LiDAR Prompted Spatio-Temporal Multi-View Stereo for Autonomous Driving

Supplementary Material

In this document, we further provide the following materials to support the statements and conclusions drawn in the main body of this paper. Please find more video results in our *supplementary video*.

- Sec. A: Model and Trining Details;
- Sec. B: Implementation Details;
- Sec. C: More Discussions of Experiments.

A. Model and Training Details

A.1. Model Architecture

Sparsity-aware Prompt Encoder. To effectively leverage sparse LiDAR points as high-fidelity metric prompts, we introduce a Sparsity-aware Prompt Encoder. A standard CNN encoder with strided convolutions or average pooling is ill-suited for this task, as it tends to dilute valid sparse signals with zeros from empty regions, leading to signal loss and feature blurring. Therefore, our prompt encoder is specifically designed to preserve the integrity of sparse signals during the downsampling process while aligning them with the network's token space.

Specifically, given a raw sparse metric prompt with corresponding valid mask $\mathbf{P} \in \mathbb{R}^{2\times H\times W}$, we first transform it into the logit space to match the network's prediction target. To incorporate spatial awareness, we concatenate normalized coordinate grids $(\mathbf{Y}, \mathbf{X})$ to the input, resulting in an augmented input feature $\mathbf{X}_{in} \in \mathbb{R}^{(2+2)\times H\times W}$. Then $\mathbf{X}_{in}$ is processed by a stem block followed by three downsampling stages. To prevent signal degradation, we replace standard pooling layers with a custom Masked Max-Pooling operation. Let $\mathbf{F}_l$ be the feature map and $\mathbf{M}_l$ be the binary validity mask at stage l. The downsampling process is defined as:

$$\mathbf{F}_{l+1}, \mathbf{M}_{l+1} = \text{MaskedMaxPool}(\text{ConvBlock}(\mathbf{F}_l), \mathbf{M}_l), \tag{10}$$

where MaskedMaxPool performs max pooling only on valid pixels (where $\mathbf{M}_l = 1$). Specifically, invalid positions in the pooling window are masked with $-\infty$ before the max operation, ensuring that even a single valid signal within a window is propagated to the next resolution. After four processing stages, the feature map is downsampled to $1/16$ of its original resolution. A 1×1 convolution projects the features to the embedding dimension D. We flatten the spatial dimensions to obtain a sequence of tokens $\mathbf{F}_{metric} \in \mathbb{R}^{N\times D}$. To retain spatial context after flattening, we add sinusoidal positional embeddings of 2D absolute positions to the tokens. Finally, to ensure the subsequent Transformer layers do not attend to empty regions, we generate a token-level attention mask $\mathbf{M}_{attn}$ derived from the final downsampled validity mask. The output features $\mathbf{F}_{metric}$ are masked, setting invalid regions to zero, forcing the network to rely solely on valid metric cues.

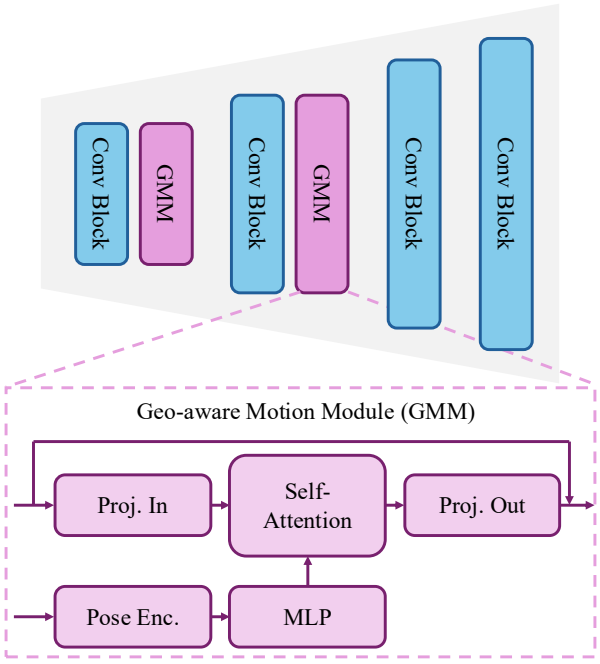

Figure 6. **Illustrations of spatio-temporal decoder.** Please refer to Sec. A for more details.

Spatio-Temporal Decoder. While the Triple-Cues Combiner (Sec. 3.3) effectively fuses multi-modal cues within a single frame, independent decoding fails to guarantee temporal consistency across the video sequence. Standard video transformers rely on blind learnable positional encodings, ignoring the explicit 3D geometric relationships provided by camera poses. To address this, we propose a Spatio-Temporal Decoder. As illustrated in Fig. 6, it builds upon the Dense Prediction Transformer (DPT) architecture [48], with Geometry-Aware Motion Modules (GMMs) that extend it from static image processing to geometry-aware video sequence modeling. Instead of asking the transformer to hallucinate 3D relationships from 2D sequences, we explicitly embed each pixel's 3D geometric state into the decoder's features. For each pixel $\mathbf{p} = (u, v)$ in frame t, we compute its camera ray origin $\mathbf{o}_t \in \mathbb{R}^3$ and normalized direction $\mathbf{d}_t \in \mathbb{R}^3$ in the world coordinate system

$$\mathbf{d}_t = \mathbf{R}_t\mathbf{K}^{-1}\mathbf{p}, \quad \mathbf{o}_t = \mathbf{C}_t, \tag{11}$$

where $\mathbf{R}_t, \mathbf{C}_t$ are the camera rotation and center derived from the pose. To allow the network to learn high-frequency geometric details (e.g., small parallax changes), we map the 6D ray coordinate $[\mathbf{o}_t, \mathbf{d}_t]$ to a high-dimensional feature space using Fourier features:

$$\gamma(\mathbf{v}) = [\dots, \sin(2^k \pi \mathbf{v}), \cos(2^k \pi \mathbf{v}), \dots]. \tag{12}$$

This embedding is then projected to the feature dimension C via a shallow MLP:

$$E_{geo}(t, u, v) = \text{MLP}(\gamma([\mathbf{o}_t, \mathbf{d}_t])). \tag{13}$$

We inject this geometric embedding directly into the decoder features before the temporal attention layers

$$\hat{F} = F + E_{geo}. \tag{14}$$

When the transformer computes attention between frame t and t', it can now explicitly compare their ray embeddings. Tokens with intersecting or converging rays (indicating the same 3D surface point) will naturally have higher affinity, thereby enforcing multi-view 3D consistency in a geometrically grounded manner.

