VEGS: View Extrapolation of Urban Scenes in 3D Gaussian Splatting using Learned Priors

Sungwon Hwang[1]*, Min-Jung Kim[1]*, Taewoong Kang[1], Jayeon Kang[2], and Jaegul Choo[1]

[1]KAIST [2]Ghent University
{shwang.14, emjay73, keh0t0, jchoo}@kaist.ac.kr,
jayeon.kang@ghent.ac.kr

Abstract. Neural rendering-based urban scene reconstruction methods commonly rely on images collected from driving vehicles with cameras facing and moving forward. Although these methods can successfully synthesize from views similar to training camera trajectory, directing the novel view outside the training camera distribution does not guarantee on-par performance. In this paper, we tackle the Extrapolated View Synthesis (EVS) problem by evaluating the reconstructions on views such as looking left, right or downwards with respect to training camera distributions. To improve rendering quality for EVS, we initialize our model by constructing dense LiDAR map, and propose to leverage prior scene knowledge such as surface normal estimator and large-scale diffusion model. Qualitative and quantitative comparisons demonstrate the effectiveness of our methods on EVS. To the best of our knowledge, we are the first to address the EVS problem in urban scene reconstruction. Link to our project page: https://vegs3d.github.io/.

Keywords: Neural Rendering · Urban Scene Reconstruction · Extrapolated View Synthesis (EVS)

1 Introduction

Advancements in neural implicit representations and their rendering methods such as NeRF [22] have enabled accurate, high-fidelity reconstruction of 3D scene and novel view synthesis [3–5, 23]. However, these methods assume certain conditions such as staticity of scene, or dense and diversely distributed training images for accurate scene reconstruction. To handle non-static scenes, a line of works [26, 27, 29] define canonical space and temporal latent vectors to encode per-frame deformation, or learn to separate transient objects via space uncertainty modeling [21, 35]. To relax the dense training set requirements, various methods have been proposed to train NeRFs given a few sparsely distributed images [16, 24, 45, 46]. However, these works mainly focus on the small number of training cameras rather than their pose distribution, which can also be problematic when it is biased toward a certain location or viewpoint.

* Authors contributed equally to this work.

Fig. 1: (a) Illustration of Extrapolated View Synthesis (EVS) problem in urban scenes reconstructed with forward-facing cameras. In contrast to conventional test cameras similar to training camera poses, we evaluate view synthesis on cameras distant from training camera distribution. (b) Qualitative comparison on EVS to baselines.

Meanwhile, some other methods raised specific solutions for urban scene reconstruction using NeRF-based methods. Most of these works either focus on reconstructing scenes with dynamic objects [10,25,41] or improving modeling capacity [25,38], as urban scenes tend to be in large-scale. Notably, Neural Scene Graph [25] and MARS [41] propose to model urban scenes with a graph that comprises multiple neural implicit models for static and dynamic objects as nodes, and 3D bounding boxes and their spatial relations as edges, followed by demonstrating their methods on common driving scene dataset such as KITTI [12]. Block-NeRF [36] proposes to effectively model large-scale scene by dividing a space into multiple blocks, each of which is represented with an independent NeRF network.

However, none of the existing methods on urban scenes address the limited view distribution of training images commonly collected from cameras on vehicles facing and moving forward. Since such characteristic is quite contrary to requiring diversely posed images for accurate scene reconstruction [22], one can easily insinuate that rendering from viewpoints far-distanced from training cameras may yield lower quality. In fact, existing works on urban scene reconstruction [10, 25, 41] construct training and test viewpoints from a single set of forward-facing posed images, which makes the test viewpoints to reside in "*interpolative*" area defined by training cameras. Thus, evaluation on these test cameras is irrelevant for view synthesis looking far on the left, right, and downward with respect to the distribution of training cameras. Considering that observation from such extrapolated views is essential for maximal use of reconstructed scenes, we intend to focus our work on observing, analyzing, and improving rendering quality from these views.

As shown in Figure 1, we formulate such problem as Extrapolated View Synthesis (EVS), and demonstrate that rendering quality does degrade on EVS over existing methods even when they render successfully on the interpolative test cameras. To address the problem, we propose three methods to improve rendering quality on EVS by distilling prior knowledge from LiDAR, surface normal estimator, and large-scale image diffusion model to our scene reconstructions.

Since many applications of view synthesis on urban scene require real-time view synthesis [17], we stem our method from 3D Gaussian Splatting [18], a point-based scene representation method that can yield high-quality rendering in real time with ≈ 144 fps. We propose a method to model and initialize a dynamic scene given point-clouds from LiDAR and off-the-shelf 3D object detectors in order to guide the model with accurate geometry to improve EVS. During scene reconstruction training with photometric loss, we also propose a method to distill surface normal estimations from training images in order to shape and orient covariances of 3D Gaussians suitable for EVS. We then propose a method to fine-tune a large-scale image diffusion model to teach the visual characteristic of the scene while keeping its generalization capability for unseen views, followed by distilling that knowledge to EVS.

In summary, the contributions of this work are four-fold:

- First to tackle extrapolated view synthesis on urban scenes reconstructed with forward-facing cameras to the best of our knowledge.
- Proposal of a dynamic urban scene modeling and reconstruction method in 3D Gaussians [18] using LiDAR.
- Proposal of a rendering and supervision method of covariances in 3D Gaussians with surface normal priors.
- Proposal of a method to training and distilling knowledge from large-scale diffusion model to unobserved views.

2 Related Works

2.1 Neural Scene Representation

Recent innovations driven by NeRF [22] and its variants [3–5] have enabled accurate 3D reconstruction by supervising MLP with densely posed images via differentiable volume rendering. While another line of works [11,23] have improved the rendering speed of NeRFs, 3DGS [18], a unique form of point-based rendering, brought another step of innovation in terms of high-fidelity real-time rendering via point-based scene representation followed by its differential, rasterization-based splatting techniques.

