

CoRoGS: Contextual Gaussian Splatting for Robust Large-Deviation View Synthesis

Supplementary Material

This supplementary material is organized into two main sections. Sec. A outlines the technical details and implementation of our method. Sec. B provides additional qualitative results and quantitative evaluations to further illustrate the effectiveness of our method. Rendered video is also included for a more comprehensive comparison.

A. Technical Details and Implementation

A.1. Implementation Details

We conduct a total of 30,000 iterations for all scenes. Specifically, we first perform the initial training adapted until 1,500 iterations, after which a scheduled graph expansion strategy is employed every 100 iterations until iteration 15,000. We employ a 2-layer MLP with a hidden dimension of 32 to learn the geometric and semantic embedding, where Fourier positional encoding with 6 and 10 frequency orders are applied to the position and normal inputs respectively. The geometric and semantic update operations are repeated $L = 2$ times. The influence of boundary guidance in Gaussian graph expansion is controlled by λ_b , which is set to 0.2. The number of Gaussian candidates predicted by each fused feature is set to $N = 3$. The predefined threshold τ_g is set to 2×10^{-4} to control the growth of new Gaussian nodes and their corresponding edges. The weights balancing the each loss components in loss function are set as $\lambda_D = 0.8, \lambda_N = 0.5, \lambda_S = 0.2$ and $\lambda_C = 0.5$. All experiments are performed on a single NVIDIA RTX A6000 based on the Pytorch framework.

A.2. 3D Gaussian Graph Initialization

We utilize SfM to estimate camera poses and a sparse point cloud, followed by dense reconstruction via conventional Multi-View Stereo (MVS). Delaunay Triangulation is then applied to generate a triangular mesh with connectivity between points. We derive triangular facet normals via vector cross-products of mesh edges, followed by interpolation to obtain vertex normals. These vertices and normals then serve as the initial positions and normals for our 3D Gaussian graph \mathcal{G} .

A.3. Computation of the Semantic Maps

Each fused feature $z_i^o \in \mathbb{R}^{32}$ is decoded into a 16-D semantic vector via an MLP, which then replaces the RGB color in the 3DGS alpha blending pipeline to synthesize semantic maps. The channel dimension of the semantic maps is configured to align with the segmentation model. We set this

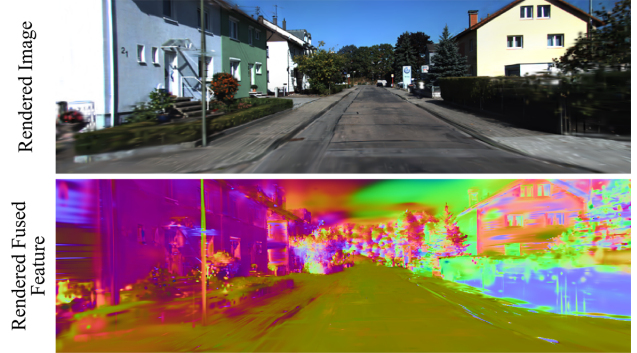


Figure 1. Visualization of the fused feature following cross-modal fusion.

dimension 16 to match the category count of LSeg, and it can be replaced by other advanced foundation models (e.g., DINO).

A.4. Visualization of Contextual Feature

We present additional visualizations of the fused context features. Through the proposed adaptive cross-modal fusion, the final output representation z_i^o for each Gaussian node achieves a balanced encoding of fine-grained geometry and semantics. As illustrated in Fig. 1, we visualize the fused features z_i^o produced by our Gaussian graph neural network. The resulting contextual feature exhibits coherent spatial structure while maintaining clear semantic discrimination across scene boundaries.

A.5. Training Process Analysis

We report the variations in PSNR and the number of Gaussian nodes during the training process. Our method demonstrates quicker convergence, enhanced robustness, and better generalization compared to baselines. As shown in Fig. 2 (Right), we achieve the highest PSNR within the same number of training iterations, outperforming DeSiRe-GS and SAGS by 2.25 dB and 4.45 dB, respectively. Meanwhile, as shown in Fig. 2 (Left), the number of Gaussians in these alternative methods increases much faster during training, with DeSiRe-GS and SAGS requiring approximately 1.36× and 1.20× more Gaussians than ours. In contrast, our approach maintains controllable Gaussians and achieves superior rendering performance with similar training steps. This improvement primarily stems from the proposed context-aware framework, which enforces consistent constraints among neighboring Gaussians.

