23.26**	**0.797**	0.179	**23.43**	**0.786**	0.202	**83.81**	10.41

Table 1. **Quantitative results on KITTI-360 and Waymo datasets** with other generalizable methods. All models are trained on the KITTI-360 dataset using drop50% sparsity level. Metrics are averaged on five validation scenes without any finetuning. Our EVolSplat generalizes better than all the baselines in terms of PSNR and SSIM on both KITTI-360 and Waymo open datasets. It is also worth noting that our method is more memory efficient compared with other 3DGS-based methods.

4. Experiment

4.1. Experimental Setup

Dataset and metrics: We pretrain our model on the KITTI-360 dataset [20]. In the case of training scenes, we collect 160 sequences from different geographic variations, with each sequence comprising 30 stereo images. We use five public validation sets from the KITTI-360 dataset, which has no overlapping with the training set. Additionally, we access the zero-shot generalization performance on the public Waymo Open Dataset [29]. In our experiments, we apply 50% drop rate for all training and validation scenarios to amplify the movement between adjacent cameras.

For novel view synthesis, we adopt existing evaluation protocols, including PSNR, SSIM, and LPIPS, for quantitative assessments. Regarding the render efficiency, we report frames per second (FPS) and inference memory usage on the same device to ensure a fair comparison.

Volume Setup: In this paper, we construct the axis-aligned bounding box to partition the foreground. We set the height (Y-axis) and width (X-axis) of the foreground volume to encompass objects in the field, with a height of 12.8m and a width of 32m. The Z-axis is set to cover the vehicle's forward trajectory with a length of 64m. The input point cloud is voxelized with a voxel size of $(0.1\text{m}, 0.1\text{m}, 0.1\text{m})$, resulting in the volume dimension of $128 \times 320 \times 640$. Notably, our 3D CNN effectively handles *arbitrary* volume size well during inference. Compared to KITTI-360, the Waymo dataset captures urban data with a wider field of view and clearer distant scenes. Therefore, we adjust the foreground volume range to $(50\text{m}, 20\text{m}, 128\text{m})$ in our zero-shot inference experiment.

Baselines: We compare EVolSplat with several representative feed-forward methods, including MVSNeRF [3], MuRF [43], EDUS [23], MVSplat [6] and PixelSplat[2]. Unless stated otherwise, we train and evaluate all generalizable models using the same data as EVolSplat. Note that We train MVSplat and PixelSplat on NVIDIA V100 since their transformer-based architecture requires a large amount of GPU memory, making it hard to train on consumer GPU (e.g., RTX4090 which we use for EVolSplat) at a full resolution. Additionally, we compare the NVS quality and reconstruction time with other fast per-scene optimization methods: Nerfacto [34] and 3DGS [15].

4.2. Comparison with Feed-Forward Methods

Rendering Quality: Tab. 1 and Fig. 4 present quantitative and qualitative comparison results for feed-forward inference, respectively. We adopt the 50% drop rate following existing evaluation protocols [20, 40, 55]. Our proposed EVolSplat achieves on par or better photorealism with the recent state-of-the-art NeRF-based method, while significantly improving rendering speed. Compared with feed-forward 3DGS prediction methods, our method achieves higher rendering quality. Specifically, MVSNeRF [3] mainly focuses on object-level scenes and struggles with unbounded street views, leading to blurry images. We note that EDUS achieves good LPIPS, but has visual artifacts in thin structure and lower PSNR. MVSplat [6] estimates depth for each reference frame using a Transformer, which results in inconsistent 3D Gaussian primitives and causes ghost artifacts under large camera movements.

Model Efficiency: We report the memory consumption during inference on a full-resolution 376×1408 image of KITTI-360 in Tab. 1. MuRF [43] and PixelSplat [2] fail to train with full-resolutionon a consumer GPU (24GB) since their transformer architecture requires computation resources. In contrast, our method leverages an efficient sparse 3DCNN for generalizable reconstruction with only 10.41GB usage, demonstrating superior memory efficiency and practical utility. Additionally, we note that EDUS [23] consumes 5.75GB of memory for inference as its NeRF-based representation only casts 4096 rays each iteration, resulting in lower memory usage but significantly slow rendering speed (0.14fps).

