GaussianDet3D: Bridging Gaussian Splatting and Sparse LiDAR Detection for Multi-View 3D Object Detection

Malaz Tamim[1,3⋆] Wenzhao Zheng[2] Johannes Meier[1,3,4]
Daniel Cremers[1,3] Kurt Keutzer[2]

[1]Technical University of Munich [2]UC Berkeley
[3]Munich Center for Machine Learning [4]DeepScenario

Abstract

Accurate 3D object detection from cameras alone remains a fundamental challenge in autonomous driving, particularly for precise localization and velocity estimation, two metrics critical for safe trajectory planning and collision avoidance. Existing camera-based methods lift image features into dense Bird's-Eye View (BEV) grids, which struggle to capture fine-grained geometry and motion cues. We present ***GaussianDet3D****, the first method, to the best of our knowledge, to apply 3D Gaussian Splatting from multi-view images to 3D object detection in the context of autonomous driving, treating predicted Gaussian primitives as a pseudo-LiDAR point cloud fed into a sparse LiDAR detector. Unlike a LiDAR point, which carries only coordinates and intensity, each Gaussian encodes parameters capturing geometry, orientation, opacity, and per-class semantic distributions. By aggregating Gaussian point clouds across multiple frames, GaussianDet3D captures temporal motion cues that enable precise velocity estimation. On the nuScenes benchmark, GaussianDet3D achieves state-of-the-art translation and velocity errors among all camera-based methods, outperforming BEVFormer by* ***8.1%*** *and* ***13.1%****, respectively, while remaining competitive in overall detection score. These results demonstrate that Gaussian Splatting provides a geometrically precise, semantically rich representation that bridges the gap between image-based perception and LiDAR-quality spatial reasoning, particularly for the localization and motion estimation tasks most critical to autonomous driving safety. The project page is available at https://malaztamim.com/GaussianDet3D.*

1. Introduction

Autonomous driving requires accurate 3D perception of surrounding objects, including their positions, sizes, orientations, and velocities, to enable safe trajectory planning and collision avoidance. While LiDAR sensors deliver this with high geometric precision, their cost and complexity have driven significant interest in camera-based 3D detection as a scalable alternative.

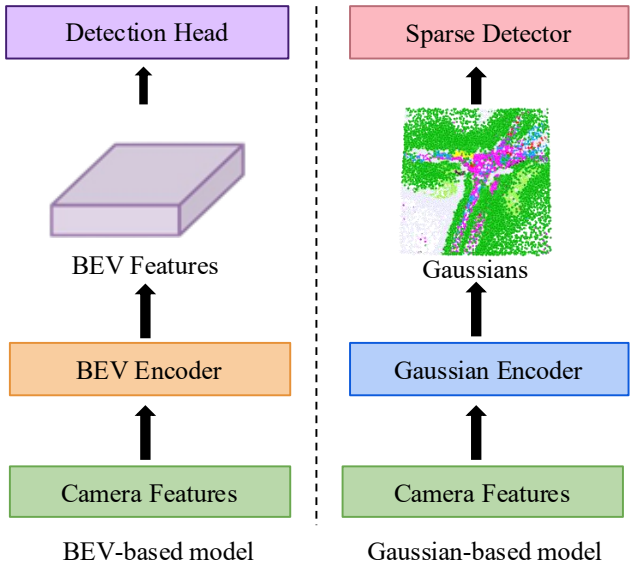


Figure 1. Comparison of BEV-based and Gaussian-based detection pipelines. BEV-based methods lift camera features into a dense grid before detection. GaussianDet3D instead predicts sparse Gaussian primitives and feeds them directly into a sparse detector, enabling richer per-point representations and precise localization.

As illustrated in Figure 1, the dominant paradigm for camera-based 3D perception is to lift image features into an intermediate 3D representation, then detect objects within it. BEV-based methods pass these dense grid features into a dedicated detection head designed specifically for BEV-structured inputs. GaussianDet3D takes a different approach: instead of a dense grid, the scene is represented as

⋆Corresponding author: malaz.tamim@tum.de

a sparse set of Gaussian primitives that can be fed directly into any LiDAR-style sparse detector without architectural modification.

Existing methods overwhelmingly adopt dense grid-based representations for this purpose: Bird's-Eye View (BEV) feature maps [8, 16], voxel grids [2], or tri-perspective views [10]. While effective, these representations share a fundamental limitation, they discretize continuous 3D space into fixed uniform grids, allocating equal resources to empty and occupied regions alike. This leads to two key limitations: (1) high memory consumption from modeling predominantly empty space, and (2) loss of fine-grained geometry at grid boundaries.

3D Gaussian Splatting [14] offers a compelling alternative intermediate representation: sparse, geometrically precise, and naturally compatible with high-performance sparse 3D detectors. Inspired by GaussianFormer [11] and GaussianFormer-2 [12], which demonstrated that scenes can be compactly described by sparse sets of 3D Gaussian primitives predicted from multi-view images, we propose treating these Gaussians directly as input to a 3D object detector, and demonstrate that they constitute a superior intermediate representation for localization and motion estimation.