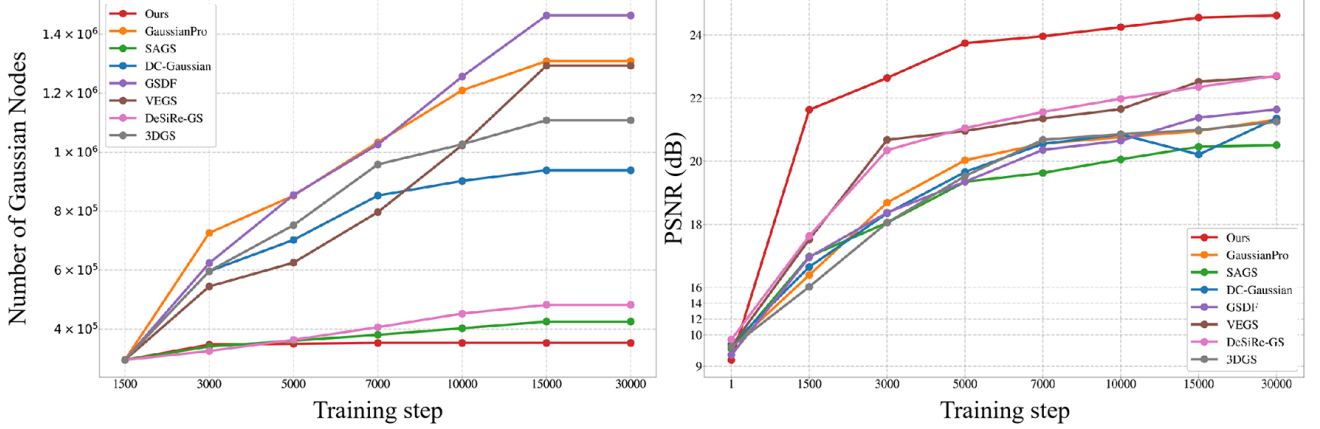


Figure 2. Comparisons on Gaussian node expansion (left) and PSNR (right) across training iterations between our method and the baselines, showing that our method exhibits more stable convergence and achieves superior rendering quality.

A.6. Graph Expansion Module

To address the structural incompleteness introduced by MVS-based initialization, we use a Gaussian Graph Expansion mechanism that adaptively densifies the graph in poorly reconstructed regions. The key idea is to monitor boundary Gaussians during training and selectively introduce new Gaussians in regions where the rendering process indicates missing structure. Furthermore, to prevent excessive growth and ensure spatial regularity, the redundant Gaussians are pruned based on inter-node distance constraints.

We introduce a boundary indicator to identify boundary Gaussians during training. A Gaussian is marked as a boundary node if any edge incident to v_i is associated with only one triangle. Such nodes typically indicate structural discontinuities and are more prone to geometric incompleteness. To compensate for this, we amplify their gradient growth signal by introducing a boundary modulation term.

Specifically, the growth criterion ε_g for Gaussians is defined as the sum of the average Gaussian gradient ∇_g and a boundary modulation term, formulated as:

$$\varepsilon_g = \nabla_g + \lambda_b \beta(p_i) \nabla_g, \quad (1)$$

where $\beta(p_i) \in \{0, 1\}$ denotes the boundary indicator and λ_b controls the degree of boundary enhancement. New nodes are generated when ε_g exceeds a predefined threshold τ_g , forming an initial nodes set denoted as \mathcal{P} .

To ensure a valid triangular topology during graph expansion, each accepted Gaussian $p_{\text{new}} \in \mathcal{P}$ is required to lie outside existing triangles and is inserted into the local neighborhood of its anchor node. The new Gaussian is then connected with the anchor and its adjacent nodes to form valid triangles. All integrated Gaussians inherit their attributes from their corresponding anchors, preserving con-

sistency in geometry and appearance throughout the expansion process.

To maintain spatial regularity, newly generated Gaussians are constrained by the inter-node distance defined by the initial Delaunay triangulation. Nodes that violate these bounds are removed to prevent over-expansion or structural distortion.

Specifically, each node p_{new} is associated with a spatial anchor point p_m , and the edge connecting the new node to this anchor has length

$$d_{mn} = \|p_m - p_{\text{new}}\|_2, \quad (2)$$

which is accepted only if $d_{mn} \leq d_{\text{max}}$, where d_{max} denotes the maximum edge length between p_m and its neighboring nodes. The nodes and corresponding edges that fail to satisfy these distance constraints are discarded.

B. More Experiments and Results

B.1. Additional Ablation Studies

Adjacency Construction Strategies between Gaussians.