Zero-shot Inference on Waymo: To further verify the

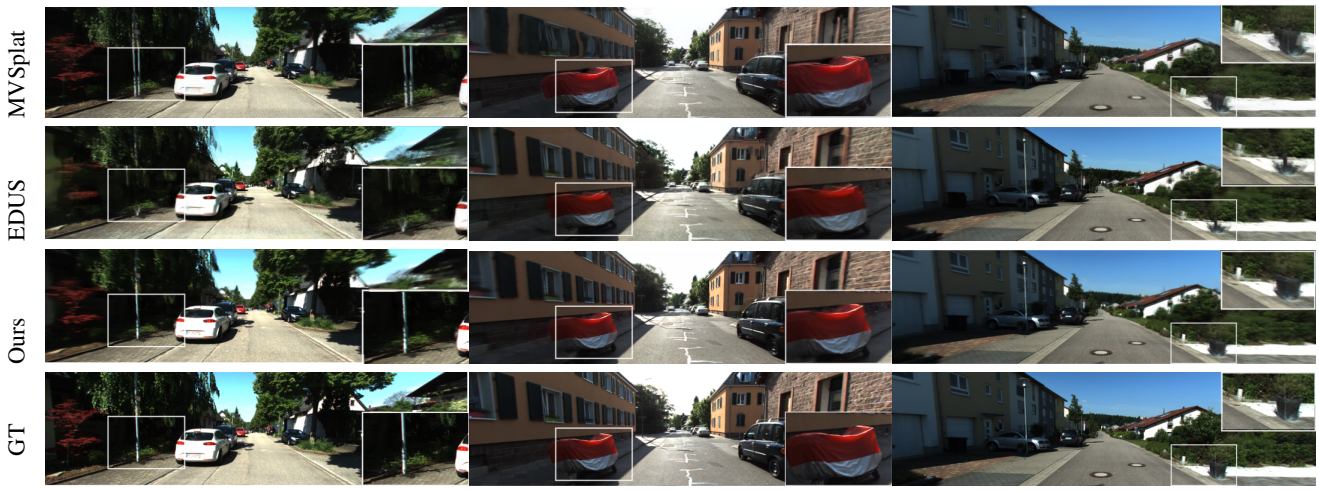
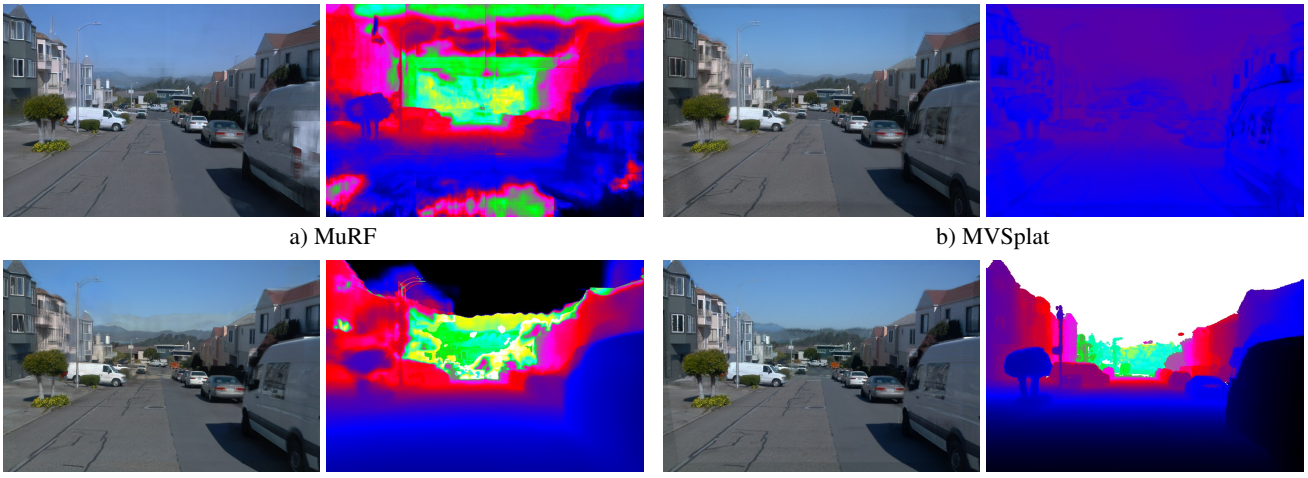


Figure 4. **Qualitative Comparison** with generalizable baselines on the KITTI-360 dataset.



a) MuRF b) MVSplat

c) EDUS d) Ours

Figure 5. **Qualitative Comparisons** with baselines for zero-shot inference on the Waymo Open dataset.

generalization capability, we evaluate different methods on an unseen dataset, the Waymo Open dataset [29]. This is an out-of-domain setting for all methods as the models are trained on the KITTI-360 dataset. We set the image resolution to 640×960 and evaluate the rendering quality and predicted geometry, see Tab. 1 and Fig. 5. As can be seen, our EVolSplat achieves state-of-the-art performance (PSNR: 23.43dB) on the unseen dataset. We observe that MuRF [43] also achieves promising rendering results on Waymo, but it struggles with poor geometric reconstruction in texture-less areas, such as road, due to its strong reliance on feature mapping (as shown in the Fig. 5), which limits its practical applicability on autonomous driving scenes. In contrast, our method benefits from the powerful geometric priors, demonstrating superior zero-shot generalization performance.

4.3. Comparison with Optimization-based Methods

We further compare the finetuning results with other test-time optimization methods and report the PSNR and LPIPS curve on the same test set in Fig. 6. 3DGS [15] and Nerfacto [34] reconstruct the scenes from scratch, requiring longer training time. By leveraging pretrained weights, our model converges with fewer training steps (steps < 1000). Note that our method outperforms 3DGS in terms of LPIPS and has comparable PSNR after convergence.

4.4. Ablation study and analysis

For fast ablation experiments, We select a subset of the training sequences from the KITTI-360 datasets for training. We use these pretrained model to evaluate their feed-forward performance on drop50% sparsity level to ablate the design choices of EVolSplat. Tab. 2 and Fig. 7 presents

	PSNR↑	SSIM↑	LPIPS↓
w/o IBR	21.06	0.706	0.274
w/o refine offset	22.76	0.780	0.195
w/o occlusion check $\mathbf{v}_k$	22.54	0.781	0.197
Full model	22.83	0.786	0.190

Table 2. **Ablation Analysis** on KITTI-360 dataset.

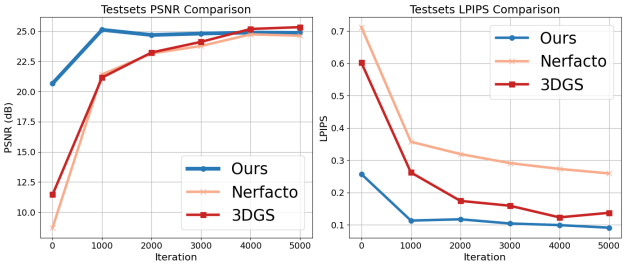


Figure 6. **Comparison with Optimization-based Methods.** We show PSNR and LPIPS on the test set at different training steps. Compared with test-time optimization baselines, our method with generalizable priors converges faster and achieves better LPIPS.

the quantitative and qualitative results respectively.