We present **GaussianDet3D**, which treats each predicted Gaussian primitive as a pseudo-LiDAR point, feeding the resulting Gaussian point cloud directly into FSD V2 [6], a state-of-the-art fully sparse LiDAR detector. Crucially, each Gaussian encodes geometry, orientation, opacity, and semantic class distributions. By aggregating Gaussian point clouds across multiple consecutive frames, GaussianDet3D further captures temporal motion cues for accurate velocity estimation without explicit object tracking.

On the nuScenes benchmark [1] test set, GaussianDet3D achieves state-of-the-art translation error (mATE = 0.580) and velocity error (mAVE = 0.378) among all camera-based methods, outperforming BEVFormer [16] by 8.1% and 13.1% respectively, while also achieving state-of-the-art attribute estimation (mAAE = 0.129).

Our contributions are:

- We present the first method, to the best of our knowledge, to apply 3D Gaussian Splatting from multi-view images to 3D object detection in autonomous driving.
- We propose treating Gaussian primitives as pseudo-LiDAR points, enabling direct use of high-performance sparse LiDAR detectors for camera-only perception.
- We demonstrate that Gaussians provide richer geometric and semantic information than conventional representations, achieving state-of-the-art localization and velocity estimation on nuScenes.

2. Related Work

Camera-based 3D object detection. Early camera-based 3D detectors operate directly in image space, predicting depth and 3D boxes from perspective features [18, 21, 22]. A more scalable paradigm lifts image features into Bird's-Eye View (BEV) grids using depth estimation or spatial transformers [8, 9, 16]. BEVFormer [16] extends this with temporal attention across frames, achieving strong overall detection scores. Alternative discretizations include voxel grids [2] and tri-perspective views [10]. Despite their effectiveness, all these methods share a fundamental limitation: dense grid representations allocate equal capacity to empty and occupied regions, losing fine-grained geometry.

3D Gaussian Splatting for scene understanding. 3D Gaussian Splatting was introduced by Kerbl et al. [14] as a real-time neural rendering technique, representing scenes as collections of anisotropic Gaussian primitives optimized for photorealistic novel view synthesis. The ability to predict and refine Gaussian primitives from images in a fully differentiable pipeline opened new directions in 3D scene understanding. In autonomous driving, SplatAD [7] demonstrated that 3D Gaussians can synthesize high-fidelity LiDAR and camera data for scene reconstruction, and DrivingGaussian [24] proposed compound Gaussian representations for dynamic driving scene generation. For perception tasks, GaussTR [13] showed that Gaussian Transformers aligned with foundation model features can perform self-supervised 3D spatial understanding, GaussianLSS [19] modeled depth uncertainty as Gaussian distributions to improve robustness in BEV perception, refined this with a probabilistic lifting strategy that improves geometric accuracy. However, all prior work in autonomous driving decodes the Gaussian representation back into dense occupancy grids or renders novel views, none have explored using Gaussian Splatting for 3D object detection from multi-view images. GaussianDet3D closes this gap.

LiDAR-based sparse 3D detection. PointPillars [15] introduced efficient pillar-based voxelization for fast 3D detection from point clouds, establishing a strong and widely adopted baseline. FocalFormer3D [4] addresses the complementary challenge of false negatives through Hard Instance Probing, a multi-stage heatmap encoding strategy that progressively identifies missed detections and guides the model to focus on hard instances, coupled with a box-level deformable decoder for candidate refinement. FSD V2 [6] advances this with a fully sparse 3D convolutional backbone and virtual voxelization, achieving state-of-the-art performance on LiDAR benchmarks. These architectures are designed to exploit the sparsity and geometric precision of LiDAR point clouds.

3. Method

We propose **GaussianDet3D**, a camera-based 3D object detection pipeline that reconstructs a scene as a set of 3D Gaussian primitives from multi-view images and feeds them directly into a sparse LiDAR detector. Figure 2 provides an overview of the complete pipeline, consisting of four components: an image encoder, a Gaussian lifter, a Gaussian encoder, and FSD V2.

3.1. Problem Formulation

Given N synchronized RGB images $\mathcal{I} = \{\mathbf{I}_i \in \mathbb{R}^{3\times H\times W}\}$ with intrinsics $\mathbf{K}_i$ and extrinsics $\mathbf{T}_i$, the goal is to predict 3D bounding boxes $\mathcal{B} = \{b_j\}_{j=1}^{M}$, each encoding center, dimensions, orientation, class, and velocity. Unlike prior methods that lift features to dense BEV grids or voxels, we introduce an intermediate sparse 3D Gaussian representation $\mathcal{G} = \{G_p\}_{p=1}^{P}$ and apply a LiDAR-style sparse detector to obtain $\mathcal{B}$.

3.2. 3D Gaussian Scene Representation

We represent the scene with P Gaussian primitives, each defined as:

$$G_i = \left[m_i,\, s_i,\, r_i,\, a_i,\, c_i\right] \in \mathbb{R}^{28}, \tag{1}$$

where $m_i \in \mathbb{R}^3$ is the 3D mean, $s_i \in \mathbb{R}^3$ the axis-aligned scale, $r_i \in \mathbb{R}^4$ a unit quaternion, $a_i \in \mathbb{R}$ the opacity, and $c_i \in \mathbb{R}^{17}$ unnormalized semantic logits. The full covariance is $\Sigma_i = R(r_i)\,\mathrm{diag}(s_i)^2 R(r_i)^\top$, giving each Gaussian an anisotropic spatial extent that adapts to local scene geometry. Unlike dense voxel grids, this representation is sparse by construction and geometrically adaptive, properties that make it naturally compatible with LiDAR detectors as demonstrated in Section 3.6.

