

# Multi-Level Neural Scene Graphs for Dynamic Urban Environments

Tobias Fischer<sup>1</sup>    Lorenzo Porzi<sup>2</sup>    Samuel Rota Bulò<sup>2</sup>

Marc Pollefeys<sup>1</sup>    Peter Kontschieder<sup>2</sup>

<sup>1</sup> ETH Zürich    <sup>2</sup> Meta Reality Labs

<https://tobiasfshr.github.io/pub/ml-nsg/>

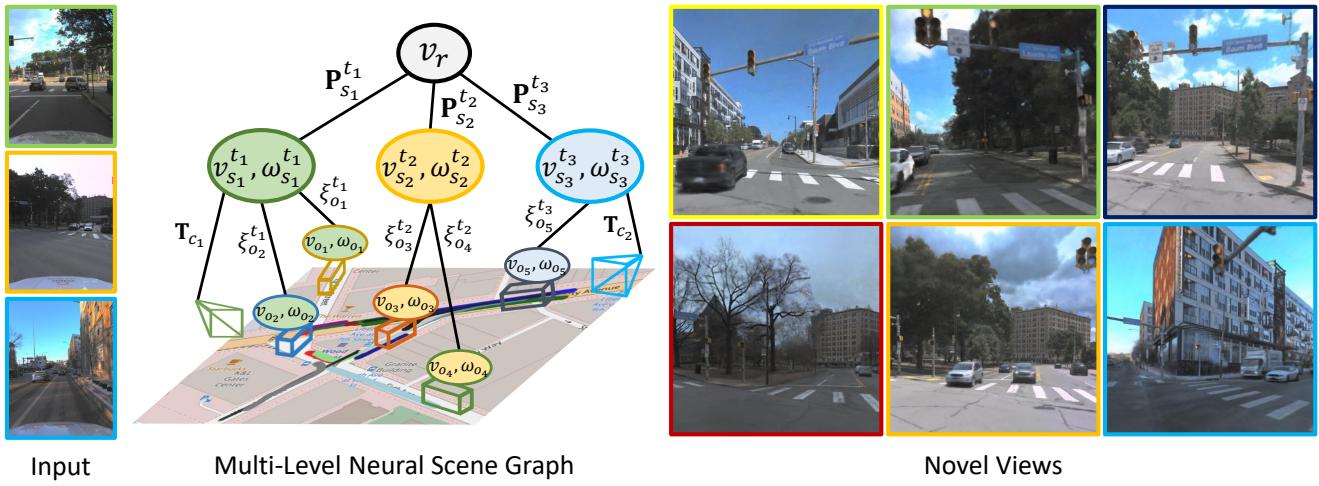


Figure 1. **Overview.** We represent sequences captured from moving vehicles in a shared geographic area with a multi-level scene graph. Each dynamic object  $v_o$  is associated with a sequence node  $v_s^t$  and time  $t$ . The sequence nodes are registered in a common world frame at the root node  $v_r$  through the vehicle poses  $P_s^t$ , while the dynamic objects are localized w.r.t. the sequence node with pose  $\xi_o^t$ . Each camera  $c$  is associated with an ego-vehicle position, i.e. node  $v_s^t$ , through the extrinsic calibration  $T_c$ . The sequence and object nodes hold latent codes  $\omega$  that condition the radiance field, synthesizing novel views in various conditions with distinct dynamic objects.

## Abstract

We estimate the radiance field of large-scale dynamic areas from multiple vehicle captures under varying environmental conditions. Previous works in this domain are either restricted to static environments, do not scale to more than a single short video, or struggle to separately represent dynamic object instances. To this end, we present a novel, decomposable radiance field approach for dynamic urban environments. We propose a multi-level neural scene graph representation that scales to thousands of images from dozens of sequences with hundreds of fast-moving objects. To enable efficient training and rendering of our representation, we develop a fast composite ray sampling and rendering scheme. To test our approach in urban driving scenarios, we introduce a new, novel view synthesis benchmark. We show that our approach outperforms prior art by a significant margin on both established and our proposed benchmark while being faster in training and rendering.

## 1. Introduction

Estimating the radiance field of a dynamic urban environment from data collected by sensor-equipped vehicles is a core challenge in closed-loop simulation for robotics and in mixed reality. It is particularly relevant for applications like autonomous driving and city-scale mapping, where capture vehicles can be frequently deployed. The growing amount of available data provides great opportunities for creating up-to-date digital twins of entire cities but also poses unique challenges as increasingly heterogeneous data sources must be processed. In particular, limited scene coverage, different lighting, weather, seasonal conditions, and varying geometry due to distinct dynamic and transient objects make radiance field reconstruction of dynamic urban environments extremely challenging.

Recently, Neural Radiance Fields (NeRFs) have enabled significant progress in achieving realistic novel view synthesis from a set of input views [2, 32–34, 51]. These methods represent a static 3D scene with fully implicit [31, 32, 51] or low-level explicit structures [11, 14, 17, 24, 33, 45],

[54, 68] such as voxels, points, surfels, meshes or 3D gaussians. In parallel, implicit [20, 39, 52, 65, 66] and low-level explicit [10, 28, 38, 40] neural representations for dynamic 4D scenes have been investigated. However, fewer works have focused on decomposing scenes into higher-level entities [19, 27, 48, 59]. In computer graphics and 3D mapping, scene graphs have been used to represent complex scenes in a multi-level hierarchy [1, 46, 53]. In view synthesis, Ost *et al.* [36] apply this concept by describing actors in a dynamic scene as entities of a scene graph. However, their representation lacks the ability of [1, 46] to represent scenes at multiple levels, thereby limiting their representation to short, single sequences.

While earlier methods for view synthesis focused on object-centric scenes with controlled camera trajectories, recent works move towards radiance field reconstruction of large-scale environments from in-the-wild captures. Among these, many methods focus on static environments, removing dynamic actors from the input data [23, 25, 30, 44, 47, 55, 60]. A few works explicitly model dynamic actors [19, 36, 70], but either do not scale to more than a single, short video [19, 36, 70], or struggle to accurately represent individual object instances [61]. This complicates the evaluation of these methods for real-world applications because existing benchmarks neither scale to large urban areas [19, 36] nor reflect realistic capturing conditions [13].

To address these issues, we propose a multi-level neural scene graph representation that spans large geographic areas with hundreds of dynamic objects. In contrast to previous works, our multi-level scene graph formulation allows us to distinguish and represent dynamic object instances effectively and further to represent a scene under varying conditions. To make our representation viable for large-scale dynamic environments, we develop a composite ray sampling and rendering scheme that enables fast training and rendering of our method. To test this hypothesis, we introduce a benchmark for radiance field reconstruction in dynamic urban environments based on [64]. We fuse data from dozens of vehicle captures under varying conditions amounting to more than ten thousand images with several hundreds of dynamic objects per reconstructed area. We summarize our contributions as follows:

- We propose a multi-level neural scene graph formulation that scales to dozens of sequences with hundreds of fast-moving objects under varying environmental conditions.
- We develop an efficient composite ray sampling and rendering scheme that enables fast training and rendering of our representation.
- We present a benchmark that provides a realistic, application-driven evaluation of radiance field reconstruction in dynamic urban environments.

We show state-of-the-art view synthesis results on both established benchmarks [4, 15] and our proposed benchmark.

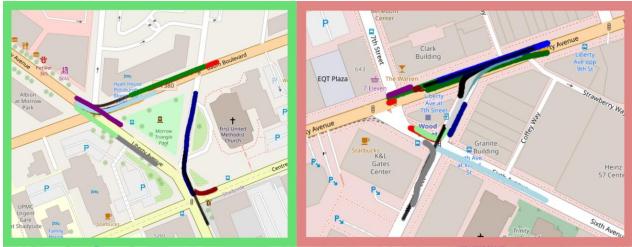
## 2. Related Work

**3D and 4D scene representations.** Finding the right scene representation is a core issue in 3D computer vision and graphics [5]. Over the years, a wide variety of options have been explored [1, 3, 9, 16, 26, 31, 42, 48–50], and, more recently, neural rendering [34] has been used to enable a new generation of scene models that support photo-realistic novel view synthesis. Scene representations for neural rendering can be roughly classified as “implicit” [31, 32, 51], which store most of the information in the weights of a neural network, or “explicit”, which use low-level spatial primitives such as voxels [11, 24, 33, 54], points [68], surfels [14], meshes [45], or 3D gaussians [17].

Similarly, different 4D dynamic scene representations for view synthesis have been investigated, including both implicit [12, 20, 39, 43, 58, 61, 65, 66] and explicit [10, 28, 38, 70] approaches. Dynamics are generally modeled as deformations of a canonical volume [10, 39, 43, 58, 65], a separate scene motion function [12, 20, 21, 61, 66], or rigid transformations of local geometric primitives [28].

Another line of work investigates the decomposition of scenes into higher-level entities [19, 27, 48, 59]. To express the composition of different entities into a complex scene, classical computer graphics literature [8] uses scene graphs. In particular, entities are represented as nodes in a hierarchical graph and are connected through edges defined by coordinate frame transformations. Thus, global transformations can be acquired by traversing the graph from its root node. For indoor 3D mapping, this concept was proposed by Armeni *et al.* [1] to represent static scenes at multiple levels of hierarchy, *i.e.* buildings, rooms and objects, and was later extended to dynamic scenes by Rosinol *et al.* [46]. Recently, Ost *et al.* [36] have revisited this concept for view synthesis, describing multi-object scenes with a graph that represents the dynamic actors in the scene. However, their representation lacks the ability of [1, 46] to represent scenes at multiple levels of hierarchy and is thus inherently limited to single, short video clips. On the contrary, we present a scalable, multi-level scene graph representation that spans large geographic areas with hundreds of dynamic objects.

