IDSplat: Instance-Decomposed 3D Gaussian Splatting for Driving Scenes

Carl Lindström[†,1,2] Mahan Rafidashti[†,1,2] Maryam Fatemi[1]
Lars Hammarstrand[2] Martin R. Oswald[3] Lennart Svensson[2]
[1]Zenseact [2]Chalmers University of Technology [3]University of Amsterdam
{firstname.lastname}@{zenseact.com, chalmers.se}

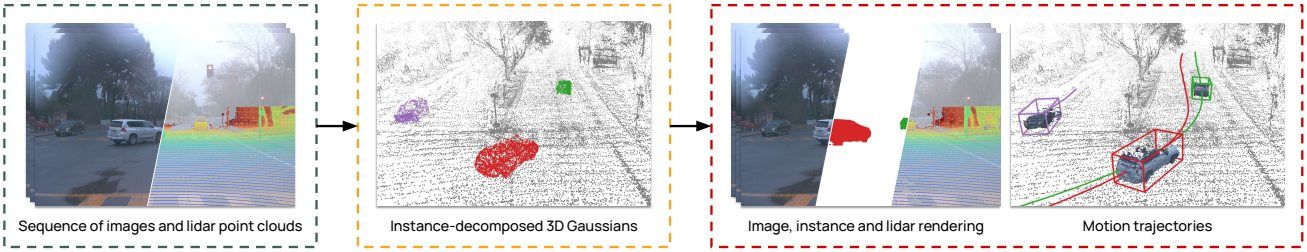

Figure 1. IDSplat performs self-supervised reconstruction of dynamic scenes with explicit instance-decomposition and learnable motion trajectories. IDSplat enables high-fidelity rendering of images, instances, and lidar point clouds without the need for human annotations.

Abstract

Reconstructing dynamic driving scenes is essential for developing autonomous systems through sensor-realistic simulation. Although recent methods achieve high-fidelity reconstructions, they either rely on costly human annotations for object trajectories or use time-varying representations without explicit object-level decomposition, leading to intertwined static and dynamic elements that hinder scene separation. We present IDSplat, a self-supervised 3D Gaussian Splatting framework that reconstructs dynamic scenes with explicit instance decomposition and learnable motion trajectories, without requiring human annotations. Our key insight is to model dynamic objects as coherent instances undergoing rigid transformations, rather than unstructured time-varying primitives. For instance decomposition, we employ zero-shot, language-grounded video tracking anchored to 3D using lidar, and estimate consistent poses via feature correspondences. We introduce a coordinated-turn smoothing scheme to obtain temporally and physically consistent motion trajectories, mitigating pose misalignments and tracking failures, followed by joint optimization of object poses and Gaussian parameters. Experiments on the Waymo Open Dataset demonstrate that our method achieves competitive reconstruction quality while maintaining instance-level decomposition and generalizes across diverse sequences and view densities without retraining, making it practical for large-scale autonomous driving applications. Code will be released.

[†]These authors contributed equally to this work.

1. Introduction

Reconstructing dynamic scenes has become an important cornerstone in the development of autonomous driving systems, enabling closed-loop training and testing through sensor-realistic renderings. In contrast to real-world testing, digital twins provide a scalable, low-cost, and safe means of exploring novel driving scenarios using already collected data [16, 17]. Recent work has improved the quality, efficiency, and sensor compatibility of such reconstructions [4, 8, 30, 31, 36, 39, 42, 48], but typically rely on human-annotated object trajectories and 3D bounding boxes, which are expensive and time-consuming to acquire at scale.

To address this challenge, several self-supervised approaches have emerged that aim to reconstruct dynamic scenes without human annotations [3, 9, 10, 20, 40, 46]. Although they produce high-quality renderings, they lack explicit instance decomposition, significantly limiting their practical use, since novel scenario generation requires manipulating individual dynamic objects.

In this paper, we address the problem of reconstructing realistic dynamic scenes without human annotations, while preserving instance-level decomposition and learning the underlying motion trajectories of individual objects. Reconstructing dynamic scenes using 3D Gaussian Splatting (3DGS) [11] presents a significant challenge, as it violates the assumption that each 3D point maintains a fixed position and appearance across viewpoints. While prior self-supervised approaches address this by introducing time-dependent Gaussian parameters [3, 9, 20], we instead preserve the geometry and appearance of coherent object in-

stances over time and optimize their rigid transformations to capture the true underlying motion. This formulation preserves instance-level decomposition and enables controllable trajectories, rather than relying on time-varying primitives whose changing visibility and appearance can lead to inconsistent scene representations.

Obtaining instance-level decomposition of 3D Gaussians without annotated poses introduces an additional challenge. Although 3D object trackers can perform well in estimating object poses and have been used effectively in dynamic scene reconstruction [30, 39], they rely on human annotations for fine-tuning on the target data and are constrained by the predefined taxonomy of those annotations. To overcome this limitation, we leverage recent advances in vision models and employ a language-grounded video tracker to extract instance masks in a zero-shot manner, allowing both generalization to new datasets and new classes without retraining or additional annotations. These masks are lifted to 3D using corresponding lidar point clouds, and object poses are estimated via RANSAC using DINOv3 [24] feature correspondences. To further address pose misalignments and missing detections, we introduce an iterative robust coordinated-turn smoothing scheme that discards outliers and refines the trajectories.

Although accurate initialization of object trajectories is crucial, inaccuracies are inevitable due to missing instance masks, inaccurate lidar poses, or tracking drift over time. To mitigate these errors, we make a final refinement of the object trajectories guided by reconstruction errors during the 3D Gaussian Splatting optimization.

We present IDSplat, a novel 3D Gaussian Splatting framework designed to handle dynamic scenes through instance-level decomposition and joint optimization of object appearance, geometry, and motion. We extensively evaluate the effectiveness of our method on Waymo Open Dataset [27], achieving state-of-the-art results across a diverse set of sequences and test protocols. In summary, our contributions are as follows:

- We propose a self-supervised framework for dynamic scene reconstruction that explicitly decomposes scenes into object instances with learnable motion trajectories, enabling joint rendering, segmentation and motion tracking.
- We introduce a zero-shot approach for 3D instance decomposition and pose estimation, enabling generalization to new datasets and object classes without retraining or human annotations.
- We present simple yet effective techniques for optimizing and refining motion trajectories, combining motion modeling and photometric consistency to obtain accurate trajectories even under sparse views.

2. Related work

Annotation-based rendering for autonomous driving: Neural radiance fields (NeRF) [19] inspired numerous NeRF-based methods for dynamic road scenes [4, 21, 30, 36, 42]. These achieve high-quality renderings and naturally extend to new sensors [21, 30], but are sample-intensive, resulting in slow rendering speeds and limiting their scalability.

3D Gaussian Splatting (3DGS) [11] provides an explicit rasterization-friendly representation, enabling orders-of-magnitude faster rendering. Automotive scene reconstruction with 3DGS has been explored in several works, including lidar-based extensions [2, 4, 8, 12, 39, 47, 48].

To model dynamic scenes, both NeRF- and 3DGS-based approaches typically rely on accurate 3D bounding boxes with temporal instance associations. These enable the scene to be decomposed into a static background and dynamic foreground components, with each dynamic object transformed according to its trajectory. To achieve high-fidelity reconstruction, state-of-the-art approaches typically rely on either human-annotated bounding boxes or predictions from high-performing 3D object trackers that have been carefully adapted to the target dataset. This reliance on curated annotations or dataset-specific trackers limits the scalability and zero-shot generalization of these methods to new datasets.

Self-supervised dynamic scene reconstruction: Self-supervised approaches also separate static and dynamic regions, using either separate hash-grids [31, 40], or time-varying 3DGS representations [3, 9, 18, 20, 26, 28, 34, 38, 41], where Gaussian attributes are allowed to change over time. Instead of bounding boxes, these methods rely on photometric and geometric consistency or foundation model outputs to guide decomposition, either via learned features or explicitly via predicted masks [9, 10, 20, 34, 40, 43, 46]. DeSiRe-GS [20] segments dynamic regions in images using features from FiT3D [44] and uses these dynamic masks to optimize time-varying Gaussians. Similarly, AD-GS [10] relies on Grounded-SAM-2 [23] masks, but uses B-splines and trigonometric functions to enforce smoother trajectories for individual Gaussians. Other works attempt more detailed decomposition, such as CoDa-4DGS [26] which enables semantic segmentation by learning per-Gaussian feature vectors. A key limitation of these models is that temporal changes are modeled at the primitive level, preventing decomposition into coherent object instances. This restricts practical applications such as novel scenario generation, auto-labeling, and simulation.

IDSplat addresses this by decomposing scenes into a static background and dynamic foreground instances that remain consistent across the sequence. Each instance's motion trajectory is explicitly represented, enabling reassignment, manipulation or removal. We achieve self-supervised instance decomposition using Grounded-SAM-2

for zero-shot masks and DINOv3 [24] for robust registration across time, and guide the refinement of motion trajectories through photometric and geometric consistency.

3. Method

Given a sequence of images and lidar point clouds from a driving scenario, our goal is to learn a 3D representation of the scene, including instance-decomposed dynamic objects with associated learnable motion trajectories. In the following, we describe our scene representation (Sec. 3.1), how the representation is decomposed into separate instances (Sec. 3.2), and how associated trajectories are estimated and refined (Secs. 3.3 and 3.4). Finally, we describe how the complete scene is optimized (Sec. 3.5). See Fig. 2 for an overview of our method.

