Real-Time Front-View Depth Estimation for Automatic Emergency Braking

Yu Han, Junjie Zhang, Xiangyu Wang, Chuanchuan Zhong, and Yulun Song

Li Auto Inc. , Shanghai, China
hanyu15@lixiang.com, zhangjunjie11@lixiang.com

Abstract. Automatic emergency braking (AEB) requires front-obstacle perception that is both accurate and low-latency. Existing systems typically rely on lidar-based perception, camera-based object detection, or image-view depth estimation. Lidar pipelines can be vulnerable to spurious returns caused by reflective surfaces or exhaust smoke, while object detectors depend on predefined categories and may miss unknown or irregular obstacles. Image-view depth estimation predicts dense per-pixel depth, but its outputs are not directly aligned with AEB and usually require projection and additional post-processing. We propose an end-to-end model that directly predicts a front-view depth map on a forward plane in the ego-vehicle coordinate frame. An attention-based encoder aggregates multi-scale image features onto a fixed front-view query grid, and a lightweight decoder outputs a compact front-view depth map. Training targets are generated automatically by projecting temporally aligned lidar point clouds onto the front-view plane, requiring no manual annotation. In deployment, the model runs at 1.7 ms on ThorU and reduces lidar-noise-induced false AEB triggers by about 40% without degrading true-positive AEB triggering.

Keywords: AEB · Front-View Depth · Depth Estimation

1 Introduction

Automatic emergency braking (AEB) is an advanced driver-assistance function that can brake without driver intervention to avoid or mitigate collisions, as shown in Fig 1. Its performance heavily depends on the perception module, which must detect front obstacles with high speed, accuracy, and stability. Current autonomous driving perception solutions are mainly based on lidar or cameras.

Lidar-based pipelines typically perform 3D object detection [14, 22, 23] or point cloud segmentation [10]. They provide accurate range and localization, but lidar point clouds are sparse and can be affected by significant "blooming" noise. In challenging conditions such as highly reflective objects, rain/fog, or exhaust smoke, abnormal returns may cause false detections, which is particularly critical for safety-critical functions with low false-positive tolerance such as AEB.

Camera-based pipelines include object detection [9, 12, 18] and depth estimation [1,2,15]. Object detectors often degrade substantially for unknown categories

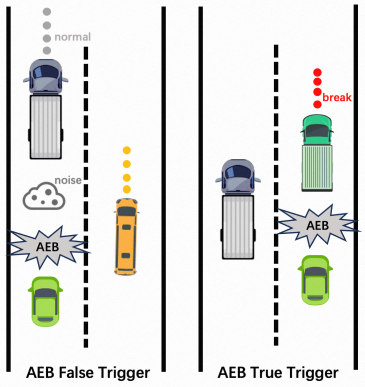

Fig. 1: Illustration of AEB triggering cases. **Left:** a false trigger caused by perception noise (no real obstacle requires braking). **Right:** a true trigger where a leading vehicle brakes, and AEB should be activated to avoid collision.

(objects not labeled in the training set) or irregular shapes. Monocular depth estimation can provide dense per-pixel depth, but the output is defined in the image coordinate and does not directly provide obstacle positions in the ego-vehicle coordinate system. In practice, it usually requires projection and additional processing (segmentation or clustering) to obtain object-level distances. Moreover, computing dense depth for large background regions wastes computation and increases latency.

To address these issues, we propose a new depth estimation paradigm: directly predicting front-view depth on a forward plane in the ego-vehicle coordinate system. Compared with monocular depth estimation, front-view depth naturally encodes obstacle positions in the ego-vehicle coordinate system and can be consumed by AEB without complex post-processing. Compared with object detection, front-view depth provides a more general representation and can estimate distances for obstacles of categories not present in the training annotations. Compared with lidar-based methods, predicting depth from images can reduce the impact of lidar blooming noise and improve robustness across scenes. We design an efficient network architecture that focuses on the driving-critical frontal region, significantly reducing redundant computation, and achieving very low inference latency while maintaining high accuracy.

Our main contributions are as follows.

1. We introduce a front-view depth estimation paradigm that directly predicts front-view depth, producing outputs better aligned with AEB requirements.
2. We design a temporal, real-time front-view depth network that achieves extremely low inference latency without sacrificing accuracy. The end-to-end inference latency is only 1.7 ms on the NVIDIA ThorU platform.
3. We further evaluate the proposed method in real-world tests and reduces lidar-noise-induced false AEB triggers by about 40% without degrading true-positive AEB triggering.