**Representing large-scale urban scenes.** Compared to controlled captures of small scenes, in-the-wild captures of large-scale scenes pose distinct challenges. Limited viewpoint coverage, inaccurate camera poses, far-away buildings and sky, fast-moving objects, complex lighting, and auto exposure make radiance field reconstruction challenging. Therefore, previous works aid the reconstruction by using depth priors from *e.g.* LiDAR, refining camera parameters, adding information about camera exposure, and using specialized sky and light modeling components [23, 30, 44, 47, 55, 63, 67]. While many of these works simply remove dynamic actors, some methods explicitly model scene dynamics [19, 36, 61, 70]. However,



**Figure 2. Ego-vehicle trajectories of our benchmark.** We show the residential (left) and the downtown (right) areas, trajectories superimposed on 2D maps obtained from OpenStreetMap [35].

these methods either struggle to scale to more than a single sequence [19, 36, 70] or to accurately represent and distinguish dynamic actors [61]. To address these issues, we present a decomposed, scalable scene representation for dynamic urban environments.

**Efficient scene rendering.** Especially for large-scale scenes, the efficiency of a scene representation in training and inference is crucial. To alleviate the burden of ray traversal in volumetric rendering, many techniques for efficient sampling [2, 11, 33, 60] have been proposed. Recently, researchers have exploited more efficient forms of rendering, *e.g.* rasterization of meshes [23, 25] or other 3D primitives [17]. While these approaches are focused on static scenes, we extend the approach of [2] to our scene graph representation and thus to dynamic scenes.

### 3. Data

We investigate how to synthesize novel views of a dynamic urban environment from a set of captures taken from moving vehicles. Specifically, we are interested in captures spanning the same geographic area under varying conditions. This data presents unique challenges due to limited viewpoint coverage, different lighting, weather, and season, and even varying geometry due to distinct dynamic and transient objects.

Previous benchmarks in this area [19, 36] are limited to short, single video clips with simple ego trajectories and a few dynamic objects. Therefore, we present a benchmark that better reflects the aforementioned challenges. We base our benchmark on Argoverse 2 [64] which provides a rich set of captures from a fleet of vehicles deployed in multiple US cities that span different weather, season, and time of day. The vehicles are equipped with a surround-view camera rig with seven global shutter cameras, a LiDAR sensor, and a GPS. Furthermore, the timings of the different sensors are provided, so that one can relate the LiDAR-based 3D bounding box annotations to camera timestamps.

We leverage the GPS information to globally align the sequences and to identify regions that we are interested in mapping. We then extract the specific vehicle captures.



**Figure 3. Sequence alignment visualization.** The initial GPS-based alignment is imprecise, as evidenced by the duplicated structures in the overlaid LiDAR point clouds (left). After our ICP alignment, the area is well reconstructed (middle) according to its real geometry (right, from Argoverse 2 [64]).

Since GPS-based localization accuracy is only coarsely precise in urban areas, we align the captures via a global, offline iterative-closest-point (ICP) procedure applied to the LiDAR point clouds of all sequences to achieve satisfactory alignment for novel view synthesis purposes (see Fig. 3).

Following this procedure, we build a benchmark that enables real-world evaluation of novel view synthesis from diverse vehicle captures. It is composed of 37 vehicle captures split into two geographic regions as illustrated in Fig. 2. The regions cover a residential and a downtown area to resemble the different characteristics of urban environments. The residential area spans 14 captures with 10493 training and 1162 testing images with more than 700 distinct moving objects. The downtown area spans 23 captures with 16933 training and 1876 testing images with more than 600 distinct moving objects.

### 4. Method

**Problem setup.** We are given a set of sequences  $S$  captured from moving vehicles in different conditions. Each sequence  $s \in S$  consists of a set of images taken from cameras  $C_s$  mounted on the vehicle at different timesteps indexed by  $T_s$ . We assume the vehicle’s sensors are calibrated with respect to a common vehicle and a global world frame. In particular, we assume given sensor extrinsic  $\mathbf{T}_c = [\mathbf{R}_c | \mathbf{t}_c] \in \mathbf{SE}(3)$  for each camera  $c \in C = \bigcup_s C_s$ , and ego-vehicle poses  $\mathbf{P}_s^t = [\mathbf{R}_s^t | \mathbf{t}_s^t] \in \mathbf{SE}(3)$  for each timestamp  $t \in T_s$  and sequence  $s \in S$ . Furthermore, we assume all cameras to have known intrinsics  $\mathbf{K}_c$ . Each sequence  $s$  entails a set  $O_s$  of dynamic objects. For each object  $o \in O_s$ , we assume to have an estimated 3D bounding box track  $\mathcal{T}_o$  consisting of object poses  $\{\xi_o^{t_0}, \dots, \xi_o^{t_n}\} \subset \mathbf{SE}(3)$  w.r.t. the ego-vehicle frame, where  $t_i \in T_s$ , and the 3D object dimensions  $\mathbf{s}_o \in \mathbb{R}_{>0}^3$ . The images span a common geographic area for which we would like to estimate a radiance field

$$f_\theta(\mathbf{x}, \mathbf{d}, t, s) = (\sigma(\mathbf{x}, t, s), \mathbf{c}(\mathbf{x}, \mathbf{d}, t, s)) \quad (1)$$

that outputs volume density  $\sigma \in \mathbb{R}_{\geq 0}$  and color  $\mathbf{c} \in [0, 1]^3$  conditioned on 3D location  $\mathbf{x}$ , viewing direction  $\mathbf{d}$ , time  $t$  and sequence  $s$ .

## 4.1. Multi-Level Neural Scene Graph

**Overview.** We illustrate our representation in Fig. 1. We decompose the scene into a graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  where the set of nodes  $\mathcal{V}$  is composed by a root node  $v_r$  that defines the global coordinate system, *camera* nodes  $\{v_c\}_{c \in C}$ , and, for each sequence  $s \in S$ , *sequence* nodes  $\{v_s^t\}_{t \in T_s}$  and dynamic *object* nodes  $\{v_o\}_{o \in O_s}$ . The nodes are connected through oriented edges  $e \in \mathcal{E}$  that represent rigid transformations between the coordinate frames of the nodes, consistently with the edge direction. In addition, each node  $v \in \mathcal{V}$  can have an associated latent vector  $\omega$ . Given the scene graph  $\mathcal{G}$ , we model  $f_\theta$  using two radiance fields, namely  $\phi$  for largely static and  $\psi$  for highly dynamic scene parts. We use the latent vectors  $\omega$  to condition the radiance fields. In particular, we use latent vectors  $\omega_s^t$  of a sequence node  $v_s^t$  to condition  $\phi$ :

$$\phi(\mathbf{x}, \mathbf{d}, \omega_s^t) = (\sigma_\phi(\mathbf{x}, \omega_s^t), \mathbf{c}_\phi(\mathbf{x}, \mathbf{d}, \omega_s^t)). \quad (2)$$

Since the graph  $\mathcal{G}$  has multiple levels, a node associated with a dynamic object will naturally fall into the sequence  $s$  it appears in. Therefore, the radiance field  $\psi$  is conditioned on latent vectors  $\omega_s^t$  and  $\omega_o$ :

$$\psi(\mathbf{x}, \mathbf{d}, \omega_s^t, \omega_o) = (\sigma_\psi(\mathbf{x}, \omega_s^t, \omega_o), \mathbf{c}_\psi(\mathbf{x}, \mathbf{d}, \omega_s^t, \omega_o)) \quad (3)$$

of the corresponding sequence and object nodes  $v_s^t$  and  $v_o$ . **Appearance and geometry variation.** Reconstructing an environment from multiple captures is challenging from two perspectives: in addition to varying dynamic objects, there is i) varying appearance across captures, and ii) slow-moving or static *transient* geometry such as tree leaves or construction sites. Since both transient geometry and appearance can vary across captures, but are usually smooth *within* a sequence, we model these phenomena as smooth functions over time

$$\omega_s^t = [\mathbf{A}_s \mathcal{F}(t), \mathbf{G}_s \mathcal{F}(t)] \quad (4)$$

where  $\mathbf{A}_s$  is an appearance matrix,  $\mathbf{G}_s$  is a transient geometry matrix and  $\mathcal{F}(\cdot)$  is a 1D basis function of sines and cosines with linearly increasing frequencies at log-scale [61, 69]. We normalize time  $t$  into the interval  $[-1, 1]$  using the maximum sequence length  $\max_{s \in S} |T_s|$ . Crucially, this allows to model near-static, but sequence-specific regions of the input, as well as appearance changes due to *e.g.* auto-exposure. The degree of variation across time can be controlled by the number of frequencies.

