Appendix

In this appendix, we begin by discussing implementation details in Appendix A, which includes information about our 3D Gaussian, metrics, and the training and inference processes. We then describe the datasets used in our experiments in Appendix B. Appendix C provides information about the baselines we compare with. Finally, Appendix D contains additional experiment results.

A. Implementation

In this section, we begin by discussing our 3D Gaussian details, encompassing semantic, opacity and depth implementation (Appendix A.1). Subsequently, we discuss the difference between 3D softmax and 2D softmax in 3D Semantic Scene Reconstruction (Appendix A.2). Finally, we elucidate the evaluation metrics we utilize (Appendix A.3). Our source code will be released.

A.1. 3D Gaussian Details

Following [18], each Gaussian has the following attributes: rotation ($\mathbf{R}_g \in \mathbb{R}^{3\times3}$), scale ($\mathbf{S}_g \in \mathbb{R}^{3\times1}$), opacity ($\alpha$) and spherical harmonics (SH). The corresponding 3D covariance matrix $\mathbf{\Sigma} \in \mathbb{R}^{3\times3}$ can be calculated using the following formula:

$$\mathbf{\Sigma} = \mathbf{R}_g\mathbf{S}_g\mathbf{S}_g^T\mathbf{R}_g^T \tag{15}$$

When provided with a viewing transformation $\mathbf{W} \in \mathbb{R}^{3\times3}$ and the Jacobian of the affine approximation of the projective transformation $\mathbf{J} \in \mathbb{R}^{3\times3}$, the covariance matrix $\mathbf{\Sigma}' \in \mathbb{R}^{3\times3}$ in camera coordinates can be expressed as:

$$\mathbf{\Sigma}' = \mathbf{J}\mathbf{W}\mathbf{\Sigma}\mathbf{W}^T\mathbf{J}^T \tag{16}$$

Following EWA splatting [54], we can skip the third row and column of $\mathbf{\Sigma}'$ to obtain a 2×2 covariance matrix with the same structure and properties. For brevity, we still use the notation $\mathbf{\Sigma}' \in \mathbb{R}^{2\times2}$ to denote the 2D covariance matrix.

By considering the projected 3D Gaussian center $\boldsymbol{\mu} \in \mathbb{R}^{2\times1}$ and an arbitrary point $\mathbf{x} \in \mathbb{R}^{2\times1}$ on camera coordinates, the opacity α' of $\mathbf{x}$ contributed by this 3D Gaussian can be computed as follows:

$$\alpha' = \alpha \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T(\mathbf{\Sigma}')^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \tag{17}$$

The color $\mathbf{c}$ of each Gaussian can be computed based on the view direction and its corresponding spherical harmonics (SH). Given a set of sorted 3D Gaussians $\mathcal{N}$ along the ray, we obtain the accumulated color via volume rendering:

$$\pi: \quad \mathbf{C} = \sum_{i\in\mathcal{N}} \mathbf{c}_i\alpha'_i \prod_{j=1}^{i-1}(1-\alpha'_j) \tag{18}$$

The same volume rendering technique can be applied to obtain semantic $\mathbf{S}$, depth $\mathbf{D}$ and optical flow $\mathbf{F}$. With the given semantic feature $\mathbf{s}_i$, depth value d_i, and Gaussian motion $\mathbf{f}_i$ relative to the camera pose, we can define the semantic rendering, depth rendering, and flow rendering as follows:

$$\mathbf{S} = \sum_{i\in\mathcal{N}} \text{softmax}(\mathbf{s}_i)\alpha'_i \prod_{j=1}^{i-1}(1-\alpha'_j) \tag{19}$$

$$\mathbf{D} = \sum_{i\in\mathcal{N}} d_i\alpha'_i \prod_{j=1}^{i-1}(1-\alpha'_j) \tag{20}$$

$$\mathbf{F} = \sum_{i\in\mathcal{N}} \mathbf{f}_i\alpha'_i \prod_{j=1}^{i-1}(1-\alpha'_j) \tag{21}$$

Note that all the projections and volume rendering techniques mentioned are implemented in CUDA. Calculating the projected 2D opacity α' on each pixel and sorting Gaussians based on their distances from the camera takes the majority of computations in the rendering process. These computations need to be performed only once for rendering all modalities, thus maintaining the real-time rendering property of the original 3D Gaussian Splatting.

A.2. 3D Semantic Scene Reconstruction

We utilize Eq. (19), referred to as 3D softmax, to render semantic maps. This is in contrast to most existing NeRF-based semantic reconstruction methods that perform softmax to the accumulated 2D logits [11, 52], described in Eq. (22), referred to as 2D softmax. The fundamental difference between these two rendering techniques lies in the fact that 3D softmax normalizes the logits of each 3D point. This normalization process helps prevent a single point with a significantly high logit value from imposing an overwhelming influence on the overall volume rendering outcome. On the other hand, it also prevents placing 3D points of low logit values in empty space. As a result, 3D softmax is effective in reducing floaters and enhancing the geometry of the reconstruction results. In Appendix D.3, we present a comprehensive analysis of the qualitative and quantitative comparison results between these two rendering methods.

$$\mathbf{S}_{\text{2D\_norm}} = \text{softmax}\left(\sum_{i\in\mathcal{N}} \mathbf{s}_i \alpha'_i \prod_{j=1}^{i-1}(1-\alpha'_j)\right) \tag{22}$$

In the following sections, we refer to our default setting obtained by Eq. (19) as $\mathbf{S}_{\text{3D\_norm}}$.