As real scenes tend to be dynamic, recent works [26, 29, 37] define a continuous deformation field that maps an observation coordinate to canonical coordinate where a template NeRF is defined. Notably, HyperNeRF [27] introduces additional high-dimensional canonical space to expand NeRF's capacity to capture topologically-varying motions. Meanwhile, scene reconstruction methods for driving scenes model dynamic objects via bounding-box detections, with an assumption that common objects in driveways such as cars are static within its bounding-box coordinate. Specially, NSG [25] proposed dynamic scene graphs to handle multiple dynamic objects in urban scenes, followed by MARS [41] with instance-aware modeling of dynamic objects.

2.2 Scene Reconstruction with Constrained Viewpoints

Many recent works on few-shot NeRFs defines a problem where there are a few sparsely posed yet well-distributed images for training. Some representative works employs fully convolutional networks [46], vision transformers [24], normalizing flow models [16], or diffusion models [42] as a prior to compensate the lack of training images.

Works closest to our problem definition tackles extrapolated view synthesis, where biased distribution of train cameras are heavily emphasized rather than their number. RapNeRF [48] assumes training cameras to be densely posed in a certain altitude, and test their model in different altitudes. However, the method assume view-agnostic color for pseudo-guidance of unseen rays, which is inappropriate to capture outdoor scenes that often include reflective surfaces or varying lighting conditions, whose images are highly view-dependent. Conversely, NeRFVS [44] enhances the approach by incorporating holistic priors, such as pseudo depth maps and view coverage, derived from neural reconstructions. The method is demonstrated for 3D indoor scenes, offering a possible solution for rendering quality across diverse appearances. Meanwhile, we tackle a new extrapolated view synthesis set-up in outdoor driving scenes where training cameras tend to face and move forwards.

2.3 Scene Reconstruction with Priors

Recent works leverage geometry prior for accurate scene reconstruction. DS-NeRF [8] harnesses free depth from SfM for neural rendering, while neural RGB-D surface reconstruction [2] integrates depth from RGB-D sensors into the NeRF framework for precise 3D models. Notably, MonoSDF [47] demonstrates that depth and normal cues significantly improve reconstruction quality and optimization time. Meanwhile, many urban scene reconstruction methods leverage LiDAR, as it is a common sensor for vehicles in driving scenes. S-NeRF [43] densifies per-frame sparse LiDAR scans via a depth completion network, which is used as a pseudo-guidance for depth renderings. Another LiDAR-based NeRF [7] builds a LiDAR map for scene model. However, their proposed rendering method yields sparse images, not to mention that dynamic objects such as cars that are commonly present in urban scenes are not handled.

3 Method

Given a sequence of frames $k \in \{1 \cdots K\}$ of dynamic urban scene images $\mathcal{I}^k$ captured from forward-facing cameras on driving vehicles, and a sequence of point-cloud set $\mathcal{P}_k$ collected from LiDAR sensor, our goal is to reconstruct a driving scene that can yield photo-realistic renderings on views that are not located in training cameras' distribution. In this article, we will refer to renderings on such views as Extrapolated View Synthesis (EVS). We specify how the camera poses for EVS are parameterized in Sec. 4.

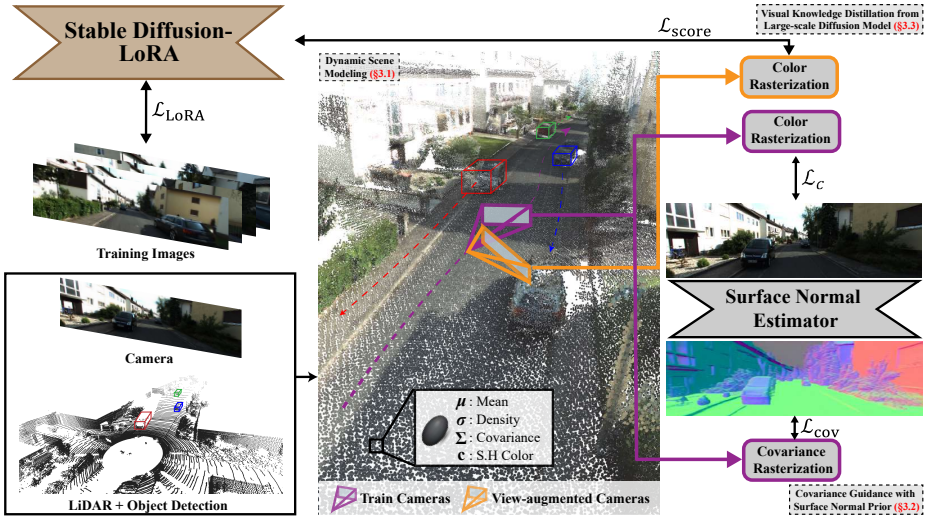

Fig. 2: Our dynamic scene model combines camera, LiDAR, and bounding box estimations with 3D Gaussian Splatting [18] Aside from reconstruction loss $\mathcal{L}_c$, we additionally supervise Gaussian covariances with surface normal priors for improved extrapolated view synthesis (EVS). We also make use of a large-scale diffusion model to distill its knowledge directly to renderings of view-augmented cameras.

Our dynamic scene model integrates camera, LiDAR, and a standard bounding box estimator, leveraging 3D Gaussian Splatting [18] to construct a static and multiple instance-wise Gaussian models (Sec. 3.1). In addition, we learned that optimizing Gaussian models with forward-facing cameras causes the covariance shapes of Gaussians to over-fit to a certain view, making the model unsuitable for EVS. For that, we propose to guide covariance orientation and shape using surface normal priors, introducing a new covariance renderer and supervision method with surface normal maps extracted from training images (Sec. 3.2). Finally, we propose a method for directly supervising extrapolated views by distilling knowledge from a large-scale diffusion model, which we fine-tune a subset of parameters to balance between scene-specific knowledge and generalization to unseen views (Sec. 3.3). We summarize our method in Fig. 2.