3.3. Image Encoder

We adopt a ResNet-101-DCN backbone with an FPN neck following SurroundOcc [23], processing six surround-view images at 1600×864 resolution into multi-scale features aggregated to $C{=}128$ channels. Deformable convolutions in the last two stages improve robustness to perspective distortion. The backbone is initialized from FCOS3D pretraining with a reduced learning rate multiplier of 0.1 during joint training.

3.4. Gaussian Lifter

The lifter converts image features into an initial Gaussian set $\mathcal{G}^{(0)}$ through three steps. First, a lightweight projection head predicts pixel-aligned occupancy distributions along each ray by sampling R reference points at equal intervals, following GaussianFormer-2 [12]. Second, each pixel is backprojected to 3D using the most probable occupied position p^* along its ray. Third, Farthest Point Sampling (FPS) selects exactly P spatially diverse anchor positions from valid backprojected points. Each anchor is paired with learnable Gaussian parameters $(s_i^{(0)}, r_i^{(0)}, a_i^{(0)}, c_i^{(0)})$ shared across anchors and refined downstream.

3.5. Gaussian Encoder

The encoder iteratively refines $\mathcal{G}^{(0)}$ through stacked blocks, each performing three operations: (1) sparse 3D convolution on voxelized Gaussian means to propagate geometric context across spatial neighbors; (2) deformable cross-attention sampling multi-view image features around each Gaussian mean to enrich geometric and semantic features; and (3) MLP refinement updating Gaussian parameters, with the mean updated residually for spatial coherence and all other parameters replaced directly:

$$m_i^{(*)} = m_i^{(0)} + \Delta m_i, \quad (s_i^{(*)}, r_i^{(*)}, a_i^{(*)}, c_i^{(*)}) = \mathrm{MLP}(h_i''). \tag{2}$$

The output $\mathcal{G}^{(*)}$ is geometrically consistent and semantically rich.

3.6. Gaussian Point Cloud Interface to FSD V2

Each refined Gaussian is reinterpreted as a pseudo-LiDAR point, preserving its full geometric and semantic parameters all the way to the final detection stage without any intermediate densification or grid projection. The mean $m_i^{(*)}$ serves as the 3D coordinate, and the remaining parameters form the feature vector:

$$f_i = \left[s_i^{(*)},\, r_i^{(*)},\, a_i^{(*)},\, c_i^{(*)}\right] \in \mathbb{R}^{25}, \tag{3}$$

yielding a Gaussian point cloud $\mathcal{P} \in \mathbb{R}^{P\times 28}$ passed directly to FSD V2 without architectural modification.

This interface is powerful for two reasons. First, while a LiDAR point carries only $(x, y, z, \text{intensity})$, each Gaussian point carries parameters encoding geometry, shape, opacity, and per-class semantics, a significantly richer input. Second, FSD V2's fully sparse architecture is natively suited to irregular sparse point sets, making it an ideal consumer of Gaussian point clouds. FSD V2 applies virtual voxelization, a sparse 3D convolutional backbone, and an anchor-free detection head predicting objectness, class, center offset, box dimensions, orientation, and velocity for each voxel, with final predictions merged via NMS in BEV.

Following the official FSD V2 loss [6], we apply Focal Loss [17] for classification and ℓ_1 losses for box regression, with an additional velocity term:

$$\mathcal{L}_{\text{det}} = \mathcal{L}_{\text{focal}} + \mathcal{L}_{\ell_1}^{\text{box}} + \mathcal{L}_{\ell_1}^{\text{vel}}. \tag{4}$$

3.7. Temporal Aggregation

To improve velocity estimation, we aggregate Gaussian point clouds from K consecutive keyframes ($\Delta t{=}0.5$s in