We compare two strategies for defining adjacency between Gaussians in our 3D Gaussian graph construction. Specifically, we define Gaussian adjacency through Delaunay triangulation, while *Ours+KNN Graph* denotes a variant that connects each Gaussian to its k -nearest neighbors based on spatial distance. As shown in Tab. 1, using a KNN graph leads to a 2.93 dB drop in PSNR, indicating degraded rendering quality. This degradation mainly arises because the KNN graph relies only on Euclidean distance, often linking Gaussians from different semantic regions or with distinct surface normals. Such connections misguide feature aggregation, leading the network to infer false similarities between geometrically or semantically distinct Gaussians. Moreover, KNN enforces each Gaussian to connect with a

Table 1. Ablation studies on the KITTI dataset. Metrics are averaged over all the sequences on the KITTI dataset.

Description	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	FID \downarrow
Ours + KNN Graph	22.03	0.7159	0.2008	90.36
Ours	24.96	0.8492	0.1805	67.12

Table 2. Ablation on depth of Gaussian graph neural network.

#Layers	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	FID \downarrow	#Params (MB)
1	24.32	0.8165	0.1828	79.68	0.45
2	24.96	0.8492	0.1805	67.12	0.96
3	25.28	0.8529	0.1796	73.25	1.42

fixed number of K neighbors (about $K \times N$ edges). In contrast, Delaunay-based graph construction removes duplicate or invalid connections, yielding only about $1.5 \times N$ edges while maintaining effective feature propagation.

Depth of Gaussian Graph Neural Network. We evaluate the impact of varying the number of layers L in our Gaussian graph neural network module. As summarized in Tab. 2, increasing the depth from 1 to 2 layers improves PSNR by 0.64 dB, and reduces LPIPS by 0.0023 and FID by 12.56. These improvements indicate that adding a second layer substantially enhances the ability to aggregate both local and global structural information, resulting in more effective aggregation. In contrast, increasing the depth from 2 to 3 layers yields only marginal gains in PSNR and SSIM, suggesting limited benefit and potential overfitting. In addition, the model size increases considerably with three layers, further highlighting the trade-off between performance and computational cost. Overall, these results demonstrate that a 2-layer GNN achieves the most favorable balance between reconstruction quality and efficiency, motivating its selection for experiments.

B.2. Detailed Quantitative Comparisons on Each Dataset

In the main paper, we report the average performance of our method on KITTI and Waymo open dataset. In this supplementary material, Tab. 3 and Tab. 4 provide detailed comparisons of our method under different deviation settings for each dataset. The results show that our method achieves the best performance across all settings.

B.3. Additional Qualitative Comparisons

We provide additional qualitative results to demonstrate the effectiveness of our method under various viewpoint deviations. Specifically, Figs. 3 to 5 show representative results on the KITTI dataset, while Figs. 6 to 8 correspond to the Waymo Open dataset. Across all tested scenarios, baseline

methods exhibit noticeable geometric artifacts, including distorted road surfaces and degraded details of vehicles and buildings, particularly under large viewpoint shifts. In contrast, our framework consistently produces high-quality renderings, effectively preserving geometric consistency and maintaining both the global scene structure and fine-grained object details.

Table 3. Quantitative results under unsupervised large-deviation settings on the KITTI dataset.

Setting		KITTI-Left-3m		KITTI-Up-1m		KITTI-Diagonal-3m	
Method	Metric	KID↓	FID↓	KID↓	FID↓	KID↓	FID↓
GaussianPro		126.11	0.0546	96.53	0.0459	139.80	0.0779
SAGS		113.27	0.0679	116.06	0.0662	136.71	0.0802
DC-Gaussian		121.20	0.0518	95.88	0.0437	139.70	0.0770
GSDF		119.32	0.0506	97.52	0.0513	148.44	0.0834
VEGS		120.96	0.0591	110.88	0.0537	139.92	0.0805
DeSiRe-GS		106.71	0.0441	99.70	0.0403	149.36	0.0823
Ours		68.25	0.0263	74.21	0.0349	109.69	0.0370

Setting		KITTI-Left-5m		KITTI-Up-2m		KITTI-Diagonal-5m	
Method	Metric	KID↓	FID↓	KID↓	FID↓	KID↓	FID↓
GaussianPro		135.46	0.0944	163.69	0.1357	179.46	0.1258
SAGS		147.96	0.0696	173.26	0.1463	172.63	0.1210
DC-Gaussian		133.00	0.0907	162.35	0.1632	164.49	0.1184
GSDF		137.93	0.0958	161.25	0.1645	173.29	0.1046
VEGS		137.28	0.0978	164.58	0.1597	178.52	0.1134
DeSiRe-GS		130.31	0.0856	169.53	0.1546	175.59	0.1084
Ours		88.22	0.0375	118.70	0.0789	126.38	0.0853

Table 4. Quantitative results under unsupervised large-deviation settings on the Waymo Open dataset.