Effectiveness of IBR Module: We evaluate the impact of the IBR module by comparing our method to a variant that uses the initial point cloud's color without an appearance learning scheme. In this variant, global points learn geometric attributes via networks but retain appearance information from the initial point cloud. Due to the misalignment that exists in the monocular depth predictions, this naïve strategy results in blurry renderings in new scenes (see Fig. 7a). In contrast, our method leverages the IBR module, enabling high-detail rendering by predicting and retrieving colors from reference images.

Effectiveness of Refine offset: We now ablate the position refinement strategy. Our model updates the primitive locations during training considering the predicted depth may be inaccurate. We compare our method with a variant that does not output the position offset $\Delta\mu_i$, using only the predicted positions as Gaussian locations. The results in Tab. 2 show a slight decrease in rendering quality, with more noticeable declines in SSIM and LPIPS metrics. This indicates that our generalized priors effectively compensate for depth estimation errors, facilitating improved rendering quality.

Effectiveness of Occlusion Check: To investigate the occlusion check importance, we perform a study that removes the visibility maps and aggregates all 2D retrieved colors. As illustrated in Fig. 7b, the rendering quality in occluded regions suffers from noticeable blurriness. This is because 3D Gaussians can retrieve inconsistent colors due to the occlusions. By incorporating a visibility map $\mathbf{v}_k$ as guidance, our model can effectively remove the impact of these inconsistencies, resulting in an overall improvement of about 0.3 dB PSNR as reported in Tab. 2.

	PSNR↑	SSIM↑	LPIPS↓
$W = 1$	22.54	0.771	0.216
$W = 3$	22.83	0.786	0.190
$W = 5$	22.84	0.785	0.201

Table 3. **Ablation Study** on different widow size.

(a) w/o IBR (b) w/o occlusion check $\mathbf{v}_k$

(c) Full model (d) Ground Truth

Figure 7. **Qualitative Results of Ablation Study.** We represent rendering images via feed-forward inference on one novel scene.

Analysis of color window size: We further study the effect of different projection window. As presented in Tab. 3, we compare three window sizes: $W = \{1, 3, 5\}$ and evaluate their performance in feed-forward inference on novel scenes. When $W = 1$, each Gaussian retrieves a single pixel in each reference image for color prediction, resulting in the lowest PSNR. This is because inaccurate Gaussian positions lead to inaccurate 2D projections. With increased $W = 3$, more color information is integrated to compensate for projection inaccuracies, resulting the better image quality. At $W = 5$, while more information is queried, the redundancy can cause the image to become blurry, leading to an increase in LPIPS and higher memory consumption. We show more qualitative results in supplementary.

5. Conclusion

This paper presents EVolSplat, a method for efficient urban scene reconstruction in a feed-forward manner. Unlike previous pixel-aligned 3DGS frameworks, we use geometric priors to construct a global volume and predict a standalone 3D representation, achieving state-of-the-art performance across several street-view datasets and enabling real-time rendering, making it well-suited for urban scenes. However, driving scenes still pose significant challenges, particularly in handling dynamic agents such as other vehicles. In future work, we will explore ways to generalize the reconstruction of dynamic street scenes.

Acknowledgements

We would like to thank Dzmitry Tsishkou and Sicheng Li for their useful suggestions and tips. This work is supported by NSFC under grant 62441223, 62202418, and U21B2004.

References

[1] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16*, pages 696–712. Springer, 2020. 2

[2] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19457–19467, 2024. 1, 2, 6

[3] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14124–14133, 2021. 2, 6

[4] Anpei Chen, Haofei Xu, Stefano Esposito, Siyu Tang, and Andreas Geiger. Lara: Efficient large-baseline radiance fields. *arXiv preprint arXiv:2407.04699*, 2024. 4

[5] Yurui Chen, Chun Gu, Junzhe Jiang, Xiatian Zhu, and Li Zhang. Periodic vibration gaussian: Dynamic urban scene reconstruction and real-time rendering. *arXiv preprint arXiv:2311.18561*, 2023. 1, 2, 5

[6] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. *arXiv preprint arXiv:2403.14627*, 2024. 1, 2, 6, 3

[7] Yun Chen, Jingkang Wang, Ze Yang, Sivabalan Manivasagam, and Raquel Urtasun. G3r: Gradient guided generalizable reconstruction. In *European Conference on Computer Vision*, pages 305–323. Springer, 2025. 1, 2

[8] Kai Cheng, Xiaoxiao Long, Wei Yin, Jin Wang, Zhiqiang Wu, Yuexin Ma, Kaixuan Wang, Xiaozhi Chen, and Xuejin Chen. Uc-nerf: Neural radiance field for under-calibrated multi-view cameras. In *The Twelfth International Conference on Learning Representations*, 2023. 2, 3

[9] Yilun Du, Cameron Smith, Ayush Tewari, and Vincent Sitzmann. Learning to render novel views from wide-baseline stereo pairs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4970–4980, 2023. 2

[10] Linus Franke, Darius Rückert, Laura Fink, Matthias Innmann, and Marc Stamminger. Vet: Visual error tomography for point cloud completion and high-quality neural rendering. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–12, 2023. 2

[11] Xiao Fu, Shangzhan Zhang, Tianrun Chen, Yichong Lu, Xiaowei Zhou, Andreas Geiger, and Yiyi Liao. Panopticnerf-360: Panoramic 3d-to-2d label transfer in urban scenes. *arXiv preprint arXiv:2309.10815*, 2023. 2

[12] Jianfei Guo, Nianchen Deng, Xinyang Li, Yeqi Bai, Botian Shi, Chiyu Wang, Chenjing Ding, Dongliang Wang, and Yikang Li. Streetsurf: Extending multi-view implicit surface reconstruction to street views. *arXiv preprint arXiv:2306.04988*, 2023. 2, 5

