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-22593245829588300 57_3767_030_3787_030	0	-1
438	segment-25478994097211971 55_1380_000_1400_000	0	-1
546	segment-44142354784453766 89_2020_000_2040_000	0	-1
581	segment-50835168790919122 47_3600_000_3620_000	0	-1
592	segment-52223367165991941 10_8940_000_8960_000	0	-1
620	segment-58350494236003031 30_180_000_200_000	0	-1
640	segment-6242822583398487496_73_000_93_000	0	-1
700	segment-76701030065805497 15_360_000_380_000	0	-1
754	segment-88225036194829266 05_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.

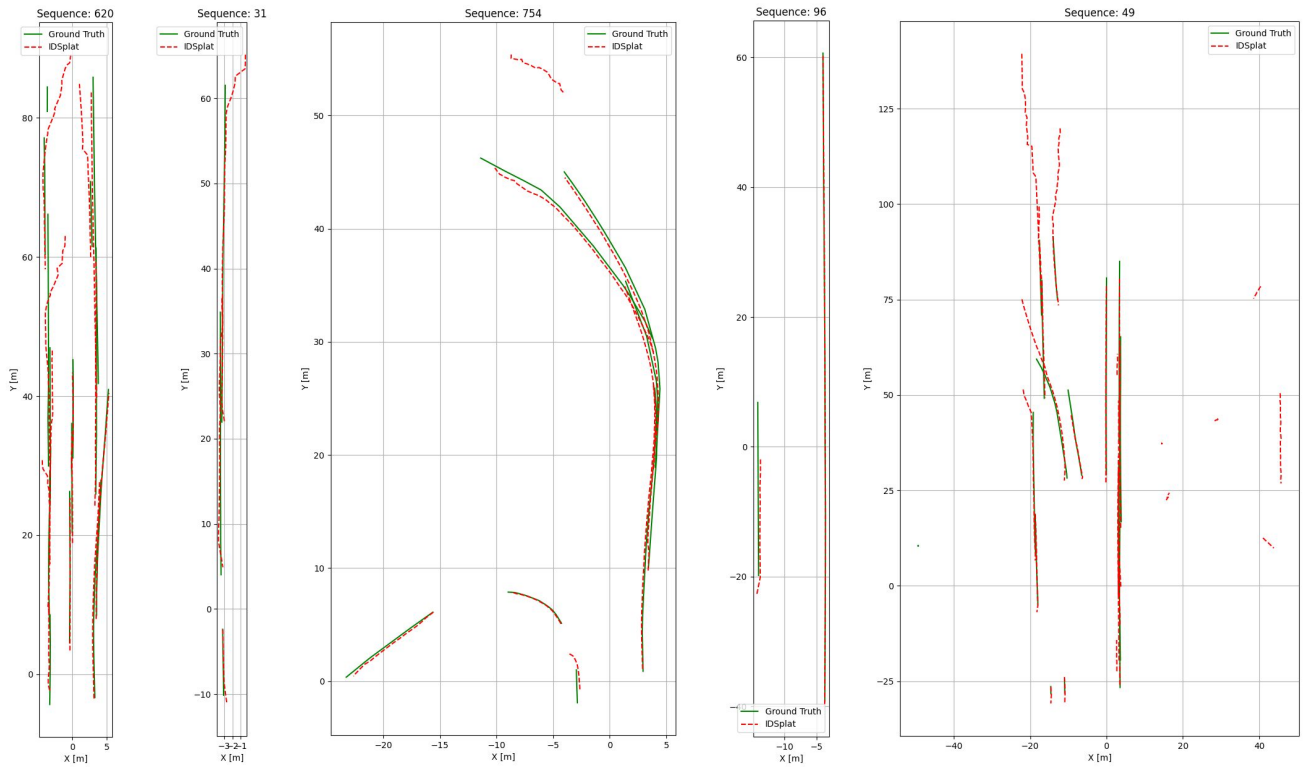

Figure 6. Plots of optimized trajectories from IDSplat compared to ground-truth, for five different sequences.