A.3. Metrics

Novel View Appearance Synthesis: To assess the quality of novel view appearance synthesis, we utilize the Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS) [50] following the common practice.

Novel View Semantic Synthesis: Following KITTI-360 [21], we evaluate the quality of novel view semantic synthesis via the mean Intersection over Union (mIoU) metric.

3D Semantic Reconstruction: We evaluate 3D semantic reconstruction quality by extracting a 3D semantic point cloud and comparing it with the ground truth LiDAR points. We evaluate both geometric and semantic metrics in the 3D space. Specifically, we evaluate geometric reconstruction quality by measuring the accuracy (*acc.*) and completeness (*comp.*). Accuracy measures the average distance from reconstructed points to the nearest LiDAR point, while completeness measures the average distance from LiDAR points to the nearest reconstructed points. In order to measure the semantic quality of the reconstructed point cloud, we map the predicted 3D semantics to the LiDAR points. Concretely, for each point in the LiDAR point cloud, we identify its closest counterpart in the predicted semantic point cloud and allocate a semantic label based on this nearest neighbor. The assigned semantic labels of all LiDAR points are then compared with the 3D semantic segmentation ground truth provided by KITTI-360, evaluated via the mIoU metric. Note that we only use the LiDAR point clouds for evaluation.

3D Tracking: To demonstrate the effectiveness of our model in rectifying noisy 3D tracking results, we evaluate the accuracy of predicted poses compared to ground truth poses in our ablation study. Considering the rotation and translation parameters of a ground truth bounding box denoted as $\hat{\mathbf{R}}$ and $\hat{\mathbf{t}}$, respectively, and the corresponding parameters of predicted poses, represented as $\mathbf{R}$ and $\mathbf{t}$, we employ two metrics for this evaluation following [8]: $e_{\mathbf{R}}$ quantifies the rotation accuracy, while $e_{\mathbf{t}}$ assesses the translation accuracy as follows

$$e_{\mathbf{R}} = \arccos\frac{Tr(\hat{\mathbf{R}}\cdot\mathbf{R}^{-1})-1}{2} \tag{23}$$

$$e_{\mathbf{t}} = \|\hat{\mathbf{t}}-\mathbf{t}\|_2 \tag{24}$$

where Tr represents the trace of a matrix.

Depth Estimation: In our ablation study, we evaluate the depth estimation quality of our different variants. This is achieved by first projecting the LiDAR points acquired at the same frame to the 2D image space, followed by measuring the L2 distance between the projected LiDAR depth and our method. Considering the projected LiDAR depth is sparse, our assessment focuses solely on pixels with valid LiDAR projections when calculating the L2 distance.

B. Data

In this section, we present details of datasets on which we conducted our experiments, including KITTI [13], Virtual KITTI 2 (vKITTI) [7] and KITTI-360 [21].

	Pre.	+ π RGB	+ Affine	+ π Semantic	+ π Flow
Speed (ms)	6.25	8.13 (+1.88)	8.54 (+0.41)	9.70 (+1.16)	10.17 (+0.47)

Table 6. **Time consumption breakdown** of our method.

KITTI: Following NSG [27] and MARS [40], we select frames 140 to 224 from Scene02 and frames 65 to 120 from Scene06 on KITTI for conducting our experiments.

vKITTI: Virtual KITTI 2 is a synthetic dataset that closely resembles the scenes present in KITTI. In line with the settings outlined in NSG and MARS, we conduct experiments on exactly the same frames from Scene02 and Scene06.

KITTI-360: In addition, we perform experiments on KITTI-360, encompassing both static and dynamic scenes. For the tasks of novel view synthesis and novel semantic synthesis on the leaderboard, we conduct experiments on the sequences provided by the official dataset. Furthermore, we explore dynamic scenes, such as frames 11322 to 11381 from sequence 00, as showcased in our teaser.

C. Baselines

In this section, we discuss the baselines against which we compare our approach, including NSG [27], MARS [40], PNF [19], and Semantic Nerfacto [34].

NSG: NSG is the pioneering method that introduces the decomposition of dynamic scenes into static background and dynamic foreground components. They propose a learned scene graph representation that enables efficient rendering of novel scene arrangements and viewpoints. However, the official source code provided by NSG often encounters issues when training on KITTI Scene02. Therefore, we utilize the version implemented by the authors of MARS, which is more stable and yields slightly improved results compared to the original version.

MARS: We utilize the latest version of the code provided by the official MARS repository. This latest version incorporates bug fixes and includes additional training iterations, resulting in improved performance. In fact, the updated version achieves a notable improvement of 3 to 4 dB on PSNR compared to the numbers reported in the original paper.