[13] Di Huang, Sida Peng, Tong He, Honghui Yang, Xiaowei Zhou, and Wanli Ouyang. Ponder: Point cloud pre-training via neural rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16089–16098, 2023. 2, 4

[14] Muhammad Zubair Irshad, Sergey Zakharov, Katherine Liu, Vitor Guizilini, Thomas Kollar, Adrien Gaidon, Zsolt Kira, and Rares Ambrus. Neo 360: Neural fields for sparse view synthesis of outdoor scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9187–9198, 2023. 4

[15] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 2, 3, 5, 6, 7, 1

[16] Bernhard Kerbl, Andreas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. A hierarchical 3d gaussian representation for real-time rendering of very large datasets. *ACM Transactions on Graphics (TOG)*, 43(4):1–15, 2024. 1, 2

[17] Mustafa Khan, Hamidreza Fazlali, Dhruv Sharma, Tongtong Cao, Dongfeng Bai, Yuan Ren, and Bingbing Liu. Autosplat: Constrained gaussian splatting for autonomous driving scene reconstruction. *arXiv preprint arXiv:2407.02598*, 2024. 1, 2

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[19] Abhijit Kundu, Kyle Genova, Xiaoqi Yin, Alireza Fathi, Caroline Pantofaru, Leonidas J Guibas, Andrea Tagliasacchi, Frank Dellaert, and Thomas Funkhouser. Panoptic neural fields: A semantic object-aware neural scene representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12871–12881, 2022. 2

[20] Yiyi Liao, Jun Xie, and Andreas Geiger. Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3292–3310, 2022. 6, 2

[21] Tianqi Liu, Guangcong Wang, Shoukang Hu, Liao Shen, Xinyi Ye, Yuhang Zang, Zhiguo Cao, Wei Li, and Ziwei Liu. Mvsgaussian: Fast generalizable gaussian splatting reconstruction from multi-view stereo. In *European Conference on Computer Vision*, pages 37–53. Springer, 2025. 1, 2

[22] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20654–20664, 2024. 5

[23] Sheng Miao, Jiaxin Huang, Dongfeng Bai, Weichao Qiu, Bingbing Liu, Andreas Geiger, and Yiyi Liao. Effi-

cient depth-guided urban view synthesis. *arXiv preprint arXiv:2407.12395*, 2024. 1, 2, 3, 6

[24] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 1, 2

[25] Luigi Piccinelli, Yung-Hsu Yang, Christos Sakaridis, Mattia Segu, Siyuan Li, Luc Van Gool, and Fisher Yu. Unidepth: Universal monocular metric depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10106–10116, 2024. 3, 2

[26] Konstantinos Rematas, Andrew Liu, Pratul P Srinivasan, Jonathan T Barron, Andrea Tagliasacchi, Thomas Funkhouser, and Vittorio Ferrari. Urban radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12932–12942, 2022. 1, 2, 5

[27] Darius Rückert, Linus Franke, and Marc Stamminger. Adop: Approximate differentiable one-pixel point rendering. *ACM Transactions on Graphics (ToG)*, 41(4):1–14, 2022. 2

[28] Brandon Smart, Chuanxia Zheng, Iro Laina, and Victor Adrian Prisacariu. Splatt3r: Zero-shot gaussian splatting from uncalibarated image pairs. *arXiv preprint arXiv:2408.13912*, 2024. 1

[29] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020. 6, 7

[30] Weiwei Sun, Eduard Trulls, Yang-Che Tseng, Sneha Sambandam, Gopal Sharma, Andrea Tagliasacchi, and Kwang Moo Yi. Pointnerf++: A multi-scale, point-based neural radiance field. In *European Conference on Computer Vision*, pages 221–238. Springer, 2025. 2

[31] Stanislaw Szymanowicz, Eldar Insafutdinov, Chuanxia Zheng, Dylan Campbell, João F Henriques, Christian Rupprecht, and Andrea Vedaldi. Flash3d: Feed-forward generalisable 3d scene reconstruction from a single image. *arXiv preprint arXiv:2406.04343*, 2024. 1, 2

[32] Stanislaw Szymanowicz, Chrisitian Rupprecht, and Andrea Vedaldi. Splatter image: Ultra-fast single-view 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10208–10217, 2024. 1, 2

[33] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8248–8258, 2022. 1

[34] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, et al. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–12, 2023. 6, 7, 2

[35] Haotian Tang, Zhijian Liu, Xiuyu Li, Yujun Lin, and Song Han. TorchSparse: Efficient Point Cloud Inference Engine. In *Conference on Machine Learning and Systems (MLSys)*, 2022. 5

[36] Qijian Tian, Xin Tan, Yuan Xie, and Lizhuang Ma. Drivingforward: Feed-forward 3d gaussian splatting for driving scene reconstruction from flexible surround-view input. *arXiv preprint arXiv:2409.12753*, 2024. 2

[37] Peng Wang, Yuan Liu, Zhaoxi Chen, Lingjie Liu, Ziwei Liu, Taku Komura, Christian Theobalt, and Wenping Wang. F2-nerf: Fast neural radiance field training with free camera trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4150–4159, 2023. 1

[38] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2021. 2

[39] Christopher Wewer, Kevin Raj, Eddy Ilg, Bernt Schiele, and Jan Eric Lenssen. latentsplat: Autoencoding variational gaussians for fast generalizable 3d reconstruction. *arXiv preprint arXiv:2403.16292*, 2024. 1, 2

[40] Zirui Wu, Tianyu Liu, Liyi Luo, Zhide Zhong, Jianteng Chen, Hongmin Xiao, Chao Hou, Haozhe Lou, Yuantao Chen, Runyi Yang, et al. Mars: An instance-aware, modular and realistic simulator for autonomous driving. In *CAAI International Conference on Artificial Intelligence*, pages 3–15. Springer, 2023. 2, 6