2 Related Work

2.1 Lidar-based Perception

Lidar-based 3D perception takes point clouds as input and outputs 3D bounding boxes or point-wise semantic segmentation. Early methods such as PointNet [14] learn point features for classification and localization. To improve efficiency and handle sparse point clouds, voxel-based methods (VoxelNet [23]) encode points into sparse voxels and apply 3D convolutions. More recent autonomous driving systems widely adopt bird's-eye-view (BEV) representations. PointPillars [8] maps point clouds into BEV pillar features to achieve a favorable speed–accuracy trade-off. CenterPoint [22] detects object centers on BEV features and regresses 3D boxes, achieving strong performance.

Lidar semantic segmentation is typically performed on range images or within sparse convolution frameworks (RangeNet++ [10]) to classify points into obstacle/ drivable-area categories. Despite lidar's inherent ranging advantages, point sparsity at long distances and abnormal returns in highly reflective surfaces, rain/fog, and exhaust smoke can induce false positives, which is especially problematic for AEB.

2.2 Monocular 3D Object Detection

Monocular 3D object detection aims to infer 3D object locations directly from images. Recent methods are predominantly end-to-end and incorporate depth cues either implicitly or explicitly. For instance, *SMOKE* [9] formulates monocular 3D detection as a simple implicit regression problem without relying on explicit geometric reconstruction, while *FCOS3D* [19] adopts an anchor-free design for 3D box prediction. *ATV3D* [6] further introduces a tri-view representation to better describe the scene geometry and appearance across different perspectives. Another line of work leverages monocular depth estimation to provide geometric priors (*DD3D* [12]). Although such depth-assisted pipelines can be effective, they often increase computational cost and may suffer from error accumulation introduced by the intermediate depth estimation stage. In addition, transformer-based architectures and temporal modeling have been explored to improve robustness and stability in dynamic scenes, as demonstrated by *DETR3D* [20] and *StreamPETR* [18].

Despite this progress, monocular 3D detectors are highly dependent on the training distribution and annotation quality, and they typically operate on known object categories. When unknown or irregular obstacles appear, the detector may fail to produce a 3D box, increasing the risk of missed AEB activation. Moreover, for AEB, the primary requirement is often the nearest obstacle distance within the frontal region, rather than category-specific box outputs.

2.3 Monocular Depth Estimation

Monocular depth estimation regresses a depth value for each image pixel. Supervised approaches on benchmarks such as KITTI [4] have steadily improved

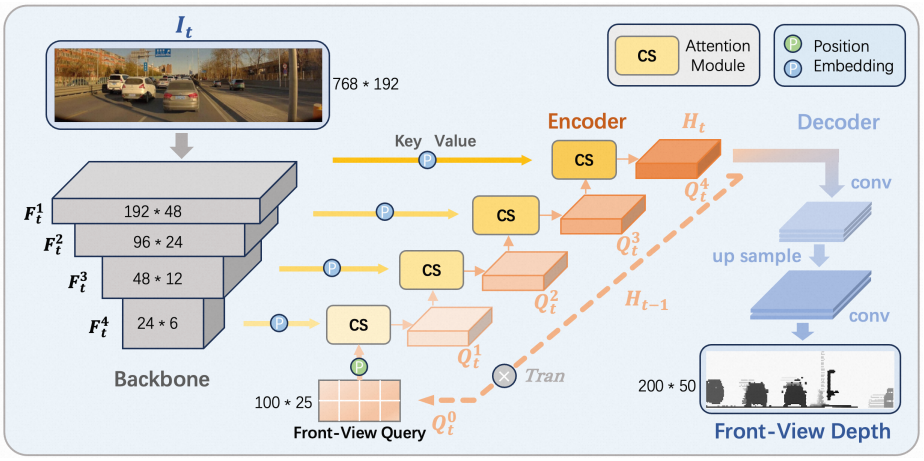

Fig. 2: Overall architecture of the proposed front-view depth estimation network.

accuracy (DORN [3] and subsequent works). Self-supervised methods (Monodepth2 [5]) learn depth and pose from adjacent frames via photometric consistency, reducing the need for ground-truth depth. Transformer-based designs (e.g., AdaBins [1], DPT [15], Depth Pro [2]) and large-scale foundation depth models (e.g., MiDaS [16], Depth Anything V2 [21]) further enhance long-range modeling, cross-domain generalization, and detail recovery.