PNF: Since PNF is not open-source, we directly compare our method to their submission on the KITTI-360 leaderboard regarding novel view appearance & semantic synthesis. To the best of our knowledge, PNF is the only work that considers the optimization of noisy 3D bounding boxes of dynamic objects. In our ablation study, we conduct a naïve baseline that optimizes the 3D bounding boxes of each frame independently, which can be considered as a re-implementation of PNF's bounding box optimization in our framework.

Semantic Nerfacto: For the evaluation of 3D semantic point cloud geometry, we compare our results with Semantic Nerfacto [34] as an alternative to PNF [19]. Nerfacto [34] is an integration of several successful methods that demonstrate strong performance on real data. It incorporates camera pose refinement, per-image appearance embedding, proposal sampling, scene contraction, and hash encoding within its pipeline. Additionally, Nerfacto includes a semantic head in its framework, enabling the generation of meaningful semantic maps, as demonstrated in Appendix D.2.

D. Additional Experiment Results

D.1. Time Consumption Breakdown

Tab. 6 shows our detailed runtime breakdown as various components are incrementally enabled. Preparation (Pre.) contains operations like tile partition and Gaussian sorting. π denotes volume rendering, and affine denotes affine transform. Other components like unicycle model, dynamic decomposition, and depth rendering are excluded as they hardly consume any additional time.

D.2. Additional Comparison Experiments

Dynamic Scene with GT 3D Bounding Boxes: Despite not being our primary focus, we additionally provide a comparison with NSG and MARS using ground truth 3D trackings. In this setting, our approach demonstrates superior performance across all test scenes, see Tab. 7.

	KITTI Scene02			KITTI Scene06			vKITTI Scene02			vKITTI Scene06		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
NSG [27]	22.51	0.653	0.397	23.38	0.717	0.243	23.50	0.718	0.352	26.42	0.811	0.170
MARS [40]	22.95	0.728	0.145	27.01	0.883	0.062	29.80	0.950	0.034	32.71	0.959	0.023
Ours	**25.89**	**0.829**	**0.092**	**28.90**	**0.925**	**0.016**	**30.73**	**0.955**	**0.018**	**33.31**	**0.963**	**0.010**

Table 7. **Novel View Appearance on Dynamic Scenes** with ground truth 3D trackings.

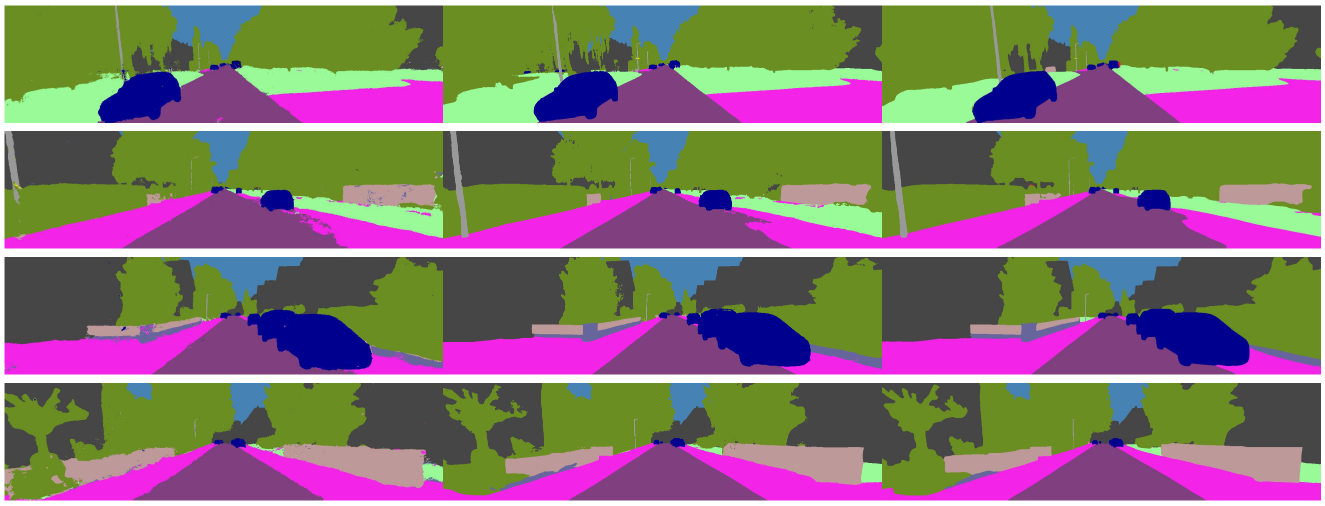

Figure 10. **Qualitative Comparison** with Nerfacto on 2D space. The Pseudo GT column represents the semantic maps that are predicted by [6] on GT RGB images.