[41] Ziyang Xie, Junge Zhang, Wenye Li, Feihu Zhang, and Li Zhang. S-nerf: Neural radiance fields for street views. *arXiv preprint arXiv:2303.00749*, 2023. 2

[42] Congrong Xu, Justin Kerr, and Angjoo Kanazawa. Splatfacto-w: A nerfstudio implementation of gaussian splatting for unconstrained photo collections. *arXiv preprint arXiv:2407.12306*, 2024. 5

[43] Haofei Xu, Anpei Chen, Yuedong Chen, Christos Sakaridis, Yulun Zhang, Marc Pollefeys, Andreas Geiger, and Fisher Yu. Murf: Multi-baseline radiance fields. *arXiv preprint arXiv:2312.04565*, 2023. 2, 6, 7, 3

[44] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5438–5448, 2022. 2

[45] Yunzhi Yan, Haotong Lin, Chenxu Zhou, Weijie Wang, Haiyang Sun, Kun Zhan, Xianpeng Lang, Xiaowei Zhou, and Sida Peng. Street gaussians for modeling dynamic urban scenes. *arXiv preprint arXiv:2401.01339*, 2024. 1, 2, 5

[46] Honghui Yang, Sha Zhang, Di Huang, Xiaoyang Wu, Haoyi Zhu, Tong He, Shixiang Tang, Hengshuang Zhao, Qibo Qiu, Binbin Lin, et al. Unipad: A universal pre-training paradigm for autonomous driving. *arXiv preprint arXiv:2310.08370*, 2023. 2

[47] Jiawei Yang, Boris Ivanovic, Or Litany, Xinshuo Weng, Seung Wook Kim, Boyi Li, Tong Che, Danfei Xu, Sanja Fidler,

Marco Pavone, et al. Emernerf: Emergent spatial-temporal scene decomposition via self-supervision. *arXiv preprint arXiv:2311.02077*, 2023. 2

[48] Ze Yang, Yun Chen, Jingkang Wang, Sivabalan Manivasagam, Wei-Chiu Ma, Anqi Joyce Yang, and Raquel Urtasun. Unisim: A neural closed-loop sensor simulator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1389–1399, 2023. 1, 2

[49] Chongjie Ye, Yinyu Nie, Jiahao Chang, Yuantao Chen, Yihao Zhi, and Xiaoguang Han. Gaustudio: A modular framework for 3d gaussian splatting and beyond. *arXiv preprint arXiv:2403.19632*, 2024. 5

[50] Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, and Angjoo Kanazawa. gsplat: An open-source library for Gaussian splatting. *arXiv preprint arXiv:2409.06765*, 2024. 5

[51] Wei Yin, Chi Zhang, Hao Chen, Zhipeng Cai, Gang Yu, Kaixuan Wang, Xiaozhi Chen, and Chunhua Shen. Metric3d: Towards zero-shot metric 3d prediction from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9043–9053, 2023. 3, 2

[52] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2021. 2

[53] Chuanrui Zhang, Yingshuang Zou, Zhuoling Li, Minmin Yi, and Haoqian Wang. Transplat: Generalizable 3d gaussian splatting from sparse multi-view images with transformers. *arXiv preprint arXiv:2408.13770*, 2024. 2

[54] Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting. In *European Conference on Computer Vision*, pages 1–19. Springer, 2025. 2

[55] Hongyu Zhou, Jiahao Shao, Lu Xu, Dongfeng Bai, Weichao Qiu, Bingbing Liu, Yue Wang, Andreas Geiger, and Yiyi Liao. Hugs: Holistic urban 3d scene understanding via gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21336–21345, 2024. 1, 2, 6

[56] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018. 1

[57] Xiaoyu Zhou, Zhiwei Lin, Xiaojun Shan, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. Drivinggaussian: Composite gaussian splatting for surrounding dynamic autonomous driving scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21634–21643, 2024. 1, 2

EVolSplat: Efficient Volume-based Gaussian Splatting for Urban View Synthesis

Supplementary Material

6. Analysis of the convergence of $\Delta\mu_i$

In this section, we will show the position offset $\Delta\mu_i$ converge to a fixed value at the infinite training iterations. Recall that the $\Delta\mu_i$ in the iteration k can be formulated as:

$$\mathbf{f}_i = \mathbf{F}(\mu_i^{init} + \Delta\mu_i^{k-1}) \tag{12}$$

$$\Delta\mu_i^k = Tanh(\mathcal{D}_{pos}(\mathbf{f}_i)) \cdot v_{\text{size}} \tag{13}$$

We can derive the $\Delta\mu_i^k$ as:

$$\Delta\mu_i^k = Tanh(\mathcal{D}_{pos}(\mathbf{F}(\mu_i^{init} + \Delta\mu_i^{k-1}))) \cdot v_{\text{size}} \tag{14}$$

We define $\mathcal{H} \triangleq Tanh(\mathcal{D}_{pos}(\mathbf{F}(\cdot))) \cdot v_{\text{size}}$ for simplicity, i.e.,

$$\Delta\mu_i^k = \mathcal{H}(\mu_i^{init} + \Delta\mu_i^{k-1}) \tag{15}$$

Given that $\Delta\mu_i$ small, we apply the first-order Taylor expansion to approximate Eq. 15:

$$\Delta\mu_i^k = \mathcal{H}(\mu_i^{init}) + \cdot \left.\frac{\partial\mathcal{H}}{\partial\mu}\right|_{\mu_i^{init}} \Delta\mu_i^{k-1} \tag{16}$$