A.2. Training Details

For network training, we utilized four synthetic datasets with precise depth annotations: TartanAir [66], VKITTI [4], TartanGround [45], and MVS-Synth [27], totaling 2.03 million samples. It is worth noting that for the VKITTI dataset, we incorporated both left- and right-camera views to accommodate multi-view inputs, effectively doubling the original VKITTI data. During training, sparse depth maps are synthetically generated by sub-sampling the dense ground truth. We simulate random 8–64 line LiDAR patterns with variations in angle and shift. To mimic the inherent measurement fluctuations found in hardware sensors, we simulate sensor precision errors by applying Gaussian noise to the radial distance of each point along its projection ray. To simulate signal loss due to reflective surfaces or distance attenuation, we randomly discard a percentage of the input points, encouraging the model to be robust against varying point densities and missing data.

To enable parallel imitation of interaction logic for multiple targets, we apply specific rules when selecting targets in the data pipeline. There is a 0.5 probability that the target exists in both modalities, a 0.25 probability that only the current frame has a prompt, and a 0.25 probability that only the source frames have prompts.

DriveMVS is trained for 240k steps at an image resolution of 640 × 480, with prompts resized to match. The network is optimized using AdamW [41] with a OneCycleLR [55] scheduler. Data augmentation includes horizontal image flipping (probability of 0.5), along with the remaining hyperparameters, following the configuration in MVSAnywhere [28].

A.3. Loss Functions

We train DriveMVS using a compound objective function that enforces both spatial geometric fidelity and temporal coherence.

Spatial Geometry Losses. To ensure precise per-frame reconstruction, we adopt a multi-scale supervision strategy inspired by [28]. First, we apply a Log-Depth $\mathcal{L}_1$ Loss to compute the absolute error between the logarithm of the ground truth D_{gt} and the predicted depth $\hat{D}$ across four decoder scales s:

$$\mathcal{L}_{depth} = \frac{1}{HW} \sum_{s=1}^{4} \sum_{i,j} \frac{1}{s^2} \left| \uparrow_{gt} \log \hat{D}r^{i,j} - \log D^{i,j} \right|, \tag{15}$$

where $\uparrow_{gt}$ denotes nearest-neighbor upsampling to the ground truth resolution. Second, to encourage sharp discontinuities and preserve high-frequency details, we utilize a Gradient Loss [28, 50]. Unlike standard approaches, we compute spatial gradients (∇) in the inverse depth space to prevent the loss from being dominated by distant regions:

$$\mathcal{L}_{grad} = \frac{1}{HW} \sum_{s=1}^{4} \sum_{i,j} \left| \nabla \downarrow_s \frac{1}{\hat{D}_r^{i,j}} - \nabla \downarrow_s \frac{1}{D^{i,j}} \right|. \tag{16}$$

Third, we enforce local surface consistency via a Surface Normal Loss [28]. We minimize the cosine distance between the ground truth normals N and the predicted normals $\hat{N}$, which are analytically derived from $\hat{D}$ using camera intrinsics:

$$\mathcal{L}_{normals} = \frac{1}{2HW} \sum_{i,j} \left(1 - \hat{N}^{i,j} \cdot N^{i,j} \right). \tag{17}$$

Temporal Consistency Loss. In order to ensure smooth depth transitions across frames, we incorporate the Temporal Gradient Matching (TGM) Loss [8]. This objective enforces temporal consistency between the predicted depth sequence and the ground truth, without requiring optical flow estimation. The temporal loss is defined as:

$$\mathcal{L}_{temporal} = \frac{1}{T-1} \sum_{t=1}^{T-1} \sum_{i,j} \left\| \left| \hat{D}_{t+1}^{i,j} - \hat{D}_t^{i,j} \right| - \left| D_{t+1}^{i,j} - D_t^{i,j} \right| \right\|_1. \tag{18}$$

Specifically, we apply a masking threshold $\tau_{temp} = 0.05$ to the ground truth temporal change ($|D_{t+1} - D_t| < \tau_{temp}$) to filter out regions with occlusions or dynamic objects, ensuring the loss focuses on static geometric consistency.

Total Loss. The final training objective is a weighted sum of the spatial and temporal components, as described in Eq. (9) of the main paper.

B. Implementation Details

B.1. Data Preprocessing

To validate the feasibility of the proposed solution, we utilize front-view camera images and LiDAR data from the Waymo [57], KITTI [19], and DDAD [21] datasets. This setup is designed to be extendable to multi-camera and LiDAR configurations in future work. It is worth noting that while KITTI provides both accumulated dense ground truth (from multi-frame stitching) and single-frame raw LiDAR, DDAD and Waymo only provide per-frame 128-beam and 64-beam LiDAR data, respectively. Given the susceptibility of multi-frame accumulation to artifacts from dynamic objects, we consistently use single-frame LiDAR as the ground truth across all datasets. For prompt generation, we back-project the single-frame LiDAR data into 3D space to calculate beam inclination angles. Based on these angles, we downsample the data to 16, 8, and 8 beams, respectively, to serve as the sparse LiDAR prompts.

B.2. Metrics

Table 6. **Depth metric definitions.** D_{gt} and $\hat{D}$ are the ground-truth and predicted depth, respectively.

