Toward Real-world BEV Perception: Depth Uncertainty Estimation via Gaussian Splatting

Supplementary Material

A. BEV Feature with Gaussian Splatting

A.1. Projection

As described in Gaussian Splatting [4], to render 3D Gaussians in image space, we project their covariance matrix Σ using a viewing transformation W and the Jacobian J of the affine approximation of the projective transformation:

$$\Sigma' = JW\Sigma W^T J^T,$$

where Σ' is the projected covariance matrix. Projecting to the Bird's Eye View (BEV) image space simplifies the process significantly because the z-axis can be ignored. The BEV scaling matrix, S_{BEV}, scales the 3D coordinates to the 2D BEV plane and is defined as:

$$S_{\text{BEV}} = \begin{bmatrix} 0 & \text{scale}_x \\ \text{scale}_y & 0 \end{bmatrix}.$$

Note that the x and y axes of the 3D coordinates are swapped when mapping to the BEV plane. The projection of the covariance matrix Σ into the BEV image space using the scaling matrix S_{BEV} is then given by:

$$\Sigma' = S_{\text{BEV}} \Sigma_{xy} S_{\text{BEV}}^T,$$

where Σ_{xy} is the 2×2 submatrix of Σ corresponding to the x and y axes. Therefore, the projected covariance matrix Σ' becomes:

$$\Sigma' = \begin{bmatrix} \Sigma_{22} \cdot \text{scale}_x^2 & \Sigma_{21} \cdot \text{scale}_x \cdot \text{scale}_y \\ \Sigma_{12} \cdot \text{scale}_x \cdot \text{scale}_y & \Sigma_{11} \cdot \text{scale}_y^2 \end{bmatrix}.$$

A.2. Gradient Computation

Next, we compute the derivative of the loss L with respect to the covariance matrix Σ, denoted as $\frac{\partial L}{\partial \Sigma}$. Let Σ' be defined as:

$$\Sigma' = \begin{bmatrix} a & b \\ b & c \end{bmatrix},$$

where:

$$\begin{aligned} a &= \Sigma_{22} \cdot \text{scale}_x^2, \\ b &= \Sigma_{21} \cdot \text{scale}_x \cdot \text{scale}_y, \\ c &= \Sigma_{11} \cdot \text{scale}_y^2. \end{aligned}$$

Using the chain rule, the gradient $\frac{\partial L}{\partial \Sigma_{ij}}$ is given by:

$$\frac{\partial L}{\partial \Sigma_{ij}} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial \Sigma_{ij}} + \frac{\partial L}{\partial b}\frac{\partial b}{\partial \Sigma_{ij}} + \frac{\partial L}{\partial c}\frac{\partial c}{\partial \Sigma_{ij}}.$$

We compute the partial derivatives:
- For Σ_{11}:

$$\frac{\partial a}{\partial \Sigma_{11}} = 0, \quad \frac{\partial b}{\partial \Sigma_{11}} = 0, \quad \frac{\partial c}{\partial \Sigma_{11}} = \text{scale}_y^2.$$

Therefore:

$$\frac{\partial L}{\partial \Sigma_{11}} = \frac{\partial L}{\partial c} \cdot \text{scale}_y^2.$$

- For Σ_{12} (since Σ is symmetric, $\Sigma_{12} = \Sigma_{21}$):

$$\frac{\partial a}{\partial \Sigma_{12}} = 0, \quad \frac{\partial b}{\partial \Sigma_{12}} = \text{scale}_x \cdot \text{scale}_y, \quad \frac{\partial c}{\partial \Sigma_{12}} = 0.$$

Therefore:

$$\frac{\partial L}{\partial \Sigma_{12}} = \frac{\partial L}{\partial b} \cdot \text{scale}_x \cdot \text{scale}_y.$$

- For Σ_{22}:

$$\frac{\partial a}{\partial \Sigma_{22}} = \text{scale}_x^2, \quad \frac{\partial b}{\partial \Sigma_{22}} = 0, \quad \frac{\partial c}{\partial \Sigma_{22}} = 0.$$

Therefore:

$$\frac{\partial L}{\partial \Sigma_{22}} = \frac{\partial L}{\partial a} \cdot \text{scale}_x^2.$$

Hence, the gradient $\frac{\partial L}{\partial \Sigma}$ is:

$$\frac{\partial L}{\partial \Sigma} = \begin{bmatrix} \frac{\partial L}{\partial c} \cdot \text{scale}_y^2 & \frac{\partial L}{\partial b} \cdot \text{scale}_x \cdot \text{scale}_y \\ \frac{\partial L}{\partial b} \cdot \text{scale}_x \cdot \text{scale}_y & \frac{\partial L}{\partial a} \cdot \text{scale}_x^2 \end{bmatrix}.$$

B. Ablation Study on Depth Prediction

We conduct an ablation study to evaluate the impact of various depth settings, including depth ranges and bin sizes, on model performance, as shown in Table 1. The result demonstrates that the model's performance is relatively stable across different depth range and bin size configurations, with the setting (1, 61) and bin size 64 providing a slightly higher IoU. This robustness simplifies parameter selection in practice.