Setting		Waymo-Left-3m		Waymo-Up-1m		Waymo-Diagonal-3m	
Method	Metric	KID↓	FID↓	KID↓	FID↓	KID↓	FID↓
GaussianPro		113.87	0.0586	101.79	0.0546	128.65	0.0802
SAGS		98.26	0.0582	98.25	0.0592	129.35	0.0812
DC-Gaussian		109.76	0.0511	94.02	0.0452	132.16	0.0803
GSDF		110.85	0.0506	94.55	0.0468	142.32	0.0823
VEGS		117.16	0.0811	106.01	0.0735	130.26	0.0796
DeSiRe-GS		102.87	0.0519	104.81	0.0554	146.37	0.0815
Ours		65.99	0.0235	71.21	0.0316	105.85	0.0456

Setting		Waymo-Left-5m		Waymo-Up-2m		Waymo-Diagonal-5m	
Method	Metric	KID↓	FID↓	KID↓	FID↓	KID↓	FID↓
GaussianPro		143.31	0.1124	171.29	0.1645	180.59	0.1242
SAGS		119.71	0.0942	178.62	0.1623	179.62	0.1150
DC-Gaussian		132.04	0.0915	170.49	0.1610	173.69	0.1201
GSDF		152.90	0.1308	169.46	0.1649	172.63	0.1153
VEGS		148.88	0.1201	173.49	0.1678	168.95	0.1132
DeSiRe-GS		156.25	0.1063	174.75	0.1637	174.59	0.1149
Ours		90.07	0.0380	119.69	0.0955	140.63	0.1190