Metric	Definition
MAE	$\frac{1}{N}\sum_{i=1}^{N}\lvert D_{gt}-\hat{D}\rvert$
AbsRel	$\frac{1}{N}\sum_{i=1}^{N}\lvert D_{gt}-\hat{D}\rvert/D_{gt}$
τ	$\frac{1}{N}\sum_{i=1}^{N}\left(\max\left(\frac{\hat{D}}{D_{gt}},\frac{D_{gt}}{\hat{D}}\right)<1.25\right)$

For depth metrics, we report MAE, AbsRel, and τ. Their definitions can be found in Tab. 6. For temporal consistency metrics, we report TAE defined as:

$$\text{TAE}=\frac{1}{2N}\sum_{k=1}^{N}(\text{AbsRel}(f(\hat{x}_d,p^k),\hat{x}_d^{k+1})+\text{AbsRel}(f(\hat{x}_d^{k+1},p_-^{k+1}),\hat{x}_d^k)), \tag{19}$$

where, f represents the projection function that maps the depth $\hat{x}_d^k$ from the k-th frame to the $(k+1)$-th frame using the transform matrix p^k. p_-^{k+1} is the inverse matrix for inverse projection. N denotes the number of frames.

B.3. Baseline Implementation

To ensure fair comparison, we evaluate all baselines on same test sequences according to their official repository and open source model, including VGGT [62][1], MapAnything [31][2], MoGe-2 [64][3], DepthPro [3][4], PromptDA [37][5], PriorDA [68][6], MVSFormer++ [7][7], MVSAnywhere [28][8] and VideoDepthAnything [8][9]. Specifically, we implement MapAnything under two different settings: one without camera poses and intrinsics as pure feed-forward setting, and one with camera poses and intrinsics as MVS-like setting. For VideoDepthAnything, we select the base model for evaluation. Notice that the original PromptDA and PriorDA require dense depth as a metric prompt. To fit our autonomous driving settings, we use the same LiDAR prompt as our method (described in Sec. B.1).

[1] https://github.com/facebookresearch/vggt
[2] https://github.com/facebookresearch/map-anything
[3] https://github.com/microsoft/MoGe
[4] https://github.com/apple/ml-depth-pro
[5] https://github.com/DepthAnything/PromptDA
[6] https://github.com/SpatialVision/Prior-Depth-Anything
[7] https://github.com/maybeLx/MVSFormerPlusPlus
[8] https://github.com/nianticlabs/mvsanywhere
[9] https://github.com/DepthAnything/Video-Depth-Anything

C. More Discussions of Experiments

C.1. Analysis of Baseline Performance

To ensure a fair comparison, we reproduced all baseline methods on a single A100 GPU, strictly adhering to their officially provided pretrained weights and configurations. It is crucial to note that, unlike MVSAnywhere [28] and MapAnything [31], for our KITTI experiments, we use the entire official validation split for evaluation. Consequently, our reported performance metrics for these baselines may differ from those presented in their respective publications, which often employ a subset or an alternative split. We present a quantitative comparison against state-of-the-art baselines on three challenging autonomous driving datasets: *KITTI*, *DDAD*, and *Waymo*. As summarized in Tab. 2, our method consistently outperforms all competing approaches across all metrics and datasets, establishing a new state of the art for robust metric depth estimation.

Comparison with Feed-forward Reconstruction. Feed-forward approaches such as VGGT [62] and MapAnything [31] prioritize speed but compromise on reconstruction accuracy. The substantial error observed in VGGT (13.19m MAE on KITTI) can be attributed to the difficulty of direct geometry regression, which overlooks the explicit constraints provided by camera intrinsics and poses. Even when MapAnything is augmented with ground-truth poses (MapAnything†), improving its MAE to 1.27m, it remains inferior to our method. This comparison validates our design choice to leverage MVS formulations, which are essential for delivering stable, metric-scale geometry in autonomous driving scenarios.

Comparison with Monocular Methods. Monocular methods without prompts, such as MoGe-2 [64] and DepthPro [3], aim to directly recover metric depth and demonstrate impressive generalization. However, they suffer from inherent scale ambiguity in complex, large-scale autonomous

driving environments, resulting in poor metric accuracy (*e.g.*, DepthPro achieves only 80.71% inlier ratio on KITTI, compared to our 98.78%). When integrating sparse LiDAR prompts, PromptDA [37] still performs inferiorly due to its strong dependency on dense depth prompts. We attribute this to its architectural design, which targets indoor scenarios and operates more like depth super-resolution, heavily relying on dense (albeit coarse) depth observations rather than the sparse inputs typical of autonomous driving. It is worth noting that, strictly following the official implementation, we applied KNN ($k = 4$) densification to the sparse prompts for PromptDA to ensure a fair comparison, yet it failed to yield satisfactory results. In contrast, while PriorDA [68] serves as a competitive baseline, our method outperforms it by a substantial margin (*e.g.*, reducing MAE by $\sim$20% on KITTI and $\sim$31% on Waymo). This significant improvement validates the indispensability of the explicit geometric cues provided by an MVS backbone for accurate depth recovery.

Comparison with MVS Baselines. Traditional MVS-based methods like MVSFormer++ [7] and the recent generalist MVSAnywhere [28] are susceptible to scale collapse in low-parallax scenarios (*e.g.*, highway scenes or ego-static scenes) or textureless regions. On Waymo, MVSAnywhere achieves a τ of 89.80%, whereas our method reaches 95.95%. This gap confirms the effectiveness of our dual-branch strategy: our method leverages MVS for geometry consistency while incorporating the TCC module to resolve ambiguities using structural and metric prompts when MVS cues are unreliable.

Cross-Domain Generalization. Crucially, our method maintains top-tier performance across diverse sensor configurations, from the 64-beam LiDAR in KITTI to the sparser setups in DDAD and Waymo (downsampled for evaluation). The consistent superiority across datasets validates that our model successfully generalizes, robustly fusing heterogeneous cues regardless of the specific domain shift.