3.1. Scene representation

We represent the scene by a set of translucent 3D Gaussians, parameterized with occupancy probability $o \in [0, 1]$, mean $\mu \in \mathbb{R}^3$, and covariance $\mathbf{\Sigma} \in \mathbb{R}^{3\times3}$. Together, the parameters describe the position, extent and visibility of the Gaussian. To facilitate both camera and lidar rendering, we follow SplatAD [8] and assign each Gaussian a feature vector $\mathbf{f}^{\text{rgb}} \in \mathbb{R}^3$ to represent its base color, and another feature vector $\mathbf{f} \in \mathbb{R}^{D_{\text{f}}}$ to represent view-dependent effects and lidar properties. Further, each Gaussian also has an associated discrete ID denoted by $z \in \{0, \ldots, N_{\text{ID}}\}$, determining which instance it belongs to, or whether it belongs to the static background ($z = 0$). We also adopt sensor-specific embeddings to model appearance shifts.

To account for scene dynamics, we parameterize the motion of Gaussians using SE(3) poses. We assume each instance corresponds to a rigid object and transform all Gaussians associated with a given ID using the same rigid transformation. The position of Gaussian i in the world at time t is given as

$$\mu_{i,t} = \mathbf{T}_{z_i,t}\mu_i \tag{1}$$

where $\mu_{i,t}$ denotes the Gaussian's position in world coordinates at time t, μ_i is its position in the canonical coordinate system of instance z_i, and $\mathbf{T}_{z_i,t} \in$ SE(3) represents the transformation from that canonical frame to the world at time t. Gaussians associated with the static background are defined directly in the world coordinate system and are not transformed.

3.2. Instance decomposition

To obtain object instances, we employ Grounded-SAM-2 [23] to generate instance masks from video frames using class prompts. As our scene is represented in 3D, we lift the 2D instance information to 3D by projecting the temporally closest lidar points onto the image plane and assigning them the corresponding instance IDs. To reduce outliers from inaccurate projections, arising from line-of-sight mismatches due to camera-lidar mounting offsets or large uncorrected object motions, we apply a two-stage filtering process. We first erode the instance masks prior to projection to reduce the influence from sensor mounting offsets, and then cluster the projected points using DBSCAN [6], retaining only the largest cluster for each instance.

3.3. Trajectory estimation

With the instance points identified, we estimate the trajectory of each object instance by registering the lidar points associated with that instance. For an instance z, we first define a canonical frame centered at the midpoint of the axis-aligned bounding box enclosing the points from the time step with the densest lidar observations. This defines the initial pose as

$$\mathbf{T}_{z,t_{\text{init}}(z)} = \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{c}_z \\ \mathbf{0}^T & 1 \end{bmatrix}, \tag{2}$$

where $t_{\text{init}}(z)$ denotes the time with the highest lidar density for instance z, and $\mathbf{c}_z$ is the center of the corresponding bounding box. Subsequent frames are registered to the canonical frame, $\mathbf{T}_{z,t_{\text{init}}(z)}$, ordered by point density. We extract DINOv3 [24] features from image projections and establish correspondences between frames based on cosine similarity. The rigid transformation is then estimated using RANSAC, where each hypothesis is computed via the Umeyama estimator [32] from three randomly sampled correspondences. The pose with the largest number of structural inliers is selected if the registration is deemed successful. We consider the registration successful when the inlier ratio of the target point cloud exceeds a predefined threshold, motivated by the fact that the target (from frame t_j) will always be smaller than the canonical source. When a registration succeeds, the resulting pose for time t_j is added to the trajectory and given by

$$\mathbf{T}_{z,t_j} = \mathbf{T}_{z,t_j \leftarrow t_{\text{init}}(z)}\mathbf{T}_{z,t_{\text{init}}(z)}, \tag{3}$$

where $\mathbf{T}_{z,t_j \leftarrow t_{\text{init}}(z)}$ is the rigid transformation estimated from RANSAC, and $\mathbf{T}_{z,t_j}$ represents the pose of instance z at time t_j. The corresponding points from frame t_j are then transformed into the canonical frame using the inverse of $\mathbf{T}_{z,t_j \leftarrow t_{\text{init}}(z)}$ and merged into the canonical point set.

3.4. Trajectory smoothing

While the RANSAC-based registration yields initial pose estimates between pairs of point clouds, our goal is to estimate a temporally and physically consistent object trajectory, and reduce the impact from imperfections due to registration errors or temporal gaps from missing instance masks. To address this, we refine the trajectories through an iterative coordinated-turn (CT) smoothing formulated as a pose graph optimization problem.

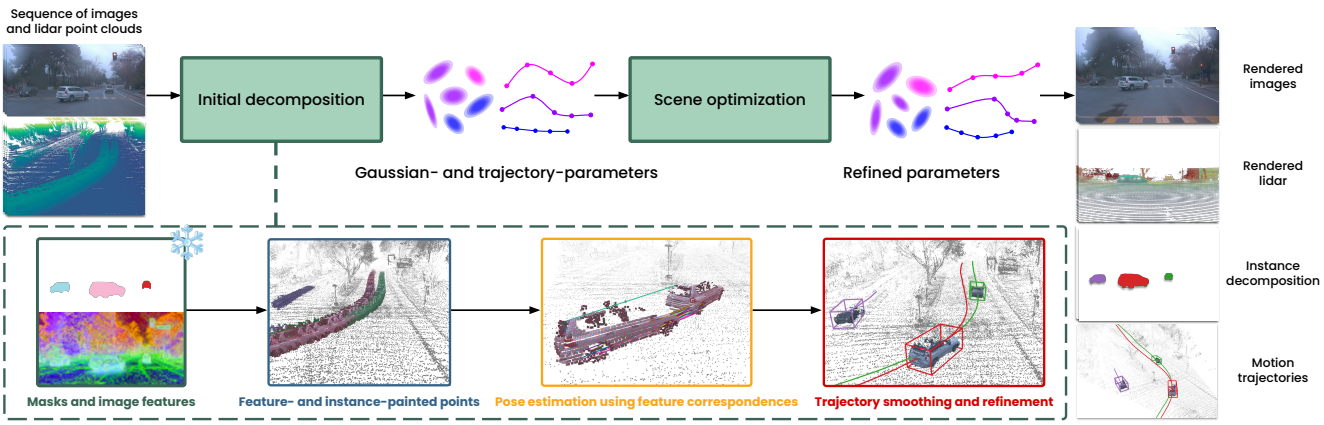

Figure 2. Overview of our method. 2D masks from Grounded-SAM-2 are lifted to 3D using corresponding lidar point clouds to initialize instances. Object poses are estimated via RANSAC using DINOv3 feature correspondences and further refined through iterative CT smoothing. Trajectories and Gaussian parameters are then optimized to render images, lidar, and instances with motion trajectories.

Optimization formulation: We employ GTSAM [5] to optimize a state vector comprising poses $\mathbf{T}_t \in \mathrm{SE}(3)$, translational speeds $v_t \in \mathbb{R}$, and curvatures $\kappa_t \in \mathbb{R}$ for all timesteps t in the sequence. The estimated poses from RANSAC are added as noisy measurement factors through Gaussian likelihoods with a Huber loss function. Since instance masks may be intermittent, measurements may only be available at a subset of timesteps. We incorporate CT motion model factors to encourage physically grounded, temporally smooth trajectory states and to reject measurement outliers. To align the axis-aligned measurements with the motion model, we also estimate a single rotation, shared across all times, that orients the local x-axis along the direction of motion. Smoothness priors on the trajectory is further added as random walk processes on speed and curvature, and additional priors are added to encourage small roll and pitch angles as well as moderate curvature values.

Outlier rejection: To address outliers, we apply iterative refinement of the trajectories by first performing a single optimization pass and then identifying and removing measurements whose residuals exceed a predefined threshold. The optimization is then re-run with the pruned measurement set, improving the robustness of trajectory estimates even in the presence of registration failures.

3.5. Scene optimization

Our scene representation consists of a set of static Gaussians and a set of dynamic Gaussians associated to instances with corresponding trajectories. All Gaussian parameters and trajectories are optimized jointly through self-supervised reconstruction of images and lidar point clouds. We adopt the rasterization proposed in [8] and optimize the entire model jointly using the reconstruction loss

$$\mathcal{L} = \lambda_r \mathcal{L}_1 + (1 - \lambda_r)\mathcal{L}_{\text{SSIM}} + \mathcal{L}_{\text{lidar}} + \lambda_{\text{MCMC}}\mathcal{L}_{\text{MCMC}}, \tag{4}$$

where $\mathcal{L}_1$ and $\mathcal{L}_{\text{SSIM}}$ are L1 and SSIM losses on the rendered images, and $\mathcal{L}_{\text{MCMC}}$ denotes the opacity and scale regularization used in [13]. $\mathcal{L}_{\text{lidar}}$ is the loss from lidar reconstruction and is defined as

$$\mathcal{L}_{\text{lidar}} = \lambda_{\text{depth}}\mathcal{L}_{\text{depth}} + \lambda_{\text{los}}\mathcal{L}_{\text{los}} + \lambda_{\text{inten}}\mathcal{L}_{\text{inten}} + \lambda_{\text{raydrop}}\mathcal{L}_{\text{BCE}}, \tag{5}$$

where $\mathcal{L}_{\text{depth}}$ and $\mathcal{L}_{\text{inten}}$ are L2 losses on the rendered expected lidar range and intensity, and $\mathcal{L}_{\text{los}}$ is a line-of-sight loss that penalize accumulated opacity before the ground truth lidar range. $\mathcal{L}_{\text{BCE}}$ is a binary cross-entropy loss on predicted ray drop probability. For ease of comparison, we adopt the hyperparameters from [8]. See Sec. A for details.