**Sequence nodes.** The sequence nodes  $v_s^t$  are connected to the root node  $v_r$  with an edge  $e_{v_s^t v_r} = \mathbf{P}_s^t$ , *i.e.* the sequences  $S$  share a common global world frame. Each sequence node holds the latent vector  $\omega_s^t$  that conditions the radiance field  $\phi$ . We model  $\phi$  with a multi-scale 3D hash grid representa-

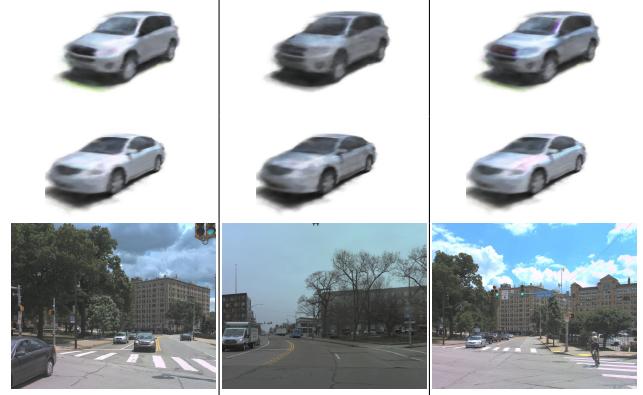


Figure 4. **Modifying car appearance with scene appearance.** We exchange  $\omega_s^t$  for different car instances. The car’s appearance in a rendered, car-centric view (top) changes according to the environmental conditions visible in the sequence  $s$  (bottom).

tion [33] and lightweight MLP heads:

$$\mathbf{f}_x = H_3(x) \quad (5)$$

$$\mathbf{h}_{\sigma_\phi}, \sigma_\phi = \text{MLP}_{\sigma_\phi}(\mathbf{f}_x) \quad (6)$$

$$\mathbf{c}_\phi = \text{MLP}_{\mathbf{c}_\phi}(\mathbf{h}_{\sigma_\phi}, \gamma_{\text{SH}}(\mathbf{d}), \mathbf{A}_s \mathcal{F}(t)) \quad (7)$$

$$\sigma_G, \mathbf{c}_G = \text{MLP}_G(\mathbf{h}_{\sigma_\phi}, \mathbf{G}_s \mathcal{F}(t)) \quad (8)$$

where  $\gamma_{\text{SH}}(\cdot)$  is a spherical harmonics encoding [33]. The final colors and densities are computed as a mixture of static and transient output (analogous to Eq. 12, see supp. mat.).

**Dynamic nodes.** The dynamic nodes  $v_o$  are connected to the sequence nodes  $v_s^t$  with edges  $e_{v_o v_s^t} = \xi_o^t$ . The dynamic node  $v_o$  associated with object  $o$  holds a latent vector  $\omega_o$  that conditions the radiance field  $\psi$ . We model  $\psi$  following [32] with an MLP conditioned on  $\omega_o$  to represent different instances with the same network [36, 70]

$$\mathbf{h}_{\sigma_\psi}, \sigma_\psi = \text{MLP}_{\sigma_\psi}(\gamma_{\text{PE}}(\mathbf{x}), \omega_o) \quad (9)$$

$$\mathbf{c}_\psi = \text{MLP}_{\mathbf{c}_\psi}(\mathbf{h}_{\sigma_\psi}, \gamma_{\text{PE}}(\mathbf{d}), \omega_o, \omega_s^t) \quad (10)$$

where  $\gamma_{\text{PE}}(\cdot)$  is a positional encoding [32]. Note that 3D position  $\mathbf{x}$  and viewing direction  $\mathbf{d}$  are transformed into the local object coordinate frame with  $(\mathbf{P}_s^t \xi_o^t)^{-1} \mathbf{I}_3(1/\max(s_o))$ . We condition  $\psi$  on both the scene and object-dependent latent vectors. This allows us to disentangle scene-dependent appearance from the object texture and thus to transfer objects across sequences. We illustrate this process in Fig. 4.

**Camera nodes.** The cameras  $c \in C_s$  are connected to the sequence nodes  $v_s^t$  through edges  $e_{v_c v_s^t} = \mathbf{T}_c$ , *i.e.* the calibration of camera  $c$  w.r.t. the ego-vehicle frame. This way, we can tie camera poses to a specific ego-vehicle pose.

## 4.2. Scene Graph Rendering

We describe how we render our scene graph  $\mathcal{G}$  for a given set of rays  $\mathcal{R}$ . The sampling locations along a ray  $(\mathbf{r}, t, s) \in \mathcal{R}$  at time  $t$  in sequence  $s$  are defined as  $\mathbf{r}(u) = \mathbf{o} + u\mathbf{d}$  with  $\mathbf{o} = \mathbf{R}_s^t \mathbf{t}_c + \mathbf{t}_s^t$  and  $\mathbf{d} = \mathbf{R}_s^t \mathbf{R}_c \mathbf{K}_c^{-1} (p_x, p_y, 1)^T$ .

**Continuous-time pose.** In order to realistically render videos at different frame rates, we treat the dynamic object poses  $\{\xi_o^{t_0}, \dots, \xi_o^{t_n}\}$  as a continuous function of time  $\xi_o(t)$ . We compute  $\xi_o(t)$  by interpolating between the two nearest poses at  $t_a \leq t < t_b$  to time  $t$ . This allows us also to synchronize estimated object poses originating from the LiDAR measurements with the camera timestamps.

**Ray-node intersection.** To render the dynamic nodes, we measure the intersection of their 3D bounding boxes with each  $(\mathbf{r}, t, s) \in \mathcal{R}$ . In particular, given the sequence  $s$  and time  $t$  the ray  $\mathbf{r}$  is associated with, we first traverse the graph  $\mathcal{G}$  to retrieve all relevant nodes  $v_o$  and their 3D bounding boxes  $\mathbf{b}_o^t = [\xi_o(t), \mathbf{s}_o]$  at time  $t$ . Then, we transform  $\mathbf{r}$  into each local node coordinate system and subsequently use AABB-ray intersection [29] to compute the entry and exit locations  $u_o^{\text{in}}, u_o^{\text{out}}$  along ray  $\mathbf{r}$ .

**Composite rendering.** To render the color  $\hat{\mathbf{C}}$  of ray  $\mathbf{r}$  in sequence  $s$  at time  $t$ , we use volumetric rendering [32]

$$\hat{\mathbf{C}}(\mathbf{r}, t, s) = \int_{u_n}^{u_f} U(u) \sigma(\mathbf{r}(u), t, s) \mathbf{c}(\mathbf{r}(u), \mathbf{d}, t, s) du$$

where  $U(u) = \exp\left(-\int_{u_n}^u \sigma(u') du'\right).$  (11)

We obtain density  $\sigma$  and color  $\mathbf{c}$  at sampling location  $\mathbf{r}(u)$  as mixture of the radiance fields  $\phi$  and  $\psi$

$$\sigma = \sigma_\phi + \sigma_\psi, \quad \mathbf{c} = \frac{\sigma_\phi}{\sigma_\phi + \sigma_\psi} \mathbf{c}_\phi + \frac{\sigma_\psi}{\sigma_\phi + \sigma_\psi} \mathbf{c}_\psi. \quad (12)$$

Crucially, we set  $\sigma_\psi$  to zero when  $\mathbf{r}(u)$  does not lie within a 3D bounding box of a dynamic node given the calculated entry and exit points  $u^{\text{in}}, u^{\text{out}}$ .

**Composite ray sampling.** Instead of densely sampling the space [19, 44] or leveraging separate ray sampling mechanisms per node [36], we use a composite ray sampling strategy illustrated in Fig. 5. We extend the proposal sampling mechanism introduced in [2] by joint sampling from a computationally efficient density field  $\sigma_{\text{prop}}$  and dynamic nodes  $v_o$  at time  $t$ . In particular, as in Eq. 12, we represent  $\sigma(\mathbf{r}(u), t, s)$  by a mixture of  $\sigma_{\text{prop}}(\mathbf{r}(u), \omega_s^t)$  and  $\sigma_\psi(\mathbf{r}(u), \omega_s^t, \omega_o)$ . However, analogous to our rendering step, we constrain sampling from  $\sigma_\psi$  to  $[u^{\text{in}}, u^{\text{out}}]$ . This allows us to skip empty space efficiently by distilling the static density  $\sigma_\phi$  into  $\sigma_{\text{prop}}$ . At the same time, sparsely querying  $\sigma_\psi$  when  $\mathbf{r}(u)$  falls into a dynamic node enables us to still accurately represent the full, dynamic  $\sigma$ .

Since there are only few samples that fall into dynamic nodes when performing uniform sampling, the computational overhead in the first ray sampling iteration is negligible. In the second iteration, we apply inverse transform sampling given the CDF  $F(u) = 1 - U(u)$  along ray  $\mathbf{r}$  and thus the samples are concentrated at the first surface intersection. Hence, only for rays that fall in the line of sight of

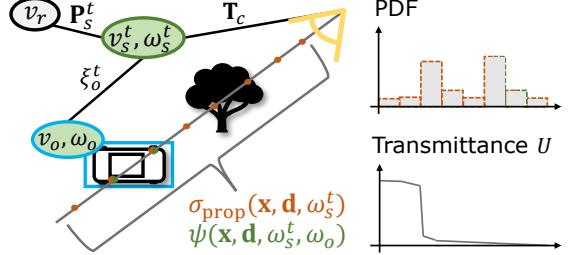


Figure 5. **Composite ray sampling.** If a ray intersects with an object  $v_o$ , we sample from both proposal network  $\sigma_{\text{prop}}$  and radiance field  $\psi$ , and  $\sigma_{\text{prop}}$  otherwise. We condition each with the latents  $\omega$  of the respective nodes. The PDF is a mixture of all node densities that intersect with the ray. The transmittance  $U$  drops at the first surface intersection (tree) where further samples will concentrate.

an object surface will we sample  $\sigma_\psi$  more than a few times. In total, we use two ray sampling iterations as in [56].

**Space contraction.** Following [2, 56], we contract the unbounded scene space into a fixed-size bounding box, normalizing the scene with bounds computed from the LiDAR point clouds and the ego-vehicle poses  $\mathbf{P}$  into a unit cube.

## 5. Optimization

We optimize the parameters  $\theta$  of the radiance field  $f_\theta$  with

$$\begin{aligned} \mathcal{L} = \sum_{(\mathbf{r}, t, s) \in \mathcal{R}} & \mathcal{L}_{\text{rgb}}(\mathbf{r}, t, s) + \lambda_{\text{dist}} \mathcal{L}_{\text{dist}}(\mathbf{r}, t, s) + \lambda_{\text{prop}} \\ & \mathcal{L}_{\text{prop}}(\mathbf{r}, t, s) + \lambda_{\text{dep}} \mathcal{L}_{\text{dep}}(\mathbf{r}, t, s) + \lambda_{\text{entr}} \mathcal{L}_{\text{entr}}(\mathbf{r}, t, s) \end{aligned} \quad (13)$$

where  $\mathcal{L}_{\text{dist}}$  and  $\mathcal{L}_{\text{prop}}$  follow [2]. We supervise  $\sigma_{\text{prop}}$  with  $\sigma_\phi$  only to learn effective composite ray sampling.