To summarize, the results in Tab. 2 validate our core hypothesis: neither MVS geometry nor sparse metric prompts are sufficient in isolation. MVS provides dense constraints but lacks robustness in degenerate motions; sparse prompts provide absolute scale but lack spatial density. By unifying these via our *Prompt-Anchored Cost Volume* and *Triple-Cue Combiner*, our framework effectively produces depth maps that are both metrically accurate (low MAE and AbsRel) and structurally precise (high τ).

We present more visualization results of different methods across various datasets. Fig. 8 and Fig. 9 show more cases on KITTI [19] dataset. Fig. 10 and Fig. 11 show more cases on DDAD [21] and Waymo [57] dataset.

Table 7. Extreme Cases IDs

Case	Scene ID	# Total Images
Static	2011_09_29_drive_0026_sync	148
	13941626351027979229	199
	14127943473592757944	197
Rainy	10448102132863604198	183
	11356601648124485814	199
	13184115878756336167	199
	13415985003725220451	199
Dark	11901761444769610243	196
	13184115878756336167	199
	13694146168933185611	193
	14107757919671295130	194

C.2. Detailed Analysis on More Experiments.

Reconstruction Performance Fig. 7 presents a qualitative comparison on the KITTI dataset. As shown in rows (b) and (c), PriorDA struggles to maintain structural integrity in complex regions, exhibiting noticeable artifacts and floating noise around vegetation and walls. While MVSA captures the general scene geometry, it tends to produce over-smoothed results with blurred boundaries. In contrast, ours effectively leverages the strengths of the MVS backbone. We achieve the most robust reconstruction with sharp boundaries and complete structural details, demonstrating superior performance in recovering both thin structures (*e.g.*, tree trunks and pedestrians) and planar surfaces (*e.g.*, road surface). Specifically, we apply the same statistical outlier-removal filter to the reconstructed point clouds of all methods.

Extreme Cases To rigorously evaluate model robustness under adverse conditions, we curated a specialized test set targeting three common yet challenging scenarios in autonomous driving: Rainy Weather, Low-light Environments, and Ego-static Situations (characterized by low-parallax motion). In Sec. 4.3, we present a quantitative evaluation of these cases to demonstrate the stability of DriveMVS. Additionally, Fig. 12 visualizes qualitative results sampled from the *Waymo* and *KITTI* datasets. Detailed dataset statistics, including specific scene IDs and the number of frames per scene, are provided in Tab. 7.

Prompt Absence To assess the robustness of our model under partial sensor coverage, we conducted an experiment on LiDAR-blind view recovery, as illustrated in Fig. 5. In this setup, sparse LiDAR prompts are provided exclusively for the front-view camera, leaving the rear-view query image as a blind spot without direct geometric guidance. As shown in the error maps, the baseline method (MVSAnywhere) fails to infer the correct metric scale for the rear view, resulting in an AbsRel error of 10.87%. This indicates a limited

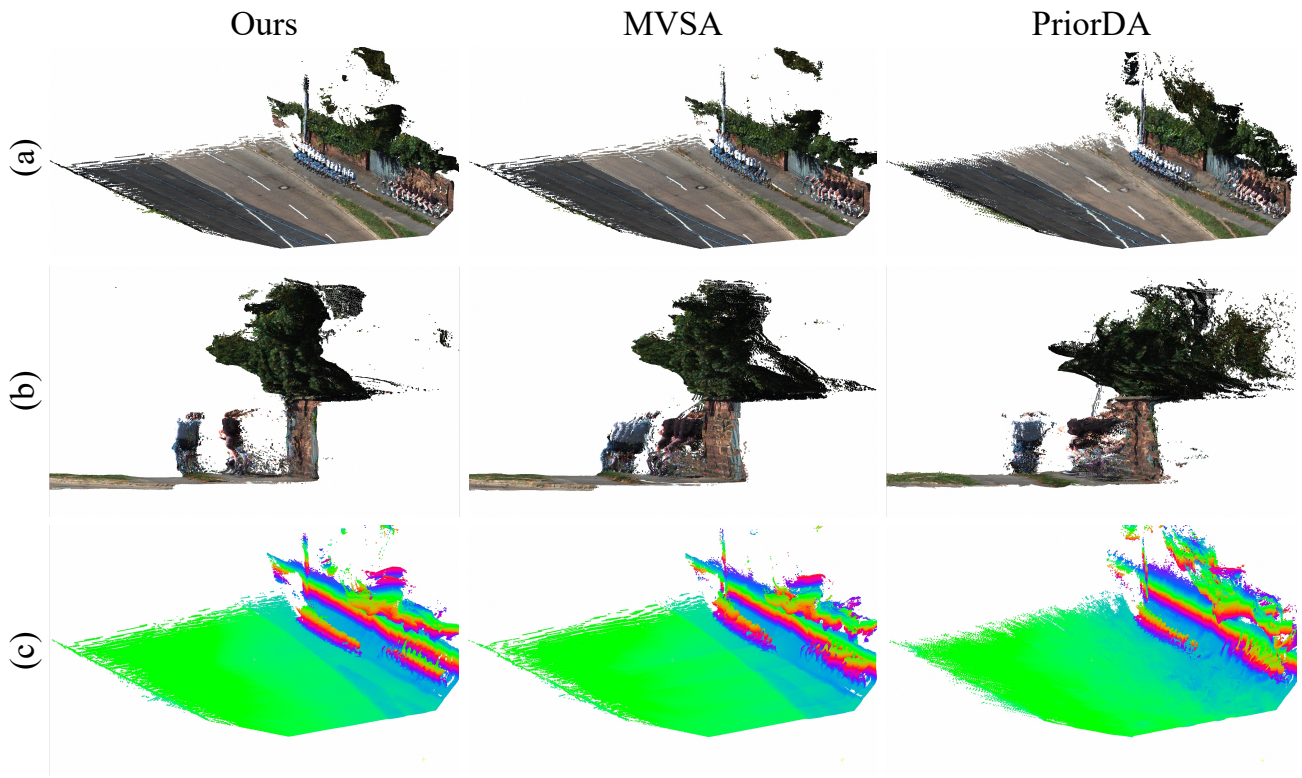

Figure 7. **Qualitative comparison of 3D reconstruction results on *KITTI* [19].** Rows show different methods: our DriveMVS model, MVSA [28], and PriorDA [68]. **(a)**: Reconstructed colored point clouds of the driving scene. **(b)**: Zoomed-in views highlighting fine-grained details of vegetation and structural boundaries. **(c)**: Corresponding z-axis visualizations.

