RaGS: Unleashing 3D Gaussian Splatting from 4D Radar and Monocular Cue for 3D Object Detection

Supplementary Material

1. Loss Calculation

Our training proceeds in two stages. The pretraining stage focuses on learning robust depth and perspective segmentation features from multi-modal inputs, while the joint training stage optimizes for 3D object detection augmented by auxiliary rendering losses.

1.1. Pretraining stage

On one hand, the pretraining stage loss consists of the depth loss $\mathcal{L}_{\text{depth}}$ and the perspective segmentation loss $\mathcal{L}_{\text{seg}}$, which can be expressed as

$$\mathcal{L}_{\text{pretrain}} = \mathcal{L}_{\text{depth}} + \mathcal{L}_{\text{seg}}. \tag{1}$$

As described in the main body, we first interact the raw image feature $\mathbf{C}$ with sparse radar depth $\mathbf{S}$ to obtain the discrete depth probability $\mathbf{D}^{\text{prob.}}$ and the enhanced image feature $\mathbf{F}^{\text{2D}}$, as $(\mathbf{F}^{\text{2D}}, \mathbf{D}^{\text{prob.}}) = \texttt{Conv}(\texttt{Concat}(\mathbf{C}, \mathbf{S}))$. Following the approach of BEVDepth, we use the Kullback-Leibler divergence loss φ between the predicted depth and the Gaussian-distributed ground-truth LiDAR depth $\mathbf{D}^{\text{prob.}}_{\text{gt}}$ to supervise depth estimation, formulated as

$$\mathcal{L}_{\text{depth}} = \sum_{u=1}^{H}\sum_{v=1}^{W} \varphi(\mathbf{D}^{\text{prob.}}(u,v), \mathbf{D}^{\text{prob.}}_{\text{gt}}(u,v)). \tag{2}$$

For the foreground mask generation, we apply the binary cross-entropy (BCE) loss for supervision, defined as

$$\begin{aligned}\mathcal{L}_{\text{seg}} = \mathcal{S}(\mathbf{L}, \mathbf{M}_{\text{gt}}) = \sum_{u=1}^{H}\sum_{v=1}^{W} -\mathbf{M}_{\text{gt}}(u,v)\cdot\log(\mathbf{L}(u,v)) \\ -(1-\mathbf{M}_{\text{gt}}(u,v))\cdot\log(1-\mathbf{L}(u,v)),\end{aligned} \tag{3}$$

where $\mathbf{M}_{\text{gt}}(u,v) = \frac{\mathbf{M}_1(u,v)+\mathbf{M}_2(u,v)}{2}$, with $\mathbf{M}_1$ and $\mathbf{M}_2$ being the processed results from Detectron2 [2] and the ground-truth 2D bounding box mask, respectively, and $\mathbf{L}$ representing the segmentation output from the lightweight segmentation network [5].

1.2. Joint Training Stage

On the other hand, for joint training stage, we adopt the 3D object detection loss $\mathcal{L}_{\text{det}}$ following [6], along with two auxiliary rendering losses, rendered depth loss in perspective view $\mathcal{L}_{\text{depth\_render}}$ and segmentation loss in bird's-eye view $\mathcal{L}_{\text{seg\_render}}$. The total training objective is defined as

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{det}} + \lambda(\mathcal{L}_{\text{depth\_render}} + \mathcal{L}_{\text{seg\_render}}), \tag{4}$$

where λ is the hyperparameter balancing detection and auxiliary tasks. In this work, we set $\lambda = 0.1$. The ground truth BEV segmentation can be easily inferred from the 3D bounding boxes. By directly using the center coordinates, dimensions (length and width), and rotation angle, the ground truth $\mathbf{M}^{\text{BEV}}_{\text{gt}}$ can be obtained. We first render the last layer of IMA output Gaussian $\mathbf{G}$ to perspective depth using `gsplat` [4], and supervise the procedure with MSE loss:

$$\mathcal{L}_{\text{depth\_render}} = \|\texttt{gsplat}(\mathbf{G}) - \mathbf{D}_{\text{gt}}\|_2^2, \tag{5}$$

where $\mathbf{D}_{\text{gt}}$ is the metric depth version of the LiDAR depth. The segmentation loss in BEV is then computed as

$$\mathcal{L}_{\text{seg\_render}} = \mathcal{S}(\mathbf{M}^{\text{BEV}}, \mathbf{M}^{\text{BEV}}_{\text{gt}}), \tag{6}$$

where $\mathbf{M}_{\text{BEV}}$ denotes the segmentation results from $\mathbf{F}^{\text{gs}}$ through the lightweight segmentation network [5].