3.1 Point-based Neural Rendering with LiDAR integration

Previous method [43] uses per-frame LiDAR scan as a sparse depth supervision. However, considering that a camera frame can also leverage scans from another frames that are visible and within view frustum, we instead propose to construct and utilize a dense point cloud map to distill concentrated scene geometry knowledge to all training views.

Dynamic Scene Modeling and Initialization Our dynamic scene model M comprises a static model M^s and multiple dynamic object models M^i, where i

refers to an i-th instance-wise object. Following 3D Gaussian [18], each model is represented with a set of Gaussian mean $\boldsymbol{\mu}$, a 3D covariance matrix $\boldsymbol{\Sigma}$, density $\boldsymbol{\sigma}$, and color $\boldsymbol{c}$. Covariance matrix is further parameterized by a diagonal scaling matrix $\mathbf{S}$ and a rotation matrix $\mathbf{R}$, so that

$$\boldsymbol{\Sigma} = \mathbf{RSS}^{\mathrm{T}}\mathbf{R}^{\mathrm{T}}. \tag{1}$$

We learned that instead of using sparse LiDAR scans as ground-truth label for optimization, initializing Gaussian means $\boldsymbol{\mu}$ with dense LiDAR maps achieves reasonable balance from over-dependence on LiDAR prior, as LiDAR scans are often prone to measurement noise [1].

Specifically, we separate per-frame LiDAR point clouds to static and instance-wise dynamic points, after which we stack each of them across frames to construct a dense static map and instance-wise point cloud objects. Formally, given P_k, we first use an off-the-shelf 3D bounding box estimator $E(\cdot)$ to yield per-instance and frame bounding box as

$$b_k^i = E(P_k). \tag{2}$$

Using b_k^i, we cull dynamic points within the box, and aggregate them across the frames to initialize means for each instance-wise dynamic Gaussian model, $\boldsymbol{\mu}^i$, that are defined in canonical bounding-box coordinate as

$$\boldsymbol{\mu}^i = \oplus^{k \in K} T_i^k P_k^i, \tag{3}$$

where P_k^i are sub-set of P_k bounded by b_k^i, T_i^k is transformation matrix from LiDAR coordinate in k-th frame to canonical bounding-box coordinate of i-th instance, and $\oplus^{k \in K}$ is concatenation across K frames. We can similarly collect static scene points as

$$\boldsymbol{\mu}^s = \oplus^{k \in K} T_{\mathrm{w}}^k P_k^s, \tag{4}$$

where P_k^s are sub-set of P_k that are bounded by none of b_k^i, and T_{w}^k is a transformation matrix from LiDAR coordinate in k-th frame to world coordinate. In addition, colors of all Gaussians are initialized by projecting P_k to camera planes to assign colors.

Dynamic Scene Rendering and Training To render our dynamic scene, dynamic Gaussian Models in box canonical space should be mapped to world coordinate using known transformation from canonical box coordinate of i-th instance to bounding box location in world coordinate at k-th frame, T_k^i. That is,

$$\boldsymbol{\mu}_k^i = T_k^i \boldsymbol{\mu}^i, \quad \mathbf{R}_k^i = R_k^i \mathbf{R}^i, \tag{5}$$

where R_k^i is a rotation matrix of T_k^i, and $\mathbf{R}^i$ is the rotation matrix that parameterizes a covariance matrix of Gaussian. Finally, all static and dynamic models in world coordinate are jointly rasterized for rendering. Specifically, Gaussian means and covariances are projected to a camera image plane to yield projected

2D mean and covariance μ and Σ using camera extrinsics Q, intrinsics K and its jacobian J as

$$\mu = KQ\boldsymbol{\mu}, \quad \Sigma = \mathrm{J}Q\boldsymbol{\Sigma}Q^{\mathrm{T}}\mathrm{J}^{\mathrm{T}}. \tag{6}$$

μ, Σ and point density $\boldsymbol{\sigma}$ are then used to calculate the probability of rasterized Gaussian to a pixel to calculate α_j [18], followed by alpha blending of Gaussians for each pixel as

$$\tilde{\mathbf{c}} = \sum_{j \in \mathcal{N}} \mathbf{c}_j \alpha_j \prod_{l=1}^{j-1} (1 - \alpha_l), \tag{7}$$

where $\mathbf{c}_j$ is a view-dependent color calculated with spherical harmonics, and $\mathcal{N}$ are indices of ordered points that overlaps the pixel. The scene renderings are then optimized with training images using a photometric loss following [18] as

$$\mathcal{L}_c = (1 - \lambda)\mathcal{L}_1 + \lambda \mathcal{L}_{\text{D-SSIM}}. \tag{8}$$

Bounding Box Optimization In fact, noisy bounding box estimation can cause a dynamic model to be transformed to inaccurate position in world coordinate that does not correspond to its images projected to training cameras. As so, we jointly optimize T_k^i, a transformation from canonical box coordinate of i-th instance to world coordinate at k-th frame, by employing an extra transformation with learnable matrix ΔT_k^i defined for every instance and frame, so that T_k^i can be replaced with

$$T'^i_k = T_k^i \Delta T_k^i, \tag{9}$$

where ΔT_k^i can be further parameterized with a quaternion vector Δq and a translation vector Δt to constrain its optimization within geometrically plausible space. In addition, we regularize ΔT_k^i to identity transformation using the loss $\mathcal{L}_{\text{box}} = ||\Delta q - q_{id.}||_2 + ||\Delta t||_2$, where $q_{id.}$ is an identity quaternion, so that T'^i_k can reside around the initial estimation.