Table 8. Speed and memory consumption. Number benchmarked on an A100 GPU.

#	Time (ms)	Memory (GB)
PriorDA	246.79	2.45
PromptDA	300.60	3.75
MVSA	38.84	4.02
Ours	65.61	4.47

capability in transferring geometric cues across disparate views. In contrast, our method successfully leverages the spatial overlap and geometric correlations between views to propagate metric information from the prompted region (front) to the unprompted query view (rear). Consequently, our approach maintains accurate metric depth recovery (AbsRel: 6.83%) even in the complete absence of current-view prompts, thereby verifying its effectiveness in handling sensor blind spots typical in autonomous driving.

C.3. Runtime Analysis

We report the inference time cost and maximum GPU memory in Tab. 8. Our framework strikes a favorable balance between performance and efficiency. Although slightly more resource-intensive than the lightweight MVSA due to enhanced feature aggregation, our method is significantly more efficient than prompt-guided methods such as PriorDA, demonstrating its potential for online autonomous driving applications.

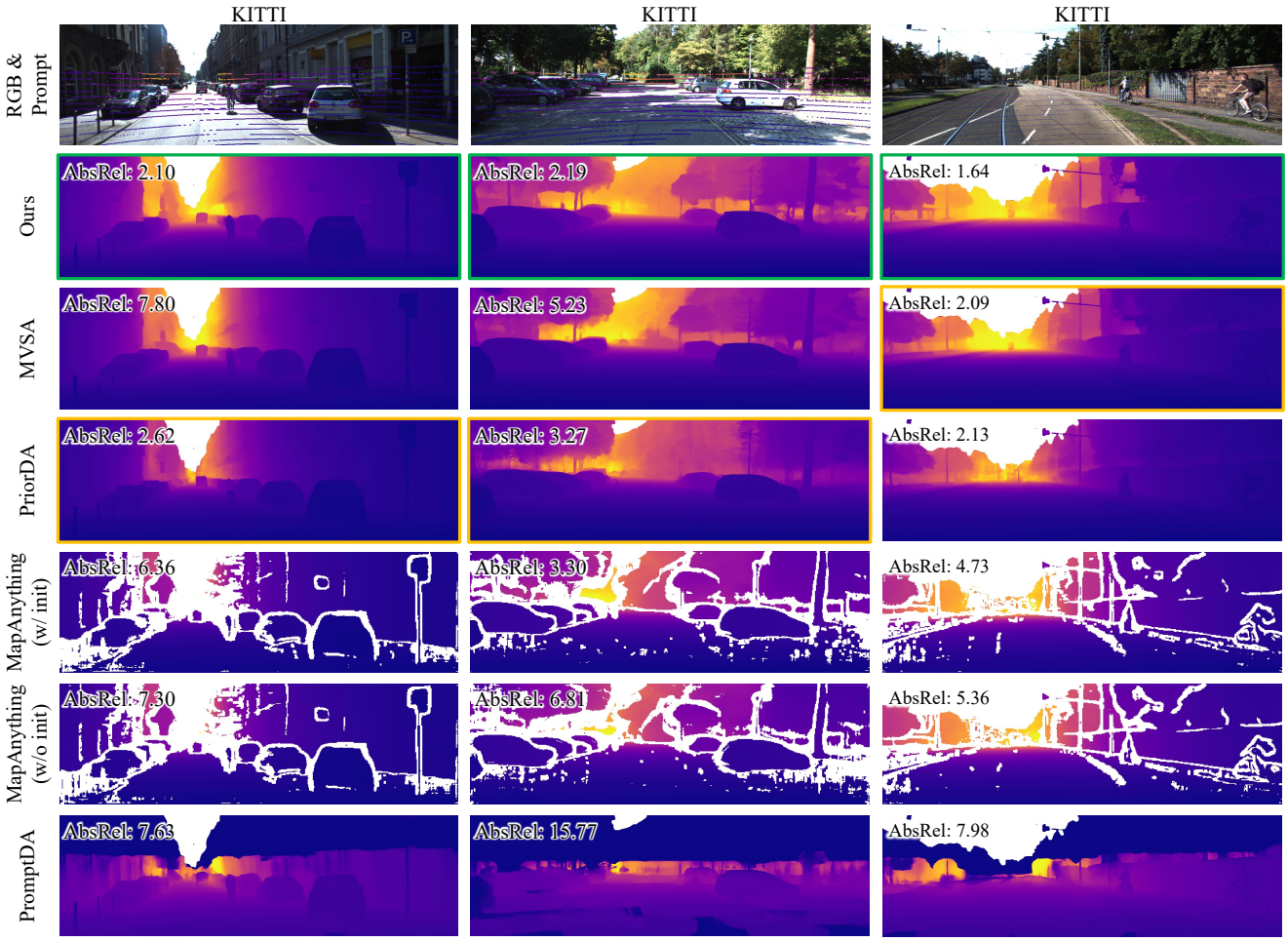

Figure 8. **Qualitative comparison of depth prediction results on *KITTI* [19].** Rows show different methods: our DriveMVS model, MVSA [28], PriorDA [68], MapAnything [31], and PromptDA [37], along with RGB and prompt inputs ($\boldsymbol{I}_r$, $\boldsymbol{P}_r$). **[left]**: *2011_09_26_drive_0095_sync*. **[Middle]**: *2011_09_26_drive_0023_sync*. **[Right]**: *2011_09_26_drive_0002_sync*. The best and second best are highlighted with green and yellow borders, respectively.