2. Implementation Details

Network Settings. For the VoD dataset, the point cloud range is limited to (0, 51.2) m, (−25.6, 25.6) m, and (−3, 2.76) m along the X-, Y-, and Z-axes, respectively. We use radar point clouds accumulated over 5 scans as input. The raw radar point feature is $\left[x, y, z, RCS, v_r, v_{rc}, t\right]^\top$, where RCS denotes radar cross section, v_r is relative radial Doppler velocity, v_{rc} is absolute radial doppler velocity, and t is the scan identifier. In this work, we use $[x, y, z, RCS, v_{rc}]$ as input feature. For the TJ4DRadSet dataset, the point cloud range is limited to (0, 69.12) m, (−39.68, 39.68) m, and (−4, 2) m along the X-, Y-, and Z-axes, respectively and single-frame radar point clouds are used as input. The raw radar point feature is $\left[x, y, z, v_r, r, SNR, \alpha, \beta\right]^\top$, where r is the detection range, SNR (in dB) represents signal-to-noise ratio, and α and β are horizontal and vertical angles, respectively. In consistence with VoD dataset, we use $[x, y, z, SNR, v_{rc}]$ as input feature. For both datasets, the voxel is set as a cube of size 0.16m. The image is resized to 800×1280 for VoD dataset and 640×800 for TJ4DRadSet dataset, while the number of discretized depth bins is set to 56 for VoD and 72 for TJ4DRadSet. The anchor size for both datasets are kept the same as in [6]. For the OmniHD-Scenes dataset, we constrain the point cloud range to $(-60, 60)$ m, $(-40, 40)$ m, and $(-3, 5)$ m along the X-, Y-, and Z-axes, respectively. The radar input consists of 3-frame accumulated point clouds, with each point characterized by a feature vector $\left[x, y, z, Power, SNR, v_{xr}, v_{yr}\right]^\top$. All six camera im-

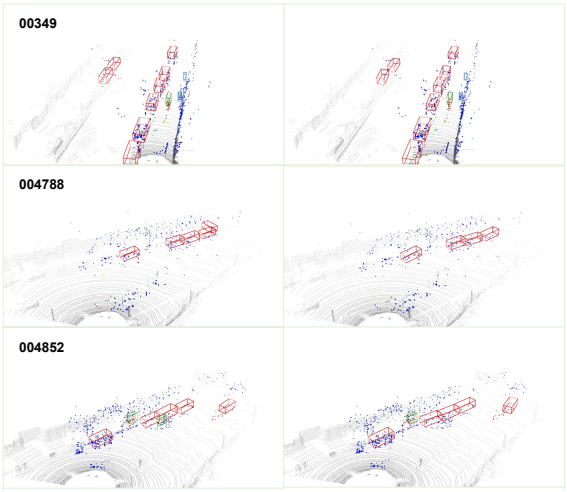

Figure 1. Visualization of RaGS on View-of-Delft dataset.

ages are resized to 544×960. The BEV feature map size $H \times W$ is 160×240, depth bins is set to 59.

Training details. We implement our model based on the MMDetection3D framework. The models are trained on 4 NVIDIA GeForce RTX 4090 GPUs with a batch size of 4 per GPU. Our training process consisted of two stages. First, we train the image branch for depth estimation and segmentation and radar branch for 3D object detection, respectively. The image branch inherites weights from the model pretrained on the COCO and KITTI datasets following [6] for 12 epoches, while we train radar branch weights from scratch. Second, we train our model using the weights inherited from the above streams with image feature extractor frozen for faster training.