Considering that $\boldsymbol{\beta} \triangleq \mathcal{H}(\mu_i^{init})$ and $\boldsymbol{\Gamma} \triangleq \left.\frac{\partial\mathcal{H}}{\partial\mu}\right|_{\mu^{init}}$ are both constants, we reformulate Eq. 16 as:

$$\Delta\mu_i^k = \boldsymbol{\beta} + \boldsymbol{\Gamma}\Delta\mu_i^{k-1} \tag{17}$$

The training converges in our experimental observations. Therefore, we assume $\Delta\mu_i^\infty$ converges as k approaches the infinite step, i.e., $k \to \infty$. This allows us to derive $\Delta\mu_i^\infty$:

$$\Delta\mu_i^\infty = \boldsymbol{\beta} + \boldsymbol{\Gamma}\Delta\mu_i^\infty \tag{18}$$

$$= (\boldsymbol{I} - \boldsymbol{\Gamma})^{-1}\boldsymbol{\beta} \tag{19}$$

This indicates that μ_i^∞ converges to a constant value. In supplementary Sec. 9.2, we show that μ_i^t converges quickly at early steps during inference.

7. Implementation Details

7.1. Remove Depth Outliers

We begin by applying a depth consistency check to filter out noisy depth data. Specifically, we unproject the depth map D_i of ith frame to 3D and reproject it to a nearby view j, obtaining a projected depth $D_{i\to j}$. Next, we compare $D_{i\to j}$ and D_j and filter out depths where the absolute relative error exceeds an empirical threshold $\sigma = 0.2m$. We formulate this process as:

$$M_d = |D_{i\to j} - D_j| < \sigma, D_{i\to j} = D_j\left(\pi_j\pi_i^{-1}(\mathbf{u}_i)\right) \tag{20}$$

where $\mathbf{u}_i$ denotes pixel coordinates of ith frame and M_d represents the geometric consistency mask. We further utilize the 3D statistical filter in Open3D [56] to remove the clear floaters in the unprojected point clouds for each frame. In our implementation, we set the number of neighbors to 20 and the standard deviation ratio to 2.0.

7.2. Foreground Gaussian Details

Following [15], $\boldsymbol{\Sigma}$ is decomposed into two learnable components: rotation matrix $\mathbf{R}$ and a scaling matrix $\mathbf{S}$ to holds practical physical significance, see the following formula:

$$\boldsymbol{\Sigma} = \mathbf{RSS}^T\mathbf{R}^T \tag{21}$$

To allow independent optimization of both factors, we use a 3D vector s representing scaling and a quaternion q for rotation separately. Instead of directly learning scales s, we initialize the s_{init} with the average distance of K nearest neighbors using the KNN algorithm and learn the scales residual Δs from the volume latent feature $\mathbf{f}_i$ via a decoder $\mathcal{D}_{cov}$. We experimentally observe that the residual learning strategy helps our network converge faster and enhances the model capacity.

$$\mathbf{s} = \mathbf{s}_{\text{init}} + \Delta\mathbf{s}, \quad \Delta\mathbf{s} = \mathcal{D}_{cov}(\mathbf{f}_i) \tag{22}$$

7.3. Hemisphere Background Details

We also develop the generalizable hemisphere model for the background which typically lies hundreds of meters away from the vehicle, as discussed in the main paper. Specifically, we initialize the background as a hemisphere with a fixed radius $r_{bg} = 100m$ as a hyperparameter, with its center positioned at the midpoint of the foreground volume. This background hemisphere moves along with the vehicle such that the relative distance is fixed wrt. the target camera. We project the points onto K reference images to retrieve their 2D color $\{\mathbf{c}_k\}_{k=1}^K$ to regress all gaussian parameters through network $\mathcal{M}_{bg}$ as mentioned in the main paper. Similar to the foreground framework, we initialize the Gaussian scales using the KNN algorithm and learn their scale residuals $\Delta\mathbf{s}_{bg}$ for each Gaussian.

$$\mathbf{s}_{bg} = \Delta\mathbf{s}_{bg} + \mathbf{s}_{bg}^{init}, \quad \Delta\mathbf{s}_{bg} = \mathcal{M}_{bg}\left(\{\mathbf{c}_k\}_{k=1}^K\right) \tag{23}$$

For opacity and rotation, we explicitly set the opacity to 1 and the rotation to a unit quaternion $[1, 0, 0, 0]$ as a reasonable initialization, without involving them in the network optimization.

7.4. SparseCNN Network Architecture

We build a generalizable efficient 3DCNN ψ^{3D} to provide the geometric priors for the foreground contents. Given the global point cloud $\mathcal{P} \in \mathbb{R}^{N_p \times 3}$, we quantize the point cloud with the voxel size $v_{\text{size}} = 0.1m$ and fed these sparse tensors into the ψ^{3D} to predict the latent feature volume$\mathbf{F}$. The sparse 3DCNN uses a U-Net like architecture with skip connections, comprising some convolution and transposed convolution layers. The details of 3DCNN are listed in the Tab. 4. We use torchsparse as the implementation of ψ^{3D}.

SparseCNN Network Architecture		
Layer	**Description**	**In/Out Ch.**
$\text{Conv}_{i=0}$	kernel = $3 \times 3 \times 3$, stride = 1	3/16
$\text{Conv}_{i=1}$	kernel = $3 \times 3 \times 3$, stride = 2	16/16
$\text{Conv}_{i=2}$	kernel = $3 \times 3 \times 3$, stride = 1	16/16
$\text{Conv}_{i=3}$	kernel = $3 \times 3 \times 3$, stride = 2	16/32
$\text{Conv}_{i=4}$	kernel = $3 \times 3 \times 3$, stride = 1	32/32
$\text{Conv}_{i=5}$	kernel = $3 \times 3 \times 3$, stride = 2	32/64
$\text{Conv}_{i=6}$	kernel = $3 \times 3 \times 3$, stride = 1	64/64
$\text{DeConv}_{i=7}$	kernel = $3 \times 3 \times 3$, stride = 2	64/32
$\text{DeConv}_{i=8}$	kernel = $3 \times 3 \times 3$, stride = 2	32/16
$\text{DeConv}_{i=9}$	kernel = $3 \times 3 \times 3$, stride = 2	16/16

Table 4. **Architecture of SparseConvNet**. Each layer consists of sparse convolution, batch normalization, and ReLU.