3.2 Covariance Guidance with Surface Normal Prior

The Lazy Covariance Optimization Problem In this section, we identify and tackle the limitation of a 3D Gaussian model optimized with forward-facing cameras. As illustrated in Fig. 3 (a), the shape and orientation of learned covariances tend to over-fit to a certain viewing angle, which we hypothesize that the covariance is trained to cover the the frustum of a training pixel with a minimal optimization effort. As a result, these covariances are prone to produce unwanted cavity on an underlying scene surface, which is revealed when viewed from unobserved angles.

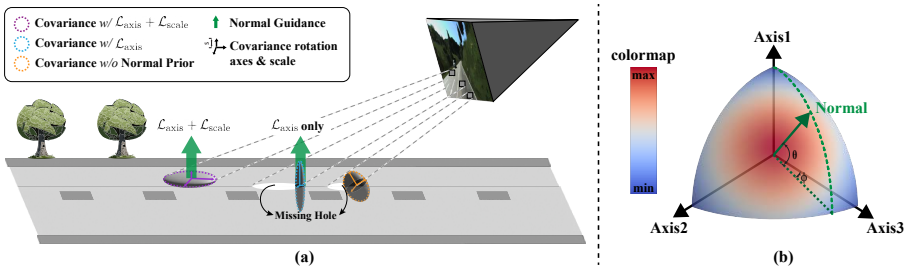

Fig. 3: (a) Working mechanism of $\mathcal{L}_{\text{cov}} = \mathcal{L}_{\text{axis}} + \mathcal{L}_{\text{scale}}$. $\mathcal{L}_{\text{axis}}$ aligns covariance axes to a surface normal vector, and $\mathcal{L}_{\text{scale}}$ minimizes the scale along the covariance axis aligned with surface normal, all of which prevents the Gaussian covariance from minimally satisfying a pixel view frustum, which causes cavity when viewed from another angle. (b) Visualizing $\mathcal{L}_{\text{axis}}$ for different alignment between normal and covariances. $\mathcal{L}_{\text{axis}}$ is minimized when an axis aligns with the normal. See supplements for detailed derivation.

Our key idea is to guide the orientation and shape of covariances to make them behave like the underlying scene surface. In fact, unlike MLP-based representations [47] that can calculate scene surface normal by taking negative gradient of density field with respect to a position via Autograd [28] library, our model cannot render a normal map due to the nature of a discrete representation of Gaussian models. Instead, we suggest a novel covariance rendering technique to approximate scene surface normal from rendered covariance map. Then, we guide the map with a surface normal estimated from training images in two steps: First, we align the orientation of covariances to surface normals using $\mathcal{L}_{\text{axis}}$, followed by flattening the covariance map toward the surface with $\mathcal{L}_{\text{scale}}$. The intuition behind this optimization goal is illustrated in Fig. 3.

Covariance Axes Loss We first propose a method to render covariances axes expressed in quaternion. As alpha-blending based on linear composition is not suitable for quaternion, we re-design Eq. (7) to render the covariance orientation map $\tilde{\mathbf{q}}$ as

$$\tilde{\mathbf{q}} = \prod_{j \in \mathcal{N}} \mathcal{S}\left(\mathbf{q}_I, \mathbf{q}_j, w_j\right), \quad w_j = \alpha_j \prod_{l=1}^{j-1}(1-\alpha_l) \tag{10}$$

where $\mathbf{q}_I$ is an identity quaternion, $\mathcal{S}(\mathbf{q}_I, \mathbf{q}_j, w_j)$ is a slerp function that spherically weights the orientation of j-th covariance $\mathbf{q}_j$ with respect to $\mathbf{q}_I$ by w_j. Weighted covariance orientations are then multiplied for cumulative application of rotations [19]. The rendered quaternion vector map is reformulated into a rotation matrix map and transformed into a training camera coordinate, which we denote as a covariance orientation map in matrix form $\tilde{\mathbf{Q}}$.

$\tilde{\mathbf{Q}}$ is then supervised with surface normal estimated from training images using an off-the-shelf normal prediction network G. Formally, our covariance axes loss is defined as,

$$\mathcal{L}_{\text{axis}} = \sum_{i \in \{0,1,2\}} |\tilde{\mathbf{Q}}[:, i] \cdot G(\mathcal{I})|/3, \tag{11}$$

where $\tilde{\mathbf{Q}}[:, i]$ represents the i-th axis of pixel-wise rendered covariance orientation matrix. As illustrated in Fig. 3 (b), $\mathcal{L}_{\text{axis}}$ is minimized when any of the three covariance axes aligns with the normal vector. We make detailed derivation of this loss in supplements.

Covariance Scale Loss Axis alignment itself, however, cannot prevent the lazy covariance optimization problem, as the scale of the axis that aligns with the normal can still increase to cover the pixel view-frustum, which can still cause the cavity problem. As so, scale of the axis aligned to normal must be minimized to finally induce the covariance to mimic an underlying surface.