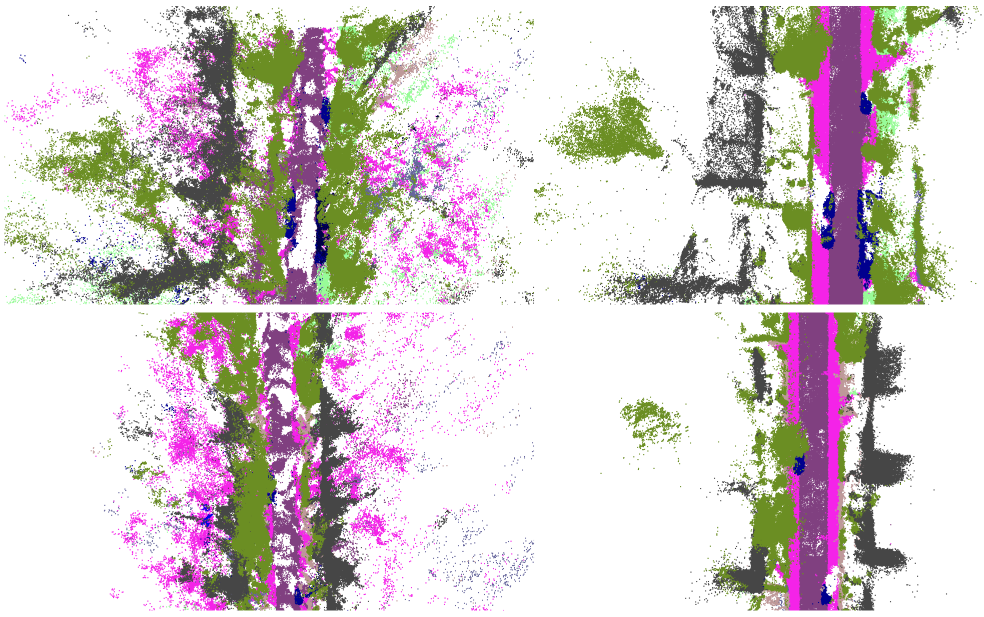

Figure 11. **Qualitative Comparison** with Nerfacto on 3D space. The semantic point cloud extracted from Semantic Nerfacto struggles to faithfully represent the geometry.

		$e_{\mathbf{R}} \downarrow$	$e_{\mathbf{t}} \downarrow$
KITTI 02	QD-3DT	0.027	0.215
	Ours	**0.018**	**0.108**
KITTI 06	QD-3DT	0.017	0.046
	Ours	**0.012**	**0.033**

Table 8. **Qualitative Comparison** with a tracking method, QD-3DT [16], on two sequences.

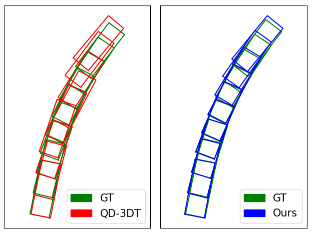

Figure 12. **Pose comparison** with QD-3DT.

	Seq01 $\text{mIoU}_{\text{cls}} \uparrow$	Seq02 $\text{mIoU}_{\text{cls}} \uparrow$	Seq03 $\text{mIoU}_{\text{cls}} \uparrow$	Average $\text{mIoU}_{\text{cls}} \uparrow$
Ours w/ $\mathbf{S}_{\text{2D\_norm}}$	0.427	0.363	0.416	0.402
Ours w/ $\mathbf{S}_{\text{3D\_norm}}$	**0.544**	**0.452**	**0.520**	**0.505**

Table 9. **Comparison on 3D and 2D Semantic Softmax** on KITTI-360.

	KITTI-360 Scene00			KITTI-360 Scene01			KITTI-360 Scene02			Average		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Random	20.84	0.784	0.150	19.40	0.705	0.171	22.55	0.800	0.136	20.93	0.763	0.457
LiDAR	25.64	0.856	0.070	22.88	0.784	**0.089**	24.04	0.836	0.080	24.19	0.825	**0.080**
COLMAP	**26.23**	**0.863**	**0.069**	**22.94**	**0.794**	0.096	**24.38**	**0.843**	**0.077**	**24.52**	**0.833**	0.081

Table 10. **Quantitative Comparison** with different initialization.

Details of Comparison with Semantic Nerfacto: While Semantic Nerfacto excels at rendering meaningful novel view semantic images (as seen in Fig. 10), Fig. 11 shows it struggling to accurately reconstruct correct geometry. Following the common practice of NeRF-based semantic reconstruction methods [34], we apply 2D softmax to Semantic Nerfacto. when we attempted to apply the 3D Softmax technique to Nerfacto, it did not yield better results compared to using 2D softmax. The results can be attributed to the incorrect of Nerfacto's 3D geometry. Instead of adjusting 2D logits with large-scale logits in 3D, the use of 3D softmax prevents the "cheating" approach by normalizing logits in 3D space. However, this normalization requirement necessitates sufficiently accurate geometry for satisfactory results.

Comparisons with Tracking Methods: To further compare with off-the-shelf tracking methods, we show the performance of QD-3DT [16] and our optimized pose initialized with [16] in Tab. 8 and qualitatively illustrate the poses of one vehicle in Fig. 12. Our method consistently improves [16] across two KITTI scenes.