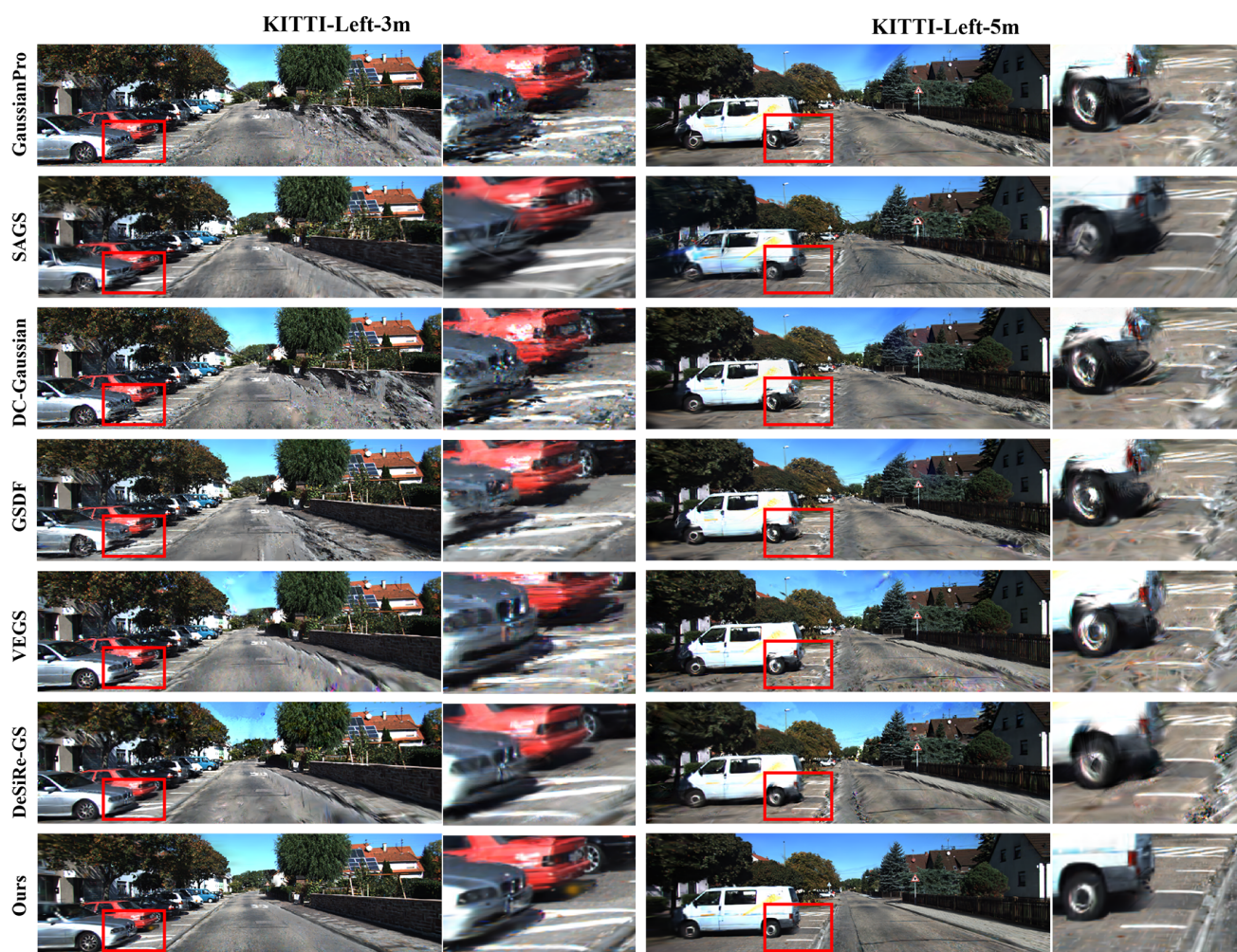


Figure 3. Qualitative results on the KITTI dataset under lateral left shift of 3m and 5m. Our method achieves higher visual quality than baseline approaches, consistent with the quantitative results in Tab. 3.