Specifically, we can render a covariance scale map $\tilde{\mathbf{S}}$ similar to Eq. (7), and minimize scales proportional to the cosine similarity of its axis with a normal vector. As a result, scale for normal-aligned axis will be minimized, while the remaining two scales can be trained more freely to satisfy the reconstruction loss $\mathcal{L}_c$. Formally, we establish the scale loss as

$$\mathcal{L}_{\text{scale}} = \sum_{i \in \{0,1,2\}} \left| \tilde{\mathbf{S}}[i] \left(\tilde{\mathbf{Q}}[:, i] \cdot G(\mathcal{I}) \right) \right| / 3, \tag{12}$$

where $\tilde{\mathbf{S}}[i]$ is the scale of i-th axis of $\tilde{\mathbf{S}}$. Also, we do not back-propagate to $\tilde{\mathbf{Q}}$ in $\mathcal{L}_{\text{scale}}$ to clearly disentangle the working mechanism of $\mathcal{L}_{\text{axis}}$ and $\mathcal{L}_{\text{scale}}$. Finally, we define our covariance guidance loss as $\mathcal{L}_{\text{cov}} = \lambda_{\text{axis}} \mathcal{L}_{\text{axis}} + (1 - \lambda_{\text{axis}}) \mathcal{L}_{\text{scale}}$.

3.3 Visual Knowledge Distillation from Large-scale Diffusion Model

Denoising Score Matching for Visual Knowledge Distillation Apart from leveraging scene priors such as LiDAR or surface normals during optimization from training cameras, we augment cameras to EVS in order to perform direct guidance to unseen views. However, as training data is not provided for EVS, we instead make use of an image diffusion model in order to distill its knowledge on visual sanity.

We leverage from [14, 39] that noise predicted from a diffusion model $\mathbf{s}_\theta$ is proportional to the log-gradient of prior distribution, or denoising score matching given noise that is small enough [34]. That is, given $\mathbf{x}_\tau = \sqrt{\bar{\alpha}_\tau}\mathbf{x} + (1 - \bar{\alpha}_\tau)\epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$, timestep τ, pre-defined noise schedule $\bar{\alpha}_\tau$, and an image $\mathbf{x}$,

$$\mathbf{s}_\theta(\mathbf{x}_\tau, \tau) \approx -\nabla_{\mathbf{x}} \log p(\mathbf{x}), \tag{13}$$

Thus, optimizing $\mathbf{x}_\tau$ to yield smaller score pushes $\mathbf{x}$ to our prior distribution $p(\cdot)$. Similar to Perturb-and-Average Scoring in Score Jacobian Chaining (SJC) [40] and DiffusioNeRF [42], we design our loss function using Eq.(13) as

$$\nabla_M \mathcal{L}_{\text{score}} = -\mathbf{s}_\theta(\hat{\mathcal{I}}_\tau, \tau), \tag{14}$$

where $\hat{\mathcal{I}}_\tau = \sqrt{\bar{\alpha}_\tau}\, \hat{\mathcal{I}} + (1 - \bar{\alpha}_\tau)\epsilon$ and $\hat{\mathcal{I}}$ is a rendering from our model M on EVS.

Large-scale Diffusion Model with Scene-Specific Adaptation Since the visual distribution of EVS is designed to resemble that of diffusion model as stated in Eq. (13), it is important for our diffusion model to have scene-specific visual understanding, yet can generalize to renderings from unseen views.

Meanwhile, recent works such as DiffusioNeRF [42] trains DDPM [14] with Hypersim [30], a synthetic indoor image dataset, in order to design a critic for visual sanity. However, guidance is conducted via 48x48 patches to prevent from over-fitting to indoor training images. As a result, the model does not strictly have scene-specific understanding, because the data used for training is not visually identical to our scene, not to mention that patch-wise supervision may not be enough to assess scene-specific visual sanity of a rendering as a whole. Meanwhile, GA-NeRF [31] proposes GAN loss between training images and renderings from augmented views. However, adversarial training mechanism is unsuitable to our scenario due to the large difference of camera distribution between training and EVS views, making discriminator hard to be deceived. As so, adversarial training may be unsuitable for guiding unseen views.

To satisfy both properties, we propose to fine-tune a large-scale diffusion model such as Stable Diffusion [33] using LoRA [15], a method commonly used in Large Language Models to fine-tune the low-rank residuals of projection layers in cross-attention. By doing so, our score matching model achieves generalization capability for unseen views by leveraging knowledge from large pretrained model, and scene-specific reconstruction capability by fine-tuning part of the model parameters using our training data. Formally, we use the following loss to fine-tune our diffusion model as

$$\mathcal{L}_{\text{LoRA}} = \mathbb{E}_{\tau,p,\epsilon}[||\epsilon - \mathbf{s}_\theta(\mathcal{I}_\tau, p)||_2^2], \tag{15}$$

where p is a text prompt appropriately chosen for the scene, and $\mathcal{I}_\tau = \sqrt{\bar{\alpha}_\tau}\,\mathcal{I} + (1-\bar{\alpha}_\tau)\epsilon$ are noised training images.

Training Strategy Prior to scene reconstruction, we first fine-tune our diffusion model $\mathbf{s}_\theta$ using Eq. (15) using our training images. Then, we freeze $\mathbf{s}_\theta$ and optimize our scene model M using the final loss formally stated as

$$\nabla\mathcal{L} = \lambda_c\nabla\mathcal{L}_c + \lambda_{\text{box}}\nabla\mathcal{L}_{\text{box}} + \lambda_{cov}\nabla\mathcal{L}_{\text{cov}} + \lambda_{\text{score}}\nabla\mathcal{L}_{\text{score}}. \tag{16}$$

4 Experiments

Dataset We conduct our experiments on KITTI-360 [20] and KITTI [12] Dataset. As KITTI-360 contains 9 voluminous sequences where each sequence contains up to 15000 frames, we divide a sequence into segments of approximately 250 frames. We randomly select 16 segments with dynamic objects and another 16 segments without dynamic objects, which is for fair comparisons on EVS with baselines that do not necessarily handle dynamic objects.