D.3. Additional Ablation Experiments

3D and 2D Semantic Softmax: We provide more 3D and 2D semantic logits softmax comparison in Fig. 13 and Tab. 9. As can be seen, normalizing semantic logits in 3D space leads to notable qualitative and quantitative improvement compared to 2D space normalization.

Improvements on Geometry: We now qualitatively examine how the optical flow loss $\mathcal{L}_{\mathbf{F}}$ and the semantic loss $\mathcal{L}_{\mathbf{S}}$ impact the geometry, as shown in Fig. 14 and Fig. 15. Both figures reveal that incorporating either the semantic loss or the optical flow loss improves the underlying geometry. While the impact of the semantic loss on geometry may be less evident, the optical flow clearly enhances geometric accuracy. This improvement is rationalized by the fact that optical flow guides correspondences across neighboring frames. It's important to note that when the semantic loss $\mathcal{L}_{\mathbf{S}}$ is active, the sky region of the depth maps in Fig. 14 is set to infinite.

Effects of Initialization: We conduct a thorough comparison of the results obtained through different initialization strategies. In particular, we consider random initialization and COLMAP-based initialization. To further investigate whether adopting LiDAR point cloud for initialization is helpful in urban scenes, we further consider LiDAR point clouds as initialization. We report the quantitative and qualitative comparison in Tab. 10 and Fig. 16, respectively. We observe that both LiDAR and COLMAP initialization outperform random initialization. Interestingly, the COLMAP-based initialization even shows a slight advantage over the LiDAR-based one. This could be attributed to the presence of points in the LiDAR

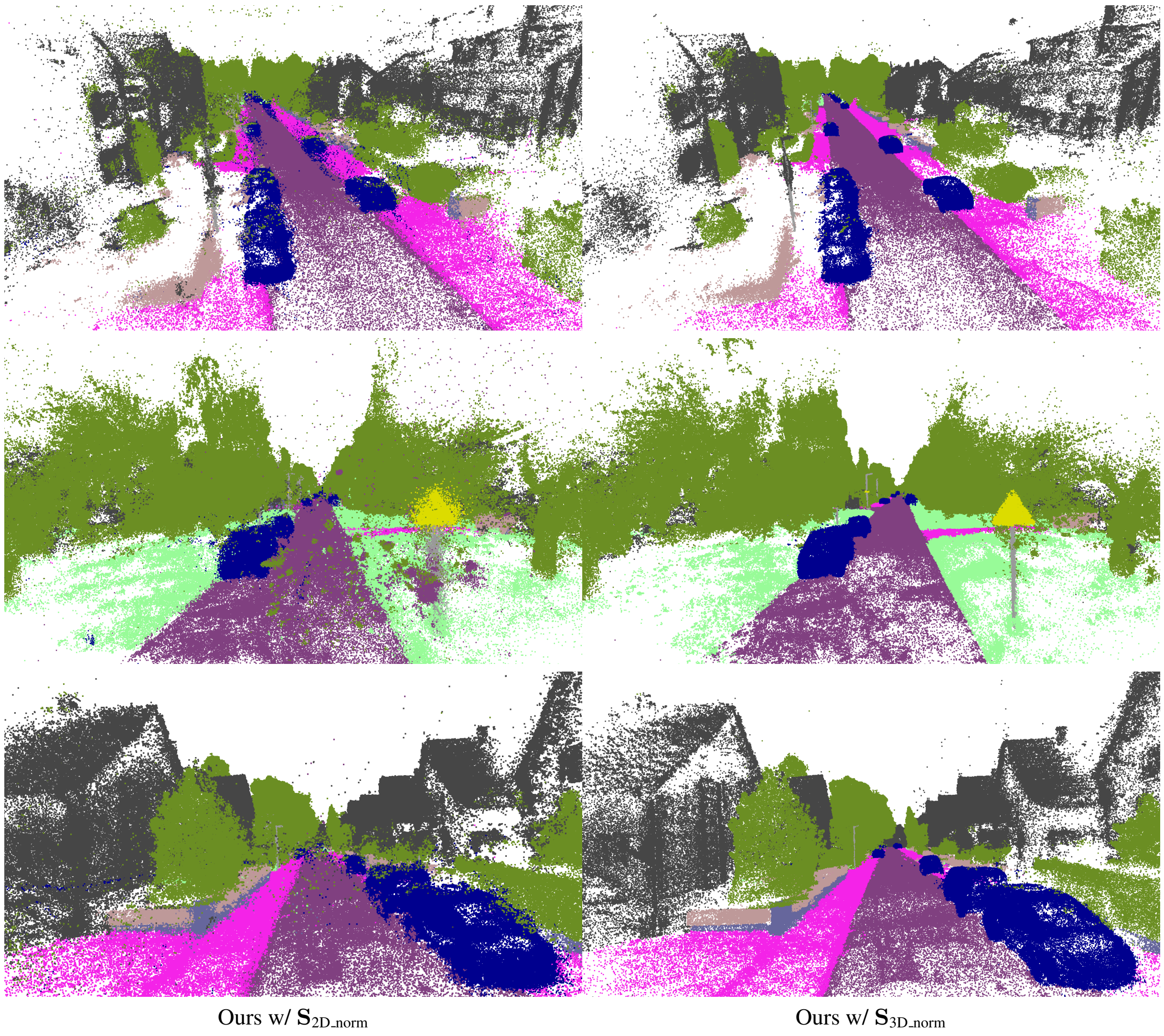
Ours w/ $\mathbf{S}_{\text{2D\_norm}}$
Ours w/ $\mathbf{S}_{\text{3D\_norm}}$