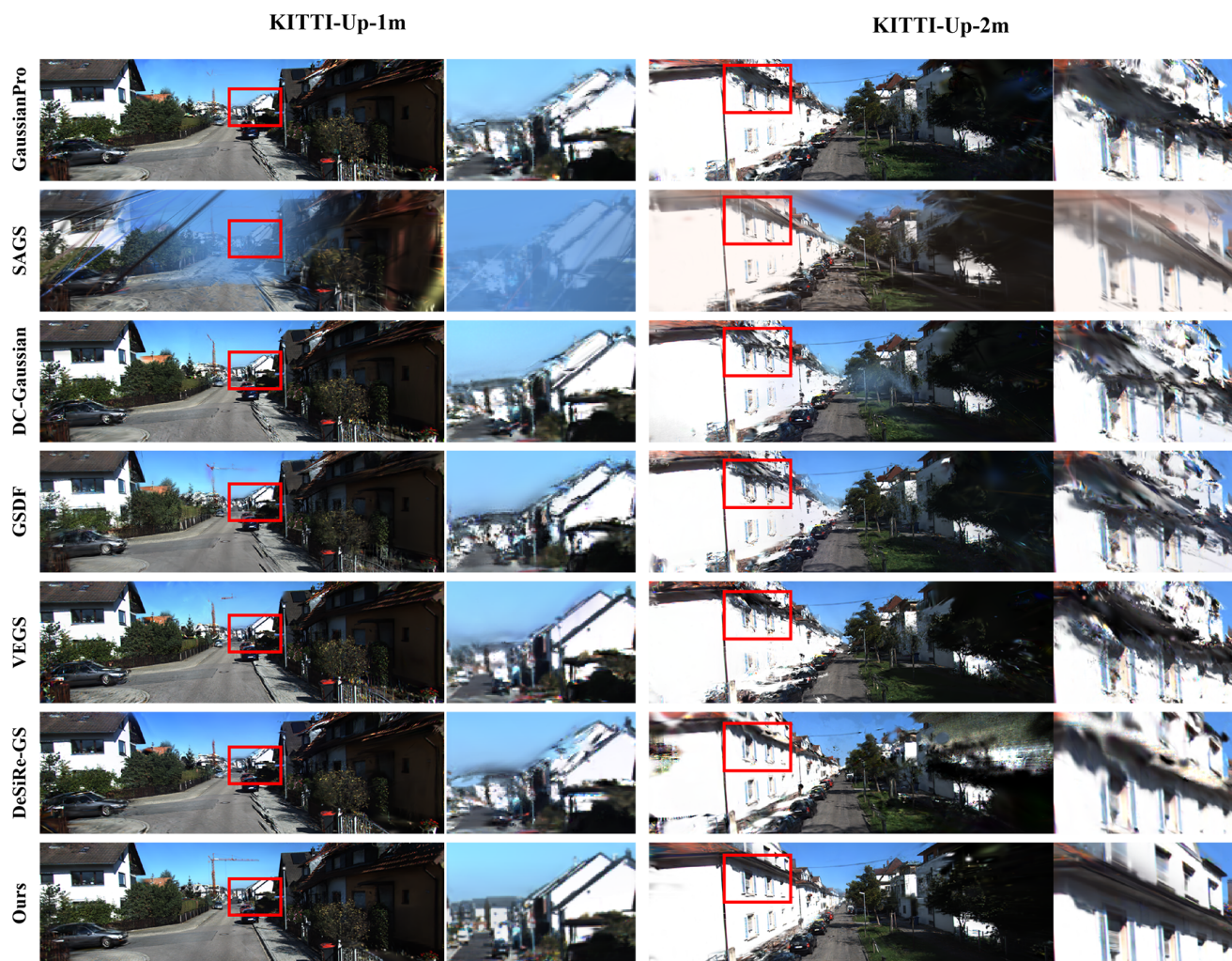


Figure 4. Qualitative results on the KITTI dataset under vertical shifts of 1m or 2m upward. Our method achieves higher visual quality than baseline approaches, consistent with the quantitative results in Tab. 3