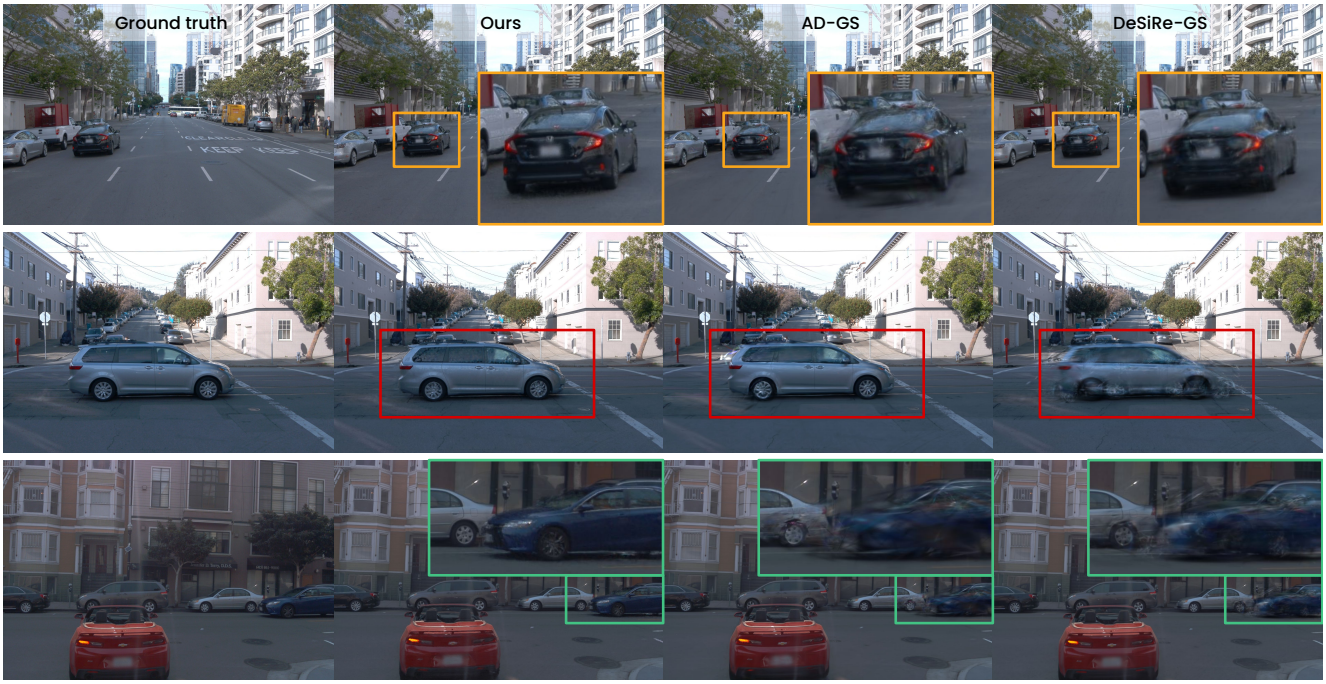

Figure 7. Qualitative comparison with baselines on the dynamic subset of Waymo NOTR, using 75% of views for training.

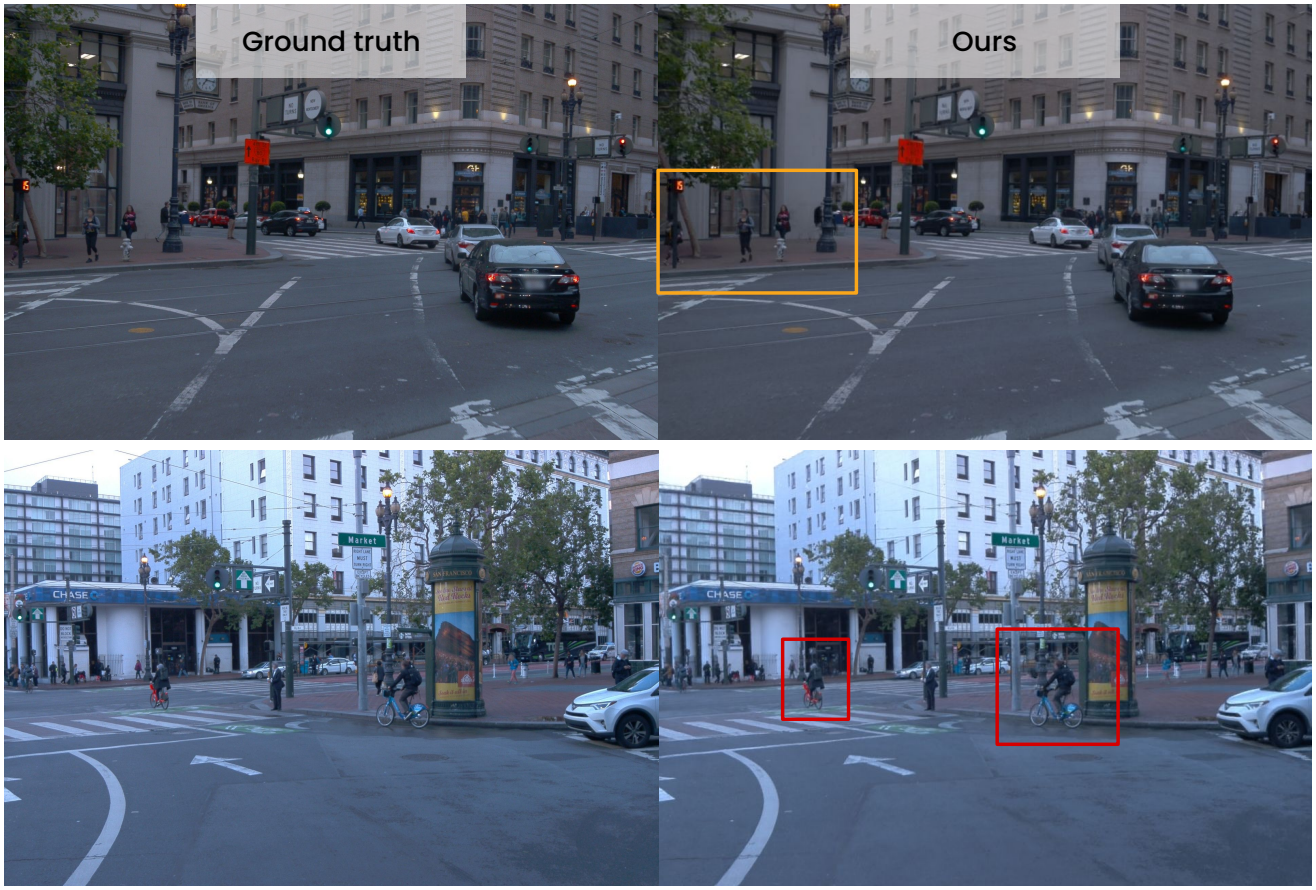

Figure 8. Qualitative results on deformable object classes. Pedestrians and cyclists are rendered with high fidelity despite being modeled as rigid.