Figure 13. **Qualitative Comparison** of 3D and 2D softmax results. Note that normalizing semantic logits in 3D space (Ours w/ $\mathbf{S}_{\text{3D\_norm}}$) clearly reduces floaters and yields better 3D semantic reconstruction than the 2D normalization counterpart (Ours w/ $\mathbf{S}_{\text{2D\_norm}}$).

point clouds that remain unobserved in any training views, leading to artifacts in test viewpoints. Furthermore, COLMAP improves the quality of objects located at far distances, which cannot be accurately captured by LiDAR. These findings underscore the potential for achieving high-fidelity novel view synthesis in urban scenes based solely on RGB images. In our main experiments, we adopt the COLMAP-based initialization by default.

D.4. Visualization of Optimization Progress

We present the visualization of the optimization progress for both the noisy bounding boxes and the background semantic point cloud in Fig. 17. Using noisy 3D bounding boxes as input, our approach optimizes both the background and the poses of the bounding boxes simultaneously. As evident, the application of physical constraints derived from the unicycle model results in a smooth trajectory for the bounding boxes.

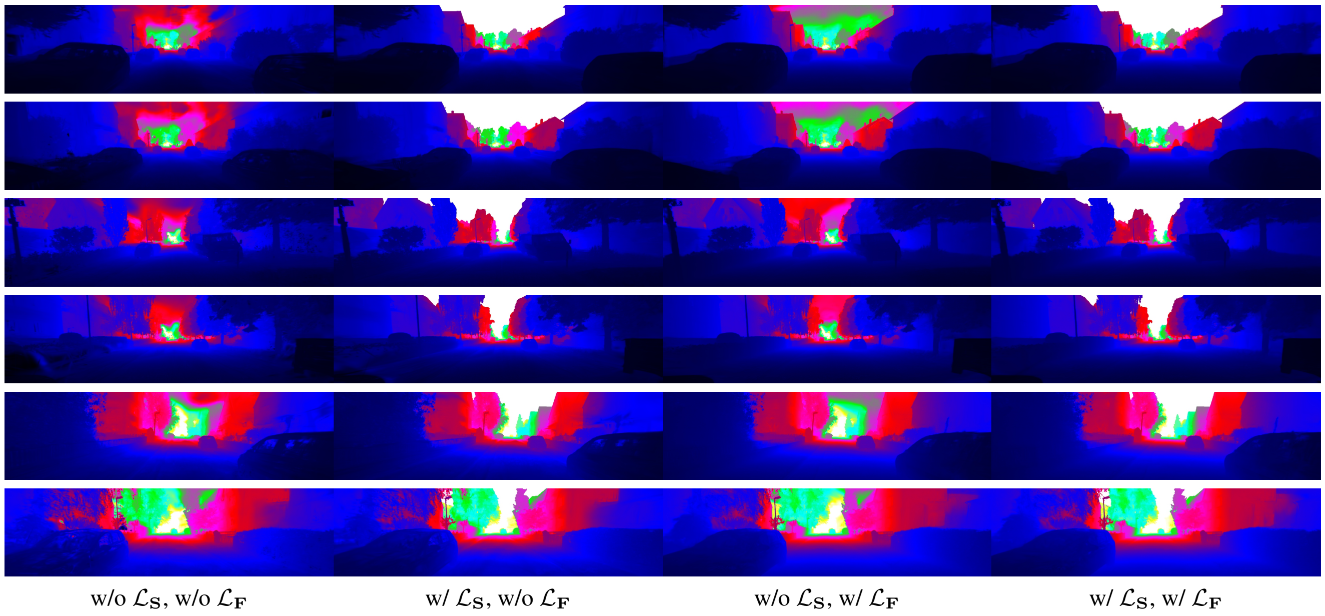
w/o $\mathcal{L}_\mathbf{S}$, w/o $\mathcal{L}_\mathbf{F}$
w/ $\mathcal{L}_\mathbf{S}$, w/o $\mathcal{L}_\mathbf{F}$
w/o $\mathcal{L}_\mathbf{S}$, w/ $\mathcal{L}_\mathbf{F}$
w/ $\mathcal{L}_\mathbf{S}$, w/ $\mathcal{L}_\mathbf{F}$

Figure 14. **Qualitative Comparison** on depth. In the presence of the semantic loss $\mathcal{L}_\mathbf{S}$ (2nd and 4th columns), we set the sky region's depth infinite based on its semantic label. Note that the activation of either the semantic loss $\mathcal{L}_\mathbf{S}$ (2nd column) or the optical loss $\mathcal{L}_\mathbf{F}$ (3rd column) yields enhancements in geometry, e.g., the left car in the bottom row, with the improvement in optical flow loss being more evident.