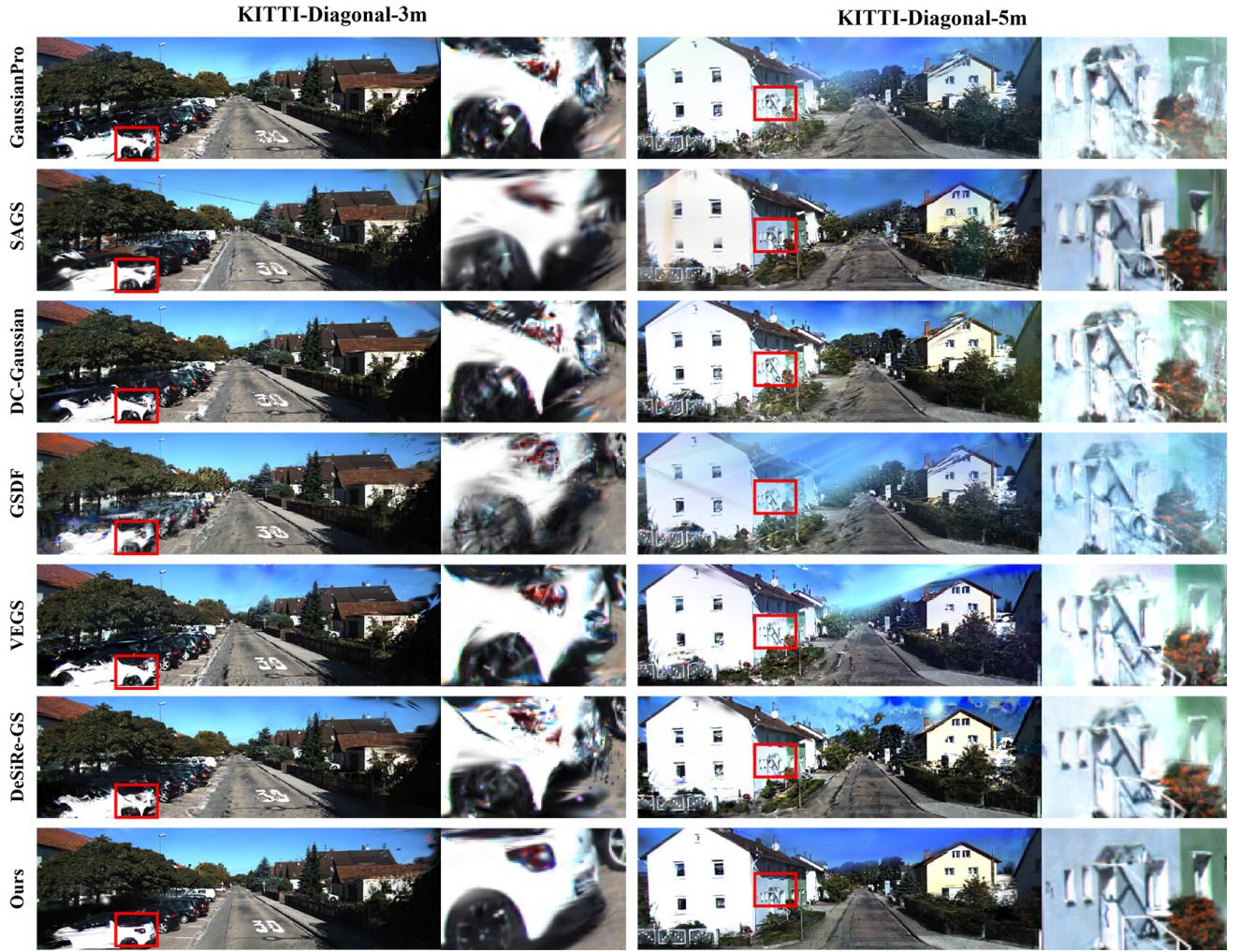


Figure 5. Qualitative results on the KITTI dataset under combined shift of left and upward. Our method achieves higher visual quality than baseline approaches, consistent with the quantitative results in Tab. 3

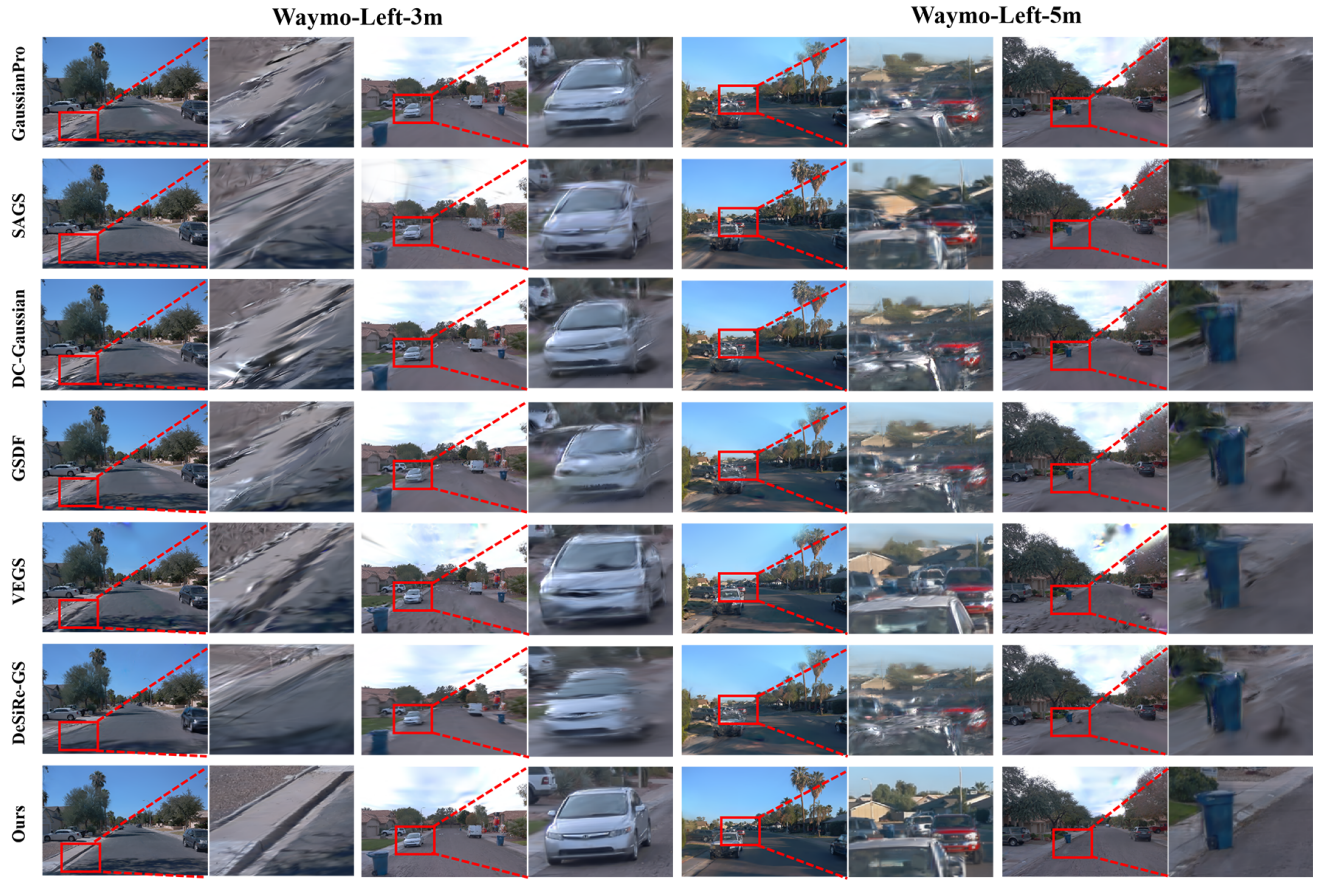


Figure 6. Qualitative comparisons on the Waymo open dataset under lateral left shift of 3m and 5m. Zoomed-in image patches demonstrate the superior rendering quality of our method compared to competing baselines, consistent with the quantitative results in Tab. 4