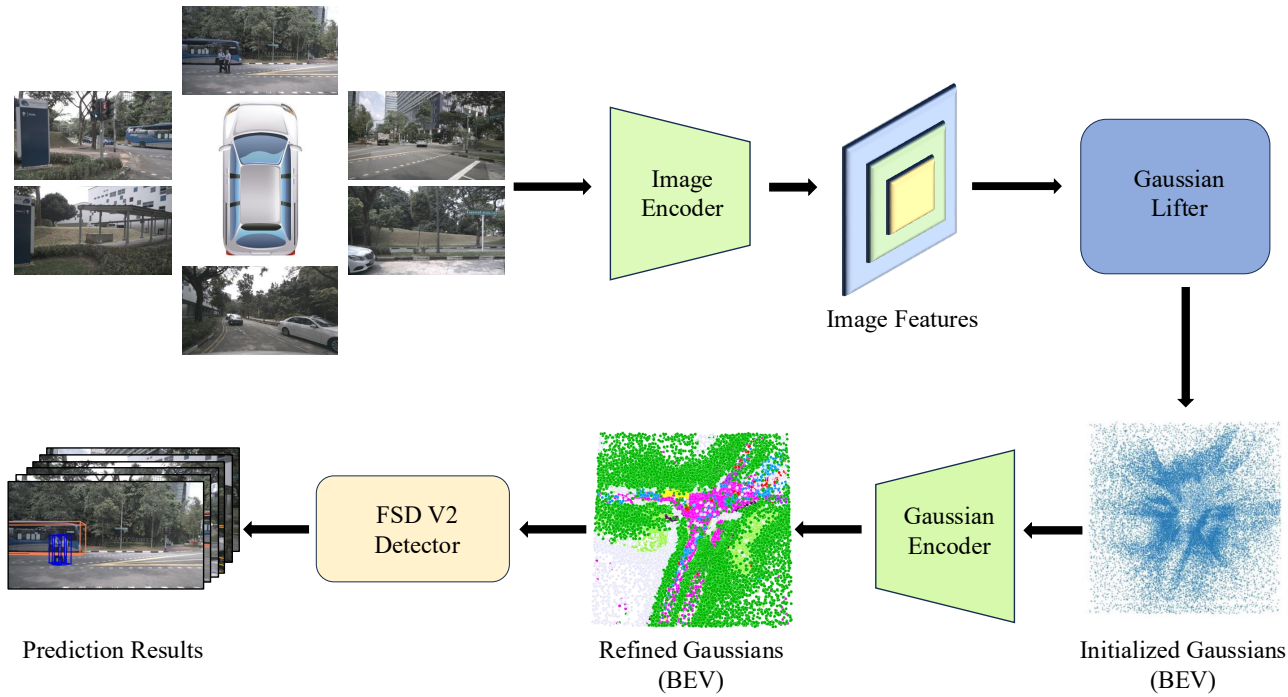


Figure 2. Overview of the GaussianDet3D pipeline. Multi-view images are encoded into deep feature maps, lifted into initialized 3D Gaussians, refined by the Gaussian encoder, and finally interpreted as a Gaussian point cloud fed to FSD V2 for 3D bounding box prediction.

nuScenes), all transformed into the current ego-vehicle frame:

$$\mathcal{P}_t^{\text{multi}} = \mathcal{P}_t \cup \mathcal{P}_{t-\Delta t} \cup \cdots \cup \mathcal{P}_{t-(K-1)\Delta t}. \tag{5}$$

A dedicated timestamp channel encodes relative frame time (0 for current, $t-t_k$ seconds for prior frames), enabling FSD V2 to learn motion-sensitive features and estimate velocities without explicit tracking. We experiment with $K \in \{1, 2, 4\}$ frames and $P \in \{6400, 12800, 25600\}$ Gaussians per frame.

4. Experiments

4.1. Dataset and Metrics

We evaluate on nuScenes [1], a large-scale autonomous driving benchmark containing 1,000 driving scenes recorded under diverse weather and lighting conditions, with 28,130 keyframes annotated at 2 Hz across 10 object classes: car, truck, bus, trailer, construction vehicle, pedestrian, motorcycle, bicycle, traffic cone, and barrier. Each scene provides synchronized data from six surround-view RGB cameras and a 32-beam LiDAR sensor operating at 20 FPS. We follow the official 700/150/150 train/val/test split in the camera-only setting, using the six surround-view RGB images at 1600×864 as input, with LiDAR used only for ground truth generation.

We report the official nuScenes metrics for 3D object detection: mean Average Precision (mAP), computed via center-distance matching at thresholds $D = \{0.5, 1.0, 2.0, 4.0\}$ meters, and the nuScenes Detection Score (NDS), a composite metric combining mAP with five error terms: translation error (mATE), scale error (mASE), orientation error (mAOE), velocity error (mAVE), and attribute error (mAAE). Lower values are better for all error metrics.

4.2. Implementation Details

The image encoder uses a ResNet-101-DCN backbone pretrained on FCOS3D, with a learning rate multiplier of 0.1 to prevent overfitting the pretrained weights. The Gaussian lifter and encoder follow the configuration of GaussianFormer-2 [12], producing between 6,400 and 25,600 Gaussians per frame depending on the experiment. We adopt a two-stage training procedure: the image encoder, Gaussian lifter, and Gaussian encoder are trained in the first stage, after which FSD V2 is trained separately on the resulting Gaussian point clouds.

FSD V2 is trained with AdamW (weight decay 0.01) and a cyclic learning rate schedule. PointPillars and FSD V2 are trained for 24 epochs, and FocalFormer3D for 20 epochs, following the schedules recommended in their original papers. Batch size is set to 16 for single-frame experiments

and 32 for multi-frame configurations. All experiments are implemented using the MMDetection3D [5] framework and trained on 4 NVIDIA L40 GPUs.

4.3. Main Results

Table 1 compares GaussianDet3D against camera-based methods on the nuScenes validation set, all using ResNet101-DCN backbones for fair comparison. GaussianDet3D achieves the lowest translation error (mATE = 0.6494) and velocity error (mAVE = 0.3400) among all compared methods, outperforming BEVFormer by 3.3% and 14.3% respectively. While BEVFormer leads in overall mAP and NDS, GaussianDet3D demonstrates a clear advantage in the metrics most safety-critical for autonomous driving. Orientation estimation remains challenging (mAOE = 0.5373), a limitation we discuss in Section 5.

Table 2 reports results on the nuScenes test set. GaussianDet3D achieves the lowest translation error (mATE = 0.580), velocity error (mAVE = 0.378), and attribute error (mAAE = 0.129) among all compared methods, outperforming BEVFormer by 8.1%, 13.1%, and 9.8% respectively. These gains are consistent with or stronger than those observed on the validation set, confirming that temporal Gaussian point clouds generalize effectively to unseen data.