Despite their value, image-view dense depth has two main limitations for AEB:

1. **Representation mismatch:** image-view depth must be projected into the ego frame using calibration, and often requires ground removal and clustering to obtain obstacle-level distances.
2. **Redundant computation:** dense depth spends significant compute on irrelevant background (sky, distant buildings), increasing end-to-end latency.

Therefore, predicting depth directly in a task-aligned ego-frame representation is attractive. Prior work on BEV feature construction (Lift-Splat-Shoot [13]) has shown that transforming image features into ego coordinates can simplify downstream tasks. Motivated by this, we propose front-view depth estimation: it leverages visual cues to reduce lidar noise sensitivity, does not rely on object categories, and focuses computation on the AEB-critical frontal region.

3 Method

3.1 Overview

As illustrated in Fig. 2, we propose a front-view depth estimation network for active safety. The input is a monocular front-facing image (extendable to video

sequences),

$$I_t \in \mathbb{R}^{768\times192\times3}, \tag{1}$$

and the output is a front-view depth map defined on a forward plane in the ego-vehicle coordinate system,

$$\hat{D}_t^{fv} \in \mathbb{R}^{W_{fv}\times H_{fv}}, \quad (W_{fv}, H_{fv}) = (200, 50). \tag{2}$$

Unlike conventional monocular depth estimation that regresses depth on the image plane, we directly predict depth on an ego-frame front-view plane. This output is naturally aligned with AEB needs (frontal occupancy and nearest distance), reducing post-processing such as projection, segmentation, and clustering, and avoiding computation on irrelevant background pixels.

The input resolution 768×192 follows the aspect ratio of our front camera after cropping the sky and hood regions that are irrelevant to AEB. The output front-view grid 200×50 is determined by the AEB decision needs: each cell corresponds to a physical extent of 0.08 m, covering the frontal region of 16 m (width) $\times$ 4 m (height). For deployment, we crop the lateral coverage to 8m while keeping the same vertical resolution, resulting in a 100×50 output grid (8 m $\times$ 4 m). The range is already sufficient for AEB, to meet the 2.0 ms latency budget (Sec. 4.7).

3.2 Image-to-Front-View Mapping

The network first extracts multi-scale image features using a backbone:

$$\{F_t^k\}_{k=1}^4 = \text{Backbone}(I_t). \tag{3}$$

To aggregate image evidence into a front-view representation, we introduce a learnable front-view query grid:

$$Q_t^0 \in \mathbb{R}^{N\times C}, \quad N = 100 \times 25. \tag{4}$$

The encoder stacks multiple cross-/self-attention (CS) blocks and updates the queries by fusing features from different scales. For the k-th scale, we project features into keys and values and add positional encodings:

$$K_t^k = \phi_k(F_t^k) + P_t^k, \qquad V_t^k = \psi_k(F_t^k) + P_t^k, \tag{5}$$

where $\phi_k(\cdot)$ and $\psi_k(\cdot)$ denote linear projections, and P_t^k is the positional encoding. We then update the queries via cross-attention followed by self-attention:

$$\tilde{Q}_t^k = \text{CrossAttn}(Q_t^{k-1}, K_t^k, V_t^k), \tag{6}$$

$$Q_t^k = \text{SelfAttn}(\tilde{Q}_t^k). \tag{7}$$

After fusing all four scales, we obtain the front-view latent representation:

$$H_t = Q_t^4. \tag{8}$$

Temporal enhancement. To improve temporal consistency and robustness in dynamic scenes, we introduce a temporal module. Using the ego-motion transform $Tran_{t-1\to t}$, we align the previous latent representation to the current coordinate:

$$\bar{H}_{t-1} = \mathrm{Warp}(H_{t-1}, Tran_{t-1\to t}), \tag{9}$$

and fuse it with the current queries:

$$Q_t^0 \leftarrow \mathrm{Fuse}(Q_t^0, \bar{H}_{t-1}). \tag{10}$$

Here, $\mathrm{Warp}(\cdot)$ is implemented via pose-guided grid sampling with bilinear interpolation, and $\mathrm{Fuse}(\cdot)$ is implemented as concatenation followed by a linear projection. This design injects temporal priors with minimal overhead, reducing prediction jitter and improving stability under occlusion and for small distant obstacles.