**Photometric loss.** We compare the rendered training rays with their ground truth color

$$\mathcal{L}_{\text{rgb}}(\mathbf{r}, t, s) = \|\mathbf{C}(\mathbf{r}, t, s) - \hat{\mathbf{C}}(\mathbf{r}, t, s)\|_2^2. \quad (14)$$

**Expected depth loss.** We render the expected depth values for each ray and compare it with the ground truth depth

$$\mathcal{L}_{\text{dep}}(\mathbf{r}, t, s) = \|\mathbf{D}(\mathbf{r}, t, s) - \hat{\mathbf{D}}(\mathbf{r}, t, s)\|_2^2 \quad (15)$$

where the expected depth is calculated via integrating the sampling values  $\hat{\mathbf{D}}(\mathbf{r}, t, s) = \int_{u_n}^{u_f} u U(u) \sigma(\mathbf{r}(u), t, s) du$ .

**Entropy regularization.** While static and dynamic scene parts can overlap, i.e. inside a 3D bounding box  $\mathbf{b}_o^t$  there could be a part of the street or sidewalk, their density should be strictly separated w.r.t. a single sampling location  $\mathbf{r}(u)$ . We leverage an entropy regularization loss [65] to encourage clear separation between entities in  $\phi$  and  $\psi$

$$\mathcal{L}_{\text{entr}}(\mathbf{r}, t, s) = \int_{u_n}^{u_f} \mathcal{H}\left(\frac{\sigma_\psi(\mathbf{r}(u), t, s)}{\sigma_\phi(\mathbf{r}(u), t, s) + \sigma_\psi(\mathbf{r}(u), t, s)}\right) du \quad (16)$$

where  $\mathcal{H}(\cdot)$  is the Shannon entropy and we use  $t, s$  as a replacement for the vectors  $\omega_s^t, \omega_o$  for ease of notation.

Method	Residential			Downtown			Mean			Train time (h)
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	
Nerfacto + Emb.	19.83	0.637	0.562	18.05	0.655	0.625	18.94	0.646	0.594	8.0
Nerfacto + Emb. + Time	20.05	0.641	0.562	18.66	0.656	0.603	19.36	0.654	0.583	13.2
SUDS [61]	21.76	0.659	0.556	19.91	0.665	0.645	20.84	0.662	0.601	54.8
<b>Ours</b>	<b>22.29</b>	<b>0.678</b>	<b>0.523</b>	<b>20.01</b>	<b>0.681</b>	<b>0.586</b>	<b>21.15</b>	<b>0.680</b>	<b>0.555</b>	17.2

Table 1. **Novel View Synthesis on Argoverse 2.** While the static Nerfacto baseline has the weakest performance, the dynamic variant Nerfacto + Time improves only marginally upon it. The state-of-the-art method SUDS exhibits stronger view synthesis results but takes more than  $3\times$  longer to train. Our method outperforms all methods across all metrics and exhibits competitive training speed.

Method	KITTI [75%]			KITTI [50%]			KITTI [25%]			
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	
NeRF [32]	18.56	0.557	0.554	19.12	0.587	0.497	18.61	0.570	0.510	
NeRF + Time	21.01	0.612	0.492	21.34	0.635	0.448	19.55	0.586	0.505	
NSG [36]	21.53	0.673	0.254	21.26	0.659	0.266	20.00	0.632	0.281	
Nerfacto + Emb.	22.75	0.801	0.156	22.38	0.793	0.160	21.24	0.758	0.178	
Nerfacto + Emb. + Time	23.19	0.804	0.155	23.18	0.803	0.155	21.98	0.777	0.172	
SUDS [61]	22.77	0.797	0.171	23.12	0.821	0.135	20.76	0.747	0.198	
<b>Ours</b>	<b>28.38</b>	<b>0.907</b>	<b>0.052</b>	<b>27.51</b>	<b>0.898</b>	<b>0.055</b>	<b>26.51</b>	<b>0.887</b>	<b>0.060</b>	

Method	VKITTI2 [75%]			VKITTI2 [50%]			VKITTI2 [25%]			
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	
NeRF [32]	18.67	0.548	0.634	18.58	0.544	0.635	18.17	0.537	0.644	
NeRF + Time	19.03	0.574	0.587	18.90	0.565	0.610	18.04	0.545	0.626	
NSG [36]	23.41	0.689	0.317	23.23	0.679	0.325	21.29	0.666	0.317	
Nerfacto + Emb.	22.15	0.847	0.145	21.88	0.843	0.148	21.28	0.827	0.155	
Nerfacto + Emb. + Time	22.11	0.849	0.144	21.78	0.844	0.147	21.00	0.825	0.157	
SUDS [61]	23.87	0.846	0.150	23.78	0.851	0.142	22.18	0.829	0.160	
<b>Ours</b>	<b>29.73</b>	<b>0.912</b>	<b>0.065</b>	<b>29.19</b>	<b>0.906</b>	<b>0.066</b>	<b>28.29</b>	<b>0.901</b>	<b>0.067</b>	

Table 2. **Novel View Synthesis on KITTI and VKITTI2.** We compare our method to prior art, following the experimental protocol in [61]. We test the view synthesis performance of the methods with varying fractions of training views and observe that fewer training views generally result in lower performance. Our method outperforms previous works by a large margin on all settings.

**Hierarchical pose optimization.** Alongside scene geometry, we optimize edges  $e_{v_s^t v_r}$  and  $e_{v_o v_s^t}$  in our graph, *i.e.* we refine ego-vehicle poses  $\mathbf{P}_s^t \in \text{SE}(3)$  and object poses  $\xi_o^t \in \text{SE}(3)$ . In particular, we optimize for pose residuals  $\delta\mathbf{P}_s^t \in \mathfrak{se}(3)$  and  $\delta\xi_o^t \in \mathfrak{se}(2)$ . We constrain the object pose residual to  $\mathfrak{se}(2)$  since objects are usually upright and move along the ground plane. Given each residual, we update the ego-vehicle pose with  $\hat{\mathbf{P}}_s^t = \text{expmap}(\delta\mathbf{P}_s^t)\mathbf{P}_s^t$  and the object pose with  $\hat{\xi}_o^t = \text{expmap}(\delta\xi_o^t)_{\text{SE}(2) \rightarrow \text{SE}(3)}\xi_o^t$ , where  $\text{expmap}(\cdot)$  is the exponential map of each lie group.

Compared to naively optimizing camera and object poses, optimizing the edges along our scene graph has two key advantages. First, we leverage multi-camera constraints, *i.e.* we keep the camera extrinsics  $\mathbf{T}_c$  fixed and optimize the ego-vehicle poses  $\mathbf{P}_s^t$  only. This is in contrast to prior art that generally treats each camera pose as independent. Second, the residual  $\delta\mathbf{P}_s^t$  propagates to the object poses since they are defined w.r.t. the ego-vehicle coordinate frame instead of the global world frame. Given that optimizing a radiance field as well as camera and object poses jointly is notoriously difficult [22], these constraints are vital to view synthesis quality (see Tab. 6).

## 6. Experiments

**Datasets.** To evaluate against competing methods on our proposed benchmark on Argoverse 2 [64], we hold out every 10th sample in uniform time intervals where a sample

corresponds to seven ring-camera images. To compare our methods against prior art on KITTI [15] and VKITTI2 [4], we follow the experimental protocol and data splits in [61]. We use the provided 3D bounding box annotations for all datasets. We provide results using an off-the-shelf 3D tracker [71] in the supplemental material.

**Metrics.** Following [61], we measure image synthesis quality with PSNR, SSIM [62], and LPIPS (AlexNet) [72].

**Implementation details.** We implement our method in PyTorch [41], accelerating the time-consuming ray-node intersection with a custom CUDA implementation. We train our model on a single RTX 3090 GPU for 250,000 steps on Argoverse 2 and 100,000 steps on KITTI and VKITTI2, with 8192 rays per batch. All model parameters are optimized using Adam [18] with  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ , and an exponential decay learning rate schedule: from  $10^{-5}$  to  $10^{-6}$  for pose parameters, and from  $10^{-3}$  to  $10^{-4}$  for all others. In order to counter pose drift, we further apply weight decay with a factor  $10^{-2}$  to  $\delta\mathbf{P}$  and  $\delta\xi$ . The static radiance field  $\phi$  is trained from scratch, while  $\psi$  is initialized from a semantic prior [7] following [19].

**Baselines.** We compare our method against prior work in dynamic outdoor scene representation, *i.e.* SUDS [61] and NSG [36]. Further, we include results obtained by augmenting Nerfacto [56], the closest state-of-the-art NeRF architecture to ours, with components designed to handle multi-capture reconstruction and scene dynamics. In particular,

	SRN [51]	NeRF [32]	NeRF + Time	NSG [36]	PNF [19]	SUDS [61]	<b>Ours</b>
PSNR $\uparrow$	18.83	23.34	24.18	26.66	27.48	28.31	<b>29.36</b>
SSIM $\uparrow$	0.590	0.662	0.677	0.806	0.870	0.876	<b>0.911</b>

Table 3. **Image reconstruction on KITTI.** We outperform prior art in image reconstruction, *i.e.* our method can better fit the training views. We follow the experimental protocol in [19, 36, 61].