4.4. Ablation Studies

Detector backbone. Table 3 compares three LiDAR-style detectors on single-frame Gaussian point clouds (6,400 Gaussians). Each detector is trained for the number of epochs recommended in its original paper: 24 epochs for PointPillars and FSD V2, and 20 epochs for FocalFormer3D. FocalFormer3D requires a two-stage training procedure as prescribed in its original paper: we first train a DeformFormer3D backbone on Gaussian point clouds to initialize the weights, then fine-tune the full FocalFormer3D detector with its multi-stage heatmap encoder and box-level deformable decoder.

All three are viable consumers of Gaussian point clouds, validating Gaussians as a general input modality compatible with different detection architectures. FSD V2 achieves the highest mAP (0.3198) and NDS (0.3676), confirming that its fully sparse 3D convolutional architecture is best suited to the irregular, feature-rich structure of Gaussian point clouds. FocalFormer3D achieves competitive NDS (0.3628) through its transformer-based attention mechanism, while PointPillars provides a strong lightweight baseline despite its simpler pillar-based voxelization.

Temporal aggregation. Table 4 shows that adding frames consistently improves detection across all metrics. The most impactful gain comes from going from 1 to 2 frames, which cuts mAVE by 64.0% (from 1.2596 to 0.4536) as the model gains access to cross-frame motion cues, while NDS improves substantially from 0.3619 to 0.4386. Extending to 4 frames further reduces mAVE to 0.3477 and improves NDS to 0.4637, confirming that longer temporal windows provide richer motion context. mATE improves steadily from 0.7326 to 0.6811 as more frames are aggregated, demonstrating that temporal Gaussian point clouds help resolve depth ambiguities inherent in camera-based perception. Orientation error (mAOE) also benefits from temporal context, dropping from 0.7084 to 0.5685 across the 1 to 4 frame progression.

Gaussian feature components. Table 5 ablates the contribution of each Gaussian parameter group incrementally. Each addition yields consistent gains: scale improves mAP by 10.8% over the position+opacity baseline by encoding object shape, and rotation adds a further 9.0% by capturing primitive orientation. Adding the full 17-channel semantic logits produces the largest single jump, bringing mAP from 0.2156 to 0.2664 (23.6% gain) and achieving the best NDS (0.3187), lowest mATE (0.7616), and lowest mAVE (1.0664). These results confirm that geometric and semantic features are complementary, and validate our design choice to pass all 28 Gaussian parameters to the detector.

Effect of Gaussian density. Increasing Gaussian density consistently improves detection across both single-frame and multi-frame settings. In the single-frame setting with PointPillars (Table 6), going from 6,400 to 25,600 Gaussians per frame improves mAP by 6.1% and NDS from 0.3188 to 0.3300, confirming that denser representations capture finer geometric detail and reduce information loss during backprojection. This trend holds in the multi-frame setting (Table 7): with 4 frames and NMS optimization (threshold 0.05), increasing from 6,400 to 25,600 Gaussians per frame improves mAP by 6.6% (from 0.3564 to 0.3801), NDS from 0.4723 to 0.4903, and mAVE from 0.3487 to 0.3400, with mATE also improving from 0.6666 to 0.6494. The 25,600 configuration requires opacity pruning (≤ 0.01) to maintain practical inference efficiency, yet still achieves our best overall results reported in Table 1.

NMS threshold. Lowering the NMS IoU threshold from 0.25 to 0.05 improves mAP by 4.4% (0.3507 $\rightarrow$ 0.3663). The gain is class-dependent: pedestrian AP improves by 16.8% and traffic cone AP by 22.4%, while large vehicles are unaffected. This reflects the Gaussian representation's tendency to produce multiple nearby hypotheses for small objects, which aggressive NMS suppresses unnecessarily.

4.5. Runtime Analysis

Table 8 provides a latency breakdown of the GaussianDet3D pipeline measured on a single NVIDIA L40 GPU

Table 1. Comparison on nuScenes **validation** set. All methods use ResNet101-DCN. **Bold** = best.

Method	Frames	mATE↓	mASE↓	mAVE↓	mAOE↓	mAAE↓	mAP↑	NDS↑
DETR3D [22]	1	0.716	0.268	0.842	0.379	0.200	0.349	0.434
Focal-PETR [21]	1	0.678	**0.263**	0.804	0.395	0.202	0.390	0.461
PETR [18]	1	0.717	0.267	0.834	0.412	**0.190**	0.366	0.441
PolarDETR [3]	2	0.707	0.269	0.518	**0.344**	0.196	0.383	0.488
BEVFormer [16]	4	0.672	0.274	0.397	0.369	0.198	**0.415**	**0.517**
GaussianDet3D (ours)	4	**0.649**	0.272	**0.340**	0.537	0.199	0.380	0.490

Table 2. Comparison on nuScenes **test** set. All methods use ResNet101-DCN. **Bold** = best.