8. Baselines

In this section, we discuss the state-of-the-art baselines used for comparison with our approach.

Feed-Forward NeRFs: We adopt the official implementations of MVSNeRF [3], MuRF [43] and EDUS [23]. For each method, we retrain the model using 160 sequences from KITTI-360 [20] under a 50% drop rate. We select the three nearest training frames of the target view as reference images for these methods. MVSNeRF and MuRF utilize multi-view stereo (MVS) algorithms to construct the cost volume and apply 3DCNN to reconstruct a neural field while EDUS leverages the depth priors to learn a generalizable scene representation.

Feed-Forward 3DGS: We adopt the official implementations of PixelSplat [2] and MVSplat [6]. We find that using three reference images caused color shifts at novel viewpoints in urban scenes, so we use the two nearest training frames to achieve optimal performance following the original paper. PixelSplat predicts 3D Gaussians with a two-view epipolar transformer and then spawns per-pixel Gaussians. MVSplat exploits multi-view correspondence information for geometry learning and predicts 3D Gaussians from image features. Both methods are trained on a single Nvidia RTX V100 using the full resolution of KITTI-360.

Additionally, as we illustrated in the teaser figure in the main paper, these pixel-align 3DGS methods predict inconsistent 3DGS when accumulating multiple local volumes. Note that to ensure a fair comparison, we conduct experiments using a single local volume following their default setting(use 2 reference images), as reported in Table 1 and Figure 4 in the main paper.

Test-Time Optimization Methods: To evaluate our fine-tuning results, we compare them against recent test-time optimization methods under the 50% drop rate. Specifically, we use the latest version of Nerfacto [34] provided by Nerfstudio and the official codebase of 3DGS [15]. Nerfacto is a combination of many published methods that demonstrate strong performance on real-world data, including pose refinement, appearance embedding, scene contraction, and hash encoding. For 3DGS, we initialize the 3DGS model with our global point cloud to ensure a fair comparison.

9. Additional Experimental Results

9.1. Monocular Depth Modularity

To further evaluate the sensitivity of our method to different depth estimation approaches, we conduct experiments using two distinct metric depth estimators: Metric3D [51] and UniDepth [25]. Specifically, we pretrain our model with Metric3D, and evaluate using depth maps of two different models for feed-forward inference on novel scenes. As shown in Tab. 5, our EVolSplat consistently outperforms the baseline methods on both depth estimators, demonstrating its robustness in handling depth predictions across varying distributions.

Depth Method	PSNR↑	SSIM↑	LPIPS↓
Metric3D [51]	24.43	0.786	0.202
UniDepth [25]	23.38	0.775	0.223

Table 5. **Depth Sensitivity Experiment** The results are averaged on five testsets from the Waymo dataset.

9.2. Additional Ablation for $\Delta\mu_i$

As mentioned in the main paper, $\Delta\mu_i$ depends on the previous estimation $\Delta\mu_i^{prev}$, but it stabilizes at a stationary point after infinite training iterations. Note that $\mathcal{D}_{pos}$ is designed to continually decode the offset Δ wrt. μ_{init}, even at the ideal location, avoiding an infinite loop caused by toggling between the ideal offset and zero. We further conduct experiments by recursively updating the offsets ($i = 0, 1, 2, 3$) during inference to verify its convergence. As reported in Tab. 7, our pretrained model successfully refines the noisy primitive's position after the first inference (increasing PSNR by approximately 0.46 dB) and maintains a stable value of 23.78dB even with additional updates.

Method	Waymo			KITTI-360		
	PSNR(dB)↑	SSIM↑	LPIPS↓	PSNR(dB)↓	SSIM↑	LPIPS↓
MuRF [43]	23.66	0.746	0.256	19.83	0.669	0.340
EDUS [23]	23.41	0.769	**0.147**	20.13	0.659	0.257
MVSplat [6]	24.08	0.758	0.197	17.80	0.581	0.361
Ours	**25.06**	**0.820**	0.189	**21.23**	**0.738**	**0.222**

Table 6. **Quantitative results on Waymo and KITTI-360 datasets** with other generalizable methods. All models are trained on the Waymo dataset using drop50% sparsity level. Metrics are averaged on five validation scenes without any finetuning.

	PSNR↑	SSIM↑	LPIPS↓
$i = 0$	23.329	0.794	0.175
$i = 1$	23.787	0.819	0.171
$i = 2$	23.778	0.819	0.171
$i = 3$	23.786	0.819	0.171

Table 7. **Ablation Study** on recursion of $\Delta\mu_i$

Layers	Width	PSNR↑	SSIM↑	LPIPS↓
2	64	22.61	0.780	0.192
3	128	22.69	0.785	0.189
4	128	22.60	0.785	0.191

Table 8. **Ablation study** of the background MLP capacity.

9.3. Training on Waymo

To verify our method's performance given different training sequences, we train our method on the Waymo dataset and evaluate its feed-forward performance on Waymo and KITTI-360, as shown in the Tab. 6. Our method consistently achieves state-of-the-art performance in terms of PSNR and SSIM metrics, indicating its robustness for different driving data distributions.

9.4. Ablation Experiments for Background MLP

Our background model primarily blends colors from nearby reference images rather than learning textures from scratch. As shown in Tab. 8, increasing the layers and width of the background MLP does not yield significant improvements. A light-weight two-layer MLP provides sufficient capacity to reconstruct backgrounds while minimizing computational overhead.