Dynamic Gaussians are initialized from the canonical point sets created during point registration (3.3), while static Gaussians are seeded from lidar points not associated with any instance. RGB values for both sets are assigned by projecting the corresponding lidar points into the temporally closest image. Additional Gaussians are sampled randomly within the lidar range, and sampled linearly in disparity beyond observed points. Following [8], we employ the MCMC densification strategy from [13].

4. Experiments

To thoroughly evaluate IDSplat, we benchmark its performance on novel view synthesis (NVS) and image reconstruction using Waymo Open Dataset [27]. We compare against state-of-the-art self-supervised methods adapted to automotive scenes under multiple evaluation protocols, spanning a range of view densities and dynamic object categories. To assess generalization, we conduct additional ex-

Table 1. NVS results on experimental settings of AD-GS and DeSiRe-GS. Results for both baselines are obtained using their official implementation. IDSplat outperforms the baselines for both settings. First , second , third .

		Anno. free	PSNR ↑	SSIM ↑	LPIPS ↓	DPSNR ↑
DeSiRe-GS setting	MARS	×	26.61	-	-	22.21
	SplatAD	×	30.80	0.900	0.160	28.97
	PVG	✓	29.77	-	-	27.19
	EmerNeRF	✓	25.14	-	-	23.49
	S3Gaussian	✓	27.44	-	-	22.92
	DeSiRe-GS	✓	28.76	0.873	0.193	26.26
	IDSplat (ours)	✓	30.83	0.900	0.160	29.20
AD-GS setting	StreetGS	×	33.97	0.926	0.227	28.50
	4DGS	×	34.64	0.940	0.244	29.77
	SplatAD	×	34.24	0.925	0.246	29.68
	SplatAD (CasTrack)	×	32.52	0.924	0.241	25.31
	PVG	✓	29.54	0.895	0.266	21.56
	EmerNeRF	✓	31.32	0.881	0.301	21.80
	Grid4D	✓	32.19	0.921	0.253	22.77
	AD-GS	✓	33.91	0.927	0.228	27.41
	IDSplat (ours)	✓	34.59	0.929	0.235	29.63
CoDa	CoDa-4DGS	✓	28.66	0.900	0.058	-
	IDSplat (ours)	✓	30.50	0.875	0.090	-
SF	SplatFlow	✓	28.71	0.874	0.239	-
	IDSplat (ours)	✓	29.95	0.879	0.183	-

Table 2. NVS results for lidar point cloud rendering. Our method obtains similar lidar rendering performance to the annotation-based SplatAD.

	Depth ↓	Intensity ↓	Drop acc. ↑	CD ↓
SplatAD	0.01	0.055	87.3	0.98
IDSplat (ours)	0.01	0.056	87.5	1.24

periments on PandaSet [37]. We further analyze the optimized trajectories using standard tracking metrics. Finally, we perform ablation studies of individual components to understand their impact on overall performance.

4.1. Implementation

IDSplat is implemented in `neurad-studio` [29], built upon the rasterization framework from SplatAD [7, 8]. For pre-processing, we use Grounded-SAM-2 [23] to generate instance masks and DINOv3 [24] for image features. We optimize IDSplat for 30,000 iterations, following the hyperparameter settings in [8] unless otherwise specified. All experiments were run on an NVIDIA A100 40GB GPU. See Sec. A for further details.

4.2. Dataset

Our experiments are conducted on three subsets of Waymo Open Dataset. One is the set of eight sequences used in StreetGS [39] and AD-GS [10], the second is the Waymo

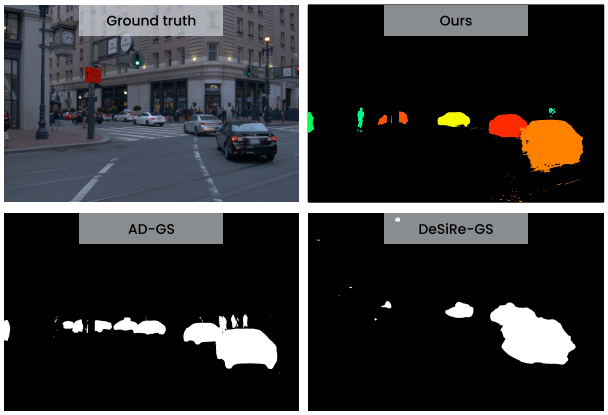

Figure 3. Dynamic mask rendering results. Beyond separating dynamic and static components, our method also renders instance masks for each dynamic object.

Table 3. NVS results under varying view densities. Even with limited training views, our method achieves high DPSNR which highlights the effectiveness of our instance-decomposed model of dynamic objects. First , second , third .

	25%		50%		75%		100%	
	PSNR ↑	DPSNR ↑	PSNR ↑	DPSNR ↑	PSNR ↑	DPSNR ↑	PSNR ↑	DPSNR ↑
DeSiRe-GS	24.37	22.97	28.78	27.34	30.04	28.04	35.11	34.99
AD-GS	26.21	22.33	29.97	26.85	30.76	28.07	34.42	35.09
IDSplat (ours)	26.83	26.35	29.19	28.74	30.11	29.25	35.04	33.67

NeRF-On-The-Road (NOTR) dataset, a curated subset of challenging sequences provided in EmerNeRF [40], and the last the set of 4 sequences used in PVG [3]. NOTR contains 32 static, 32 dynamic, and 56 diverse sequences that cover various weather conditions and road types. See Sec. B for more details.

4.3. Baseline comparisons

We compare IDSplat with state-of-the-art self-supervised rendering methods designed for dynamic automotive scenes, focusing our comparison with the following best performing methods, DeSiRe-GS [20], AD-GS [10], CoDa-4DGS [26], and SplatFlow [28]. In addition, we also report results for several supervised methods that rely on bounding box annotations to model dynamic objects. Among these, SplatAD [7] is the most closely related to our approach and serves as a strong reference for assessing how closely IDSplat approaches supervised performance. For further reference, we run SplatAD using tracks from CasTrack [35], a high-performing tracker on the Waymo 3D tracking leaderboard.

NVS results: We report PSNR, SSIM [33], and LPIPS [45] as our primary evaluation metrics, with LPIPS computed using the VGG network [25]. To specifically assess performance in dynamic regions, we also compute dynamic

Figure 4. Qualitative comparisons of novel view synthesis over different view densities (25%, 50%, and 75% of training frames) on the dynamic subset of Waymo NOTR. Our instance-decomposed representation enables high-quality rendering of dynamic objects even when trained with sparse viewpoints.

PSNR (DPSNR) using dynamic object masks. Since both IDSplat and SplatAD support lidar rendering, we additionally evaluate lidar metrics for these methods, reporting median squared depth error, RMSE intensity error, ray drop accuracy, and chamfer distance (CD).

For a fair comparison with prior work, we adopt the data splits and evaluation protocols of each baseline. For DeSiRe-GS, we follow their setup of using 90% of the frames for optimization and evaluating on every tenth frame. We conduct these experiments on the dynamic NOTR subset, using the three front cameras. Due to computational constraints of DeSiRe-GS, we use half-resolution images and only the first 50 frames in each sequence. For AD-GS, we follow their protocol of using 75% of the frames for optimization and evaluating on every fourth frame. Consistent with their settings, we use the eight sequences presented in StreetGS, the front camera, and full-resolution images. We use the same setting as DeSiRe-GS for CoDa-4DGS, but run the experiments on the complete NOTR dataset. To compare our results to SplatFlow, we use the same 4 Waymo sequences used in [3] using 75% of frames for training. To calculate DPSNR for each setting, we use the same dynamic masks originally used by each respective method as the ground truth.

We present comparisons with baseline methods in Tab. 1. IDSplat achieves competitive or superior performance to prior work on both full-frame and dynamic-region evaluations. Notably, IDSplat performs comparably to its supervised counterpart, SplatAD, and achieves similar performance on LiDAR metrics (Tab. 2). IDSplat further outperforms SplatAD when the latter uses tracks from CasTrack, demonstrating that our approach can outperform dataset-specific trackers.

Decomposition quality: To better understand the differences in DPSNR across methods, Fig. 3 visualizes the dynamic object masks generated by DeSiRe-GS, AD-GS, and our method. AD-GS yields reasonably accurate dynamic regions, while DeSiRe-GS often over-segments static areas. Both methods only segment dynamic regions and cannot decompose individual object instances. In contrast, IDSplat generates instance masks that closely align with ground truth. Since our model maintains a consistent set of Gaussians for each object, the appearance and geometry of each actor remain stable, leading to more coherent reconstructions and high rendering quality in dynamic regions. Further, Fig. 5 illustrates how our instance-decomposition enables targeted modification of objects.

A side effect of our dynamic object model is that instance Gaussians may also represent nearby environmental elements that move with the object such as shadows or adjacent appearance effects. This can also be seen in Fig. 3, where the orange vehicle mask slightly extends into the road surface.

4.4. View density comparisons

To assess the robustness of our approach, we conduct additional experiments under varying view densities. Specifically, we evaluate IDSplat alongside DeSiRe-GS and AD-GS, using 25%, 50%, 75%, and 100% of the frames for optimization, while evaluating on the remaining frames, except for 100% where all frames are also used for evaluation. The frames are selected linearly spaced. To enable a consistent comparison across methods, we adopt a unified setup, using the dynamic subset of NOTR with a single camera at full resolution and the first 50 frames of each sequence. We report PSNR and DPSNR in Tab. 3.