Method	Frames	mATE↓	mASE↓	mAVE↓	mAOE↓	mAAE↓	mAP↑	NDS↑
DETR3D [22]	1	0.641	0.255	0.845	**0.394**	0.133	0.412	0.479
Focal-PETR [21]	1	0.617	**0.250**	0.862	0.398	0.146	0.426	0.486
PETR [18]	1	0.647	0.251	0.933	0.433	0.143	0.391	0.455
PolarDETR [3]	1	0.588	0.253	0.845	0.408	**0.129**	0.431	0.493
BEVFormer [16]	4	0.631	0.257	0.435	0.405	0.143	**0.445**	**0.535**
GaussianDet3D (ours)	4	**0.580**	0.253	**0.378**	0.500	**0.129**	0.398	0.514

Table 3. Single-frame detector comparison (1 frame, 6,400 Gaussians). FSD V2 uses NMS threshold of 0.05.

Model	mAP	NDS
PointPillars	0.2677	0.3188
FocalFormer3D	0.3016	0.3628
FSD V2	**0.3198**	**0.3676**

Table 4. Temporal aggregation results with FSD V2, sequential sampling, batch 32, 24 epochs, and NMS threshold of 0.25.

Fr.	Gauss.	Total	mATE↓	mASE↓	mAVE↓	mAOE↓	mAP↑	NDS↑
1	6400	6400	0.7326	0.2772	1.2596	0.7084	0.3115	0.3619
2	6400	12800	0.7182	0.2774	0.4536	0.5981	0.3284	0.4386
4	6400	25600	**0.6811**	**0.2772**	**0.3477**	**0.5685**	**0.3423**	**0.4637**

Table 5. Gaussian feature ablation using PointPillars, 6,400 Gaussians, single frame.

Ch.	Components	mAP	NDS	mATE	mAVE
4	mean + opacity	0.1786	0.2655	0.7762	1.1194
7	+ scale	0.1979	0.2842	0.7888	1.1088
11	+ rotation	0.2156	0.2880	0.7751	1.1967
28	+ semantics	**0.2664**	**0.3187**	**0.7616**	**1.0664**

using PyTorch without runtime optimizations such as model quantization or kernel fusion, averaged over 2,000 samples,

Table 6. Effect of Gaussian density on detection performance (PointPillars, 1 frame, 24 epochs).

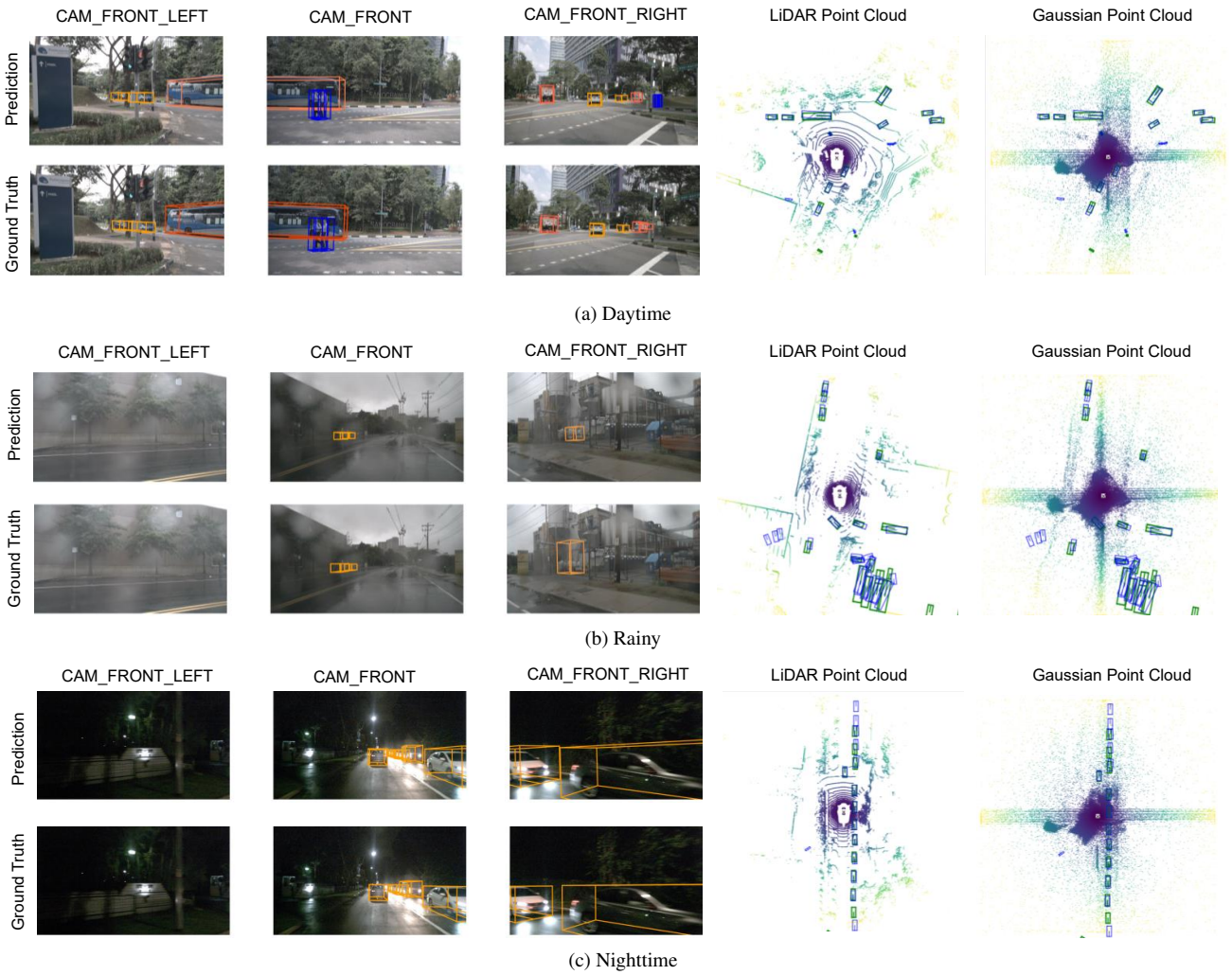
Gaussians/frame	mAP	NDS
6,400	0.2677	0.3188
25,600	**0.2840**	**0.3300**