Figure 6. **Qualitative results on KITTI.** With only 25% of views (approx. 15-20) for training, we can still synthesize sharp and realistic novel views with dynamic objects rendered at high quality.

we consider two variants: “Nerfacto + Emb.”, where we incorporate our appearance embedding  $\mathbf{A}_s\mathcal{F}(t)$  and the expected depth loss; and “Nerfacto + Emb. + Time”, where we further include time modeling with 4D hash tables following [40, 66]. For both Nerfacto variants, we set hash table and MLP sizes to be aligned with our method.

## 6.1. Comparison to state-of-the-art

We first compare to prior work on our proposed benchmark in Tab 1. We observe that the static Nerfacto + Emb. has the weakest performance, which slightly increases by adding time modeling in Nerfacto + Emb. + Time. Meanwhile, the current state-of-the-art method SUDS [61] outperforms the Nerfacto variants, while being much slower to train. Our method performs the best on all metrics. The improvement is particularly pronounced in the perceptual quality metrics (SSIM and LPIPS). The training speed of our method is competitive to Nerfacto + Emb. + Time and more than 3 $\times$  faster than SUDS.

In addition, we show a qualitative comparison in Fig. 7. We observe major differences in both the rendered images and depth maps. In particular, all other methods struggle to recover the dynamic objects in the scene, while our method produces realistic renderings and accurate depth maps for both static and dynamic areas. We also observe that, thanks to our transient geometry embedding  $\mathbf{G}_s\mathcal{F}(t)$ , we are able to accurately recover the geometry of the trees and their leaves which notably are not present in every sequence of the area that is reconstructed due to seasonal changes. At the same time, other methods struggle to recover those details. Specifically, other methods exhibit artifacts in the depth maps and degraded rendering quality of the trees left and right of the street in columns five and six of Fig. 7, while our method produces accurate color and depth renderings.

Next, we evaluate our method on established benchmarks, namely KITTI and VKITTI2, in Tab. 2 following the experimental protocol in [61]. In particular, we compare novel view synthesis quality at different fractions of

$\mathbf{A}_s\mathcal{F}(t)$	$\mathbf{G}_s\mathcal{F}(t)$		PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
-	-		19.70	0.653	0.588
✓	-		22.22	0.670	0.546
✓	✓		<b>22.49</b>	<b>0.671</b>	<b>0.541</b>

Table 4. **Ablation study on graph structure.** We show that using the multi-level graph structure of sequences and objects is crucial, *i.e.* omitting sequence vectors  $\omega_s^t$  results in degraded quality since there is no conditioning on scene-specific appearance. Combining appearance and transient geometry embeddings performs best.

Sampling	Samples per ray	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	Rays / sec.
Uniform [19]	192	25.72	0.730	0.456	28K
Uniform [19]	1024	25.84	0.734	0.449	4K
Separate [36]	1024+64+(32+64)	26.65	<b>0.762</b>	<b>0.351</b>	2.5K
<b>Ours</b>	1024+64	<b>27.07</b>	0.759	0.362	30K

Table 5. **Ray Sampling schemes.** Our composite ray sampling is about 12 $\times$  more efficient to train than separate ray sampling [36] with comparable performance. Uniform sampling [19] exhibits lower performance and is slow when sampled more densely.

Pose optimization	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
-	<b>22.49</b>	0.671	0.541
Naive	21.28	0.663	<b>0.519</b>
Hierarchical (Ours)	22.29	<b>0.678</b>	0.523

Table 6. **Pose optimization.** Compared to naively optimizing camera and object poses, optimizing vehicle and object poses hierarchically on the edges of our graph benefits view synthesis quality more consistently, *i.e.* both in SSIM and LPIPS. Note that PSNR is very sensitive to pose drift, and thus does not improve.

training views, *i.e.* testing different levels of view supervision. We run our method and our baselines and observe substantially better view synthesis quality for our method, both compared to our baselines and prior works. All methods exhibit a similar performance drop when the fraction of training views decreases. We illustrate the rendering quality of our method with only 15-20 training views in Fig. 6. Finally, we also report the results on the task of image reconstruction, *i.e.* reconstruction of images seen during training, following [19, 36, 61] in Tab. 3 on the KITTI dataset. We significantly outperform previous works in both metrics.

## 6.2. Ablation studies

We verify our design through ablation studies. We perform these on the residential area of our benchmark unless otherwise noted. First, we ablate on the multi-level structure of our scene graph. In particular, in Tab. 4, we ablate the latent codes of the sequence nodes, reducing to a representation similar to [36] and observe that the quality of the synthesized views drops significantly. In contrast, our scene graph representation achieves better view synthesis quality, *i.e.* achieves almost three points higher PSNR. Further, adding our transient geometry embeddings in addition to sequence-level appearance embeddings improves the view synthesis quality. See supplemental material for a qualitative comparison. This adds to the fact that through our multi-level graph structure, we show the ability to modulate car appear-

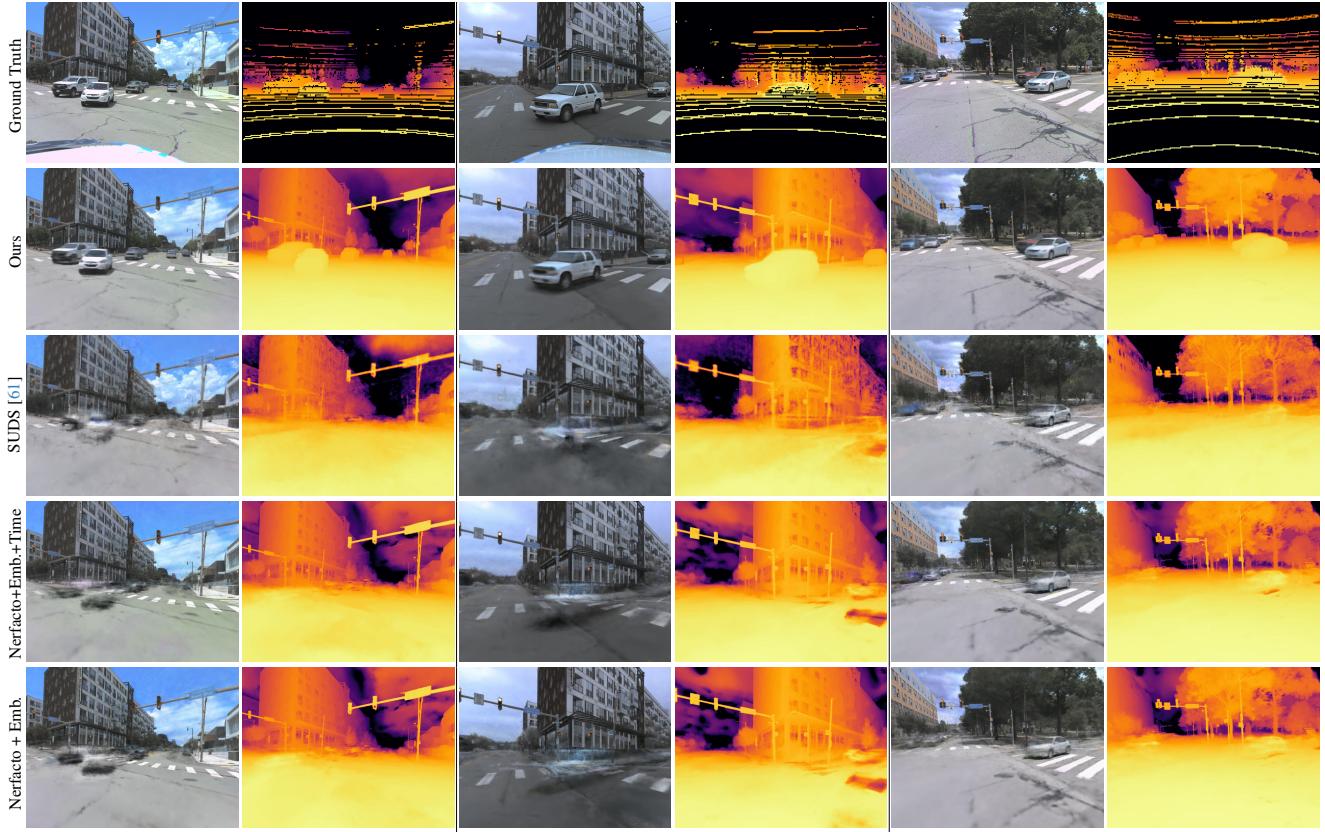


Figure 7. **Qualitative results on Argoverse 2.** While prior art struggles with dynamic actors in the scene, our representation renders both realistic novel views and plausible depth maps. Further, our method models complex transient geometry subject to, e.g., seasonal changes such as tree leaves. Note that prior art produces depth artifacts and exhibits degraded view quality in those regions (columns 5 and 6).

ance through sequence appearance in Fig. 4.

In Tab. 5, we compare our composite ray sampling scheme to uniformly sampled rays as in [19] and rays sampled separately per node as in [36]. We compare the schemes under the assumption that all other model parameters are equal, implementing them in our method. Ours is about  $12\times$  more efficient to train than densely sampling a ray in uniform intervals or separately sampling the ray per node while producing similar results to the latter and being significantly better than the former. Sparsely sampling a ray at uniform intervals is similarly fast as our method, but yields degraded view synthesis quality. Note that we run this ablation study on only a single sequence of the residential area in our benchmark since densely sampling the rays and separately sampling the rays for all nodes are prohibitively expensive to train in large-scale urban areas.

Finally, in Tab. 6, we compare our hierarchical pose optimization to naive camera pose optimization employed in previous works [22, 56]. While naive camera pose optimization degrades the results significantly in pixel-wise metrics, our hierarchical pose optimization improves the SSIM and maintain a comparable PSNR. Meanwhile, our hierarchical pose optimization exhibits similar gains to

naive pose optimization in terms of the perceptual LPIPS metric. This shows that our pose optimization mitigates pose drift while also enabling a more accurate reconstruction. Pose drift usually causes a misalignment between the evaluation viewpoint and scene geometry, which degrades pixel-wise metrics in particular.