10. More Qualitative Results

10.1. More Qualitative Results in Ablation Study

Removing offset refinement and occlusion check leads to visible artifacts in small regions, such as the car in Fig. 7(b) and Fig. 8. While these components don't significantly affect overall quantitative results, they improve local visual quality. Similarly, the color projection window, which compensates for inaccurate Gaussian positions, also improves local visual quality, see Fig. 9.

Figure 8. Qualitative Results of offset refine strategy

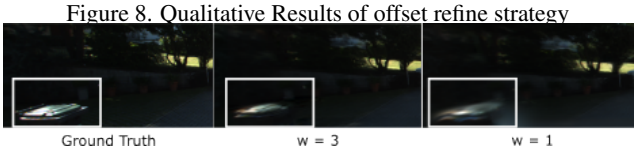


Figure 9. Qualitative Results of windows size strategy

10.2. More Feed-Forward Inference Qualitative Results

Our method enables efficient reconstruction and real-time photorealistic NVS from flexible sparse street view images. We provide more qualitative results on the KITTI-360 dataset via a feed-forward inference under drop50% setting, as shown in Fig. 11.

11. Limitions

We present some failure cases in Fig. 10. A key limitation of the proposed approach is its dependence on the metric depth estimation. Our method suffers degeneration when the depth model struggles to provide fine-grained depth estimates for thin structures.

Another limitation is that our generalizable hemisphere background only roughly approximates the geometry of distant landscapes, leading to artifacts on background region. However, high-quality rendering of the foreground is generally more critical for autonomous driving applications. A potential solution may be to leverage image-based rendering techniques to model the background, though this would reduce rendering efficiency.

Figure 10. **Limitations in thin structure and background**.

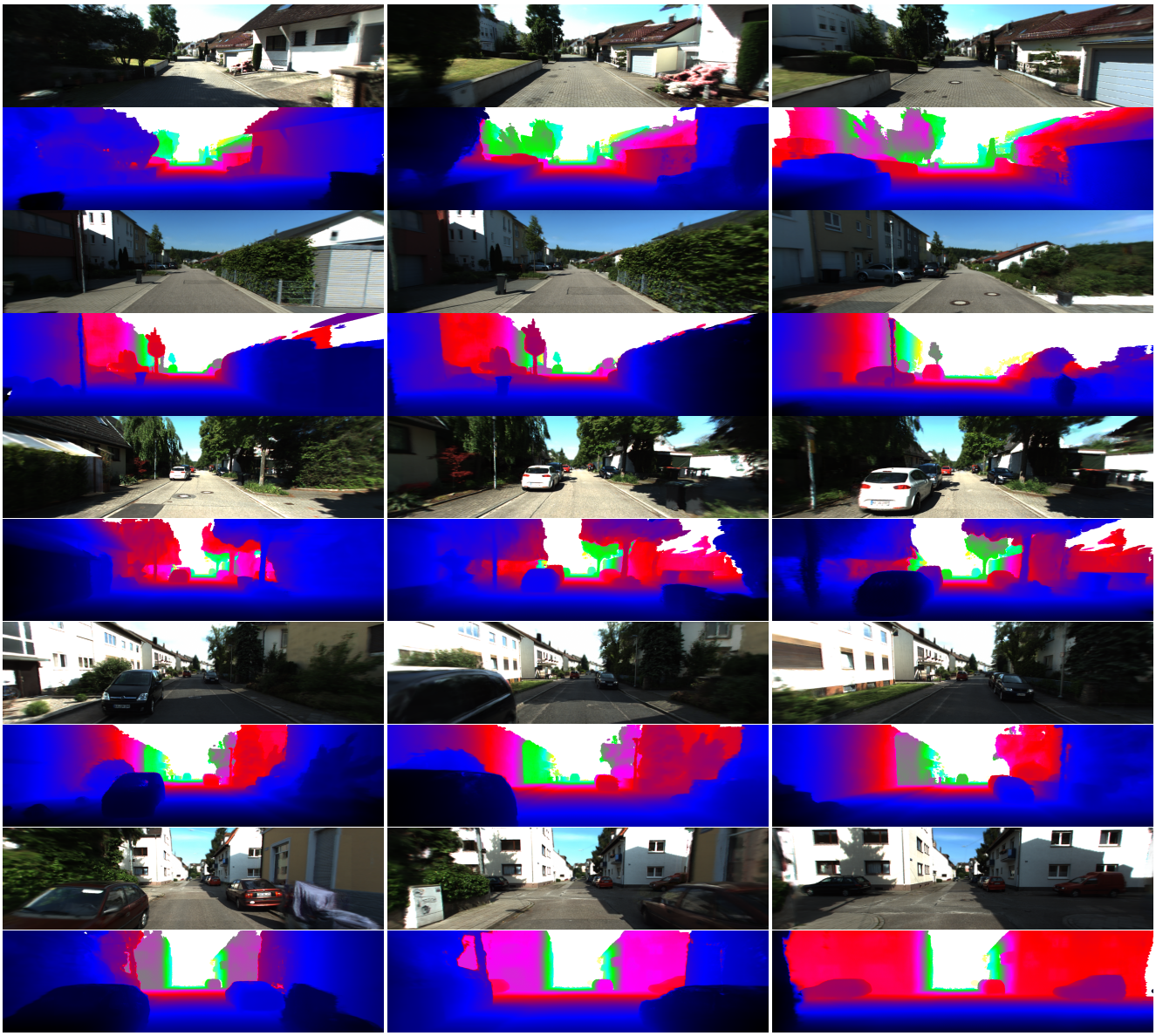


Figure 11. **More Qualitative Results on KITTI-360**. We visualize synthesized images (odd rows) and the corresponding proxy geometry (even rows) on novel scenes generated by our pretrained model through a feed-forward inference.