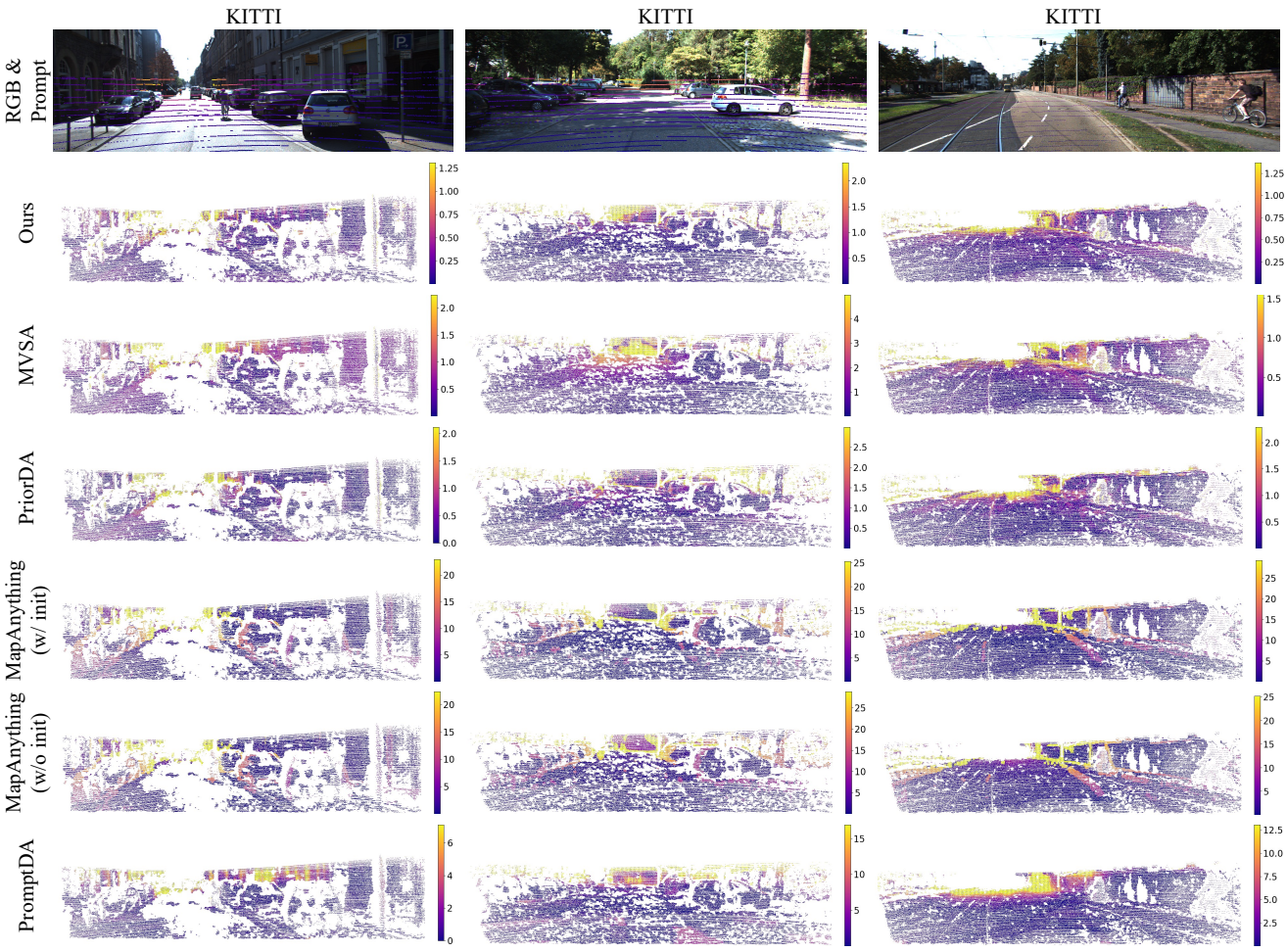

Figure 9. **Qualitative comparison of depth prediction errors on *KITTI* [19]**, corresponding to Fig. 8. Rows show different methods: our DriveMVS model, MVSA [28], PriorDA [68], MapAnything [31], and PromptDA [37], along with RGB inputs ($\boldsymbol{I}_r$) and prompt ($\boldsymbol{P}_r$).