3.3 Front-View Depth Prediction

The encoder output tokens are reshaped into a 2D front-view feature map and decoded into the final depth prediction:

$$\hat{D}_t^{fv} = \mathrm{Decoder}\big(\mathrm{Reshape}(H_t)\big). \tag{11}$$

The decoder is a convolutional module that aggregates local context and recovers spatial details, producing a 200×50 depth map. In deployment, the output resolution can be adjusted according to the AEB region-of-interest and compute budget to balance accuracy and latency.

3.4 Supervision and Loss

We train the model with supervised learning. The ground-truth front-view depth D_t^{fv} is generated automatically by projecting lidar point clouds into the ego frame and then onto the front-view plane, without manual annotation. To avoid supervision noise from unobserved regions, we use a validity mask $M(u, v)$ and only regress on reliable cells:

$$\mathcal{L} = \sum_{u,v} M(u,v) \cdot \rho\Big(\hat{D}_t^{fv}(u,v) - D_t^{fv}(u,v)\Big), \tag{12}$$

where $\rho(\cdot)$ is a robust regression loss. Specifically, we use the $\mathcal{L}_1$ loss in Stage 1 (depth regression) and the Huber loss in Stage 2 (inverse-depth fine-tuning). The Huber loss in Stage 2 is defined as

$$\rho_\delta(x) = \begin{cases} \dfrac{1}{2}\,x^2, & |x| < \delta, \\ \delta\left(|x| - \dfrac{1}{2}\delta\right), & |x| \geq \delta, \end{cases} \tag{13}$$

with $\delta = 0.1$.

Table 1: Dataset statistics

Item	Value
video clips (total)	10,000
frames per clip	150
training clips	8,000
validation clips	1000
test clips	1000
Sensors	Image + Lidar + Odometry

4 Experiments

4.1 Dataset

We use an in-house driving dataset collected by a leading electric vehicle manufacturer, consisting of 10,000 video clips covering diverse real-world scenarios (urban roads, highways, night-time, backlight, rain/fog, heavy traffic, tunnels). Each clip contains 150 consecutive frames. For every frame, we provide a synchronized front-view image and a lidar point cloud. We further leverage odometry to temporally align lidar points to the same timestamps as the images, forming paired image–pointcloud samples.

We split the 10,000 clips at the *clip level* (not the frame level) to prevent temporal leakage: 8,000 clips for training, 1,000 clips for validation, and 1,000 clips for testing (an 8:1:1 split by clip count, corresponding to 1,200,000 / 150,000 / 150,000 frames). The three splits are collected on disjoint driving sessions and share no overlapping clips or geographically adjacent trajectories, so the training, validation, and test sets are strictly independent. All numbers reported in Sec. 4.5–4.8 are computed on the held-out test set unless otherwise stated. Table 1 summarizes the statistics.

4.2 Ground Truth Construction

We train our model with supervised learning, where the front-view depth ground truth is automatically generated from lidar without manual annotation. Specifically, we first transform raw lidar points into the ego-vehicle coordinate system. With odometry-based temporal alignment, each point cloud is synchronized to the corresponding image timestamp. We then project the ego-frame point cloud onto a predefined front-view plane to form a 200×50 depth map.

Each pixel corresponds to a physical resolution of 0.08m, covering a region of 16m in lateral width (left–right) and 4m in height above the ground. The pixel value stores the distance from the target point to the ego vehicle. Pixels without valid lidar projections are marked invalid and excluded from loss computation via a binary mask. This pipeline produces scalable supervision at low cost.

Table 2: Two-stage training schedule.

Stage	Target	Loss	Epochs	Valid-pixel mask
1	Depth	$\mathcal{L}_1$	40	Yes
2	Inverse depth	Huber	5	Yes

4.3 Evaluation Metrics

We report errors over multiple distance ranges to reflect performance under different driving-relevant conditions. Following our logging convention, we compute:

Absolute error (Abs).

$$\text{Abs} = \frac{1}{N}\sum_{i=1}^{N}\left|\hat{d}_i - d_i\right|, \tag{14}$$

Relative error (Rel).

$$\text{Rel} = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|\hat{d}_i - d_i\right|}{d_i}, \tag{15}$$

where d_i and $\hat{d}_i$ denote the ground-truth and predicted depth, respectively, and N is the number of valid pixels in the evaluated region.