Table 7. Gaussian density in the 4-frame setting with NMS optimization (threshold 0.05). *opacity $\leq$ 0.01 pruned to maintain practical inference efficiency.

Gauss./frame	Total	mATE↓	mAVE↓	mAOE↓	mAP↑	NDS↑
6,400	25,600	0.6666	0.3487	0.5644	0.3564	0.4723
25,600*	∼80,000	**0.6494**	**0.3400**	**0.5373**	**0.3801**	**0.4903**

for both the 6,400 and 25,600 Gaussian configurations.

The Gaussian lifter is the most computationally expensive component, taking 291.72 ms and 582.70 ms at 6,400 and 25,600 Gaussians respectively, reflecting the cost of per-pixel occupancy distribution estimation and FPS-based anchor selection. The image encoder contributes 185 ms for both configurations, invariant to Gaussian density, while the Gaussian encoder scales more noticeably from 35.10 ms to 113.26 ms. Converting refined Gaussians to a point cloud (reported as *other* in Table 8) is negligible ($\leq$0.04 ms) in both configurations. FSD V2 inference adds 68.56 ms and 98.73 ms respectively, demonstrating that the sparse detector remains highly efficient even when receiving the full 28-



(a) Daytime

(b) Rainy

(c) Nighttime

Figure 3. Qualitative results under diverse conditions. BEV predictions in **blue**, ground truth in **green**.

Table 8. Latency breakdown of GaussianDet3D on a single NVIDIA L40 GPU, averaged over 2,000 samples. No runtime optimizations applied.

Gaussians/frame	Latency (ms)					Total (ms)	NDS↑
	Image Enc.	Lifter	Gaussian Enc.	FSD V2	other		
6,400	185.95	291.72	35.10	68.30	0.03	581.10	0.4723
25,600	188.35	582.70	113.26	98.73	0.04	983.08	0.4903

parameter Gaussian point cloud as input. The increased latency of the 25,600-Gaussian configuration (983.08 ms vs. 581.10 ms) yields a meaningful NDS gain from 0.4723 to 0.4903, reflecting a favorable accuracy–efficiency tradeoff for applications where detection quality outweighs inference speed. Thus, the Gaussian lifter represents the most promising target for future speedup through efficient depth estimation or learned sparse sampling strategies.

4.6. Qualitative Results

Figure 3 shows detection results across three conditions: daytime, rainy, and nighttime. GaussianDet3D performs robustly under favorable and moderate conditions, with strong BEV localization alignment. Under rain and night, detection range decreases and distant objects are more frequently missed, consistent with the inherent sensitivity of camera-based methods to illumination and weather. Across all conditions, the Gaussian point cloud representation preserves sufficient geometric and semantic structure to enable accurate localization of nearby objects, with predicted boxes showing close alignment to ground truth in BEV. This confirms that GaussianDet3D generalizes well beyond controlled settings, demonstrating the promise of Gaussian Splatting as a robust foundation for camera-only 3D perception.

5. Conclusion

We presented GaussianDet3D, the first method to apply 3D Gaussian Splatting from multi-view images to 3D object detection in autonomous driving. By treating refined Gaussian primitives as pseudo-LiDAR points and feeding them directly into a fully sparse detector, we demonstrated that the Gaussian representation provides richer geometric and semantic information than conventional BEV or voxel-based intermediate representations. On the nuScenes benchmark, GaussianDet3D achieves state-of-the-art translation error, velocity error, and attribute estimation among all camera-based methods, outperforming BEVFormer by 8.1% and 13.1% in mATE and mAVE respectively. Ablation studies confirm that temporal aggregation of Gaussian point clouds is the primary driver of velocity estimation accuracy, and that all the Gaussian parameters contribute meaningfully to detection quality. More broadly, this work demonstrates that sparse, adaptive scene representations are a viable and promising foundation for camera-only 3D detection, opening new directions for efficient and scalable perception systems.

The main limitation of our approach is orientation estimation, where GaussianDet3D lags behind prior methods. We attribute this to the quaternion rotation representation of Gaussians, which introduces angular ambiguity that the detector struggles to resolve from camera-only inputs. Addressing this through alternative rotation parameterizations such as axis-angle or Euler angles, combined with geometric and temporal consistency constraints, is an important direction for future work.

Beyond orientation, we identify four further directions: learned pruning strategies that replace fixed opacity thresholds with detection-loss-guided masks; improved robustness under adverse conditions through weather-aware data augmentation; inference speedup through a more efficient Gaussian lifter; and extension to non-automotive domains such as robotics and industrial safety, where camera-based perception must remain reliable under occlusion and varying object distances [20].