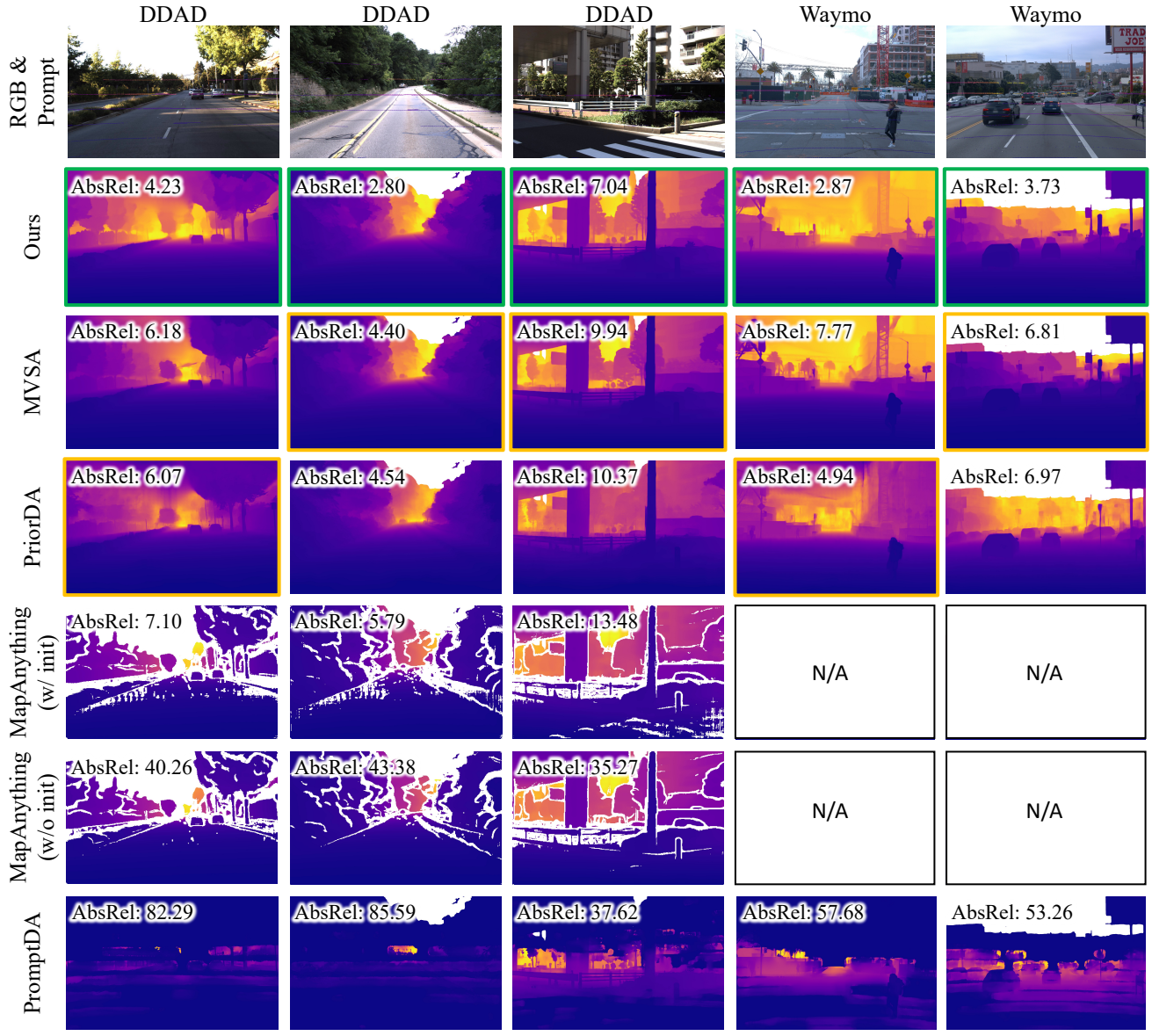

Figure 10. **Qualitative comparison of depth prediction results on *DDAD* [21] and *Waymo* [57].** Rows show different methods: our DriveMVS model, MVSA [28], PriorDA [68], MapAnything [31], and PromptDA [37], along with RGB and prompt inputs ($\boldsymbol{I_r}$, $\boldsymbol{P_r}$). For left to right: *000156*, *000155*, *000194*, *1024360143612057520*, *1286817684252793621*. The best and second best are highlighted with green and yellow borders, respectively.