Both DeSiRe-GS and AD-GS perform well in the full reconstruction (100%) setting, demonstrating the effectiveness of their time-varying parameterizations in fitting the data. However, evaluations under sparser settings reveal that these parameterizations struggle with larger interpolation gaps, as they are not explicitly constrained to capture the underlying dynamics. DeSiRe-GS degrades significantly on the sparser settings, especially in dynamic regions. AD-GS exhibits more stable performance across different frame densities, but also suffers from a drop in dynamic regions for more sparse settings. In contrast, the performance of IDSplat in dynamic regions is much more consistent with the full-image results across all view densities, and surpasses the baselines with more than 4.8 PSNR in dynamic regions on the most sparse setting.

This is further illustrated in Fig. 4, which shows qualitative comparisons of novel view reconstruction on the dynamic subset of Waymo NOTR for different view densities. As also observed in quantitative results, IDSplat maintains stable reconstruction quality and consistent geometry even when trained with fewer viewpoints.

4.5. Object class comparisons

For further analysis of our method, we present DPSNR for three different classes of road-users; vehicles, pedestrians, and cyclists. These experiments were run using the 50% setting of the setup presented in Sec. 4.4. Masks of the dynamic regions were obtained by projected 3D bounding box annotations, and filtered based on speed and semantic class. As shown in Tab. 4, IDSplat attains the highest accuracy on vehicles, the only class that is modeled as dynamic instances in our approach. IDSplat also shows strong results on pedestrians and cyclists, even though the rigid motion assumption is only partially valid for these classes.

4.6. Generalization

To assess the generalization and robustness of our approach, we further evaluate it on PandaSet [37], without any hyperparameter tuning. Following [7], we use the same 10 sequences and all six available cameras at full resolution, and compute LPIPS using AlexNet [15]. We perform novel view synthesis using 50% of the frames for optimization and the remaining for evaluation, and compare our results against prior state-of-the-art methods. As shown in Tab. 5, IDSplat achieves performance competitive with the best-performing method, SplatAD, despite not relying on any annotations.

Table 4. NVS results filtered on different dynamic object classes. IDSplat demonstrates strong performance for vehicles and pedestrians, with cyclists being more challenging. First, second, third.

	PSNR	$\text{DPSNR}_{\text{Vehicle}}$	$\text{DPSNR}_{\text{Pedestrian}}$	$\text{DPSNR}_{\text{Cyclist}}$
DeSiRe-GS	28.78	25.04	27.65	29.34
AD-GS	29.97	26.80	27.22	26.52
IDSplat (ours)	29.19	29.02	28.31	27.23

Table 5. NVS results on PandaSet, using all six cameras at full-resolution. IDSplat achieves performance on par with annotation-based methods.

	Anno. free	PSNR ↑	SSIM ↑	LPIPS ↓
PVG	×	24.01	0.712	0.452
Street-GS	×	24.73	0.745	0.314
SplatAD	×	26.76	0.815	0.193
IDSplat (ours)	✓	26.78	0.814	0.174

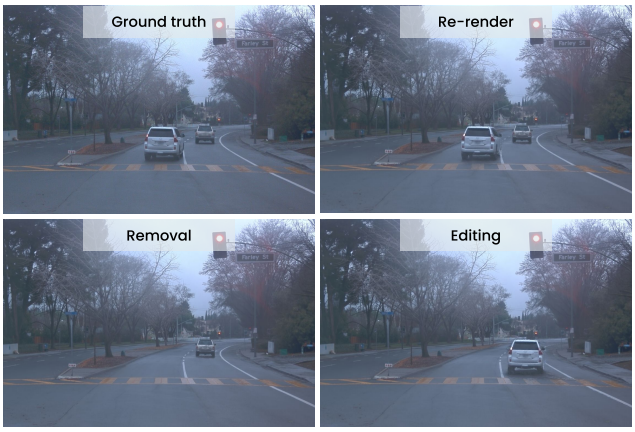

Figure 5. Instance editing. Our instance-decomposed representation enables targeted modifications of individual instances, such as their complete removal or the editing of their trajectories.

4.7. Ablations

We ablate the effectiveness of key components in our method by analyzing their impact on NVS performance, using the AD-GS setting described in Sec. 4.3 but with only 50% of views used for optimization. The results of the ablations are presented in Tab. 6. We observe that the initial filtering of lidar points using eroded masks (a) and DBSCAN

Table 6. NVS results when removing different model components.

		PSNR ↑	SSIM ↑	LPIPS ↓	DPSNR ↑
	Full model	33.59	0.920	0.240	29.35
a)	Eroded masks	33.50	0.920	0.241	29.32
b)	DBSCAN	33.48	0.920	0.242	28.07
c)	Registration	32.60	0.915	0.247	26.67
d)	Image features	32.7	0.917	0.243	25.70
e)	Smoothing	32.96	0.918	0.244	28.01
f)	Outlier rejection	33.38	0.919	0.242	29.44
g)	Refinement	33.32	0.918	0.245	27.36

Table 7. NVS results from using image features from different models when establishing point correspondences. Gray row marks the image features used in our method.

	PSNR ↑	SSIM ↑	LPIPS ↓	DPSNR ↑
RGB	32.76	0.917	0.243	25.72
SAM2	33.24	0.918	0.245	27.74
DINOv3 layer 7	33.59	0.920	0.240	29.35
DINOv3 layer 9	33.57	0.919	0.240	29.17
DINOv3 layer 12	33.52	0.920	0.242	28.95

(b) helps improve the final rendering results. Removing the RANSAC-based registration of points (c), and instead naively estimating pose translations by the point clouds centroids, has a severe impact on the performance. Selecting point correspondences using RGB colors instead of image features (d) also degrades the performance notably. Further, we see that the smoothing (e) is a key component for attaining good results in dynamic regions, while the outlier rejection from a first initial smoothing iteration (f) has a small impact on the final rendering results. Finally, we note the importance of refining the trajectories using gradients from the rendering losses during optimization (g). Additionally, we analyze the impact of image features in Tab. 7. Specifically, we run the registration using different layers of DINOv3, image features from SAM2 [22], and RGB colors. We observe that DINOv3 has the most descriptive features for successful registration, with earlier layers giving a slight increase in performance in dynamic regions.

4.8. Tracking

While IDSplat primarily targets high-quality scene reconstruction and rendering, we also evaluate the quality of the resulting motion trajectories. Specifically, we compare our optimized trajectories against human-annotated ground truth trajectories on a combined set of the 32 sequences from the dynamic subset of NOTR and eight sequences from the AD-GS split. We report Multiple Object Tracking Accuracy (MOTA) [1] and Multiple Object Tracking Precision (MOTP) [1] across different distance thresholds, along with detailed metrics such as false positives, ID switches, recall, and precision. The results are provided in Sec. D along with qualitative examples and evaluation details. Since none of the baseline methods perform instance decomposition, tracking evaluation is not applicable to them. Nevertheless, we include our tracking results to establish a reference point and to encourage future research in this direction.

Our trajectories exhibit occasional ID switches, as IDSplat does not explicitly handle identity association across frames. Additional errors arise from stationary objects or incorrectly classified masks generated by Grounded-SAM-2. Furthermore, this evaluation does not account for potential constant offsets between the predicted and ground-truth trajectories, which can occur even under perfect motion tracking, due to partial or incomplete point cloud representations of objects.

4.9. Limitations

While IDSplat provides strong reconstruction performance and enables instance-level decomposition, several limitations remain. First, our object initialization and trajectory estimation are dependent on lidar measurements. Therefore, objects that fall outside of the lidar field of view but are still visible in the cameras are excluded and considered a part of the static scene. Second, our framework assumes that dynamic actors behave as rigid bodies. This works well for vehicles but is less suitable for highly deformable classes such as pedestrians and cyclists, which can lead to reduced reconstruction quality for these categories. Third, because each dynamic object is represented by a single set of Gaussians, nearby environmental effects such as shadows or reflections, which occur consistently in the scene and move with the object, may be represented by the instance. Finally, the current system lacks explicit track management mechanisms, as we do not merge overlapping tracks or handle ID switches, which may result in duplicated or incomplete instances in challenging scenarios. Addressing these limitations presents a promising direction for future work.

5. Conclusion

We presented IDSplat, a zero-shot framework for instance decomposition and neural rendering in dynamic driving scenes. Extensive experiments show that IDSplat achieves competitive or superior performance compared to state-of-the-art self-supervised approaches, and even matches the performance of annotation-based baselines. By representing each actor as a rigid instance, the method establishes a clear separation not only between dynamic and static scene elements, but also between individual dynamic objects, enabling precise scene editing such as object repositioning or removal.

Acknowledgements

We thank Adam Lilja and William Ljungbergh for valueable feedback. This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. This work was also partially supported by Vinnova, the Swedish Innovation Agency. Computational resources were provided by NAISS at NSC Berzelius, partially funded by the Swedish Research Council, grant agreement no. 2022-06725.