During the fusion training, we use the AdamW optimizer with an initial learning rate of 8×10^{-4} and trained the model for 24 epochs. We adopt image data augmentations including random cropping, random scaling, random flipping, and random rotation, and also adopt BEV data augmentations including random scaling, random flipping, and random rotation.

3. Detection Performance Comparison.

Table 1 presents a comprehensive comparison between our RaGS and the baseline LXL [3] on the validation set of VoD [1]. It can be clearly observed that RaGS consistently outperforms LXL across all object categories (Car, Pedestrian, Cyclist) and under all spatial evaluation formats (Image, BEV, 3D). In the Image-level evaluation, our method achieves higher true positives (TP ↑) and notably fewer false negatives (FN ↓), indicating stronger recognition and recall of small or distant targets. Under the BEV format, RaGS still maintains clear advantages, especially for Pedestrian detection, which demonstrates that our Gaussian-field representation yields more accurate horizontal localization and contour delineation than grid-based fusion. Furthermore, in the most challenging 3D evaluation, RaGS surpasses LXL by a significant margin (e.g., 1126 vs. 1076 TP for Cars and 970 vs. 930 TP for Pedestrians), verifying its effectiveness in achieving spatially consistent depth reasoning from radar-camera cues. Overall, these improvements highlight that our design effectively enhances both semantic understanding and geometric alignment, leading to superior detection accuracy and robustness across different object scales and viewpoints.

Metric	Eval	Car			Pedestrian			Cyclist		
		TP↑	FP↓	FN↓	TP↑	FP↓	FN↓	TP↑	FP↓	FN↓
Image	LXL	2263	2416	2028	2052	2218	2239	1221	2869	3070
	Ours	**2513**	**1972**	**1778**	**2148**	**2130**	**2143**	**1400**	**2842**	**2891**
BEV	LXL	1848	1631	1901	**1331**	2172	2418	917	**2534**	2832
	Ours	**1871**	**1602**	**1878**	1318	**2155**	**2431**	**933**	2540	**2816**
3D	LXL	1076	**497**	358	930	**442**	504	785	**527**	649
	Ours	**1126**	533	**308**	**970**	689	**464**	**840**	536	**594**

Table 1. Comparison results of our RaGS and the baseline [3] on the validation set of VoD [1]. In the Eval column, 3D, BEV, and Image refer to different spatial evaluation formats.

References

[1] Andras Palffy, Ewoud Pool, Srimannarayana Baratam, Julian FP Kooij, and Dariu M Gavrila. Multi-Class Road User Detection with 3+1D Radar in the View-of-Delft Dataset. *IEEE Robotics and Automation Letters*, 7(2):4961–4968, 2022. 2

[2] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 1

[3] Weiyi Xiong, Jianan Liu, Tao Huang, Qing-Long Han, Yuxuan Xia, and Bing Zhu. LXL: LiDAR Excluded Lean 3D Object Detection with 4D Imaging Radar and Camera Fusion. *IEEE Transactions on Intelligent Vehicles*, 9(1):79–92, 2024. 2

[4] Vickie Ye et al. gsplat: An open-source library for gaussian splatting. *Journal of Machine Learning Research*, 26(34):1–17, 2025. 1

[5] Xiaohan Zhang, Xue Zhang, Si-Yuan Cao, Beinan Yu, Chenghao Zhang, and Hui-Liang Shen. MRF^{3}Net: An Infrared Small Target Detection Network Using Multireceptive Field Perception and Effective Feature Fusion. *IEEE Transactions on Geoscience and Remote Sensing*, 2024. 1

[6] Lianqing Zheng, Sen Li, Bin Tan, Long Yang, Sihan Chen, Libo Huang, Jie Bai, Xichan Zhu, and Zhixiong Ma. RCFusion: Fusing 4-D Radar and Camera with Bird's-Eye View Features for 3-D Object Detection. *IEEE Transactions on Instrumentation and Measurement*, 72:1–14, 2023. 1, 2