We leave the development of more efficient Gaussian generation strategies to future work on fast 3D Gaussian reconstruction.

Acknowledgments This work was supported by the ERC Advanced Grant SIMULACRON, the Georg Nemetschek Institute project AI4TWINNING, and the DFG project 4D-YouTube CR 250/26-1. It is a result of the joint research project STADT:up. The project was also supported by the German Federal Ministry for Economic Affairs and Climate Action (BMWK), based on a decision of the German Bundestag. The author is solely responsible for the content of this publication.

References

[1] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019. 2, 4

[2] Anh-Quan Cao and Raoul de Charette. Monoscene: Monocular 3d semantic scene completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3991–4001, 2022. 2

[3] Shaoyu Chen, Xinggang Wang, Tianheng Cheng, Qian Zhang, Chang Huang, and Wenyu Liu. Polar parametrization for vision-based surround-view 3d detection. *arXiv:2206.10965*, 2022. 6

[4] Yilun Chen, Zhiding Yu, Yukang Chen, Shiyi Lan, Anima Anandkumar, Jiaya Jia, and Jose M Alvarez. Focalformer3d: Focusing on hard instance for 3d object detection. 2023. 2

[5] MMDetection3D Contributors. MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. https://github.com/open-mmlab/mmdetection3d, 2020. 5

[6] Lue Fan, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Fsd v2: Improving fully sparse 3d object detection with virtual voxels. *arXiv preprint arXiv:2308.03755*, 2023. 2, 3

[7] Georg Hess, Carl Lindström, Maryam Fatemi, Christoffer Petersson, and Lennart Svensson. Splatad: Real-time lidar and camera rendering with 3d gaussian splatting for autonomous driving. *arXiv preprint arXiv:2411.16816*, 2024. 2

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

[9] Junjie Huang, Xinzhu Wang, Guan Huang, Zhuang Liu, and Zheng Zhu. Bevdet4d: Exploit temporal cues in multi-camera 3d object detection. In *arXiv preprint arXiv:2203.17054*, 2022. 2

[10] Yuanhui Huang, Wenzhao Zheng, Yunpeng Zhang, Jie Zhou, and Jiwen Lu. Tri-perspective view for vision-based 3d semantic occupancy prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9223–9232, 2023. 2

[11] Yuanhui Huang, Wenzhao Zheng, Yunpeng Zhang, Jie Zhou, and Jiwen Lu. Gaussianformer: Scene as gaussians for vision-based 3d semantic occupancy prediction. *arXiv preprint arXiv:2405.17429*, 2024. 2

[12] Yuanhui Huang, Amonnut Thammatadatrakoon, Wenzhao Zheng, Yunpeng Zhang, Dalong Du, and Jiwen Lu. Gaussianformer-2: Probabilistic gaussian superposition for efficient 3d occupancy prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 27477–27486, 2025. 2, 3, 4

[13] Haoyi Jiang, Liu Liu, Tianheng Cheng, Xinjie Wang, Tianwei Lin, Zhizhong Su, Wenyu Liu, and Xinggang Wang. Gausstr: Foundation model-aligned gaussian transformer for self-supervised 3d spatial understanding. In *CVPR*, 2025. 2

[14] 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. 2

[15] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12697–12705, 2019. 2

[16] Zhiqi Li, Wenhai Yu, Li Yuan, Chunjing Peng, Hang Xu, and Jifeng Yan. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In *European Conference on Computer Vision (ECCV)*, pages 1–18, 2022. 2, 6

[17] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 3

[18] Yingfei Liu, Tiancai Wang, Xiangyu Zhang, and Jian Sun. Petr: Position embedding transformation for multi-view 3d object detection. *arXiv preprint arXiv:2203.05625*, 2022. 2, 6

[19] Shu-Wei Lu, Yi-Hsuan Tsai, and Yi-Ting Chen. Toward real-world bev perception: Depth uncertainty estimation via gaussian splatting. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pages 17124–17133, 2025. 2

[20] Malaz Tamim, Andrea Matic-Flierl, and Karsten Roscher. A comparative study of 3d person detection: Sensor modalities and robustness in diverse indoor and outdoor environments. In *Proceedings of the 21st International Conference on Computer Vision Theory and Applications - Volume 3: VISAPP*, pages 66–74. INSTICC, SciTePress, 2026. 8

[21] S. Wang, X. Jiang, and Y. Li. Focal-petr: Embracing foreground for efficient multi-camera 3d object detection. *ArXiv*, 2022. 2, 6

[22] Yue Wang, Vitor Guizilini, Tianyuan Zhang, Yilun Wang, Hang Zhao, and Justin M. Solomon. Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. In *The Conference on Robot Learning (CoRL)*, 2021. 2, 6

[23] Yi Wei, Linqing Zhao, Wenzhao Zheng, Zheng Zhu, Jie Zhou, and Jiwen Lu. Surroundocc: Multi-camera 3d occupancy prediction for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 21729–21740, 2023. 3

[24] Xiaoyu Zhou, Zhiwei Lin, Xiaojun Shan, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. Drivinggaussian: Composite gaussian splatting for surrounding dynamic autonomous driving scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21634–21643, 2024. 2