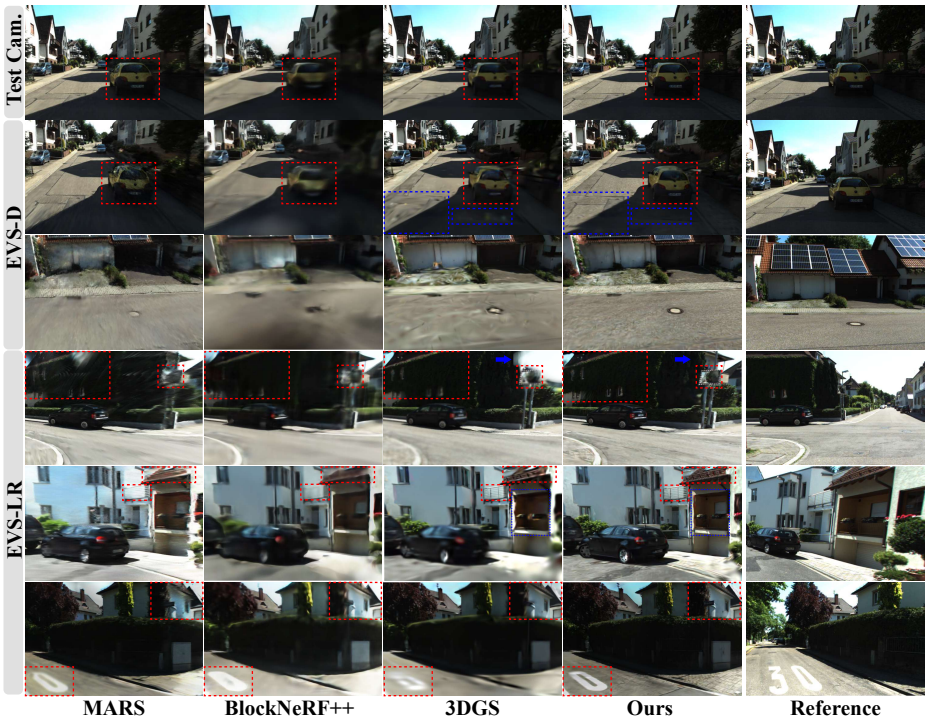

Fig. 4: Qualitative comparison on KITTI-360 [20] for extrapolated view synthesis. EVS-D and EVS-LR refers to extrapolated views facing downwards and left/right, respectively. Test Cam. refers to the conventional test camera sampled from a set of forward-facing cameras. We also report training images for reference that maximally covers the view space of EVS from another location for comparison. Ours outperforms the baselines in terms of geometry and visual sanity.

Evaluation Cameras We first select every 8th frame as conventional test cameras. Then, we construct a EVS camera set that look left and right (EVS-LR) via rotating the test cameras by $\pm 60°$ around Z-axis of world coordinate pointing upward, and another set that look downward (EVS-D) via rotating the test cameras by $10°$ around the x-axis of camera coordinate pointing to the right and translating camera upward in world coordinate by 1.0 in world unit. For EVS-LR, the cameras often cover under-reconstructed spaces on the side of the frame. This nature comes from the forward-facing camera movement, which is quite common in urban scenes. We elaborate more on this phenomenon in supplementary material. Since the renderings from unobserved space disturbs the quantitative results, we remove half of the image plane of EVS-LR camera farther away from the direction of train camera trajectory for experimental comparisons, and resize the cameras for EVS-D and conventional test camera to have the same image plane size with EVS-LR while keeping the same principal point.

Baselines We made our own baseline using BlockNeRF [36], a state-of-the-art large urban scene reconstruction method, with additional supervision with

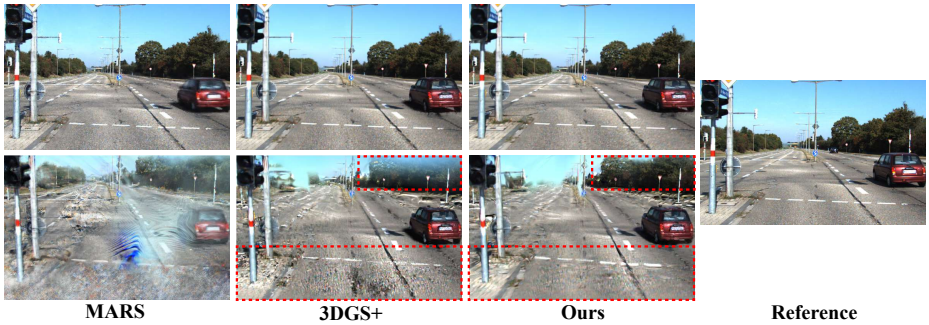

Fig. 5: Qualitative comparison on KITTI [12] dataset from conventional test camera (*top*) and EVS-D (*bottom*).

	FID↓	KID↓	PSNR↑	SSIM↑	LPIPS↓	PSNR*↑	FPS↑
Mip-NeRF 360 [4]	181.5	0.1431	21.59	0.739	0.203	-	0.08
MARS [41]	131.1	0.0617	23.13	0.814	0.125	21.98	0.17
BlockNeRF++ [8,36,47]	245.1	0.1914	21.03	0.723	0.223	-	0.13
3DGS [18]	211.8	0.1382	21.68	0.772	0.192	-	121
3DGS+	126.3	0.0565	23.76	0.814	0.106	22.48	108
VEGS (*ours*)	124.4	0.0561	23.71	0.812	0.106	22.44	108

Table 1: Quantitative results on KITTI-360. FID [13] and KID [6] are measured between EVS and training images. PSNR, SSIM and LPIPS are measured from conventional test cameras on static scenes where ground-truth images are available. PSNR* measures PSNR from conventional test cameras on dynamic object reconstructions.