We report both metrics on several distance bins: $[0, 10]$ m, $[10, 20]$ m, $[20, 30]$ m, $[30, 50]$ m, and $[50, 100]$ m.

4.4 Baseline Methods

To place our method in the context of state-of-the-art monocular metric depth models, we compare against two representative baselines evaluated under both zero-shot and fine-tuned regimes:

Depth Pro [2]. A high-resolution ViT-based metric depth model with a two-stream 1536×1536 backbone. We evaluate it *zero-shot* using the official released checkpoint.

Depth Anything V2 (DA-V2) [21]. The metric variant pre-trained on Virtual KITTI. We evaluate two encoder sizes (ViT-S and ViT-L) in two regimes: (i) *Zero-shot (ZS):* applied directly with released weights; (ii) *Fine-tuned (FT):* initialized from the ZS weights and fine-tuned on our training split with a SiLog loss (encoder LR 1×10^{-6}, decoder LR 5×10^{-5}; 10 epochs). We select the checkpoint with the lowest validation Abs error.

Since none of these baselines predict the front-view (FV) depth grid directly, we project their dense image-plane depth onto the FV plane using the same calibrated LiDAR-to-camera extrinsics that generate our supervision. All methods therefore share an identical evaluation protocol on the same 200×50 FV grid.

Table 3: Comparison with state-of-the-art models on the FV grid

Method	Abs Error (m) ↓					Rel Error ↓				
	≤10 m	10–20	20–30	30–50	50–100	≤10 m	10–20	20–30	30–50	50–100
Depth Pro (ZS) [2]	5.59	6.39	10.63	17.95	37.20	0.727	0.450	0.433	0.464	0.557
DA-V2 ViT-S (ZS) [21]	2.70	5.83	9.17	16.39	36.77	0.328	0.418	0.375	0.428	0.555
DA-V2 ViT-L (ZS) [21]	3.65	6.09	8.19	10.22	19.25	0.452	0.441	0.338	0.272	0.288
DA-V2 ViT-S (FT)	1.86	3.75	7.54	13.90	30.23	0.226	0.266	0.307	0.361	0.459
DA-V2 ViT-L (FT)	1.45	3.14	6.48	12.59	26.97	0.175	0.223	0.263	0.327	0.412
Ours (Stage-1)	<u>0.62</u>	<u>1.21</u>	**2.73**	**5.87**	<u>15.5</u>	<u>0.076</u>	<u>0.087</u>	**0.111**	**0.151**	<u>0.227</u>
Ours (Stage-2)	**0.55**	**1.16**	<u>2.85</u>	<u>6.02</u>	**15.3**	**0.067**	**0.083**	<u>0.116</u>	<u>0.154</u>	**0.225**

Abs error (m ↓) and Rel error (↓) are reported per distance bin. All baselines are evaluated on the same 200×50 FV grid by projecting their image-plane depth through the calibrated LiDAR-camera extrinsics. **Bold** = best, <u>underline</u> = second best.

4.5 Comparison with Prior Work

Table 3 compares our model against the five baselines on the FV grid. We make the following observations.

Zero-shot generalization is insufficient for AEB. Neither Depth Pro nor DA-V2 (ViT-S/ViT-L) generalizes acceptably to our FV setting in zero-shot mode: even the strongest ZS variant (DA-V2 ViT-L) yields Abs = 3.65 m in the 0–10 m bin, an order of magnitude worse than our model. Depth Pro suffers the most, largely because its FOV head is calibrated for full-frame inputs, whereas our AEB pipeline consumes a fisheye-cropped 192×768 strip, breaking metric-scale recovery. Together with its ∼100× slower inference, Depth Pro is unsuitable for the AEB deployment target.

Fine-tuning closes the near-range gap but not the far-range gap. Fine-tuning DA-V2 on our data cuts short-range Abs error by more than 2× (ViT-L: 3.65 → 1.45 m at 0–10 m), but the long-range regime (≥ 30 m) degrades below the zero-shot models (DA-V2 ViT-L Abs at 50–100 m: 19.25 → 26.97 m). This is a direct consequence of LiDAR-projected supervision sparsity at distance: only ∼2% of GT cells lie beyond 50 m, making the fine-tuning gradient at long range unreliable.