## 7. Conclusion

We introduce a novel multi-level scene graph representation for radiance field reconstruction in dynamic urban environments that scales to large geographic areas with more than ten thousand images from dozens of sequences and hundreds of dynamic objects. We train our representation with an efficient composite ray sampling and rendering scheme and introduce latent variables that enable modeling complex phenomena like varying environmental conditions and transient geometry present across different vehicle captures. We leverage our representation to refine camera and object poses hierarchically using multi-camera constraints. Finally, we propose a new view synthesis benchmark for dynamic urban driving scenarios. Our approach yields substantially improved results compared to prior art while allowing for flexible de- and recomposition of the scene.

## References

- [1] Iro Armeni, Zhi-Yang He, Jun Young Gwak, Amir R Zamir, Martin Fischer, Jitendra Malik, and Silvio Savarese. 3d scene graph: A structure for unified semantics, 3d space, and camera. In *ICCV*, 2019. 2
- [2] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, 2022. 1, 3, 5, 12
- [3] Jules Bloomenthal and Brian Wyvill. Introduction to implicit surfaces. 1997. 2
- [4] Yohann Cabon, Naila Murray, and Martin Humenberger. Virtual kitti 2, 2020. 2, 6
- [5] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6):1309–1332, 2016. 2
- [6] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giacomo Baldan, and Oscar Beijbom. nuscenes: A multi-modal dataset for autonomous driving. In *CVPR*, 2020. 14
- [7] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 6
- [8] Steve Cunningham and Michael J Bailey. Lessons from scene graphs: using scene graphs to teach hierarchical modeling. *Computers & Graphics*, 25(4):703–711, 2001. 2
- [9] Nathaniel Fairfield, George A. Kantor, and David S. Wettergreen. Real-time slam with octree evidence grids for exploration in underwater tunnels. *Journal of Field Robotics*, 24, 2007. 2
- [10] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. In *SIGGRAPH Asia 2022 Conference Papers*, 2022. 2
- [11] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022. 1, 2, 3
- [12] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *CVPR*, 2021. 2
- [13] Hang Gao, Ruilong Li, Shubham Tulsiani, Bryan Russell, and Angjoo Kanazawa. Monocular dynamic view synthesis: A reality check. *NeurIPS*, 2022. 2
- [14] Yiming Gao, Yan-Pei Cao, and Ying Shan. Surfelnerf: Neural surfel radiance fields for online photorealistic reconstruction of indoor scenes. In *CVPR*, 2023. 1, 2
- [15] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 2, 6
- [16] Michael Kaess. Simultaneous localization and mapping with infinite planes. *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4605–4611, 2015. 2
- [17] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (ToG)*, 42(4):1–14, 2023. 1, 2, 3
- [18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6
- [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 *CVPR*, 2022. 2, 3, 5, 6, 7, 8
- [20] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *CVPR*, 2021. 2
- [21] Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely. Dynibar: Neural dynamic image-based rendering. In *CVPR*, 2023. 2
- [22] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *ICCV*, 2021. 6, 8, 13
- [23] Jeffrey Yunfan Liu, Yun Chen, Ze Yang, Jingkang Wang, Sivabalan Manivasagam, and Raquel Urtasun. Real-time neural rasterization for large scenes. In *ICCV*, 2023. 2, 3
- [24] Lingjie Liu, Jitao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *NeurIPS*, 2020. 1, 2
- [25] Fan Lu, Yan Xu, Guang Chen, Hongsheng Li, Kwan-Yee Lin, and Changjun Jiang. Urban radiance field representation with deformable neural mesh primitives. In *ICCV*, 2023. 2, 3
- [26] Yan Lu and Dezhen Song. Visual navigation using heterogeneous landmarks and unsupervised geometric constraints. *IEEE Transactions on Robotics*, 31:736–749, 2015. 2
- [27] Jonathon Luiten, Tobias Fischer, and Bastian Leibe. Track to reconstruct and reconstruct to track. *IEEE Robotics and Automation Letters*, 5(2):1803–1810, 2020. 2
- [28] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. *arXiv preprint arXiv:2308.09713*, 2023. 2
- [29] Alexander Majercik, Cyril Crassin, Peter Shirley, and Morgan McGuire. A ray-box intersection algorithm and efficient dynamic voxel rendering. *Journal of Computer Graphics Techniques Vol*, 7(3):66–81, 2018. 5
- [30] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *CVPR*, 2021. 2
- [31] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019. 1, 2
- [32] 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, 4, 5, 6, 7
- [33] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022. 1, 2, 3, 4
- [34] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *CVPR*, 2020. 1, 2
- [35] OpenStreetMap contributors. Planet dump retrieved from <https://planet.osm.org>. <https://www.openstreetmap.org>, 2017. 3
- [36] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes. In *CVPR*, 2021. 2, 3, 4, 5, 6, 7, 8, 14
- [37] Ziqi Pang, Zhichao Li, and Naiyan Wang. Simpletrack: Understanding and rethinking 3d multi-object tracking. *arXiv preprint arXiv:2111.09621*, 2021. 14
- [38] Byeongjun Park and Changick Kim. Point-dynrf: Point-based dynamic radiance fields from a monocular video, 2023. 2
- [39] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *arXiv preprint arXiv:2106.13228*, 2021. 2
- [40] Sungheon Park, Minjung Son, Seokhwan Jang, Young Chun Ahn, Ji-Yeon Kim, and Nahyup Kang. Temporal interpolation is all you need for dynamic neural radiance fields. In *CVPR*, 2023. 2, 7
- [41] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS*, 2019. 6
- [42] Marc Pollefeys, David Nistér, Jan-Michael Frahm, Amir Abarzadeh, Philippos Mordohai, Brian Clipp, Chris Engels, David Gallup, Seon Joo Kim, Paul C. Merrell, C. Salmi, Sudipta N. Sinha, B. Talton, Liang Wang, Qingxiong Yang, Henrik Stewénius, Ruigang Yang, Greg Welch, and Herman Towles. Detailed real-time urban 3d reconstruction from video. *IJCV*, 78:143–167, 2007. 2
- [43] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *CVPR*, 2021. 2
- [44] Konstantinos Rematas, Andrew Liu, Pratul P Srinivasan, Jonathan T Barron, Andrea Tagliasacchi, Thomas Funkhouser, and Vittorio Ferrari. Urban radiance fields. In *CVPR*, 2022. 2, 5
- [45] Gernot Riegler and Vladlen Koltun. Stable view synthesis. In *CVPR*, 2021. 1, 2
- [46] Antoni Rosinol, Arjun Gupta, Marcus Abate, Jingnan Shi, and Luca Carlone. 3d dynamic scene graphs: Actionable spatial perception with places, objects, and humans. *arXiv preprint arXiv:2002.06289*, 2020. 2
- [47] Viktor Rudnev, Mohamed Elgharib, William Smith, Lingjie Liu, Vladislav Golyanik, and Christian Theobalt. Nerf for outdoor scene relighting. In *ECCV*, 2022. 2
- [48] Renato F Salas-Moreno, Richard A Newcombe, Hauke Strasdat, Paul HJ Kelly, and Andrew J Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In *CVPR*, 2013. 2
- [49] Aron Schmied, Tobias Fischer, Martin Danelljan, Marc Pollefeys, and Fisher Yu. R3d3: Dense 3d reconstruction of dynamic scenes from multiple cameras. In *ICCV*, 2023.
- [50] Chen Shen, James F. O’Brien, and Jonathan R. Shewchuk. Interpolating and approximating implicit surfaces from polygon soup. In *ACM SIGGRAPH 2004 Papers*, page 896–904, 2004. 2
- [51] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *NeurIPS*, 2019. 1, 2, 7
- [52] Liangchen Song, Anpei Chen, Zhong Li, Zhang Chen, Lele Chen, Junsong Yuan, Yi Xu, and Andreas Geiger. Nerf-player: A streamable dynamic scene representation with decomposed neural radiance fields. *IEEE Transactions on Visualization and Computer Graphics*, 29(5):2732–2742, 2023. 2
- [53] Henry Sowizral. Scene graphs in the new millennium. *IEEE Computer Graphics and Applications*, 20(1):56–57, 2000. 2
- [54] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022. 2
- [55] 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 *CVPR*, 2022. 2
- [56] 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. 5, 6, 8, 12, 13
- [57] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020. 13
- [58] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *ICCV*, 2021. 2
- [59] Shubham Tulsiani, Saurabh Gupta, David F Fouhey, Alexei A Efros, and Jitendra Malik. Factoring shape, pose, and layout from the 2d image of a 3d scene. In *CVPR*, 2018. 2
- [60] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs. In *CVPR*, 2022. 2, 3
- [61] Haithem Turki, Jason Y Zhang, Francesco Ferroni, and Deva Ramanan. Suds: Scalable urban dynamic scenes. In *CVPR*, 2023. 2, 3, 4, 6, 7, 8, 13, 14, 15

- [62] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 6
- [63] Zian Wang, Tianchang Shen, Jun Gao, Shengyu Huang, Jacob Munkberg, Jon Hasselgren, Zan Gojcic, Wenzheng Chen, and Sanja Fidler. Neural fields meet explicit geometric representations for inverse rendering of urban scenes. In *CVPR*, 2023. 2
- [64] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemeyer Pontes, et al. Argoverse 2: Next generation datasets for self-driving perception and forecasting. *arXiv preprint arXiv:2301.00493*, 2023. 2, 3, 6, 12, 14
- [65] Tianhao Wu, Fangcheng Zhong, Andrea Tagliasacchi, Forrester Cole, and Cengiz Oztireli. D<sup>^</sup>2nerf: Self-supervised decoupling of dynamic and static objects from a monocular video. *NeurIPS*, 2022. 2, 5
- [66] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *CVPR*, 2021. 2, 7
- [67] 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
- [68] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *CVPR*, 2022. 2
- [69] Gengshan Yang, Minh Vo, Natalia Neverova, Deva Ramanan, Andrea Vedaldi, and Hanbyul Joo. Banmo: Building animatable 3d neural models from many casual videos. In *CVPR*, 2022. 4
- [70] 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 *CVPR*, 2023. 2, 3, 4, 14
- [71] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *CVPR*, 2021. 6, 14
- [72] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 6

## Appendix

This supplementary material provides details on our method, our experimental setup, and more quantitative and qualitative results and comparisons. In Sec. A, we provide further details on our benchmark data. In Sec. B, we provide further details on our method. In Sec. C, we provide details on our experimental setup, conduct additional experiments, and show more qualitative comparisons.