Table 1. **Ablation study on depth settings.** The first depth range, $(1, 61)$, where the minimum depth is set to 1 meter and the maximum depth is set to 61 meters, corresponds to settings used in the paper. The second depth range, $(0.5, 71)$, represents an extended range capturing the minimum and maximum depth values in the BEV plane. Bin sizes $(32, 64, 128)$ are also evaluated for their impact on IoU for the vehicle class.

Depth range	Bin size B	IoU Vehicle
$(1, 61)$	32	37.7
$(1, 61)$	64	**38.0**
$(1, 61)$	128	37.8
$(0.5, 71)$	32	37.7
$(0.5, 71)$	64	37.9
$(0.5, 71)$	128	37.6

Table 2. **Ablation study on multi-scale BEV features.** We vary the number of multi-scale stages and evaluate the effect on IoU (vehicle class), FPS, and memory usage (in GiB). Each stage corresponds to a specific BEV resolution: 50×50, 100×100, and 200×200.

50×50	100×100	200×200	FPS	Mem (GiB)	IoU Vehicle
		✓	**85.0**	0.374	36.6
	✓	✓	77.0	0.363	37.4
✓	✓	✓	67.9	**0.360**	**38.0**

C. Multi-scale BEV Features

The multi-scale BEV feature extraction layer enhances the representation of BEV features by progressively increasing resolution across stages. The process starts with a learnable BEV embedding initialized at the lowest resolution. At each stage, the rendered BEV features are fused with the BEV embedding and upsampled by a factor of 2. This hierarchical approach allows the model to capture finer spatial details while balancing computational efficiency. An ablation study was conducted to evaluate the effect of using one, two, or three multi-scale stages, as shown in Table 2. The results demonstrate that increasing the number of stages improves performance, with the highest IoU of 38.0 achieved using three stages. However, this comes at the cost of reduced FPS, which drops from 85.0 for one stage to 67.9 for three stages. Memory usage slightly decreases with more stages, from 0.374 GiB to 0.360 GiB, due to the hierarchical refinement reducing reliance on larger resolution BEV embeddings.

These results suggest that three stages offer the best accuracy while balancing memory and speed. However, configurations with fewer stages can be selected for applications prioritizing higher FPS.

D. More Results on Map Segmentation and 3D Object Detection

Beyond instance-level BEV segmentation, we further evaluate our method on **map segmentation** and **3D object detection** to assess its generalization capability. For map segmentation, we predict drivable areas, pedestrian crossings, walkways, and road dividers, following the same experimental setup as prior works. As shown in Table 3, our method achieves competitive performance.

For 3D object detection, we integrate a detection head directly into the BEV features, following the approach of BEVFormer. We evaluate performance using mean Average Precision (mAP) and nuScenes Detection Score (NDS). As shown in Table 4, our method extends beyond BEV segmentation and is also applicable to detection tasks. These results further demonstrate the versatility of our approach across different autonomous driving tasks.

Table 3. **Map Segmentation Comparisons.** We evaluate our method for common map classes on nuScenes.

Method	Drivable	Ped. Cross.	Walkway	Divider
LSS [6]	75.4	38.8	46.3	36.5
CVT [7]	74.3	36.8	39.9	29.4
Ours	**76.3**	**46.3**	**50.2**	**38.7**

Table 4. **3D Detection Performance on nuScenes Validation.**

Method	NDS	mAP
BEVDet [3]	35.0	28.3
BEVFormer [5]	35.4	25.2
Ours	34.0	26.6

Table 5. **Comparison of Submodule Execution Time.** All times are measured in milliseconds. The "VT" column represents the view-transformation module (BEV encoder). All measurements are conducted on an RTX 4090 GPU.

Method	Backbone	Neck	VT	Head	Total
PointBEV [1]	5.6	0.21	13.47	1.95	21.23
FIERY static [2]	5.71	—	36.46	1.18	43.35
CVT [7]	5.68	—	2.17	0.71	8.02
Ours	5.92	0.51	1.46	1.13	9.02

E. Submodule Speed Analysis

To further evaluate the efficiency of our approach, we analyze the runtime performance of key submodules and compare them with baseline methods. We break down the inference time into different processing stages, including the backbone, neck, view transformation, and head. Table 5

reports the speed of each submodule. Our method demonstrates a significant speed advantage in the view transformation stage while maintaining comparable efficiency to projection-based methods.

References

[1] Loick Chambon, Eloi Zablocki, Mickael Chen, Florent Bartoccioni, Patrick Perez, and Matthieu Cord. Pointbev: A sparse approach to bev predictions. In *CVPR*, 2024. 2

[2] Anthony Hu, Zak Murez, Nikhil Mohan, Sofía Dudas, Jeffrey Hawke, Vijay Badrinarayanan, Roberto Cipolla, and Alex Kendall. FIERY: Future instance segmentation in bird's-eye view from surround monocular cameras. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. 2

[3] Junjie Huang, Guan Huang, Zheng Zhu, Ye Yun, and Dalong Du. Bevdet: High-performance multi-camera 3d object detection in bird-eye-view. *arXiv preprint arXiv:2112.11790*, 2021. 2

[4] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023. 1

[5] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Yu Qiao, and Jifeng Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers, 2022. 2

[6] Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *Proceedings of the European Conference on Computer Vision*, 2020. 2

[7] Brady Zhou and Philipp Krähenbühl. Cross-view transformers for real-time map-view semantic segmentation. In *CVPR*, 2022. 2