Our task-aligned design dominates across all bins. Even the smallest encoder (ResNet-34, Stage-2) achieves Abs = 1.16 m at 10–20 m and 6.02 m at 30–50 m, outperforming DA-V2 ViT-L (FT) by 1.98 m and 6.57 m respectively, while running at 1.7 ms on ThorU (vs. ∼ 30 ms for DA-V2 ViT-L). This supports our central hypothesis: directly regressing on the AEB-critical front-view representation is far more effective than repurposing an image-plane monocular depth model, both in accuracy and in latency.

Table 4: On-road summary

Item	Value / Observation
Deployment output size	100 ×50
Platform	NVIDIA ThorU
Inference latency	1.7 ms
Fleet mileage	1×10^7 km
AEB True Positive Retention	100%
AEB False Positive Reduction	40%

4.6 Two-Stage Training

We train the network on 12 NVIDIA A100 GPUs using the two-stage strategy summarized in Table 3. *Stage 1 (depth regression)* directly regresses metric depth with an $\mathcal{L}_1$ loss for 40 epochs; invalid pixels are excluded via the validity mask. *Stage 2 (inverse-depth fine-tuning)* continues training for 5 more epochs on the inverse-depth target with a Huber loss, which emphasizes short-range errors that matter most for AEB.

Rows *Ours (Stage-1)* and *Ours (Stage-2)* in Table 3 quantify the benefit of this schedule: Stage-2 lowers the $\leq$10 m Abs error from 0.62 m to 0.55 m (–11%) and Rel from 0.076 to 0.067 (–12%), while keeping the $\geq$30 m regime essentially unchanged. This matches our design intent – inverse-depth supervision biases capacity toward the short-range regime that governs AEB triggering.

4.7 On-Road Results

We deploy the proposed front-view depth estimator in a production-like in-vehicle stack and evaluate both runtime and system-level impact through real-world fleet driving. For deployment, we use an output resolution of 100×50. On the NVIDIA ThorU, the end-to-end inference latency is 1.7 ms.

Table 4 summarizes the on-road validation results. We further validate the model over approximately 10 million kilometers of fleet driving. Adding the depth model does not affect true-positive AEB triggering. Meanwhile, it reduces false AEB triggers caused by lidar noise (exhaust artifacts) by about 40%, demonstrating practical robustness gains.

4.8 Ablation Studies

As shown in Table 5, we study the impact of backbone choice, temporal module, feature size, and training configuration on both accuracy and runtime. We evaluate several backbones together with different settings of the CS module, where *Heads* and *FFN dim* denote the number of attention heads and the feed-forward network (FFN) hidden dimension in the CS attention blocks, respectively. The inference time (*Time*) is measured on the NVIDIA ThorU platform. We observe that configurations with similar latency generally achieve comparable accuracy.

Table 5: Ablation

Backbone	Heads	FFN dim	Time(ms)↓	Abs(0–100)↓	Rel(0–100)↓
ViT-B [11]w/o temp	8	2048	3.8	4.79	0.119
ViT-B [11]w/o temp	8	1024	3.5	4.81	0.121
ViT-B [11]w/o temp	4	1024	3.4	4.85	0.122
Cspnet-M [17]w/o temp	4	1024	3.1	4.78	0.118
ResNet-50 [7]w/o temp	4	1024	3.0	4.83	0.121
ResNet-34 [7]w/o temp	4	1024	2.1	5.19	0.129
ResNet-34 [7]w/ temp	**4**	**1024**	**2.1**	**5.13**	**0.126**

Heads and *FFN dim* are the number of attention heads and the feed-forward hidden dimension in each CS attention block, *temp* denotes the temporal enhancement module.

Moreover, under smaller model budgets, CNN backbones can be more effective than ViT-based backbones. As shown by the comparison between w/ temp and w/o temp, the temporal module improves accuracy without increasing inference latency. In our safety-critical AEB system, maintaining a strict end-to-end latency budget is critical, making ResNet-34 the optimal choice despite a slight accuracy drop. Note that the 1.7 ms latency in Sec. 4.7 is measured under the deployed 100×50 resolution, while Table 5 reports the inference latency for the default 200×50 resolution.