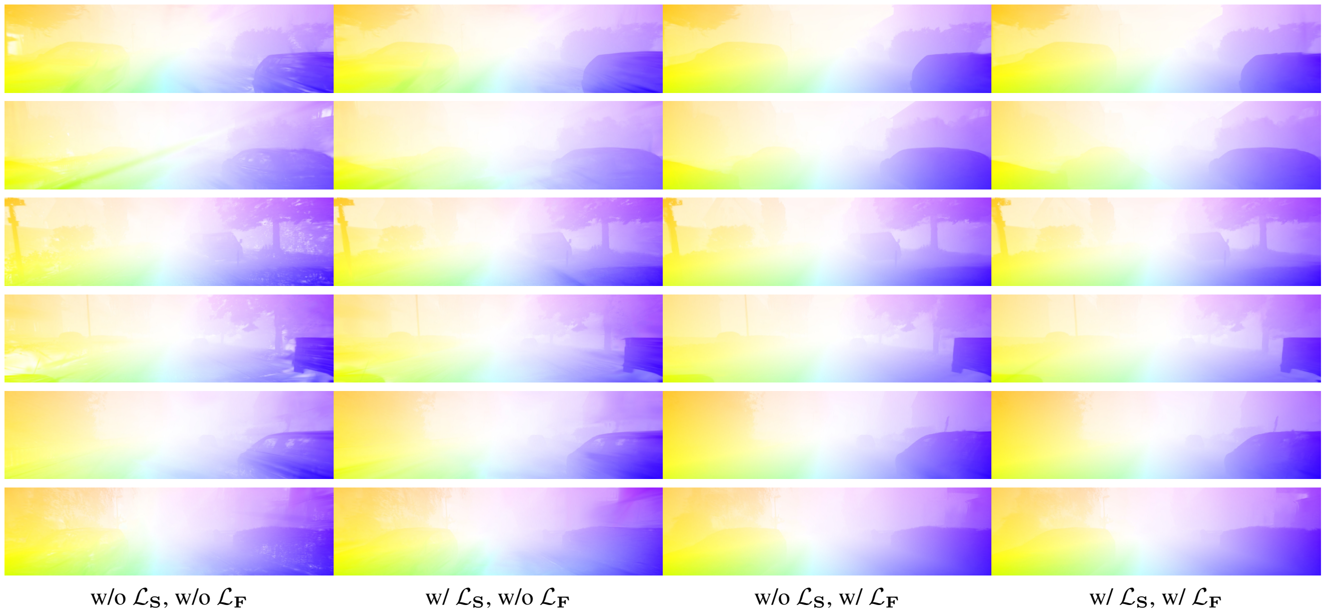
w/o $\mathcal{L}_\mathbf{S}$, w/o $\mathcal{L}_\mathbf{F}$
w/ $\mathcal{L}_\mathbf{S}$, w/o $\mathcal{L}_\mathbf{F}$
w/o $\mathcal{L}_\mathbf{S}$, w/ $\mathcal{L}_\mathbf{F}$
w/ $\mathcal{L}_\mathbf{S}$, w/ $\mathcal{L}_\mathbf{F}$

Figure 15. **Qualitative Comparison** on optical flow. While 3D Gaussians can enable the rendering of optical flow without additional supervision on semantic or optical flow, the rendered flow maps exhibit clear artifacts (1st column). These artifacts are particularly noticeable on the cars and the ground. Interestingly, the incorporation of semantic supervision $\mathcal{L}_\mathbf{S}$ mitigates the artifacts to some extent (2nd column). Additionally, introducing pseudo-optical flow supervision $\mathcal{L}_\mathbf{F}$ contributes to further improvement in the optical flow results (3rd and 4th columns).