References

[1] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: the CLEAR MOT metrics. *EURASIP Journal on Image and Video Processing*, 2008(1): 246309, 2008. 8, 4, 5

[2] Yun Chen, Jingkang Wang, Ze Yang, Sivabalan Manivasagam, and Raquel Urtasun. G3R: Gradient guided generalizable reconstruction. In *Computer Vision – ECCV 2024*, page 305–323, Cham, 2025. Springer Nature Switzerland. 2

[3] Yurui Chen, Chun Gu, Junzhe Jiang, Xiatian Zhu, and Li Zhang. Periodic vibration gaussian: Dynamic urban scene reconstruction and real-time rendering. *International Journal of Computer Vision*, 2026. 1, 2, 5, 6

[4] Ziyu Chen, Jiawei Yang, Jiahui Huang, Riccardo de Lutio, Janick Martinez Esturo, Boris Ivanovic, Or Litany, Zan Gojcic, Sanja Fidler, Marco Pavone, Li Song, and Yue Wang. OmniRe: Omni urban scene reconstruction. In *The Thirteenth International Conference on Learning Representations*, 2025. 1, 2

[5] Frank Dellaert and GTSAM Contributors. borglab/gtsam, 2022. 4

[6] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Second International Conference on Knowledge Discovery and Data Mining (KDD'96).*, pages 226–231, 1996. 3

[7] Georg Hess and Carl Lindström. splatad. https://github.com/carlinds/splatad, 2025. 5, 7, 1

[8] Georg Hess, Carl Lindström, Maryam Fatemi, Christoffer Petersson, and Lennart Svensson. SplatAD: Real-time lidar and camera rendering with 3D Gaussian splatting for autonomous driving. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 11982–11992, 2025. 1, 2, 3, 4, 5

[9] Nan Huang, Xiaobao Wei, Wenzhao Zheng, Pengju An, Ming Lu, Wei Zhan, Masayoshi Tomizuka, Kurt Keutzer, and Shanghang Zhang. S^3gaussian: Self-supervised street gaussians for autonomous driving. *arXiv preprint arXiv:2405.20323*, 2024. 1, 2

[10] Xu Jiawei, Deng Kai, Fan Zexin, Wang Shenlong, Xie Jin, and Yang Jian. AD-GS: Object-aware B-Spline Gaussian splatting for self-supervised autonomous driving. *International Conference on Computer Vision*, 2025. 1, 2, 5

[11] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 1, 2

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

[13] Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Yang-Che Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3D Gaussian splatting as markov chain monte carlo. In *Advances in Neural Information Processing Systems*, 2024. 4, 1, 2

[14] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 2012. 7

[16] Carl Lindström, Georg Hess, Adam Lilja, Maryam Fatemi, Lars Hammarstrand, Christoffer Petersson, and Lennart Svensson. Are NeRFs ready for autonomous driving? Towards closing the real-to-simulation gap. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4461–4471, 2024. 1

[17] William Ljungbergh, Adam Tonderski, Joakim Johnander, Holger Caesar, Kalle Åström, Michael Felsberg, and Christoffer Petersson. NeuroNCAP: Photorealistic closed-loop safety testing for autonomous driving. In *European Conference on Computer Vision*, pages 161–177. Springer, 2024. 1

[18] Yunxuan Mao, Rong Xiong, Yue Wang, and Yiyi Liao. Unire: Unsupervised instance decomposition for dynamic urban scene reconstruction, 2025. 2

[19] 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. 2

[20] Chensheng Peng, Chengwei Zhang, Yixiao Wang, Chenfeng Xu, Yichen Xie, Wenzhao Zheng, Kurt Keutzer, Masayoshi Tomizuka, and Wei Zhan. DeSiRe-GS: 4D street gaussians for static-dynamic decomposition and surface reconstruction for urban driving scenes. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 6782–6791, 2025. 1, 2, 5, 4

[21] Mahan Rafidashti, Ji Lan, Maryam Fatemi, Junsheng Fu, Lars Hammarstrand, and Lennart Svensson. NeuRadar: Neural radiance fields for automotive radar point clouds. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2479–2489, 2025. 2

[22] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. SAM 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. 8

[23] Tianhe Ren and Shuo Shen. Grounded-SAM-2. https://github.com/IDEA-Research/Grounded-SAM-2, 2024. 2, 3, 5

[24] Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. DINOv3. *arXiv preprint arXiv:2508.10104*, 2025. 2, 3, 5

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 5

[26] Rui Song, Chenwei Liang, Yan Xia, Walter Zimmer, Hu Cao, Holger Caesar, Andreas Festag, and Alois Knoll. Coda-4dgs: Dynamic gaussian splatting with context and deformation awareness for autonomous driving. In *IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE/CVF, 2025. 2, 5

[27] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2, 4

[28] Su Sun, Cheng Zhao, Zhuoyang Sun, Yingjie Victor Chen, and Mei Chen. SplatFlow: Self-supervised dynamic gaussian splatting in neural motion flow field for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 27487–27496, 2025. 2, 5

[29] Adam Tonderski, Carl Lindström, Georg Hess, and William Ljungbergh. neurad-studio. https://github.com/georghess/neurad-studio, 2024. 5

[30] Adam Tonderski, Carl Lindström, Georg Hess, William Ljungbergh, Lennart Svensson, and Christoffer Petersson. NeuRAD: Neural rendering for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14895–14904, 2024. 1, 2

[31] Haithem Turki, Jason Y Zhang, Francesco Ferroni, and Deva Ramanan. SUDS: Scalable urban dynamic scenes. In *Computer Vision and Pattern Recognition (CVPR)*, 2023. 1, 2

[32] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on pattern analysis and machine intelligence*, 13(4): 376–380, 2002. 3

[33] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004. 5

[34] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4D Gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20310–20320, 2024. 2

[35] Hai Wu, Wenkai Han, Chenglu Wen, Xin Li, and Cheng Wang. 3d multi-object tracking in point clouds based on prediction confidence-guided data association. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):5668–5677, 2022. 5

[36] Zirui Wu, Tianyu Liu, Liyi Luo, Zhide Zhong, Jianteng Chen, Hongmin Xiao, Chao Hou, Haozhe Lou, Yuantao Chen, Runyi Yang, Yuxin Huang, Xiaoyu Ye, Zike Yan, Yongliang Shi, Yiyi Liao, and Hao Zhao. MARS: An instance-aware, modular and realistic simulator for autonomous driving. In *Artificial Intelligence*, pages 3–15, Singapore, 2024. Springer Nature Singapore. 1, 2

[37] Pengchuan Xiao, Zhenlei Shao, Steven Hao, Zishuo Zhang, Xiaolin Chai, Judy Jiao, Zesong Li, Jian Wu, Kai Sun, Kun Jiang, Yunlong Wang, and Diange Yang. PandaSet: Advanced sensor suite dataset for autonomous driving. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 3095–3101, 2021. 5, 7

[38] Jiawei Xu, Zexin Fan, Jian Yang, and Jin Xie. Grid4D: 4d decomposed hash encoding for high-fidelity dynamic gaussian splatting. *Advances in Neural Information Processing Systems*, 37:123787–123811, 2024. 2

[39] Yunzhi Yan, Haotong Lin, Chenxu Zhou, Weijie Wang, Haiyang Sun, Kun Zhan, Xianpeng Lang, Xiaowei Zhou, and Sida Peng. Street gaussians for modeling dynamic urban scenes. In *Computer Vision – ECCV 2024*, 2024. 1, 2, 5

[40] Jiawei Yang, Boris Ivanovic, Or Litany, Xinshuo Weng, Seung Wook Kim, Boyi Li, Tong Che, Danfei Xu, Sanja Fidler, Marco Pavone, et al. EmerNeRF: Emergent spatial-temporal scene decomposition via self-supervision. *arXiv preprint arXiv:2311.02077*, 2023. 1, 2, 5

[41] Jiawei Yang, Jiahui Huang, Yuxiao Chen, Yan Wang, Boyi Li, Yurong You, Maximilian Igl, Apoorva Sharma, Peter Karkus, Danfei Xu, Boris Ivanovic, Yue Wang, and Marco Pavone. STORM: Spatio-temporal reconstruction model for large-scale outdoor scenes. *arXiv preprint arXiv:2501.00602*, 2025. 2

[42] 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 *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1389–1399, 2023. 1, 2

[43] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3D Gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20331–20341, 2024. 2

[44] Yuanwen Yue, Anurag Das, Francis Engelmann, Siyu Tang, and Jan Eric Lenssen. Improving 2D feature representations by 3D-aware fine-tuning. In *Computer Vision – ECCV 2024*, page 57–74, Cham, 2025. Springer Nature Switzerland. 2

[45] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 5

[46] Ruida Zhang, Chengxi Li, Chenyangguang Zhang, Xingyu Liu, Haili Yuan, Yanyan Li, Xiangyang Ji, and Gim Hee Lee. Street gaussians without 3D object tracker. *arXiv preprint arXiv:2412.05548*, 2024. 1, 2

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

[48] Xiaoyu Zhou, Zhiwei Lin, Xiaojun Shan, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. DrivingGaussian: Composite gaussian splatting for surrounding dynamic autonomous driving scenes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21634–21643, 2024. 1, 2

IDSplat: Instance-Decomposed 3D Gaussian Splatting for Driving Scenes

Supplementary Material

In this supplementary material, we provide additional details of our implementation, datasets, and baselines, as well as extended quantitative and qualitative results. In Sec. A we describe the details of our implementation, including all hyperparameters and training settings. Sec. B outlines the dataset splits used in our experiments, while Sec. C provides details regarding the baselines included for comparison. In Sec. D, we present further quantitative and qualitative evaluations of our tracking performance. Sec. E reports inference and training times, along with a preprocessing breakdown across the components of our method. Finally, Sec. F showcases additional qualitative examples of novel view synthesis for both our approach and the baselines.

A. Implementation details

Scene representation: We initialize Gaussians using up to a maximum of 5M lidar points, using the color from projecting the points into the temporally closest camera to initialize the base color of each Gaussian. We add 2M additional points with random colors and positions, where half are sampled uniformly within lidar range, and half are sampled linearly in inverse distance outside lidar range. Using the densification strategy in [13], we allow the representation to grow up to a maximum of 10M Gaussians. All Gaussians are initialized with 50% opacity and scaled to 20% of the average distance to their three nearest neighbors. We use a feature dimension of 13 for each Gaussian's corresponding feature vector, and a feature dimension of 8 for the sensor-embeddings. Following [7], we decode lidar intensity and ray drop probability using a small MLP of 2 layers and a hidden dimension of 32. Similarly, we use a small CNN for decoding the view-dependent effects when rendering images. This CNN is implemented using two residual blocks with a hidden dimension 32, kernel size 3, and a linear output layer.