LiDAR using methods proposed by S-NeRF [43] and normal loss proposed by MonoSDF [47], which we will denote as BlockNeRF++ in this article. We also compare our works with existing urban scene reconstruction methods such and MARS [41] that extends NSG [25] by modeling static scene with NeRF with additional depth prior supervision, as well as MipNeRF 360 [4]. We also compare with 3DGS [18] to compare relative performance between the state-of-the art point-based rendering method, and 3DGS+ that includes our dynamic scene modeling, LiDAR initialization and box optimization method to make 3DGS suitable for dynamic scenes.

5 Results

Comparison to Baselines We report qualitative results of our method and baselines on KITTI-360 in Fig. 4. Our method outperforms the baselines in both EVS-LR and EVS-D. Note that we additionally report renderings on conventional test cameras, which shows that our method is on-par with MARS and better than 3DGS and BlockNeRF++. However, comparison with MARS indicates *that reconstruction quality on the conventional test cameras does not necessarily correspond to the quality on EVS*. Similar analysis can be done on qualitative results of KITTI in Fig. 5. Here, we built and compared with 3DGS+, where we

Fig. 6: Qualitative ablation results on (a) $\mathcal{L}_{\text{cov}}$ and (b) $\mathcal{L}_{\text{score}}$ on EVS. $\mathcal{L}_{\text{cov}}$ effectively guides the Gaussian covariances to faithfully cover the scene surface, yielding noticeably less cavity and better geometry. $\mathcal{L}_{\text{score}}$ effectively improves broken textures, geometry, and removes floating artifacts.

included our dynamic scene modeling method with LiDAR and bounding-box detector, since SfM cannot initialize dynamic object points.

We report quantitative results of our method with baselines in Tab. 1. FID [13] and KID [6] are measured with respect to training images to measure the reconstruction qualities on EVS renderings. Even though small FID/KID cannot be expected due to the large difference of camera distribution between training

Fig. 7: Scene editing results. Since our method models dynamic objects on its own canonical space separated from world coordinate, the reconstructed object can be relocated or removed by manual adjustments.

images and EVS renderings, we use them as an approximation for visual sanity and closeness to the scene. We also measure PSNR, SSIM and LPIPS [49] to evaluate renderings on the conventional test cameras. Ours outperforms Block-NeRF++ and 3DGS in all metrices. However, ours outperform MARS in PSNR and LPIPS, while MARS performs slightly better in SSIM, indicating that performance on conventional test cameras are on par. However, ours out-performs MARS on FID and KID measured from EVS-D and EVS-LR, which aligns with the analysis from the qualitative results in Fig. 4. We also measure PSNR for dynamic objects only, which we denote as PSNR* in Tab. 1, and compare it with MARS. Ours yield slightly better performance in dynamic object reconstruction.

Ablations We report qualitative ablation results on $\mathcal{L}_{\text{cov}}$ and $\mathcal{L}_{\text{score}}$ in Fig. 6a and Fig. 6b, respectively. As can be seen, the lazy covariance optimization problem is effectively ameliorated with $\mathcal{L}_{\text{cov}}$ by removing cavities on surfaces such as floor, wall, and car hood. In addition, $\mathcal{L}_{\text{score}}$ brings noticeable improvement in visual quality such as refining broken texture, geometry, and floater that we conjecture to be originated from Gaussians of ill-posed space such as sky. We report quantitative ablation results in supplements.

Scene Editing In order to demonstrate the effectiveness of our dynamic scene modeling, we conducted scene editing experiments such as removing, translating or rotating the reconstructed dynamic object. We report our editing results in Fig. 7. The result indicates that the dynamic object is well-modeled and separated from the static background model.

6 Conclusion

This work introduces VEGS, a urban scene reconstruction method for improved Extrapolated View Synthesis (EVS) given training images from forward-facing cameras. We introduced techniques to modeling a dynamic scene in 3D Gaussians and integrating dense LiDAR map to the model. We also proposed methods to render and supervise covariances of the Gaussians with surface normal estimations to orient and shape Gaussian covariances suitable for EVS, followed by distilling knowledge from a fine-tuned image diffusion models for better visual sanity. Our comparative studies demonstrated the efficacy of our approaches in addressing the EVS problem.

Acknowledgments This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.RS-2019-II190075 Artificial Intelligence Graduate School Program(KAIST)), the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2022R1A2B5B02001913), and the Air Force Office of Scientific Research under award number FA9550-23-S-0001.

References

1. Adams, M.D.: Lidar design, use, and calibration concepts for correct environmental detection. IEEE Transactions on Robotics and Automation **16**(6), 753–761 (2000)
2. Azinović, D., Martin-Brualla, R., Goldman, D.B., Nießner, M., Thies, J.: Neural rgb-d surface reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6290–6301 (June 2022)
3. Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5855–5864 (2021)
4. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5470–5479 (2022)
5. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Zip-nerf: Anti-aliased grid-based neural radiance fields. arXiv preprint arXiv:2304.06706 (2023)
6. Bińkowski, M., Sutherland, D.J., Arbel, M., Gretton, A.: Demystifying mmd gans. arXiv preprint arXiv:1801.01401 (2018)
7. Chang, M., Sharma, A., Kaess, M., Lucey, S.: Neural radiance field with lidar maps. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 17914–17923 (2023)
8. Deng, K., Liu, A., Zhu, J.Y., Ramanan, D.: Depth-supervised NeRF: Fewer views and faster training for free. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2022)
9. Eftekhar, A., Sax, A., Malik, J., Zamir, A.: Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10786–10796 (2021)
10. Fu, X., Zhang, S., Chen, T., Lu, Y., Zhu, L., Zhou, X., Geiger, A., Liao, Y.: Panoptic nerf: 3d-to-2d label transfer for panoptic urban scene segmentation. In: 2022 International Conference on 3D Vision (3DV). pp. 1–11. IEEE (2022)
11. Garbin, S.J., Kowalski, M., Johnson, M., Shotton, J., Valentin, J.: Fastnerf: High-fidelity neural rendering at 200fps. arXiv preprint arXiv:2103.10380 (2021)
12. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2012)
13. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. Advances in neural information processing systems **30** (2017)

14. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Advances in neural information processing systems **33**, 6840–6851 (2020)
15. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685 (2021)
16. Jain, A., Tancik, M., Abbeel, P.: Putting nerf on a diet: Semantically consistent few-shot view synthesis. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5885–5894 (2021)
17. Kaur, P., Taghavi, S., Tian, Z., Shi, W.: A survey on simulators for testing self-driving cars. In: 2021 Fourth International Conference on Connected and Autonomous Driving (MetroCAD). pp. 62–70. IEEE (2021)
18. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4) (2023)
19. Kuipers, J.B.: Quaternions and rotation sequences: a primer with applications to orbits, aerospace, and virtual reality. Princeton university press (1999)
20. Liao, Y., Xie, J., Geiger, A.: KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. Pattern Analysis and Machine Intelligence (PAMI) (2022)
21. Martin-Brualla, R., Radwan, N., Sajjadi, M.S., Barron, J.T., Dosovitskiy, A., Duckworth, D.: Nerf in the wild: Neural radiance fields for unconstrained photo collections. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7210–7219 (2021)
22. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: ECCV (2020)
23. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Transactions on Graphics (ToG) **41**(4), 1–15 (2022)
24. Niemeyer, M., Barron, J.T., Mildenhall, B., Sajjadi, M.S., Geiger, A., Radwan, N.: Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5480–5490 (2022)
25. Ost, J., Mannan, F., Thuerey, N., Knodt, J., Heide, F.: Neural scene graphs for dynamic scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2856–2865 (2021)
26. Park, K., Sinha, U., Barron, J.T., Bouaziz, S., Goldman, D.B., Seitz, S.M., Martin-Brualla, R.: Nerfies: Deformable neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5865–5874 (2021)
27. Park, K., Sinha, U., Hedman, P., Barron, J.T., Bouaziz, S., Goldman, D.B., Martin-Brualla, R., Seitz, S.M.: Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. arXiv preprint arXiv:2106.13228 (2021)
28. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017)
29. Pumarola, A., Corona, E., Pons-Moll, G., Moreno-Noguer, F.: D-nerf: Neural radiance fields for dynamic scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10318–10327 (2021)
30. Roberts, M., Ramapuram, J., Ranjan, A., Kumar, A., Bautista, M.A., Paczan, N., Webb, R., Susskind, J.M.: Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In: International Conference on Computer Vision (ICCV) 2021 (2021)

31. Roessle, B., Müller, N., Porzi, L., Bulò, S.R., Kontschieder, P., Nießner, M.: Ganerf: Leveraging discriminators to optimize neural radiance fields. ACM Trans. Graph. **42**(6) (nov 2023)
32. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 10684–10695 (June 2022)
33. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models (2021)
34. Song, Y., Ermon, S.: Generative modeling by estimating gradients of the data distribution. Advances in neural information processing systems **32** (2019)
35. Sun, J., Chen, X., Wang, Q., Li, Z., Averbuch-Elor, H., Zhou, X., Snavely, N.: Neural 3D reconstruction in the wild. In: SIGGRAPH Conference Proceedings (2022)
36. Tancik, M., Casser, V., Yan, X., Pradhan, S., Mildenhall, B., Srinivasan, P.P., Barron, J.T., Kretzschmar, H.: Block-nerf: Scalable large scene neural view synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8248–8258 (2022)
37. Tretschk, E., Tewari, A., Golyanik, V., Zollhöfer, M., Lassner, C., Theobalt, C.: Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In: IEEE International Conference on Computer Vision (ICCV). IEEE (2021)
38. Turki, H., Ramanan, D., Satyanarayanan, M.: Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 12922–12931 (June 2022)
39. Vincent, P.: A connection between score matching and denoising autoencoders. Neural computation **23**(7), 1661–1674 (2011)
40. Wang, H., Du, X., Li, J., Yeh, R.A., Shakhnarovich, G.: Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12619–12629 (2023)
41. Wu, Z., Liu, T., Luo, L., Zhong, Z., Chen, J., Xiao, H., Hou, C., Lou, H., Chen, Y., Yang, R., et al.: Mars: An instance-aware, modular and realistic simulator for autonomous driving. arXiv preprint arXiv:2307.15058 (2023)
42. Wynn, J., Turmukhambetov, D.: Diffusionerf: Regularizing neural radiance fields with denoising diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4180–4189 (2023)
43. Xie, Z., Zhang, J., Li, W., Zhang, F., Zhang, L.: S-nerf: Neural radiance fields for street views. In: The Eleventh International Conference on Learning Representations (2022)
44. Yang, C., Li, P., Zhou, Z., Yuan, S., Liu, B., Yang, X., Qiu, W., Shen, W.: Nerfvs: Neural radiance fields for free view synthesis via geometry scaffolds (2023)
45. Yang, J., Pavone, M., Wang, Y.: Freenerf: Improving few-shot neural rendering with free frequency regularization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8254–8263 (2023)
46. Yu, A., Ye, V., Tancik, M., Kanazawa, A.: pixelnerf: Neural radiance fields from one or few images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4578–4587 (2021)

47. Yu, Z., Peng, S., Niemeyer, M., Sattler, T., Geiger, A.: Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. Advances in Neural Information Processing Systems (NeurIPS) (2022)
48. Zhang, J., Zhang, Y., Fu, H., Zhou, X., Cai, B., Huang, J., Jia, R., Zhao, B., Tang, X.: Ray priors through reprojection: Improving neural radiance fields for novel view extrapolation (2022)
49. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: CVPR (2018)