### A. Data Details

We describe further details on our benchmark data taken from [64]. The LiDAR is sampled at 10Hz, and the 3D bounding box annotations are annotated with the LiDAR, *i.e.* they are also provided at 10Hz. The cameras are synchronized with the LiDAR which yields seven images at 10Hz for each sequence. Each camera has a resolution of  $1550 \times 2048$  pixels, where all cameras besides the front camera are oriented in landscape mode. Each sequence in Argoverse 2 [64] is approximately 15 seconds long.

Since the original data contains regions where the ego-vehicle is visible in some of the cameras (cf. Fig. 7 of the main paper), we annotate each camera view with an ego-vehicle mask which we use in all experiments for all methods to constrain the ray sampling process. We release the full data splits and the sequence alignment transformations with our source code.

### B. Method Details

In this section, we provide more details on our method.

**Appearance and transient geometry embeddings.** To condition our sequence-level appearance and transient geometry matrices  $\mathbf{A}_s$  and  $\mathbf{G}_s$  on the time  $t$ , we use the 1D basis function  $\mathcal{F}(t)$  as mentioned in Sec. 4 of the main paper. We use six as the number of frequencies of  $\mathcal{F}(t)$  for both appearance and transient geometry embeddings. The resulting vectors  $\omega_s^t$  and  $\omega_o$  are in  $\mathbb{R}^{64}$ . For  $\omega_s^t$ , we learn  $\mathbf{A}$  and  $\mathbf{G}$  per sequence both in  $\mathbb{R}^{32 \times 6 \cdot 2+1}$ , *i.e.* desired latent vector size by output dimension of  $\mathcal{F}(t)$ . For  $\omega_o$ , we learn separate geometry and appearance codes per object both in  $\mathbb{R}^{32}$ .

**Transient density  $\sigma_G$  and color  $c_G$ .** In Eq. 8 of the paper, we define the output of the transient geometry branch which is used to calculate the final static color. We blend the transient color  $c_G$  with the predicted color  $c_\phi$  in Eq. 7 weighted by the densities  $\sigma_\phi$  and  $\sigma_G$  analogous to Eq. 12:

$$\sigma_\phi = \sigma_\phi + \sigma_G, \quad c_\phi = \frac{\sigma_\phi}{\sigma_\phi + \sigma_G} c_\phi + \frac{\sigma_G}{\sigma_\phi + \sigma_G} c_G. \quad (17)$$

**Proposal network  $\sigma_{\text{prop}}$ .** We align with [56] and use two separate proposal networks, one for each proposal sampling

iteration. These proposal networks and our final static radiance field  $\phi$  have increasing hash table sizes, acting as a coarse-to-fine representation of the scene geometry. In contrast to previous works [2, 56], we condition the proposal networks on the sequence-specific geometry codes to account for varying transient geometry across sequences in the proposal sampling stage.

**Dynamic object radiance field  $\psi$ .** For our dynamic object radiance field  $\psi$ , we use separate shape and appearance latent vectors that condition the radiance field. In particular, we use a shape code at the network input that we concatenate with the input coordinate  $\mathbf{x}$  and further an appearance code that we concatenate with the direction  $\mathbf{d}$  at the bottleneck after density prediction. We concatenate sequence and object appearance latent vectors to propagate sequence appearance to the individual objects.

**Space contraction.** As mentioned in Sec. 4 of the main paper, we follow [2, 56] and contract the unbounded scene space into a unit cube. In particular, we use the following function for space contraction:

$$\chi(\mathbf{x}) = \begin{cases} \mathbf{x}, & \|\mathbf{x}\|_\infty \leq 1 \\ (2 - \frac{1}{\|\mathbf{x}\|_\infty}) \frac{\mathbf{x}}{\|\mathbf{x}\|_\infty}, & \|\mathbf{x}\|_\infty > 1 \end{cases}.$$

**Limitations.** While our method sets a new state-of-the-art for radiance field reconstruction in dynamic urban environments under varying environmental conditions, the extremely challenging nature of the problem persists and further research in this area is needed. For example, we can much better represent highly dynamic, rigid objects such as cars, vans, trucks, and buses. Still, objects with highly intricate motions such as pedestrians or cyclists continue to be a challenge. Another limitation stems from the inherent problem of insufficient view coverage. For areas that were not clearly visible to the ego-car, we find that the rendering quality is significantly lower. This is particularly pronounced for dynamic objects since they are only present in a single sequence. However, we note that this problem is attenuated by the initialization of radiance field  $\psi$  with a semantic prior. Overall, views farther away from the training trajectories would constitute an interesting addition to the evaluation setup. Yet, utilizing (partial) hold-out sequences is not suitable for our task as these would contain distinct transient geometry and dynamic objects, and possibly appearance unknown to the model. Thus, a different capturing setup would be required which is outside the scope of our work but is an interesting area for future research.

Finally, while our method improves over naive pose optimization, we note that this is a challenging problem and that large pose errors are hard to correct during reconstruction. We thus tackled this issue by pre-aligning the sequences with our offline ICP procedure. We hope that our proposed benchmark can spark further research that addresses these issues.

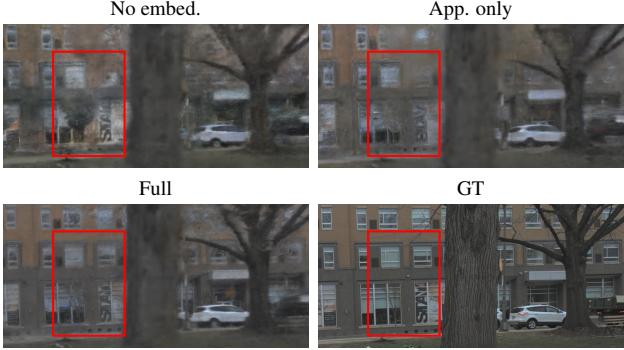


Figure 8. **Qualitative comparison of graph structure.** We show a qualitative illustration of our ablation study in Tab. 4. In particular, we show the results of our method without any sequence-dependent latent vectors  $\omega_s^t$ , with only the appearance vectors  $\mathbf{A}_s\mathcal{F}(t)$  and of our full method.

Split	3D Box Type	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
Single Seq.	GT	27.07	0.759	0.362
	Prediction	26.71	0.756	0.365
Residential	GT	22.29	0.678	0.523
	Prediction	21.28	0.667	0.538

Table 7. **Ablation on 3D bounding boxes.** We show results on a single sequence of the residential area in our benchmark and the full residential area. We train our method with either the provided 3D bounding box annotations or predictions acquired from an off-the-shelf 3D tracker.



Figure 9. **Failure case of predicted 3D bounding boxes.** While the car in the foreground is well reconstructed with both ground truth and predicted 3D bounding boxes, the van in the background is rendered with incorrect orientation and is slightly too big.

## C. Additional Experiments

We discuss further details on our experimental setup, additional experiments, and qualitative results.

**Implementation details.** We compute the scene bounds from the LiDAR point cloud with a maximum distance of 80 meters per sweep, *i.e.* we filter each point cloud so that only points less than 80 meters from the ego-vehicle remain, and use the ego-vehicle poses to register all point clouds in the global world frame. With this global, world-frame point cloud we compute the scene bounds. We use the following loss weights for Eq. 13 of the main paper:  $\lambda_{\text{dist}} = 0.002$ ,  $\lambda_{\text{prop}} = 1.0$ ,  $\lambda_{\text{dep}} = 0.05$ , and  $\lambda_{\text{entr}} = 0.0001$ . For  $\mathcal{L}_{\text{dep}}$ ,

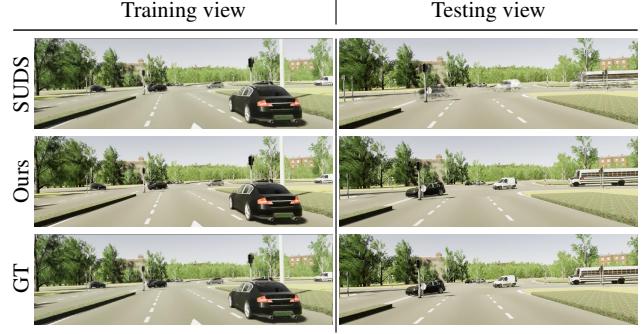


Figure 10. **Qualitative comparison on VKITTI2.** We observe that SUDS [61] can render realistic training views, but cannot properly recover the dynamic actors in the testing views. In contrast, our method shows no quality difference in rendering training and novel testing views.

we use the LiDAR measurements as ground truth. We only take depth measurements at the time of the camera sensor recording to ensure dynamic objects receive valid depth supervision. Following previous works [22, 56], we optimize the evaluation camera poses during the validation phase to compensate for pose errors introduced by drifting geometry through optimized training poses that would otherwise contaminate the view synthesis quality measurement. For this, we use  $\mathcal{L}_{\text{rgb}}$  only. For the depth map visualizations shown in our qualitative results, we use linear scaling with a maximum depth of 82.5 meters and a minimum depth of 1 meter. Rendering a  $1920 \times 1080$  image takes  $\sim 16.4$  seconds. The training speed is 30K rays per second.