Instance decomposition: To generate object masks, we query Grounded-SAM-2 for `car`, `truck`, `van`, `bus`, `train`, `human`, `cyclist`, `bicycle`, and `pedestrian` instances and track these masks across the sequence to assign consistent instance IDs. We prompt every frame to reduce frames with missing masks. Each mask is eroded by 3 pixels before projecting lidar points, as described in Sec. 3.2. We only consider lidar points within 80 meters of the sensor. The DBSCAN-clustering in Sec. 3.2 is performed with a maximum neighborhood distance of 0.5 meters and a minimum of 10 points in a neighborhood to determine a core point. We select the largest cluster as the initial 3D representation for that instance. The DINOv3 features used in Sec. 3.3 are taken from layer 7 (of the ViT-B/16 version), upsampled to the image size using bilinear interpolation, and associated to the points in the cluster by projection.

Trajectory estimation: For computational efficiency, we sub-sample both the source and target point clouds in our registration to a maximum of 5000 points, selected randomly. The pose between the point clouds is estimated from 100,000 RANSAC iterations. Each iteration samples three point correspondences, obtained from cosine-similarity using DINOv3 features. We only use matches with a similarity higher than 0.8. We define registration fitness as the ratio of target points that are within 10 cm of a source point after transformation, and use a fitness threshold of 0.5 to determine whether a registration is successful or not.

Trajectory smoothing: Object instances with total trajectory displacement below 1 meter are converted to the static representation. For all remaining instances, we refine their RANSAC-derived poses via an iterative smoothing process using a GTSAM factor graph. The states include poses, velocities, and curvatures, while the RANSAC estimates serve as measurements. As described in Sec. 3.4, we additionally estimate a single rotation shared across all timestamps to align the axis-aligned measurements with the motion model (which assumes the local x-axis is forward). Measurement factors are implemented as absolute-pose factors with a Huber loss (threshold 1.0) and diagonal noise with standard deviations $[0.1,\ 0.1,\ 0.1]$ rad for rotation and $[0.2,\ 0.2,\ 0.2]$ m for translation. The motion model is defined as

$$\theta = \kappa v \Delta t \tag{6}$$

$$\Delta x = \frac{\sin(\theta)}{\kappa} \tag{7}$$

$$\Delta y = \frac{1-\cos(\theta)}{\kappa} \tag{8}$$

$$\Delta z = 0.0 \tag{9}$$

$$\Delta R = R_z(\theta), \tag{10}$$

where

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{11}$$

Note that pose states are first rotated by the shared rotation before applying the motion-model prediction. Motion-model factors connect successive states, where the residual is defined as the deviation from the predicted motion. These factors use diagonal noise with standard deviations $[0.1,\ 0.1,\ 0.1]$ rad for rotation and $[0.2,\ 0.2,\ 0.2]$ m for

Table 8. Learning rates (LR) for each parameter group. Exponential decay is used for scheduling when applicable.

Parameters	Initial LR	Final LR	Warm-up steps
Gaussian means	1.6e-4	1.6e-6	0
Gaussian opacities	5.0e-2	5.0e-2	0
Gaussian scales	5.0e-3	5.0e-3	0
Gaussian quaternions	1.0e-3	1.0e-3	0
Gaussian features	2.5e-3	2.5e-3	0
Decoders	1.0e-3	1.0e-3	500
Dynamic object positions	1.0e-3	1.0e-4	1500
Dynamic object rotations	5.0e-5	1.0e-6	1500
Sensor embeddings	1.0e-3	1.0e-3	500
Sensor vel. linear	1.0e-3	1.0e-6	1000
Sensor vel. angular	2.0e-4	1.0e-7	1000
Cam. time to center	2.0e-4	1.0e-7	10000

translation. Velocity and curvature states follow random-walk priors with discretized standard deviations $\sqrt{0.5\Delta t}$ and $\sqrt{10^{-5}\Delta t}$, respectively. Pose states are further regularized to maintain moderate roll and pitch angles via magnitude-based residuals (computed after applying the shared rotation), with diagonal noise and standard deviation 0.4 for both roll and pitch. A curvature prior with standard deviation of 0.01 rad is also applied at every timestep.

We optimize using GTSAM´s Levenberg-Marquardt solver. We run a single iteration to identify outlier measurements whose whitened error exceeds 1.345, and then run the final optimization without those measurements for a maximum of ten iterations.

Scene optimization: We jointly optimize all model and pose parameters for 30,000 steps with the Adam optimizer [14]. Learning rates and scheduling parameters are reported in Tab. 8. Following [11], we adopt a resolution-scheduling scheme in which the first 3,000 optimization steps use images downsampled by a factor of 4, the next 3,000 steps use a downsampling factor of 2, and the remaining steps are performed with the original image size.

We use the loss formulation and hyperparameter settings from [8], without additional tuning. While these baseline values proved robust, we note that dataset-specific tuning may yield further improvements. As in [8], we employ the MCMC described in Eq. (4) and adapted from [13], which include opacity and scale regularization:

$$\lambda_{\text{MCMC}}\mathcal{L}_{\text{MCMC}} = \lambda_o \sum_i |o_i| + \lambda_\Sigma \sum_{ij} \left| \sqrt{\text{eig}_j(\Sigma_i)} \right| . \quad (12)$$

All loss-related hyperparameters are reported in Tab. 9.

B. Dataset details

In this section, we provide additional details about the datasets used in our experiments. We evaluate on three subsets of the Waymo Open Dataset, chosen to match the baseline settings, and include PandaSet to demonstrate the robustness of our method.

Table 9. Hyperparameters used for loss weighting.

Loss parameter	Weight
λ_r	0.8
λ_{depth}	0.1
λ_{los}	0.1
λ_{inten}	1.0
λ_{raydrop}	0.1
λ_o	5e-3
λ_Σ	1e-3

StreetGS split: This subset of Waymo, used for the AD-GS evaluation setting, contains 8 sequences of roughly 100 frames, each featuring a variety of moving vehicles. These sequences do not include pedestrians or cyclists. We followed instructions in the official implementation of StreetGS [39] to preprocess this subset. The original segments used to construct this split, along with their start and end frames, are listed in Tab. 10.

NOTR: NeRF-On-The-Road (NOTR) is a curated set of 120 driving sequences from Waymo spanning a broad range of challenging conditions. It includes 32 *Static* scenes, 32 *Dynamic* scenes with multiple moving actors, and 56 *Diverse* scenes capturing variations in driving speed, weather, and lighting conditions. We preprocessed this data following the official code of [40]. Tab. 11 lists all Waymo data segments included in this subset. We use *Dynamic* NOTR sequences for the DeSiRe-GS setting and the complete NOTR dataset for the CoDa-4DGS setting.

PVG split: This set of 4 sequences from Waymo were used in [3] and adopted by SplatFlow in their experiments. Further details of these sequences is presented in Tab. 12.

Pandaset: PandaSet is a multimodal autonomous driving dataset containing camera and lidar data captured in diverse urban environments. We used the following 10 sequences from PandaSet: `001`, `011`, `016`, `028`, `053`, `063`, `084`, `106`, `123`, and `158`.

Across our experiments, we selected the appropriate data based on the baselines under comparison and the corresponding cameras, image resolution and train-test splits. Full details are given in Tab. 13.

Dynamic ground truth masks: Both the StreetGS split and NOTR provide preprocessed dynamic masks obtained by projecting Waymo's 3D bounding box annotations onto the image plane and filtering objects based on their speed. These masks, however, are binary segmentation masks and do not include object class information. To evaluate the DP-SNR over different classes of road-users, we generate our own 2D dynamic object masks from the same 3D bounding boxes using the same procedure. We create 3 sets of masks

Table 10. StreetGS sequences and the corresponding Waymo Open Dataset segments.

Sequence ID	Segment Name	Start Frame	End Frame
006	segment-10448102132863604198_472_000_492_000	0	85
026	segment-12374656037744638388_1412_711_1432_711	0	100
090	segment-17612470202990834368_2800_000_2820_000	0	102
105	segment-1906113358876584689_1359_560_1379_560	20	186
108	segment-2094681306939952000_2972_300_2992_300	20	115
134	segment-4246537812751004276_1560_000_1580_000	106	198
150	segment-5372281728627437618_2005_000_2025_000	96	197
181	segment-8398516118967750070_3958_000_3978_000	0	160

Table 11. Dynamic NOTR sequences and the corresponding Waymo Open Dataset segments (-1 denotes the last frame).