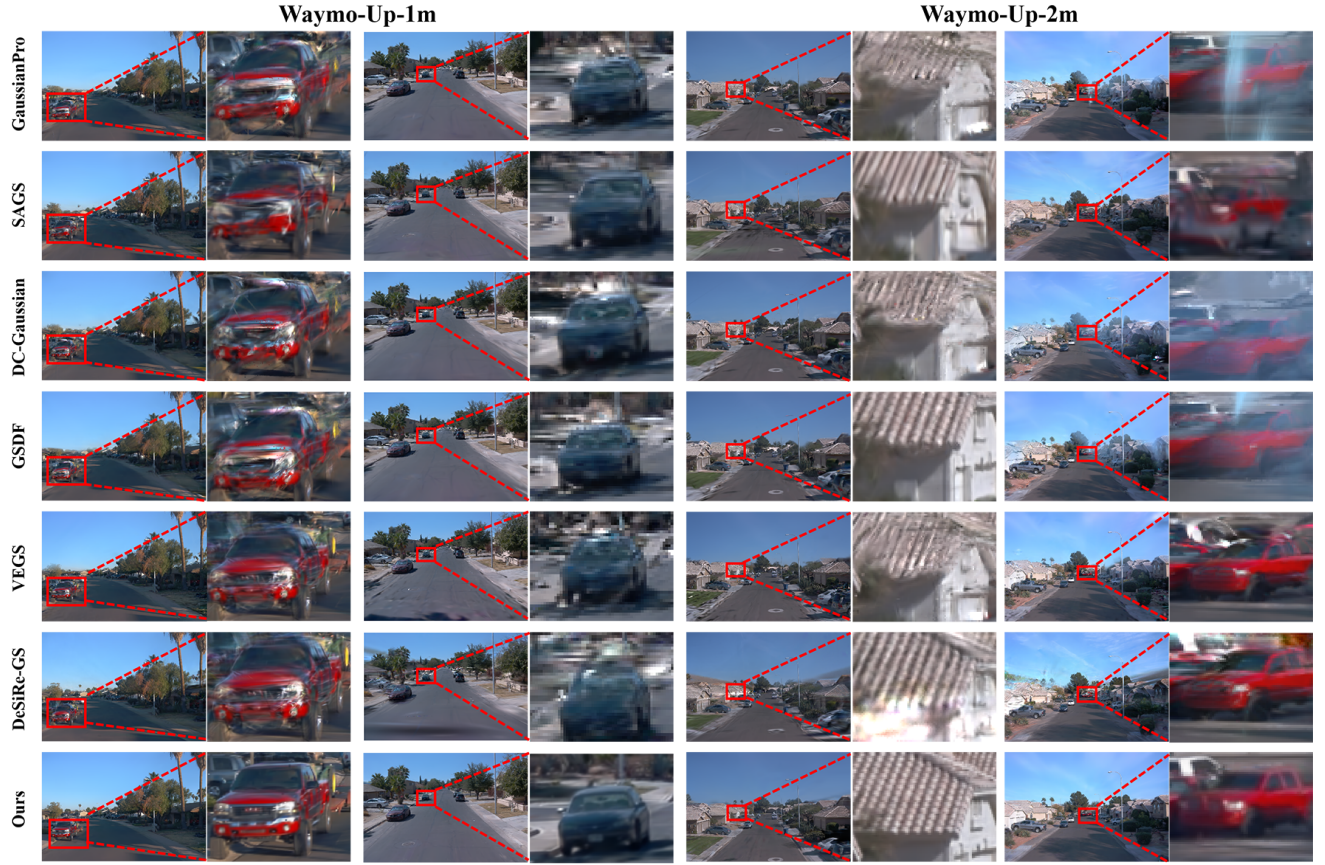


Figure 7. Qualitative comparisons on the Waymo open dataset under vertical shifts of 1m or 2m upward. Zoomed-in image patches demonstrate the superior rendering quality of our method compared to competing baselines, consistent with the quantitative results in Tab. 4



Figure 8. Qualitative comparisons on the Waymo open dataset under combined shift of left and upward. Zoomed-in image patches demonstrate the superior rendering quality of our method compared to competing baselines, consistent with the quantitative results in Tab. 4