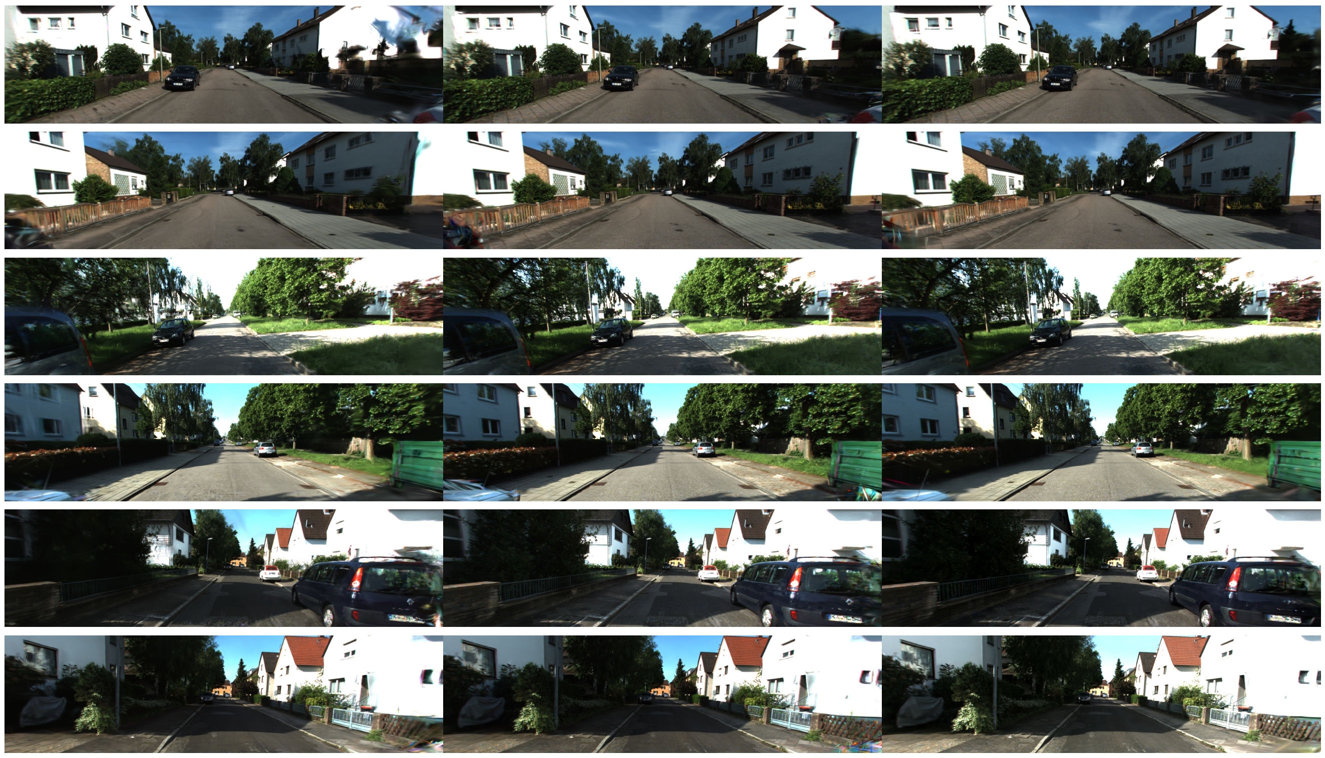

Figure 16. **Qualitative Comparison** with different initialization strategies. The superiority of both LiDAR-based and COLMAP-based initialization over random initialization is evident. Random initialization occasionally results in significant artifacts, as illustrated by the right building in the 1st row. LiDAR-based initialization, while generally effective, introduces artifacts in areas very close to the ego car, such as the bottom right corner of the 4th-6th rows. These regions typically encompass LiDAR points unseen by any training views. The COLMAP-based initialization further demonstrates an improvement over the LiDAR-based approach in distant regions, exemplified by the trees in the 1st row.

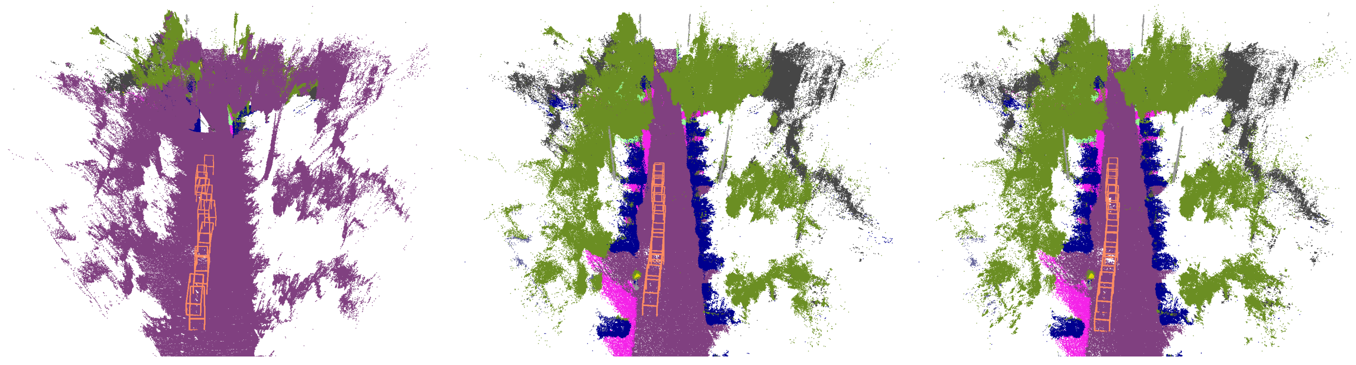

Figure 17. **Visualization of Optimization Progress**. Our method jointly optimizes the static background and the trajectory of the dynamic foreground objects. By integrating physical constraints using the unicycle model, our method allows for recovering a smooth trajectory from noisy 3D bounding boxes. To prevent visual clutter, we exclude point clouds of the dynamic object and only visualize the bounding boxes.