Sequence ID	Segment Name	Start Frame	End Frame
016	segment-10231929575853664160_1160_000_1180_000	0	-1
021	segment-10391312872392849784_4099_400_4119_400	0	-1
022	segment-10444454289801298640_4360_000_4380_000	0	-1
025	segment-10498013744573185290_1240_000_1260_000	0	-1
031	segment-10588771936253546636_2300_000_2320_000	0	-1
034	segment-10625026498155904401_200_000_220_000	0	-1
035	segment-10664823084372323928_4360_000_4380_000	0	-1
049	segment-10963653239323173269_1924_000_1944_000	0	-1
053	segment-11017034898130016754_697_830_717_830	0	-1
080	segment-11718898130355901268_2300_000_2320_000	0	-1
084	segment-11846396154240966170_3540_000_3560_000	0	-1
086	segment-11917887606306240 72_3880_000_3900_000	0	-1
089	segment-11928449532664718059_1200_000_1220_000	0	-1
094	segment-12027892938363296829_4086_280_4106_280	0	-1
096	segment-12161824480686739258_1813_380_1833_380	0	-1
102	segment-12251442326766052580_1840_000_1860_000	0	-1
111	segment-12339284075576056695_1920_000_1940_000	0	-1
222	segment-14810689888487451189_720_000_740_000	0	-1
323	segment-16801666784196221098_2480_000_2500_000	0	-1
382	segment-18111897798871103675_320_000_340_000	0	-1
402	segment-19187642209842096 54_5680_000_5700_000	0	-1
427	segment-2259324582958830057_3767_030_3787_030	0	-1
438	segment-2547899409721197155_1380_000_1400_000	0	-1
546	segment-4414235478445376689_2020_000_2040_000	0	-1
581	segment-5083516879091912247_3600_000_3620_000	0	-1
592	segment-5222336716599194110_8940_000_8960_000	0	-1
620	segment-5835049423600303130_180_000_200_000	0	-1
640	segment-6242822583398487496_73_000_93_000	0	-1
700	segment-7670103006580549715_360_000_380_000	0	-1
754	segment-8822503619482926605_1080_000_1100_000	0	-1
795	segment-9907794657177651763_1126_570_1146_570	0	-1
796	segment-9909146853379551 14_980_000_1000_000	0	-1

for vehicles, pedestrians, and cyclists. In addition, we apply an extra filter: bounding boxes with fewer than 10 associated lidar points are discarded to ensure that evaluation only considers objects that are observed by lidar. In Tab. 1 and Tab. 2, we use the class-agnostic dynamic masks provided by each dataset for computing DPSNR. In all other tables, we employ our own generated dynamic masks. Unless stated otherwise, DPSNR is computed using ground truth masks for moving vehicles.

C. Baseline details

To obtain results for our two main baselines, DeSiRe-GS and AD-GS, we use their official implementation for train-

Table 12. PVG sequences and the corresponding Waymo Open Dataset segments.

Sequence ID	Segment Name	Start Frame	End Frame
017	segment-10235335145367115211_5420_000_5440_000	61	109
022	segment-13186511704021307558_2000_000_2020_000	26	74
050	segment-13207915841618107559_2980_000_3000_000	6	54
081	segment-13506499849906169066_120_000_140_000	26	74

Table 13. Evaluation settings for tables in the main paper. * denotes that we use the full segments from StreetGS instead of using the start and end frames reported in Tab. 10.

Table	Data	Num. Sequences	Cameras	Image Res.	Train. views
Tab. 1 (DeSiRe-GS)	Dynamic NOTR	32	front, front_left, front_right	$[640 \times 960]$	90%
Tab. 1 (AD-GS)	StreetGS Split	8	front	$[1280 \times 1920]$	75%
Tab. 1 (CoDa)	StreetGS Split	120	front, front_left, front_right	$[640 \times 960]$	90%
Tab. 1 (SF)	PVG Split	4	front, front_left, front_right	$[640 \times 960]$	75%
Tab. 2	Dynamic NOTR	32	front, front_left, front_right	$[640 \times 960]$	90%
Tab. 3	Dynamic NOTR	32	front	$[1280 \times 1920]$	25%, 50%, 75%
Tab. 4	Dynamic NOTR	32	front	$[1280 \times 1920]$	50%
Tab. 5	PandaSet	10	all 6 cameras	$[1920 \times 1080]$	50%
Tab. 6	StreetGS Split*	8	front	$[1280 \times 1920]$	50%
Tab. 7	StreetGS Split*	8	front	$[1280 \times 1920]$	50%

Table 14. DeSiRe-GS NVS results for Dynamic NOTR with 90% training views. SSIM and LPIPS were not reported in the paper.

	PSNR ↑	DPSNR ↑
Reported results (Tab. 4 in [20])	30.45	28.66
Reproduced results (Tab. 1)	28.76	26.26

ing, evaluation, and visualization. We only modify configuration parameters for camera selection, image resolution, and train-test data split to match each experiment setting. All other hyperparameters remain the same, and the training follows the schedules reported in the original papers. For both methods, we adapt the official rendering scripts to extract the dynamic masks shown in Fig. 3 from their rendered output.

Despite using the official code and training parameters, we could not reproduce the DeSiRe-GS results that they reported in *Table 4* in their paper. We therefore report both the paper's numbers and our reproduced results in Tab. 14 for transparency.

D. Tracking results

We report the tracking performance of IDSplat in Tab. 15, evaluated over the combined 40 sequences from the NOTR dynamic subset and the AD-GS split. We provide Multiple Object Tracking Accuracy (MOTA) [1] and Multiple Object Tracking Precision (MOTP) [1], along with the full set of underlying metrics used to compute them. *Frames* corresponds to the total number of processed frames, while *Objects* and *Predictions* represent the total number of ground-truth and predicted object appearances, respectively (i.e., not counts of unique object identities). *Matches* refers to the number of ground-truth-prediciton pairs that fall inside the distance threshold and are assigned via the Hungarian algorithm. *Switches* counts the number of cases where a ground-truth identity is associated to different predicted identities over time. *FP* denotes false positives (predictions with no corresponding ground-truth object), and *FN* denotes false negatives (ground-truth objects with no corresponding prediction). MOTA is derived from these as

$$\text{MOTA} = 1 - \frac{\text{FP} + \text{FN} + \text{Switches}}{\text{Objects}}, \tag{13}$$

while MOTP measures the localization error for matched pairs, averaged over all matches. *Recall* quantifies the fraction of ground-truth objects that were detected, and *Precision* quantifies the fraction of correct predictions. To provide a comprehensive view of performance, we compute these metrics over six distance thresholds: 0.5 m, 1.0 m, 2.0 m, 3.0 m, 5.0 m and 10.0 m. Qualitative examples are shown in Fig. 6.

As expected, increasing the distance threshold yields more matches, reducing both the number of false positives and false negatives and thereby improving MOTA, precision, and recall. However, this comes at the cost of more identity switches and reduced localization accuracy (higher MOTP error). The low number of matches at smaller thresholds can partly be attributed to constant offsets between

Table 15. Tracking performance of IDSplat on the combined 40 sequences from the NOTR dynamic subset and the AD-GS split, evaluated over different distance thresholds for the matching. MOTA and MOTP denotes Multiple Object Tracking Accuracy [1] respectively Multiple Object Tracking Precision [1].

Dist. threshold [m]	# Frames	# Objects	# Predictions	# Matches	# Switches ↓	# FP ↓	# FN ↓	MOTA ↑	MOTP ↓	Recall ↑	Precision ↑
0.5	2024	10295	11160	2769	83	8308	7443	-0.54	0.27	0.28	0.26
1.0	2024	10295	11160	3835	119	7206	6341	-0.33	0.41	0.38	0.35
2.0	2024	10295	11160	6487	203	4470	3605	0.20	0.87	0.65	0.60
3.0	2024	10295	11160	7528	223	3409	2544	0.40	1.07	0.75	0.69
5.0	2024	10295	11160	7717	232	3211	2346	0.44	1.16	0.77	0.71
10.0	2024	10295	11160	7903	270	2987	2122	0.48	1.71	0.79	0.73

	Inference rate [MP/s]	Train time [min]
DeSiRe-GS	0.9	413.1
IDSplat	**29.6**	**63.7**
AD-GS	12.2	150.6
IDSplat	**51.5**	**116.5**

Table 16. Inference rate in megapixels per second (MP/s) and total training time in minutes, compared against DeSiRe-GS and AD-GS in their respective settings.

Exp. setting	Tot. time [s]	Mask gen.	Point paint.	Registration	Smoothing
AD-GS	127.6	79.8 (62.5%)	12.7 (9.9%)	33.5 (26.3%)	1.6 (1.3%)
DeSiRe	98.9	50.1 (50.6%)	26.4 (26.7%)	20.3 (20.5%)	2.1 (2.2%)

Table 17. Per-component preprocessing time breakdown for IDSplat, reported in seconds with percentage of total. This preprocessing time is included in the total training times of Tab. 16.

the predicted and ground-truth trajectories, which may arise even under perfect motion tracking due to partial or incomplete point cloud representations of predicted objects. Nevertheless, the results also highlight several opportunities for future work, including inter-frame identity association and more advanced strategies for modeling object births and deaths.

E. Runtime analysis

Tab. 16 reports the median inference rate and median total training time for our method compared to DeSiRe-GS and AD-GS, each evaluated in their respective setting using official implementations. Tab. 17 further breaks down preprocessing time across the components of IDSplat. The preprocessing time is included in the total training times reported above.

F. Qualitative results

We provide additional qualitative examples in Fig. 7, depicting NVS results for IDSplat, AD-GS and DeSiRe when using 75% of the views for training. All examples show validation views, and all sequences are from the dynamic subset of NOTR.

Additional qualitative results on deformable object classes are shown in Fig. 8. Despite rigid-body modeling, pedestrians and cyclists are rendered with high fidelity. Rigid motion captures the dominant movement while small deformations are absorbed by view-dependent effects.