5 Conclusion

In this work, we present a front-view depth estimation approach tailored for AEB. Our method directly predicts, in an end-to-end manner, a depth representation on the ego-vehicle forward plane from camera images. The proposed front-view output is better aligned with AEB requirements, thereby reducing downstream post-processing overhead. In addition, an efficient architecture enables real-time inference, while lidar-projected supervision provides scalable training signals without manual annotation. As a result, our approach shows robustness to unknown obstacles and challenging real-world driving conditions.

Fig. 3: Qualitative results on real driving scenes. The left column shows the input RGB images. For each example, the right column visualizes the predicted front-view depth map: the top image is a colorized depth rendering, and the bottom image is a grayscale visualization. The red rectangle indicates the evaluation region of interest (ROI), covering a 4 m width and a height range of 0.4–2.4 m in the ego frame. The magenta rectangle denotes the ego-vehicle body footprint.

References

1. Bhat, S.F., Alhashim, I., Wonka, P.: Adabins: Depth estimation using adaptive bins. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4009–4018 (2021)
2. Bochkovskii, A., Delaunoy, A., Germain, H., Santos, M., Zhou, Y., Richter, S.R., Koltun, V.: Depth pro: Sharp monocular metric depth in less than a second. arXiv preprint arXiv:2410.02073 (2024)
3. Fu, H., Gong, M., Wang, C., Batmanghelich, K., Tao, D.: Deep ordinal regression network for monocular depth estimation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2002–2011 (2018)
4. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. The international journal of robotics research **32**(11), 1231–1237 (2013)
5. Godard, C., Mac Aodha, O., Firman, M., Brostow, G.J.: Digging into self-supervised monocular depth estimation. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 3828–3838 (2019)
6. Han, Y., Chen, Y., Li, H., Zhao, Y., Zhao, Z., Hu, P.: Atv3d: 3d object detection from attention-based three-view representation. In: 2024 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2024)
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
8. Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: Pointpillars: Fast encoders for object detection from point clouds. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 12697–12705 (2019)
9. Liu, Z., Wu, Z., Tóth, R.: Smoke: Single-stage monocular 3d object detection via keypoint estimation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops. pp. 996–997 (2020)
10. Milioto, A., Vizzo, I., Behley, J., Stachniss, C.: Rangenet++: Fast and accurate lidar semantic segmentation. In: 2019 IEEE/RSJ international conference on intelligent robots and systems (IROS). pp. 4213–4220. IEEE (2019)
11. Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al.: Dinov2: Learning robust visual features without supervision. arXiv preprint arXiv:2304.07193 (2023)
12. Park, D., Ambrus, R., Guizilini, V., Li, J., Gaidon, A.: Is pseudo-lidar needed for monocular 3d object detection? In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 3142–3152 (2021)
13. Philion, J., Fidler, S.: Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In: European conference on computer vision. pp. 194–210. Springer (2020)
14. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 652–660 (2017)
15. Ranftl, R., Bochkovskiy, A., Koltun, V.: Vision transformers for dense prediction. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 12179–12188 (2021)
16. Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., Koltun, V.: Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. IEEE transactions on pattern analysis and machine intelligence **44**(3), 1623–1637 (2020)

17. Wang, C.Y., Liao, H.Y.M., Wu, Y.H., Chen, P.Y., Hsieh, J.W., Yeh, I.H.: Cspnet: A new backbone that can enhance learning capability of cnn. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops. pp. 390–391 (2020)
18. Wang, S., Liu, Y., Wang, T., Li, Y., Zhang, X.: Exploring object-centric temporal modeling for efficient multi-view 3d object detection. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 3621–3631 (2023)
19. Wang, T., Zhu, X., Pang, J., Lin, D.: Fcos3d: Fully convolutional one-stage monocular 3d object detection. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 913–922 (2021)
20. Wang, Y., Guizilini, V.C., Zhang, T., Wang, Y., Zhao, H., Solomon, J.: Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. In: Conference on robot learning. pp. 180–191. PMLR (2022)
21. Yang, L., Kang, B., Huang, Z., Zhao, Z., Xu, X., Feng, J., Zhao, H.: Depth anything V2. In: Advances in Neural Information Processing Systems (NeurIPS) (2024)
22. Yin, T., Zhou, X., Krahenbuhl, P.: Center-based 3d object detection and tracking. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11784–11793 (2021)
23. Zhou, Y., Tuzel, O.: Voxelnet: End-to-end learning for point cloud based 3d object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4490–4499 (2018)