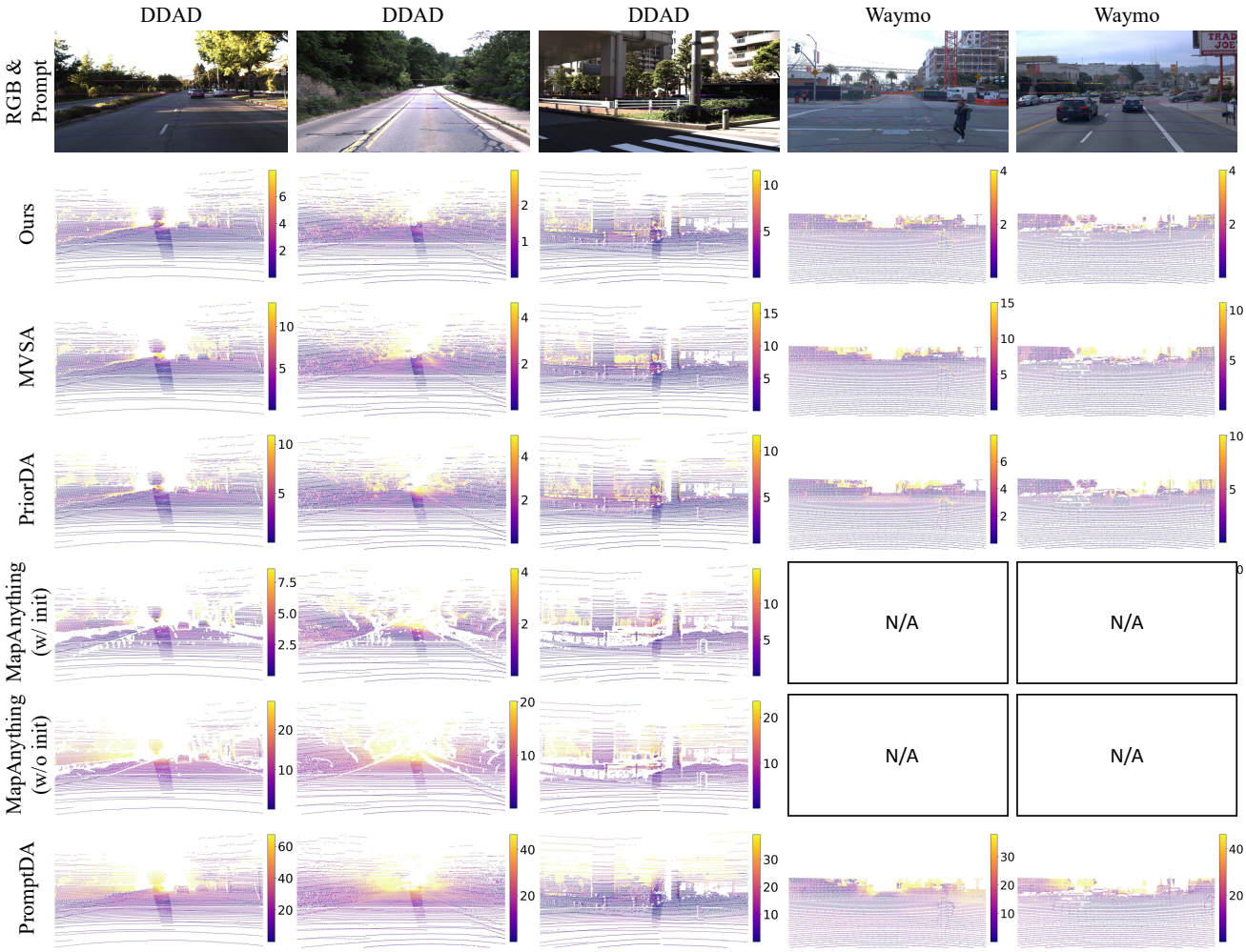

Figure 11. **Qualitative comparison of depth prediction errors on *DDAD* [19] and *Waymo* [57]**, corresponding to Fig. 10. Rows show different methods: our DriveMVS model, MVSA [28], PriorDA [68], MapAnything [31], and PromptDA [37], along with RGB inputs ($\boldsymbol{I}_r$) and prompt ($\boldsymbol{P}_r$).

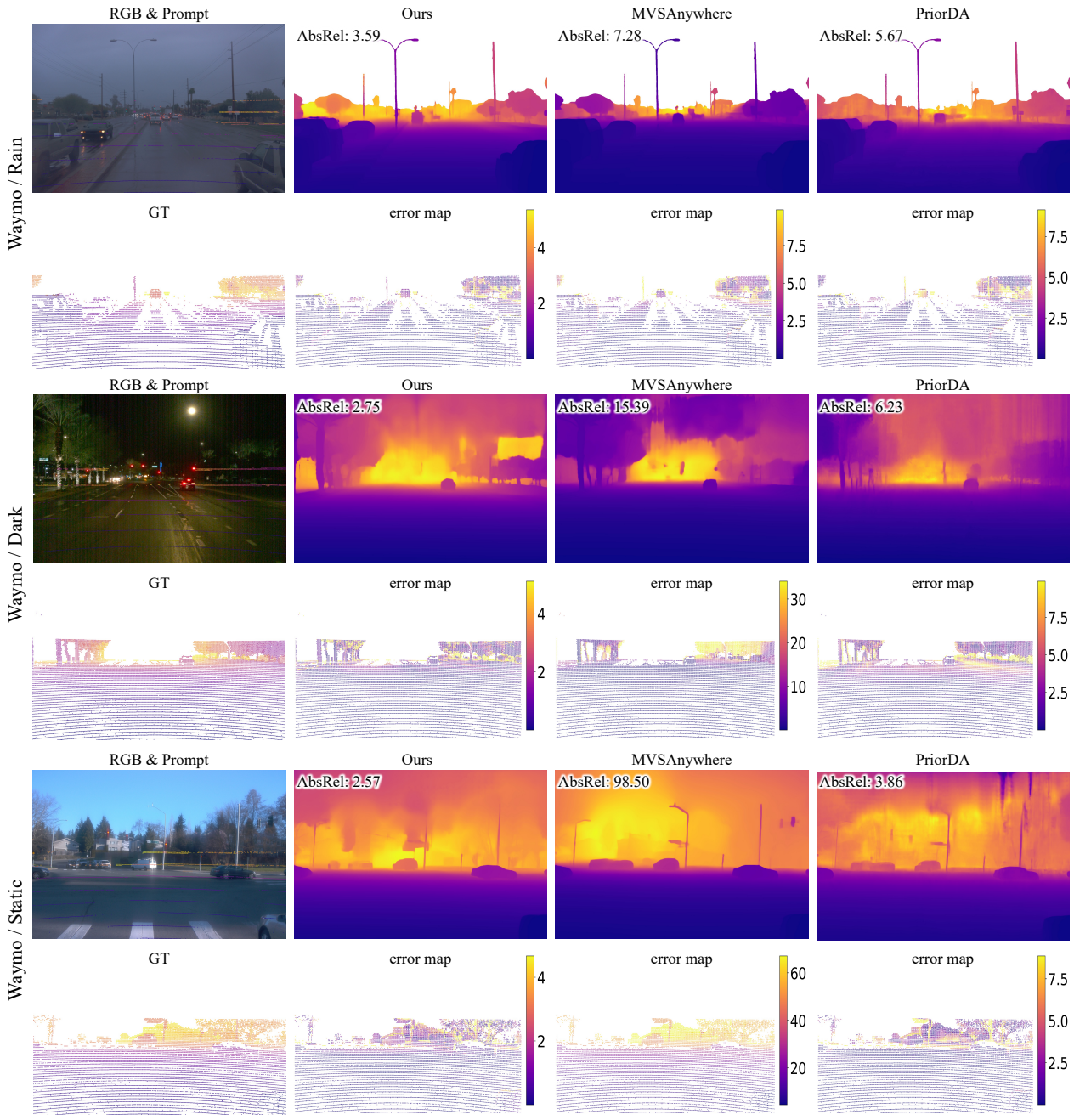

Figure 12. **Qualitative comparison of depth prediction results and depth prediction errors across extreme cases for autonomous driving in our sampled dataset (most *Waymo* [57]).** Columns show different methods: our DriveMVS model, MVSAnywhere [28], and PriorDA [68], along with RGB & prompt inputs ($\boldsymbol{I}_r$, $\boldsymbol{P}_r$) and ground-truth depths (GT). **[Top]**: *11356601648124485814*. **[Middle]**: *14107757919671295130*. **[Bottom]**: *14127943474592757944*.