**Baselines.** We run SUDS [61] on our benchmark using their official code release. Since it requires several additional inputs such as LiDAR depth and optical flow predictions, we compute the optical flow of all sequences with RAFT [57] following the experimental setup of SUDS [61] on their City-1M benchmark. We align all other auxiliary inputs such as depth with our method. We deactivate the DINO feature distillation branch that is used in SUDS for semantic reconstruction.

**Influence of graph structure on view quality.** In Fig. 8, we show a qualitative comparison of our method without the node latent codes in our graph, *i.e.*  $\mathbf{A}_s\mathcal{F}(t)$  and  $\mathbf{G}_s\mathcal{F}(t)$ , our method with only appearance codes  $\mathbf{A}_s\mathcal{F}(t)$ , and our full method.

We observe that without any conditioning on the sequence  $s$ , there are strong artifacts, *e.g.* the smaller tree highlighted in red is being rendered with leaves and the wall of the building behind it has a significantly different color than in the ground truth image. With the appearance embedding only, the color problem is alleviated, but the texture of the wall is highly distorted since there is no way for the model to distinguish between sequences where the tree leaves are present and the wall is occluded and where

the wall is visible. In contrast, our full method recovers a faithful rendering of the tree, the wall and also the windows above.

**3D bounding box predictions.** While we follow previous works [36, 70] and use provided 3D bounding boxes in our experiments, we also report results using off-the-shelf algorithms to predict the 3D bounding boxes of dynamic objects. In particular, we use an off-the-shelf LiDAR-based 3D object tracker [37, 71] to generate 3D bounding box tracks. We use those tracks instead of the provided 3D bounding box annotations. Note that neither the 3D object detector nor the tracking algorithm is adjusted or fine-tuned for the Argoverse 2 [64] dataset. We take the officially provided models trained on the nuScenes dataset [6]. This dataset notably has different LiDAR sensor properties than Argoverse 2.

In Tab. 7, we compare the results of our method with annotated 3D bounding boxes and with the predicted bounding boxes. We train our method both on a single sequence, *i.e.* the same sequence as in Tab. 5 of the main paper, and the full residential area in our benchmark. When trained on a single sequence, we observe that the difference in view quality is marginal. Trained on the full residential split of our benchmark, the gap becomes slightly larger while the performance is still competitive. We analyze this more closely in Fig. 9, where we observe some failure cases when predicted boxes are inaccurate, *e.g.* when the orientation of an object is not correctly predicted and can also not be recovered through our pose optimization. Still, the synthesized views look realistic. Overall, this shows that our approach can be scaled to large vehicle fleet data without the need for manual data annotation simply through employing off-the-shelf LiDAR 3D tracking algorithms without much loss in realism.

**Analysis of KITTI and VKITTI2 results.** We observe a large gap between previous state-of-the-art methods and our method in terms of novel view synthesis quality in Tab. 2. At the same time, our image reconstruction quality is superior, but the gap is significantly smaller (cf. Tab. 3). Motivated by this observation, we retrain SUDS [61] on the VKITTI2 dataset and visualize its renderings for example training and testing views alongside the results of our method in Fig. 10. We observe that indeed the reconstruction quality of the training views is comparable for SUDS and our method, but SUDS fails to recover dynamic objects in the testing view properly, while our method produces high-quality renderings also for novel views. This shows that our scene graph-based, high-level decomposition excels at representing scenes with highly dynamic objects while previous work struggles with this.

**Ablation of  $\mathcal{L}_{\text{dep}}$ ,  $\mathcal{L}_{\text{entr}}$ .** In Tab. 8, we observe that while depth and entropy losses have a limited effect on evaluation view quality, the depth loss helps in improving geometry

$\mathcal{L}_{\text{rgb}}$	$\mathcal{L}_{\text{dep}}$	$\mathcal{L}_{\text{entr}}$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	AbsRel $\downarrow$
✓	-	-	22.22	0.675	0.524	0.321
✓	✓	-	22.23	0.677	<b>0.523</b>	0.219
✓	-	✓	22.20	0.676	<b>0.523</b>	0.333
✓	✓	✓	<b>22.29</b>	<b>0.678</b>	<b>0.523</b>	<b>0.218</b>

Table 8. **Loss term ablation.** We report both the view and depth quality of our model when ablating different loss terms.



Figure 11. **Free viewpoint renderings.** Without  $\mathcal{L}_{\text{entr}}$ , the separation between dynamic and static content is ambiguous (red). Without  $\mathcal{L}_{\text{dep}}$ , the traffic pole exhibits artifacts (green).



Figure 12. **Object renderings.** We illustrate object instances in sunny conditions (top) and cloudy conditions (bottom).

accuracy (AbsRel). Intuitively, this improves view quality farther from the training trajectory. The entropy loss encourages static and dynamic radiance separation, improving object renderings and scene decomposition. We illustrate these effects in Fig. 11.

**Additional qualitative results of object-centric renderings.** In Fig. 12, we show additional object-centric renderings conditioned on different scene appearances. In particular, we depict objects with more intricate textures, showing the ability of  $\psi$  to generalize to a wide variety of object instances. Note that the instance reconstruction quality varies with the observed training views (cf. Fig. 12 right). Yet, we note that our method models objects much better than existing works.

**Additional qualitative comparison.** We include further qualitative results of our method compared to the state-of-the-art in Fig. 13. We observe that, similar to Fig. 7 of the main paper, our method exhibits superior view synthesis of dynamic areas and better captures seasonal variations in terms of, for example, tree leaves. Further, the synthesized views of our method are sharper compared to prior art, and the depth maps are less noisy. We include qualitative results from both the residential and downtown areas of our benchmark.

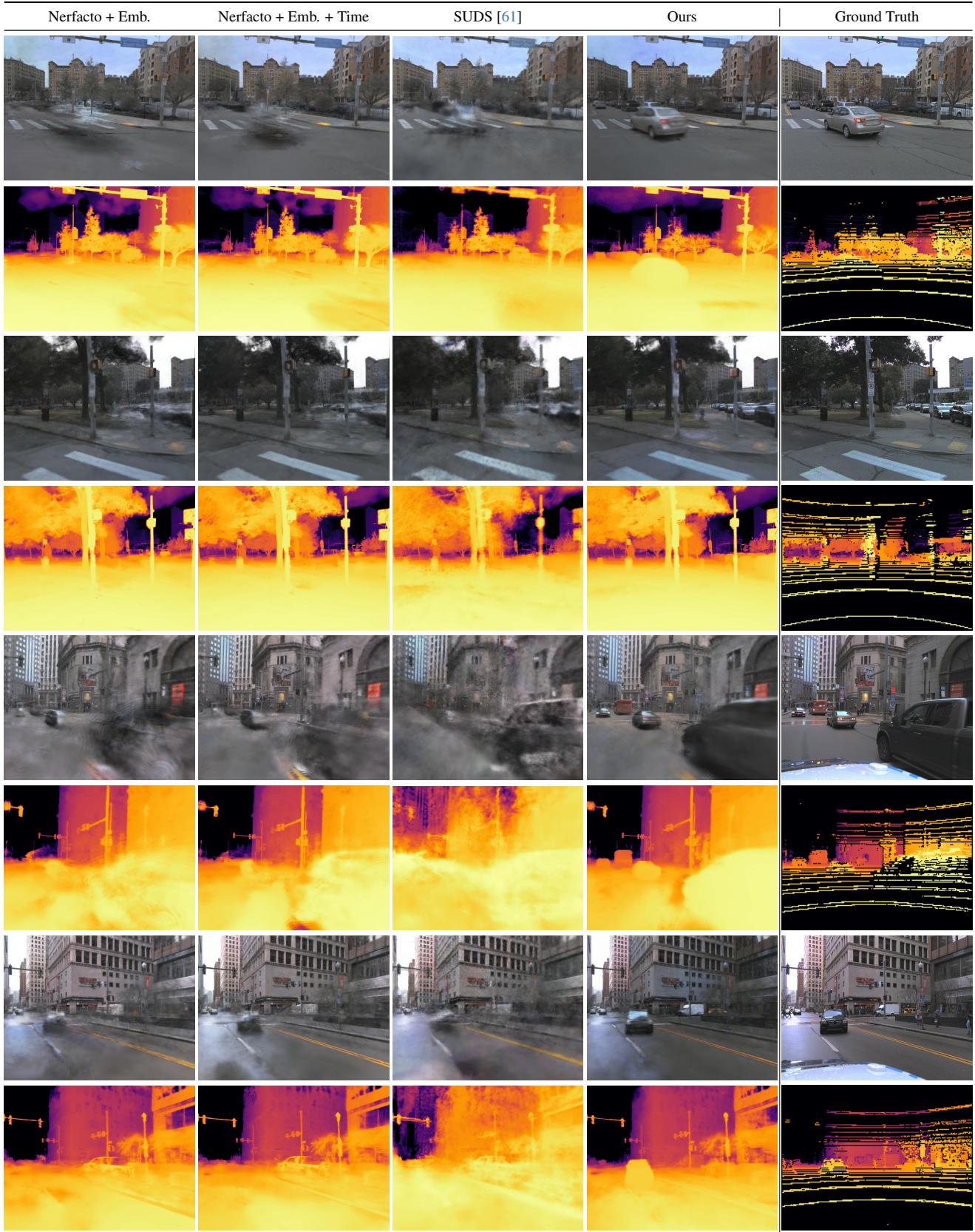


Figure 13. **Additional qualitative results on Argoverse 2.** We illustrate four examples, where the upper two are from the residential area and the lower two are from the downtown area in our benchmark. We observe that our method exhibits better view quality and cleaner depth maps, particularly in dynamic areas.