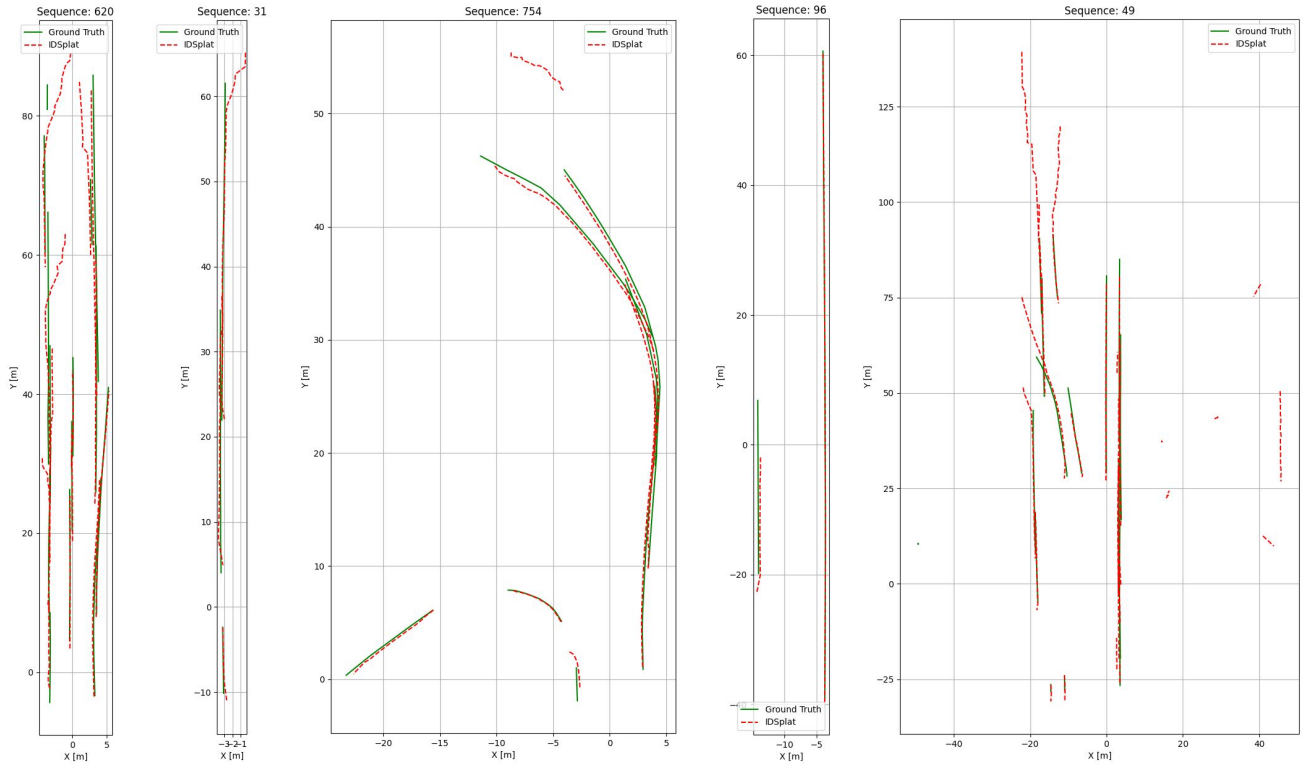

Figure 6. Plots of optimized trajectories from IDSplat compared to ground-truth, for five different sequences.

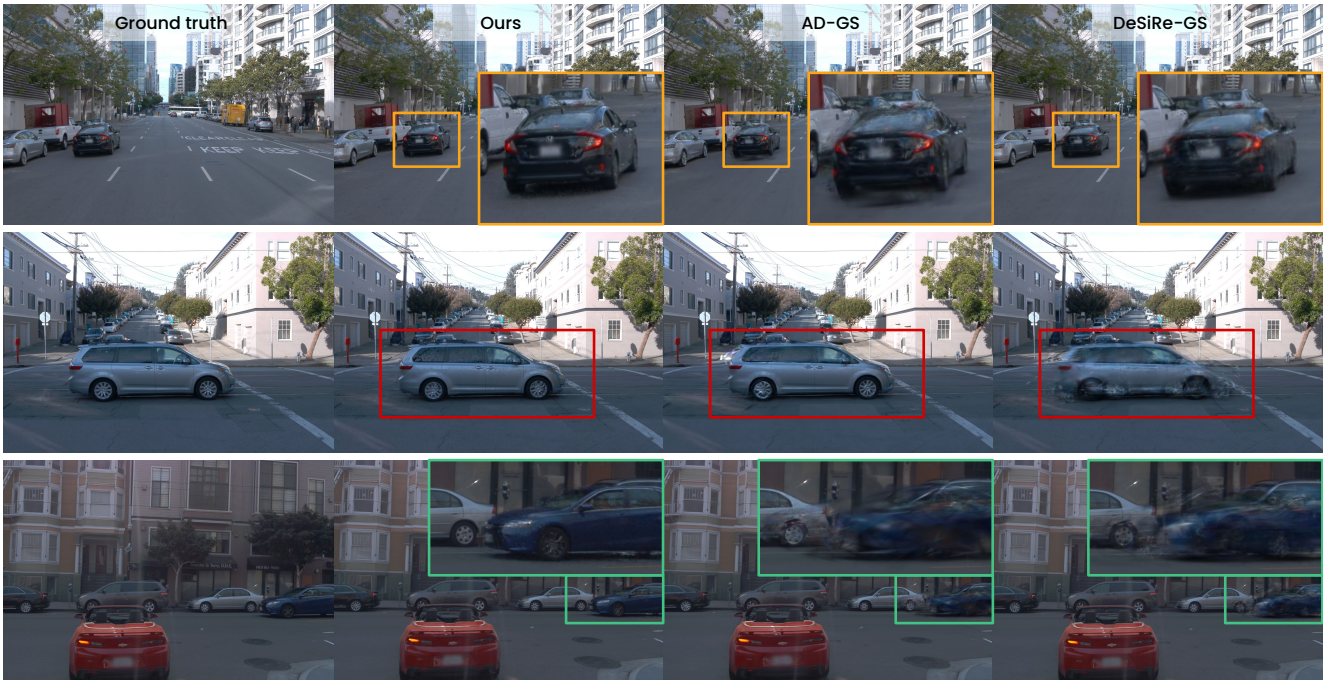

Figure 7. Qualitative comparison with baselines on the dynamic subset of Waymo NOTR, using 75% of views for training.

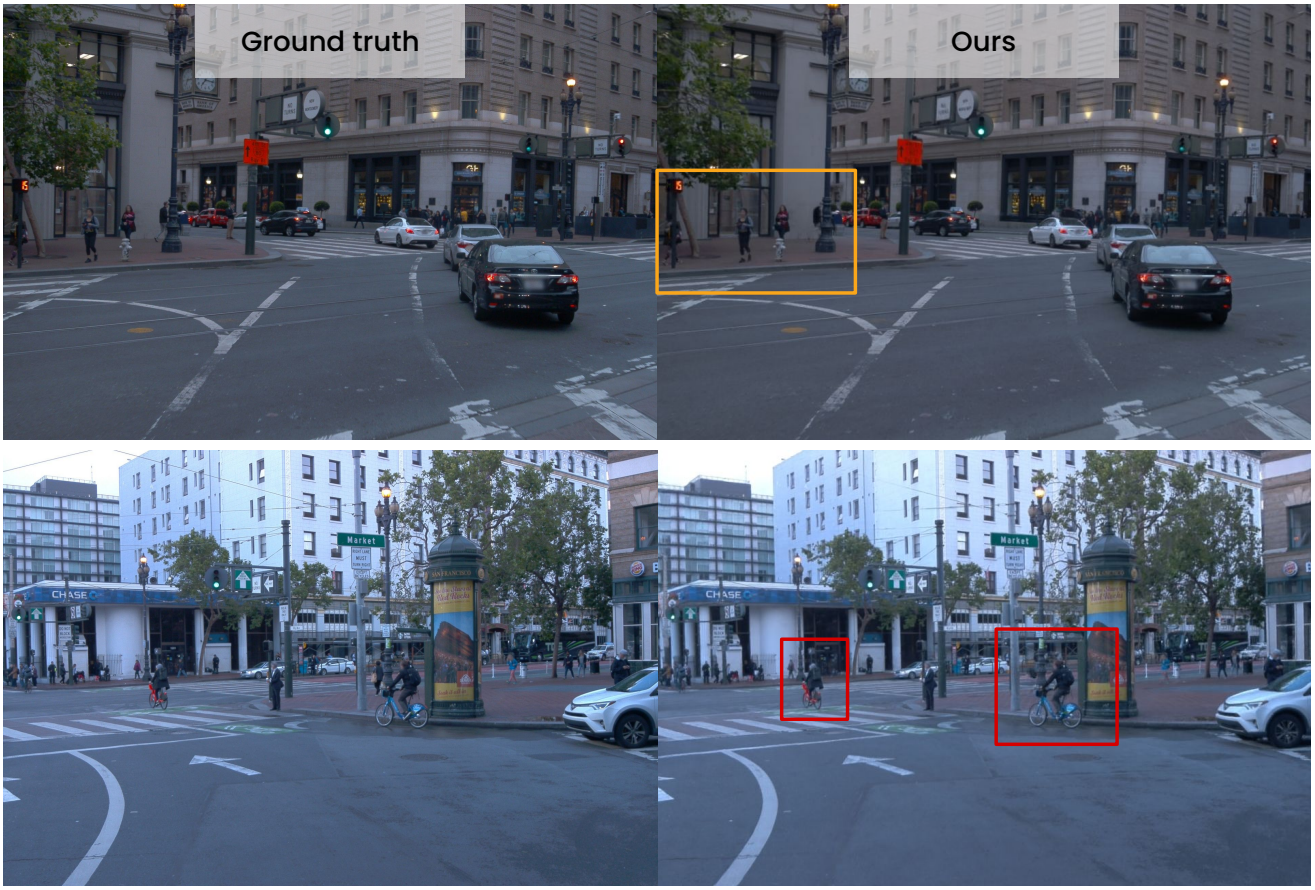

Figure 8. Qualitative results on deformable object classes. Pedestrians and cyclists are rendered with high fidelity despite being modeled as rigid.